# Proceedings of the 21ˢᵗ International Society for Music Information Retrieval Conference

Montréal, Québec, Canada
Virtual Conference
11 to 16 October 2020

978-0-9813537-0-8

ISMIR Organizers

McGill
Schulich School of Music
École de musique Schulich

Université
de Montréal

CIR
MMT
Centre for Interdisciplinary Research
in Music Media and Technology

WOMEN in
MUSIC
information
RETRIEVAL

ISMIR

ISMIR Sponsors

Funder

SSHRC = CRSH

Platinum Sponsors

Spotify®

Sony CSL
SONY

Gold Sponsors

ByteDance

Adobe

Silver Sponsors

steinberg

DOLBY.

chordify

Bronze Sponsors

cochl.

musixmatch

ACRCloud

YOUSICIAN

MUSIIO

SiriusXM pandora

Google

WiMIR Sponsors

Patrons

Contributors

Supporters

Conference Chairs

General Chairs
Audrey Laplante *(Université de Montréal, Montréal, Canada)*
Ichiro Fujinaga *(McGill University, Montréal, Canada)*

Scientific Program Chairs
Julie Cumming *(McGill University, Montréal, Canada)*
Jin Ha Lee *(University of Washington, Seattle, USA)*
Brian McFee *(New York University, New York, USA)*
Markus Schedl *(Johannes Kepler University, Linz, Austria)*

Publication Chairs
Johanna Devaney *(Brooklyn College and the CUNY Graduate Center, New York, USA)*
Cory McKay *(Marianopolis College, Montréal, Canada)*
Eva Zangerle *(University of Innsbruck, Innsbruck, Austria)*

Tutorial Chairs
Blair Kaneshiro *(Stanford University, Stanford, USA)*
Mohamed Sordo *(Pandora, Oakland, USA)*

Late-Breaking / Demo Chairs
Christian Frisson *(McGill University, Montréal, Canada)*
Xiao Hu *(University of Hong Kong, Hong Kong, China)*

Satellite Events Chairs
Jordan B. L. Smith *(ByteDance / TikTok, London, UK)*
Finn Upham *(McGill University, Montréal, Canada)*

Women in MIR Chairs
Claire Arthur *(Georgia Institute of Technology, Atlanta, USA)*
Katherine M. Kinnaird *(Smith College, Northampton, USA)*

Sponsorship Chairs
Elena Georgieva *(New York University, New York, USA and Stanford University, Stanford, USA)*
Zhengshan Shi *(Stanford University, Stanford, USA)*

Industry Chairs
Rachel Bittner *(Spotify, New York, USA)*
Thor Kell *(Spotify, New York, USA)*

Music Chairs
Guillaume Boutard *(Université de Montréal, Montréal, Canada)*
Gabriel Vigliensoni *(McGill University, Montréal, Canada and Goldsmiths, University of London, London, UK)*

Virtual Technology Chair
Finn Upham *(McGill University, Montréal, Canada)*

Website and Conference Management Software Chairs
Néstor Nápoles López *(McGill University, Montréal, Canada)*
Evan Savage *(McGill University, Montréal, Canada)*

Local Arrangements Chairs
Emily Hopkins *(McGill University, Montréal, Canada)*
Gabriel Vigliensoni *(McGill University, Montréal, Canada and Goldsmiths, University of London, London, UK)*

Volunteers Chair
Caroline Traube *(Université de Montréal, Montréal, Canada)*

Program Committee

Meta-Reviewers

Kat Agres, National University of Singapore
Claire Arthur, Georgia Institute of Technology
Andreas Arzt, Apple
Christine Bauer, Johannes Kepler University Linz
Emmanouil Benetos, Queen Mary University of London
Rachel Bittner, Spotify
John Ashley Burgoyne, University of Amsterdam
Carlos Eduardo Cancino-Chacón, Austrian Research Institute for Artificial Intelligence
Estefania Cano, Fraunhofer IDMT
Mark Cartwright, New York University
Kahyun Choi, Indiana University Bloomington
Ching-Hua Chuan, University of Miami
Roger Dannenberg, School of Computer Science, Carnegie Mellon University
Matthew Davies, CISUC - Centre for Informatics and Systems of the University of Coimbra
Christian Dittmar, Fraunhofer IIS
Simon Dixon, Queen Mary University of London
Chris Donahue, Stanford
Stephen Downie, University of Illinois at Urbana-Champaign
Zhiyao Duan, University of Rochester
Andreas Ehmann, Pandora
Gyorgy Fazekas, Queen Mary University of London
Arthur Flexer, Johannes Kepler University Linz
Emilia Gomez, Universitat Pompeu Fabra
Masataka Goto, National Institute of Advanced Industrial Science and Technology (AIST)
Fabien Gouyon, Pandora/SiriusXM
Dorien Herremans, Singapore University of Technology and Design
Andre Holzapfel, KTH Royal Institute of Technology in Stockholm
Xiao Hu, The University of Hong Kong
Cheng-Zhi Anna Huang, Google Brain
Ozgur Izmirli, Connecticut College
Dietmar Jannach, University of Klagenfurt
Blair Kaneshiro, Stanford University
Katherine Kinnaird, Smith College
Peter Knees, TU Wien
Audrey Laplante, Université de Montréal
Olivier Lartillot, RITMO, University of Oslo
Alexander Lerch, Georgia Tech Center for Music Technology
Florence Leve, Université de Picardie Jules Verne - Lab. MIS - Algomus
Cynthia Liem, Delft University of Technology
Michael Mandel, Brooklyn College, CUNY
Cory McKay, Marianopolis College
Andrew McLeod, EPFL
Matt McVicar, Apple
Meinard Müller, International Audio Laboratories Erlangen
Hema Murthy, IIT Madras
Oriol Nieto, Pandora
Johan Pauwels, Queen Mary University of London
Geoffroy Peeters, LTCI – Télécom Paris, IP Paris
Jordi Pons, Dolby Laboratories
Marcelo Queiroz, University of São Paulo
Preeti Rao, IIT Bombay
David Rizo, Universidad de Alicante
Justin Salamon, Adobe Research
Alexander Schindler, Austrian Institute of Technology
Jan Schlüter, JKU Linz
Xavier Serra, Universitat Pompeu Fabra

Jordan Smith, TikTok
Mohamed Sordo, Pandora
Bob Sturm, KTH Royal Institute of Technology
Li Su, Academia Sinica
Douglas Turnbull, Ithaca College
George Tzanetakis, University of Victoria
Julián Urbano, Delft University of Technology
Peter Van Kranenburg, Utrecht University; Meertens Institute
Gabriel Vigliensoni, McGill University
Cheng-i Wang, Smule, Inc.
Christof Weiss, International Audio Laboratories Erlangen
Frans Wiering, Utrecht University
Yi-Hsuan Yang, Academia Sinica
Kazuyoshi Yoshii, Kyoto University
Eva Zangerle, University of Innsbruck

Reviewers

Sajjad Abdoli, École de technologie supérieure, Montréal
Jakob Abeßer, Fraunhofer IDMT
Stefanie Acevedo, University of Dayton
Byron Alejandro Acuña Acurio, Universidade Estadual de Campinas
Taketo Akama, Sony CSL
Islah Ali-Maclachlan, Birmingham City University
Anna Aljanaki, University of Tartu
Nazareno Andrade, Universidade Federal de Campina Grande
Tom Arjannikov, University of Victoria
Vipul Arora, IIT Kanpur
Stefan Balke, Johannes Kepler University Linz
Camila Barros, Federal University of Santa Catarina (UFSC), Brazil
Dogac Basaran, Audible Magic
Oded Ben-Tal, Kingston University
Gilberto Bernardes, INESC TEC & University of Porto, Faculty of Engineering
Paolo Bientinesi, Umeå Universitet
Louis Bigo, Université de Lille
Merlijn Blaauw, Universitat Pompeu Fabra
Ashlae Blume, Recurse Center
Dmitry Bogdanov, Universitat Pompeu Fabra
Geoffray Bonnin, LORIA
Juanjo Bosch, UPF Barcelona
Nicolas Bouillot, Société des arts technologiques (Montreal, Canada)
Paul Brossier, aubio
Dan Brown, University of Waterloo
Fred Bruford, Queen Mary University of London / Roli
Nicholas J. Bryan, Adobe Research
Michele Buccoli, BdSound S.r.l. (former Politecnico di Milano)
Bryony Buck, Institute of Musicians' Medicine
Marcelo Caetano, CNRS-PRISM
Jorge Calvo-Zaragoza, University of Alicante
Pavel Campr, pex.com; before University of West Bohemia, Pilsen, Czech Republic
Giorgia Cantisani, Telecom ParisTech
Julio Carabias, University of Jaen

Rafael Caro Repetto, Music Technology Group, Universitat Pompeu Fabra, Barcelona
Benjamin Carterette, Spotify
Hugo Carvalho, Federal University of Rio de Janeiro
Pritish Chandna, Universitat Pompeu Fabra
Santosh Chapaneri, St. Francis Institute of Technology, Mumbai
Bo-Yu Chen, Academia Sinica
Ning Chen, East China University of Science and Technology
Tsung-Ping Chen, Academia Sinica
Yu-Hua Chen, National Taiwan University
Kai-Hsiang Cheng, Academia Sinica
Tian Cheng, National Institute of Advanced Industrial Science and Technology (AIST)
Diana Estefania Chérrez Barragn, Universidade Estadual de Campinas
Kin Wai Cheuk, Singapore University of Technology and Design
Paulo Chiliguano, Audio Engineering Society
Keunwoo Choi, Spotify
Shreyan Chowdhury, Johannes Kepler University Linz
Chia-Hao Chung, GICE, NTU
Ana Clemente, University of the Balearic Islands
Nathaniel Condit-Schultz, Georgia Institute of Technology
Bas Cornelissen, University of Amsterdam
Albin Correya, Universitat Pompeu Fabra
Helena Cuesta, Universitat Pompeu Fabra
Antonio Deusany de Carvalho Junior, USP
David de Matos, INESC ID Lisboa
Timothy de Reuse, McGill University
Reinier de Valk, Moodagent
Alessio Degani, UNIBS
Ken Déguernel, EPFL
Andrew Demetriou, Delft University of Technology
Junqi Deng, Alibaba
Subodh Deolekar, Lead Research Engineer, REDX Innovation Lab
Diego Di Carlo, INRIA
Bruno Di Giorgi, Apple

Makris Dimos, Singapore University of Technology and
    Design
Hao-Wen Dong, UC San Diego
Guillaume Doras, Ircam
Jonathan Driedger, Chordify
Konstantinos Drossos, Tampere University
Jake Drysdale, Birmingham City University
Anders Elowsson, KTH Royal Institute of Technology
Jeffrey Ens, Simon Fraser University
Elena Epure, Deezer Research
Philippe Esling, Institut de Recherche et Coordination
    Acoustique/Musique (IRCAM)
Behnam Faghih, Computer science department,
    Maynooth University, Maynooth, Co Kildare, Ireland
Ashkan Fakhrtabatabaie, University of Utah
Felipe Falcão, Universidade Federal de Campina
    Grande
Jianyu Fan, Simon Fraser University
Delia Fano Yela, Chordify
Andres Ferraro, Music Technology Group - Universitat
    Pompeu Fabra
Flavio Figueiredo, UFMG
Frederic Font, Music Technology Group - Universitat
    Pompeu Fabra
Klaus Frieler, University of Music FRANZ LISZT
    Weimar
Magdalena Fuentes, New York University
Satoru Fukayama, National Institute of Advanced
    Industrial Science and Technology (AIST)
Diego Furtado Silva, Universidade Federal de São
    Carlos
Nick Gang, Apple, Inc.
Kaustuv Kanti Ganguli, New York University Abu
    Dhabi
Roman Gebhardt, Cyanite / Audio Communication
    Group, TU Berlin
Elena Georgieva, Stanford University
François Germain, iZotope, Inc.
Jon Gillick, UC Berkeley
Mathieu Giraud, CNRS, Université de Lille
Aggelos Gkiokas, Universitat Pompeu Fabra
Juan Gómez-Cañón, Universitat Pompeu Fabra
Samuel Goree, Indiana University
Mark Gotham, Universität des Saarlandes
Richard Groult, Université de Picardie Jules Verne
Ryan Groves, Self-employed
Catherine Guastavino, McGill University
Carlos Guedes, NYU Abu Dhabi
Chitralekha Gupta, National University of Singapore
Siddharth Gururani, Georgia Institute of Technology
Ranjani H G, IISc
Gaétan Hadjeres, Sony Computer Science Laboratories
Jan Hajič, jr., Charles University
Yoonchang Han, Cochlear.ai
Yun Hao, University of Illinois at Urbana-Champaign
Peter Harrison, Max Planck Institute for Empirical
    Aesthetics
Verena Haunschmid, Johannes Kepler University Linz
Curtis Hawthorne, Google Brain
Florian Henkel, Johannes Kepler University Linz
Romain Hennequin, DEEZER
Jorge Herrera, Shazam

Peyman Heydarian, University of Waikato
Jason Hockman, Birmingham City University
Ulf A. S. Holbrook, University of Oslo
Yu-Fen Huang, Academia Sinica
Chris Hubbles, NEDCC
Karim Ibrahim, Telecom-Paristech
Jose Inesta, University of Alicante
Charles Inskip, University College London
Florent Jacquemard, Inria, CNAM-Cédric lab, Paris
Berit Janssen, Utrecht University
Dasaem Jeong, KAIST
Il-Young Jeong, Cochlear.ai
David Johnson, Fraunhofer IDMT
Daniel Jones, Sonos
Anna Jordanous, School of Computing, University of
    Kent
Yaolong Ju, McGill University
Maximos Kaliakatsos-Papakostas,
Stefano Kalonaris, Riken
Tejaswinee Kelkar, University of Oslo
Rainer Kelz, Austrian Research Institute for Artificial
    Intelligence (OFAI)
Jaehun Kim, Delft University of Technology
Jong Wook Kim, OpenAI
Minje Kim, Indiana University
Phillip Kirlin, Rhodes College
Rainer Kleinertz, Saarland University
Eunjeong Koh, UC San Diego
Qiuqiang Kong, ByteDance
Radha Kopparti, City, University of London
Filip Korzeniowski, Pandora Media, Inc.
Amanda Krause, University of Melbourne
Kosmas Kritsis, Athena Research Center
Frank Kurth, Fraunhofer FKIE
Pierre Laffitte, Moodagent
Mathieu Lagrange, École Centrale de Nantes
Robin Laney, Open University
Stefan Lattner, Sony Computer Science Laboratories,
    Paris
Sylvain Le Groux, Voicea.ai / Cisco Systems
Jongpil Lee, KAIST
Kyogu Lee, Seoul National University
Simon Leglaive, CentraleSupelec
Mark Levy, Apple
David Lewis, University of Oxford eResearch Centre
Elisabeth Lex, TU Graz
Bochen Li, University of Rochester
Fanjie Li, The University of Hong Kong
Shengchen Li, Beijing University of Posts and
    Telecommunications
Wei-Hsiang Liao, Sony
Elad Liebman, SparkCognition Research
Baihan Lin, Columbia University
Kin Wah Edward Lin, National Institute of Advanced
    Industrial Science and Technology (AIST), Japan
Yuan-Pin Lin, National Sun Yat-sen University
Meijun Liu, University of Hong Kong
Yi-Wen Liu, National Tsing Hua University
Antoine Liutkus, Inria
Patricio López-Serrano, zplane development
Carlos Lordelo, Queen Mary University of London /
    DoReMIR Music Research AB

Preface

Welcome to ISMIR 2020, the 21st International Society for Music Information Retrieval Conference. ISMIR is the world's leading research forum on processing, searching, organizing, and accessing music-related data. Our community reflects a diversity of scientific disciplines, seniority levels, professional affiliations, and cultural backgrounds. It always has been explicitly interested in fostering and stimulating this diversity, leading to better science and better music services. Due to Covid-19, our 21st ISMIR conference has become our first fully online conference. We hope that this new form of interaction will allow more people, from more different places and communities, to participate. The organizing team in Montreal, and the international team of organizers, reviewers, and meta-reviewers, welcome you to our virtual meeting place.

Scientific Program

A total of 326 abstracts were registered, of which 300 were submitted as full papers eligible for review. The reviewing model was similar to that of previous years, consisting of a two-tier, double-blind process performed by 313 (anonymous) reviewers and 71 meta-reviewers. The scientific program chairs (SPC) anticipated limited availability of reviewers due to the COVID-19 crisis. This, combined with a 20% increase in the number of papers relative to the previous year, led the SPC to actively recruit a 50% larger reviewing pool to reduce the number of papers for which each individual reviewer is responsible. The SPC was happy to welcome a large number of first-time reviewers to the pool, but to ensure that each paper is adequately reviewed, we increased the number of reviewers assigned to each paper. The SPC would also like to thank all community members for their overwhelming response to our requests for additional reviewers and meta-reviewers.

Each paper was assigned to a single meta-reviewer and 4 reviewers, to ensure that each would eventually receive at least 3 completed reviews, accounting for the foreseen limited availability of some reviewers. Meta-reviewers were also instructed to complete a full review of each of their assigned papers, in addition to the final meta-review summarizing the individual reviews. Each meta-reviewer was responsible for between 2 and 5 papers, and each reviewer was responsible for no more than 4 papers. The initial reviewing phase was followed by a discussion period, in which reviewers and meta-reviewers could discuss and revise their assessments of each paper. Meta-reviewers were then instructed to summarize the discussion and reviews in the final report. The SPC finally rendered decisions on each paper.
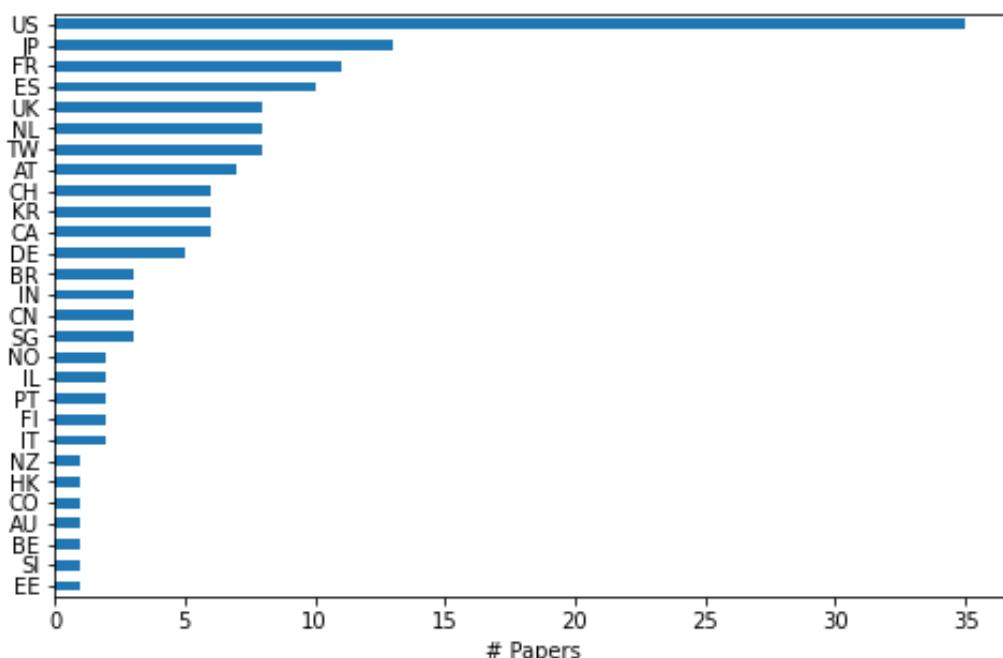
Although the ISMIR 2020 conference is virtual and physical space no longer provided, a constraint on the number of papers we could accept and the allocated time for the conference program resulted in a program of comparable size to previous years. Of the 300 papers reviewed, 115 were accepted for publication, resulting in an acceptance rate of 38.3%. At submission time, authors selected primary and secondary subject areas for their paper. The following table summarizes the number of eligible submissions and accepted papers for each primary category.

Subject Area	Submitted	Accepted	Accept %
Applications	42	15	36
Domain knowledge	61	22	36
Evaluation, datasets, reproducibility	33	10	30
Human-centered MIR	18	8	44
MIR fundamentals	16	7	44
MIR tasks	83	29	35
Musical features and properties	43	22	51
Philosophical and ethical discussions	4	2	50
TOTAL	300	115	38

The table below summarizes the publication statistics over the 21-year-history of the conference.

Year	Location	Oral	Poster	Total	Authors	Unique Authors	Authors / Paper	Unique Authors / Paper
2000	Plymouth	19	16	35	68	63	1.9	1.8
2001	Indiana	25	16	41	100	86	2.4	2.1
2002	Paris	35	22	57	129	117	2.3	2.1
2003	Baltimore	26	24	50	132	111	2.6	2.2
2004	Barcelona	61	44	105	252	214	2.4	2.0
2005	London	57	57	114	316	233	2.8	2.0
2006	Victoria	59	36	95	246	198	2.6	2.1
2007	Vienna	62	65	127	361	267	2.8	2.1
2008	Philadelphia	24	105	105	296	253	2.8	2.4
2009	Kobe	38	85	123	375	292	3.0	2.4
2010	Utrecht	24	86	110	314	263	2.0	2.4
2011	Miami	36	97	133	395	322	3.0	2.4
2012	Porto	36	65	101	324	264	3.2	2.6
2013	Curitiba	31	67	98	395	236	3.0	2.4
2014	Taipei	33	73	106	343	271	3.2	2.6
2015	Málaga	24	90	114	370	296	3.2	2.6
2016	New York	25	88	113	341	270	3.0	2.4
2017	Suzhou	24	73	97	324	248	3.3	2.6
2018	Paris			104	337	265	3.2	2.5
2019	Delft			114	390	315	3.4	2.8
2020	**Montréal / Virtual**			**115**	**426**	**343**	**3.7**	**3.0**

The figure below illustrates the number of papers accepted with at least one contributing author from each country. Geographic affiliations were inferred from self-reported author affiliations and email addresses.

Best Paper Awards

The SPC established 6 paper award categories: Best Research, Best Application, Best New Direction, Best Multi/Interdisciplinary Research, Best Evaluation, and Best Reproducibility. The papers considered for an award were those with the highest review scores, as well as papers nominated for an award by meta-reviewers. We formed a best-paper-award committee comprising 4 highly renowned members of the MIR community and the SPC. Each member of the committee identified their conflicts of interest and judged only the remaining other papers, as well as reviews and meta-reviews of those papers. They then proposed papers from that group for each award category and provided justifications for their decisions. Once all members of the committee had completed their nominations, the whole committee met to make the final decisions.

Best Reviewer Awards

Based on the scores provided by meta-reviewers on the quality of individual reviews, in relation to the number of papers reviewed by each reviewer, the SPC selected a total of 13 awardees.

Organizing a Conference During a Pandemic

The organizing team had to face several challenges while planning this conference. After having planned for a regular, face-to-face meeting in Montreal, we had to come to the conclusion that this year's ISMIR Conference would have to be entirely virtual when it became clear that the COVID-19 pandemic would not be over in time for allowing people to travel and safely attend the conference. After the initial disappointment, however, the team was very enthusiastic to plan for the first virtual ISMIR Conference, and saw this as an opportunity to innovate and create a positive experience for all participants, while ensuring that the conference would be more affordable, accessible, inclusive, and sustainable thanks to this new format.

WiMIR

WiMIR is a group of people dedicated to promoting the role of, and increasing opportunities for, women in the MIR field. WiMIR's initiatives started as informal gatherings around breakfast or lunch during ISMIR conferences (2011–2013), and moved to formal WiMIR events included in the conference program (2015–today) garnering a high turnout of both women and allies. These events provide occasions for people to network and to discuss several important issues ranging from mentorship and conference support, to improving the representation of women and, more broadly, diversity, in the community. In 2018, WiMIR started hosting its own workshop as a satellite event, in which attendees of all genders

participated in high numbers. These workshops aim to offer participants an opportunity for networking, put the spotlight on the work done by women in the field, and foster collaboration between women and allies by proposing group work led by project guides to try to solve small research problems or to undertake new research projects that could lead to longer-term collaborations.

WiMIR and the ISMIR 2020 organizing team worked together to increase women participation and ensure a good representation of women in the conference program. Both conference keynote speakers are esteemed female researchers with an impressive career path. The program also includes 9 meetup sessions with notable women in MIR that were organized by the ISMIR 2020 WiMIR chairs. Additionally, 3 types of grants were offered to female participants: registration grants, publishing fee grants, and, for the first time for an ISMIR conference, childcare grants.

Like for the main conference, the WiMIR workshop organizers had to rethink the event in the context of the pandemics. The workshop consisted this year in 4 sessions that took place over 8 weeks, featured speakers from India, Australia, California, and Europe, and attracted over 450 participants. WiMIR also ran this year the fifth round of its mentoring program, which aims at connecting "women students, postdocs, early-stage researchers, industry employees, and faculty to more senior women and men in MIR who are dedicated to increasing opportunities for women in the field" (https://wimir.wordpress.com/mentoring-program/).

Diversity & Inclusion

Diversity and inclusion are values that are dear to the ISMIR 2020 organizing team. The Black Live Matters movement and research on algorithmic racism have only increased our awareness of the need for more diversity in our field. Knowing that people from the Black community were severely underrepresented among attendees at previous ISMIR conferences, efforts were made to reach out to Black people who could be interested in MIR. Therefore, in addition to the WiMIR grants mentioned above, fee registration waivers were offered to people who self-identified as Black. ISMIR 2020 also featured the first African American keynote speaker in an ISMIR conference, Dr. Safiya U. Noble.

The virtual format of the conference also allowed us to significantly lower the registration fees for the conference. Additionally, it was decided that not only would students be able to benefit from a reduced fee, but also people from low GDP countries, or with low income.

Keynote Speakers

For ISMIR 2020, we are honored to host 2 distinguished keynote speakers:

Safiya U. Noble, Associate Professor, Department of Information Studies, University of California, Los Angeles, presenting "Taking on big tech: New paradigms for new possibilities."

Johanna Devaney, Assistant Professor of Music, Brooklyn College, and Graduate Center, CUNY, presenting "Beyond the current conception of musical performance in MIR" (WiMIR keynote).

Tutorials

ISMIR 2020 tutorial chairs have selected 5 tutorials that will be presented on Sunday, October 11:

Tutorial 1: Prototyping and scaling audio research with Klio by Fallon Chen and Lynn Root;

Tutorial 2: Analysis of expressive timing in recorded music performances by Nico Schüler;

Tutorial 3: Metric learning for music information retrieval by Brian McFee, Jongpil Lee, and Juhan Nam;

Tutorial 4: Open-source tools & data for music source separation: A practical guide for the MIR practitioner by Ethan Manilow, Prem Seetharaman, and Justin Salamont;

Tutorial 5: Version identification in the 20s by Furkan Yesiler, Christopher Tralie, and Joan Serrà.

Late Breaking/Demo and Industry Session

The Late Breaking/Demo (LBD) and Industry session will take place on Friday, October 15. LBD posters feature prototype systems, initial concepts, and early results that have not yet fully matured but are of interest to the MIR community. Proposals went through a light review by the LBD chairs, at the end of which 27 were admitted for presentation. Extended abstracts are available online on the conference website.

Meetup with Industry

As part of ISMIR 2020, a virtual Meetup with Industry series will be hosted. This series includes Interview masterclasses, where industry interviewers will perform mock interviews with volunteers and provide real time feedback, and code review masterclasses on good coding practices from an industry standpoint during which experienced developers will live-review volunteer submitted repositories.

Music

ISMIR music chairs invited composers, technologists, and performers to submit pieces from any musical genre and in any style of electronic, acoustic, or mixed electroacoustic music that explored the notion of music information in the widest sense of the term. The musical works that were submitted were subjected to blind review by the ISMIR 2020 music chairs and an international panel of musicians. The following compositions were selected and will be presented in the 3 music sessions scheduled throughout the conference:

#otherbeats, by Marcel Zaes;

A Room with Chaconne (Bach), by Seth Thorn;

Attempts at Stillness, by Pierre Alexandre Tremblay;

bell / boom, by Jeremy Hyrkas;

Blue Sky Catastrophe, by Seth Shafer;

cecia, by the CECIA team;

DigiTral, by Matheos Zaharopoulos and Georgios Varoutsos;

Generative Sibelius, by Juan Carlos Vasquez;

I'll Marry You, Punk Come, by CJ Carr;

Moon via Spirit, by Lauren Hayes;

Mosaicing, by Panayiotis Kokoras;

Sound | Figuration, by Hongshuo Fan;

Spectre (for processed solo voice), by Max Addae;

Super Colliders, by Takuto Fukuda;

The Pulse of The Sunrise, by Peyman Heydarian;

Transcognition, by Fernando Egido;

Trees, by Alexandra Uitdenbogerd.

Satellite Events

In addition to the main conference, 4 satellite events are offered to participants:

WiMIR 3rd Annual Workshop, held before the conference in 4 sessions across 8 weeks, from August 22 to October 3;

7th International Conference on Digital Libraries for Musicology (DLfM), held on October 16 as a full-day virtual conference;

NLP4MusA: First Workshop on NLP for Music and Audio, held on October 16 and 17 as a virtual conference with live presentations on the 16th that will be streamed on the 17th, and live Q&A sessions on both days;

HAMR Hackathon, held asynchronously over October 16 and 17.

Acknowledgements

Julie Cumming
Jin Ha Lee
Brian McFee
Markus Schedl
Scientific Program Chairs

Ichiro Fujinaga
Audrey Laplante
General Chairs

Contents

Keynote Talks

Keynote Talk

Taking on Big Tech: New Paradigms for New Possibilities

Dr. Safiya U. Noble
Associate Professor in the Departments of Information Studies and African American Studies
Co-Director of the Center for Critical Internet Inquiry (C2i2)
University of California, Los Angeles (UCLA)

Abstract

In her recent best-selling book *Algorithms of Oppression,* Dr. Safiya Noble challenges the idea that "Big Tech" offers an equal playing field for all forms of ideas, identities, and activities. Her work argues that the combination of private interests, along with the monopoly status of a relatively small number of internet companies, leads to a limited understanding of how racism is created, maintained, and disseminated in everyday digital engagements.

Data discrimination is a real social problem, and in this talk, Noble offers a powerful set of data points, examples, and provocations. She asserts we are just at the beginning of creating new paradigms of justice with the tech sector.

Biography

Dr. Safiya Umoja Noble is an Associate Professor at the University of California, Los Angeles (UCLA) in the Department of Information Studies where she serves as the Co-Founder and Co-Director of the UCLA Center for Critical Internet Inquiry (C2i2). She also holds appointments in African American Studies and Gender Studies. She is a Research Associate at the Oxford Internet Institute at the University of Oxford and has been appointed as a Commissioner on the Oxford Commission on AI & Good Governance (OxCAIGG). She is a board member of the Cyber Civil Rights Initiative, serving those vulnerable to online harassment.

Previously, she was a visiting faculty member to the USC Annenberg School for Communication and Journalism, and began her academic career as an Assistant Professor in the College of Media and the Institute of Communications Research at the University of Illinois at Urbana-Champaign.

Dr. Noble is the recipient of a Hellman Fellowship and the UCLA Early Career Award. Her academic research focuses on the design of digital media platforms on the internet and their impact on society. Her work is both sociological and interdisciplinary, marking the ways that digital media impacts and intersects with issues of race, gender, culture, and technology. She is regularly quoted for her expertise on issues of algorithmic discrimination and technology bias by national and international press including *The Guardian,* the BBC, CNN International, *USA Today, Wired, Time, Full Frontal with Samantha Bee, The New York Times,* and Virginia Public Radio, and a host of local news and podcasts, including *Science Friction,* and *Science Friday* to name a few. Recently, she was named in the "Top 25 Doers, Dreamers, and Drivers of 2019" by *Government Technology* magazine.

Dr. Noble is the co-editor of two edited volumes: *The Intersectional Internet: Race, Sex, Culture and Class Online* and *Emotions, Technology & Design*. She currently serves as an Associate Editor for the *Journal of Critical Library and Information Studies,* and is the co-editor of the Commentary & Criticism section of the *Journal of Feminist Media Studies*. She is a member of several academic journal and advisory boards, and holds a Ph.D. and M.S. in Library & Information Science from the University of Illinois at Urbana-Champaign, and a B.A. in Sociology from California State University, Fresno where she was recently awarded the Distinguished Alumni Award for 2018.

WiMIR Keynote Talk

Performance Matters

Beyond the Current Conception of Musical Performance in MIR

Dr. Johanna Devaney
Assistant Professor, Conservatory of Music, Brooklyn College
Faculty in the Data Analysis and Visualization MA program, Graduate Center
Faculty in the Music PhD program, Graduate Center
City University of New York (CUNY)

Abstract

This talk will reflect on what we can observe about musical performance in the audio signal and where MIR techniques have succeeded and failed in enhancing our understanding of musical performance. Since its foundation, ISMIR has showcased a range of approaches for studying musical performance. Some of these have been explicit approaches for studying expressive performance while others implicitly analyze performance with other aspects of the musical audio. Building on my own work developing tools for analyzing musical performance, I will consider not only the assumptions that underlie the questions we ask about performance but what we learn and what we miss in our current approaches to summarizing performance-related information from audio signals. I will also reflect on a number of related questions, including what do we gain by summarizing over large corpora versus close reading of a select number of recordings. What do we lose? What can we learn from generative techniques, such as those applied in style transfer? And finally, how can we integrate these disparate approaches in order to better understand the role of performance in our conception of musical style?

Biography

Johanna Devaney is an Assistant Professor at Brooklyn College and the CUNY Graduate Center. At Brooklyn College she teaches primarily in the Music Technology and Sonic Arts areas and at the Graduate Center she is appointed to the Music and the Data Analysis and Visualization programs. Previously, she was an Assistant Professor of Music Theory and Cognition at Ohio State University and a postdoctoral scholar at the Center for New Music and Audio Technologies (CNMAT) at the University of California at Berkeley. Johanna completed her PhD in music technology at the Schulich School of Music of McGill University. She also holds an MPhil degree in music theory from Columbia University and an MA in composition from York University in Toronto.

Johanna's research focuses on interdisciplinary approaches to the study of musical performance. Primarily, she examines the ways in which recorded performances can be used to study performance practice and develops computational tools to facilitate this. Her work draws on the disciplines of music, computer science, and psychology, and has been funded by the Social Sciences and Humanities Research Council of Canada (SSHRC), the Google Faculty Research Awards program and the National Endowment for the Humanities (NEH) Digital Humanities program.

Tutorials

Tutorial 1

Prototyping and Scaling Audio Research with Klio

Fallon Chen, Lynn Root, Dan Simon and Shireen Kheradpey

Abstract

This tutorial will walk attendees through the use of a Python framework called klio that makes use of the Apache Beam Python SDK to parallelize the execution of audio processing algorithms over a large dataset. Apache Beam is a portable and extensible programming model that unifies distributed batch and streaming processing. It manages the I/O and parallelized execution needed for large-scale data processing. Any audio processing algorithm that can be executed by a Python process and has dependencies that can be installed on machines supported by Apache Beam runners can be run with klio. Audio processing algorithms that have been added to a klio data processing job can be run locally on the practitioner's machine, before being run on large-scale data processing systems like Google Cloud Dataflow. This enables the practitioner to make quick local changes to their algorithm and test it on a few files before deploying a longer running job on more files.

The intended audience of this tutorial are audio processing practitioners who have wrestled with the complexity of iterating upon and coordinating the execution of algorithms that both consume and produce large audio datasets. klio provides best-practice standards and abstractions, encoded in its Python-based command line interface and API, that help audio practitioners prototype, organize and scale their work.

During the tutorial, attendees will receive:

- An overview of klio that establishes core concepts and features
- Guidance through building a klio audio processing graph and running it on an audio dataset

Biographies

Fallon Chen is a Senior Engineer at Spotify, where she builds libraries and tools for audio processing. She holds a M.S. in Computer Science from the University of California, San Diego, and a B.S. in Computer Science from the University of California, Davis. Her favorite genre is industrial techno.

Lynn Root is a Staff Engineer at Spotify and resident FOSS evangelist. She is a seasoned speaker on building and maintaining distributed systems, and maintains Spotify's audio processing framework. Lynn is a global leader of diversity in the Python community, and the former Vice Chair of the Python Software Foundation Board of Directors. When her hands are not on a keyboard, they are usually holding a bass guitar.

Tutorial 2

Analysis of Expressive Timing in Recorded Music Performances
Nico Schüler

Abstract

This tutorial will briefly summarize research on expressive timing in music, present an original research project (as an example) on rubato in four performances of Bach's Invention No. 9, explain and demonstrate how to use the freeware Sonic Visualiser as well as Excel for the analysis of expressive timing in music, and participants will, with the help of the tutorial leader, pursue their own analysis of other performances of Bach's Invention No. 9. (Recordings will be provided.) We will combine the data collected (in Excel files) to look for similarities and differences in the various performances and how expressive timing correlates to certain musical features. (An analytical score of the piece will be provided.) We will collectively formulate research findings.

This tutorial is suitable for anyone who is curious about the topic. Beyond curiosity, participants do not need to have a music or computer science background. Those interested in participating in the analyses should bring a laptop (Windows computer or Mac) to the tutorial, with Sonic Visualiser (http://sonicvisualiser.org) and the VAMP plugin "Note Onset Detector" (http://www.vamp-plugins.org) installed.

Biography

Professor Dr. **Nico Schüler** is University Distinguished Professor of Music Theory & Musicology at Texas State University. His main research interests are computer applications in music research, methods and methodology of music research, interdisciplinary aspects of 19th/20th century music, music theory pedagogy, and music historiography. He has given numerous international workshops on computer-applications in music. He is the editor of the research book series Methodology of Music Research, the author and / or editor of 21 books, and the author of more than 120 articles. Among his most recent books are Musical Listening Habits of College Students (2010) and Computer-Assisted Music Analysis (2014). http://www.nicoschuler.com, mnico.schuler@txstate.edu.

Tutorial 3

Metric Learning for Music Information Retrieval

Brian McFee, Jongpil Lee and Juhan Nam

Abstract

Metric learning is a paradigm of representation learning, in which proximity between the representations of items is optimized to correspond to a notion of similarity. Compared to the classification, metric learning can leverage more flexible forms of supervision, for example, two audio clips belong to the same artist or not, or have the same tempo or not. This enables the learning model to take a large or indefinite number of classes. Moreover, metric learning handles the embedding space directly by measuring the distance between the transformations of different examples. This facilitates usage of different domains or modalities of inputs in the same framework (e.g., audio embedding in one input and word embedding in another input). Such flexible and adaptable characteristics of metric learning have been enjoyed in many of machine learning tasks, particularly, similarity-based content retrieval. In recent years, interest in metric learning from the MIR community has also increased. Considering the multi-faceted and hierarchical-level of notions in similarity (e.g., semantic-level, score-level or audio-level) and diverse forms of data (e.g., audio, MIDI, text labels, lyrics, album covers, and user data), we see a great potential of metric learning in music. Therefore, introducing the method in an educational manner and surveying recent progress will be timely and helpful to relevant researchers. In this tutorial, we plan to present three lectures as follows:

1. **Metric learning foundations:** This lecture introduces mathematical foundations of metric learning.
2. **Deep metric learning and applications to MIR (1): core tasks -** This lecture introduces recent deep metric learning methods and their applications to music classification and similarity-based retrieval tasks.
3. **Deep metric learning and applications to MIR (2): variations -** This lecture introduces various applications of deep metric learning in MIR, showing how researchers have bridged diverse domains and modalities of input in metric learning.

Biographies

Brian McFee is Assistant Professor of Music Technology and Data Science New York University. He received the B.S. degree (2003) in Computer Science from the University of California, Santa Cruz, and M.S. (2008) and Ph.D. (2012) degrees in Computer Science and Engineering from the University of California, San Diego. His work lies at the intersection of machine learning and audio analysis. He is an active open source software developer, and the principal maintainer of the librosa package for audio analysis.

Jongpil Lee received the B.S. degree in electrical engineering from Hanyang University, Seoul, South Korea, in 2015, the M.S. degree, in 2017, from the Graduate School of Culture Technology, Korea Advanced Institute of Science and Technology, Daejeon, South Korea, where he is currently working toward the Ph.D. degree. He interned at Naver Clova Artificial Intelligence Research in the summer of 2017 and at Adobe Audio Research Group in the summer of 2019. His current research interests include machine learning and signal processing applied to audio and music applications.

Juhan Nam is an Associate Professor of the Graduate School of Culture Technology at the Korea Advanced Institute of Science and Technology (KAIST), South Korea. Before joining KAIST, he was a staff research engineer at Qualcomm. Before his research career, he was a software/DSP engineer at Young Chang (Kurzweil). He received the Ph.D. degree (2013) in Music from Stanford University, studying at the Center for Computer Research in Music and Acoustics (CCRMA). He is interested in various research topics at the intersection of music, signal processing, and machine learning.

Tutorial 4

Open-Source Tools & Data for Music Source Separation: A Practical Guide for the MIR Practitioner

Ethan Manilow, Prem Seetharaman and Justin Salamon

Abstract

Musical source separation has become increasingly effective in recent years. As such, applications of music source separation have the potential to touch many aspects of MIR research. However, from a user's perspective, either in doing source separation research from scratch or in applying source separation to other tasks (e.g. polyphonic music transcription), there are still significant roadblocks. Code and data are released on a paper-by-paper basis, making it difficult to compare, use and extend existing techniques. This limits the usefulness of source separation for researchers not actively steeped in its many nuances, and hinders its applicability to broader MIR research.

In this tutorial, we present a set of complementary, easy-to-use, open-source tools and datasets for source separation research, evaluation, and deployment. We show how they interlock with one another, and how they can be used in concert to structure source separation within a project for research or deployment. Finally, we propose a generic and well-tested project structure for efficiently doing modern source separation research, from sweeping over hyperparameters, to setting up competitive baselines, to augmenting your datasets.

Participants of this tutorial will leave with:

1. A practical overview of source separation including history and current research trends.
2. The ability to make educated decisions about how to best include source separation in their workflow.
3. The ability to select the proper separation algorithm or a pre-trained model for their research.
4. The ability to effectively train a custom model for their research using open-source tools.

Our tutorial is aimed at researchers and practitioners that are familiar with audio and machine learning but have little or no experience with source separation. Our primary resources will be the following open-source/data projects: [nussl], [scaper], [Slakh2100], and [MUSDB18]. References to additional tools and datasets (including for non-music audio) will be provided.

Biographies

Ethan Manilow is a PhD candidate in Computer Science at Northwestern University under advisor Prof. Bryan Pardo. His research lies in the intersection of signal processing and machine learning, with a focus on source separation, automatic music transcription, and open source datasets and applications. Previously he was an intern at Mitsubishi Electric Research Labs (MERL) and at Google Magenta. He is one of the lead developers of nussl, an open source audio separation library.

Prem Seetharaman is a research scientist at Descript in San Francisco. Previously, he was a teaching fellow at Northwestern University, where he received his PhD in 2019 advised by Bryan Pardo. The objective of his research is to create machines that can understand the auditory world. He works in computer audition, machine learning, and human computer interaction. He is one of the lead developers of nussl, an open source audio separation library, and Scaper, a library for soundscape generation & augmentation.

Justin Salamon is a research scientist and member of the Audio Research Group at Adobe Research in San Francisco. Previously he was a senior research scientist at the Music and Audio Research Laboratory and Center for Urban Science and Progress of New York University. His research focuses on the application of machine learning and signal processing to audio & video, with applications in machine listening, representation learning & self-supervision, music information retrieval, bioacoustics, environmental sound analysis and open-source software & data. He is the lead developer of Scaper, a library for soundscape generation & augmentation.

Tutorial 5

Version Identification in the 20s

Furkan Yesiler, Christopher Tralie and Joan Serrà

Abstract

The version identification (VI) task concerns detecting and retrieving a set of songs that originate from the same underlying musical composition. Versions (or cover songs) convey the same musical entity while incorporating differences in several musical characteristics, including the differences in timbre, tempo, key, lyrics, and even added/deleted sections. The main applications include digital rights management and music catalog organization.

For more than a decade, VI systems suffered from the accuracy-scalability trade-off, with attempts to increase accuracy resulting in cumbersome, non-scalable systems. Recent years however have witnessed an increase in deep learning-based VI approaches that take a step toward bridging the accuracy-scalability gap, and we start seeing the possibility to deploy such systems in real-world applications. Although this trend positively influences the number of researchers and institutions working on VI, it may also result in obscuring the literature before the deep learning era. To appreciate the 20 years of novel ideas in VI and to facilitate building better systems in the next decade, we believe that now may be the right time to review some of the successful ideas and applications proposed in VI literature and connect them to current systems and ideas.

We will start the tutorial by explaining common input representations and feature post-processing steps. We will continue with comparing the pros and cons of alignment-based and embedding-based approaches, which constitute the two main perspectives for similarity estimation in VI. Lastly, after discussing a number of ideas that can be incorporated into any VI system, we will conclude by presenting the current challenges and future directions in VI research. Our goal is for the audience to leave with a thorough appreciation of both the history of the task and current directions, and that this context will allow them to jump into conducting novel research in the area.

Biographies

Furkan Yesiler is a PhD candidate at the Music Technology Group (MTG) of Universitat Pompeu Fabra (Barcelona). His research is focused on leveraging deep learning techniques to build accurate and scalable music version identification (VI) systems for industrial use cases. His recent contributions include MOVE, a state-of-the-art VI system based on musically-motivated principles; Da-TACOS, a large-scale VI benchmark set; and acoss, an open-source framework for feature extraction and benchmarking designed for VI. He received his MSc in Sound and Music Computing also from the MTG, with a focus on singing voice research. He graduated summa cum laude with two BSc degrees in computer engineering and industrial engineering from Koc University (Istanbul), where he was accepted with a full scholarship. During his bachelor's studies, he did internships in management consulting and M&A advisory companies in Istanbul, managed a student club with 200+ members, participated in a number of rowing competitions and musical theater shows, and spent a trimester at the University of California, Santa Barbara.

Christopher Tralie is an assistant professor in Math and Computer Science at Ursinus College in Collegeville, Pennsylvania, USA. He works in applied geometry/topology and geometric signal processing, and his work spans shape-based music structure analysis and version identification, video analysis, multimodal time series analysis, and geometry-aided data visualization. He received a B.S.E. from Princeton University 2011, a master's at Duke University in 2013, and a Ph.D. at Duke University in 2017, all in Electrical Engineering. His Ph.D. was primarily supported by an NSF Graduate Fellowship, and his dissertation is entitled "Geometric Multimedia Time Series." He did a postdoc at Duke University in Mathematics and a postdoc at Johns Hopkins University in Complex Systems. He was awarded a Bass Instructional Teaching fellowship at Duke University, and he maintains an active interest in pedagogy and outreach, including longitudinal mentoring of underprivileged youths in STEAM (STEM + arts) education.

Joan Serrà is a staff researcher with Dolby Labs in Barcelona since 2019, where he works on deep learning and audio processing. He did his MSc (2007) and PhD (2011) in automatic version identification at the Music Technology Group of Universitat Pompeu Fabra. He also did a postdoc in artificial intelligence at IIIA-CSIC (2011-2015). After that, he was a machine learning researcher at Telefónica R&D (2015-2019). He has had research stays at the Max Planck Institute for the Physics of Complex Systems (2010) and the Max Planck Institute for Computer Science (2011). He has been involved in more than 10 research projects, funded by National and European institutions, and co-authored over 100 publications, many of them highly-cited and in top-tier venues, including NeurIPS, ICLR, ICML, InterSpeech, ICASSP, and ISMIR.

Paper Session 1

MUSIC STRUCTURE ANALYSIS BASED ON AN LSTM-HSMM HYBRID MODEL

Go Shibata   Ryo Nishikimi   Kazuyoshi Yoshii

Graduate School of Informatics, Kyoto University, Japan
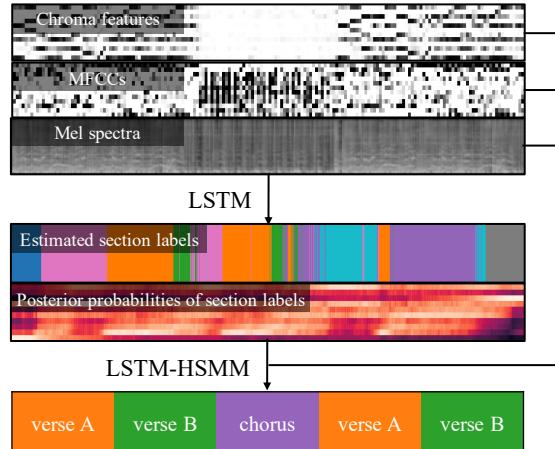
`{gshibata, nishikimi, yoshii}@sap.ist.i.kyoto-u.ac.jp`

ABSTRACT

This paper describes a statistical music structure analysis method that splits an audio signal of popular music into musically meaningful sections at the beat level and classifies them into predefined categories such as *intro*, *verse*, and *chorus*, where beat times are assumed to be estimated in advance. A basic approach to this task is to train a recurrent neural network (*e.g.*, long short-term memory (LSTM) network) that directly predicts section labels from acoustic features. This approach, however, suffers from frequent musically unnatural label switching because the homogeneity, repetitiveness, and duration regularity of musical sections are hard to represent explicitly in the network architecture. To solve this problem, we formulate a unified hidden semi-Markov model (HSMM) that represents the generative process of homogeneous mel-frequency cepstrum coefficients, repetitive chroma features, and mel spectra from section labels, where the emission probabilities of mel spectra are computed from the posterior probabilities of section labels predicted by an LSTM. Given these acoustic features, the most likely label sequence can be estimated with Viterbi decoding. The experimental results show that the proposed LSTM-HSMM hybrid model outperformed a conventional HSMM.

1. INTRODUCTION

Music structure analysis is the fundamental task in the field of music information retrieval (MIR) [1] because the musical structure, which consists of several sections including intro, verse, bridge, and chorus, is one of the most important elements of popular music. Most studies have tackled the segmentation task, which splits audio signals into several sections [2–12], the clustering task, which categorizes such sections into several classes [13–23], or both. Beyond the clustering task that gives arbitrary labels such as "A" and "B" to detected sections, we tackle the labeling task that gives concrete labels such as "verse A", "verse B", and "chorus" [4, 24] because such musically meaningful labels are useful for playback navigation [25]. Because section

Figure 1. Proposed music structure analysis method.

labels are subjective features of music, the labeling task is still challenging. Although deep neural networks (DNNs) have widely been used for frame-level classification tasks in MIR, they often suffer from frequent musically unnatural label switching.

In music structure analysis, the *homogeneity* and *repetitiveness* of acoustic features, the *regularity* of section durations, and the *novelty* of section boundaries, have considered as the four main noticeable aspects of musical sections [1, 11]. Using a sufficient amount of music signals with section label annotations, one could train a labeling DNN in a supervised manner such that the four aspects are learned implicitly. Another approach is to formulate a probabilistic generative model of acoustic features that can explicitly represent the four aspects and infer latent sections from observed features. A hierarchical hidden semi-Markov model (HSMM) based on the homogeneity, repetitiveness, and regularity, for example, has recently been proposed for joint segmentation and clustering [26]. The complementary properties of these approaches call for a hybrid approach for joint segmentation and labeling.

In this paper, we propose a deep generative approach to music structure analysis that integrates the labeling capability of a bidirectional long short-term memory (BLSTM) network into the classical shallow generative framework of the HSMM (Fig. 1). The unified model represents the generative processes of mel-frequency cepstrum coefficients (MFCCs) that are *homogeneous* in each section, chroma features that are *repeated* in sections of the same label, and mel spectra, from sections having *regular* durations. The

BLSTM network that estimates section labels from mel spectra at the frame level is trained in a supervised manner. The emission probabilities of mel spectra from sections are computed at run-time by referring to the posterior probabilities of section labels estimated by the network and the empirical prior distributions of section labels. Given acoustic features, the latent section sequence as well as the initial, transition, duration, and terminal probabilities of sections are estimated in a Bayesian manner with Gibbs sampling followed by Viterbi decoding, where the latent section sequence is initialized by the network to avoid bad local optima.

The main contribution of this paper is to propose a statistical joint segmentation and labeling method based on a Bayesian LSTM-HSMM hybrid model that can be adapted to each musical piece. Because the statistical characteristics of sections are specific to each musical piece, Bayesian inference based on the empirical prior distributions of those characteristics plays an essential role for improving the performance of music structure analysis. We experimentally show that the proposed method significantly outperformed a cascade model using the labeling results of the BLSTM in the post-processing and an LSTM-HSMM model using only Viterbi decoding.

2. RELATED WORK

This section reviews music structure analysis methods in terms of segmentation, clustering, and labeling.

2.1 Segmentation

In the segmentation task, the novelty plays a central role. Foote [2] detected peaks from a novelty curve obtained by convoluting a checkerboard kernel with the diagonal elements of a self-similarity matrix (SSM). Jensen [3] detected section boundaries such that a homogeneity- and novelty-aware cost function is minimized. Goto [4] and Serrà *et al.* [5] proposed novelty curves computed from lag SSMs showing repetitions as vertical lines. These methods were integrated for better segmentation [6] and the method [5] was extended for clustering [7]. Recently, Ullrich *et al.* [8] proposed a supervised method based on a convolutional neural network, which was extended to deal with coarse and fine boundary annotations [9]. Smith *et al.* [10] emphasized the importance of considering the regularity in the main analysis step, not in the post-processing step. Sargent *et al.* [11] focused on the regularity to favor comparable-size sections. Maezawa [12] used an LSTM network based on a cost function considering the homogeneity, repetitiveness, novelty, and regularity.

2.2 Clustering and Labeling

Cooper *et al.* [13] sequentially performed segmentation [2] and clustering based on intra- and inter-section characteristics. Goodwin *et al.* [14] efficiently detected off-diagonal stripes as repetitions from an SSM using dynamic programming. To deal with the repetitiveness and homogeneity, Grohganz *et al.* [15] converted a repetitiveness-aware SSM with off-diagonal stripes into a homogeneity-aware SSM with a block-diagonal structure. Nieto *et al.* [16] used a convex variant of nonnegative matrix factorization for segmentation and clustering. McFee *et al.* [17] encoded repetitive structures into a graph and performed spectral clustering for graph partitioning. Cheng *et al.* [18] converted a path-enhanced SSM into a block-enhanced SSM using nonnegative matrix factor deconvolution as in [15].

Several studies have taken a statistical approach based on generative models for joint segmentation and clustering. Aucouturier *et al.* [19] used a standard HMM. Levy *et al.* [20] proposed an HSMM based on the regularity of section durations. Ren *et al.* [21] proposed a nonparametric Bayesian HMM that can estimate an appropriate number of sections. Barrington *et al.* [22] proposed a nonparametric Bayesian switching linear dynamical system (LDS) that has the ability of automatic model complexity control.

Only a few studies have attempted to estimate musically meaningful labels. Maddage *et al.* [27] proposed a labeling method based on a typical music structure and the role of each section. Paulus *et al.* [24] performed segmentation, clustering, and labeling using a probabilistic fitness measure for the N-grams of sections.

3. PROPOSED METHOD

This section describes the proposed method for music structure analysis.

3.1 Problem Specification

The task we tackle in this paper is specified as follows:

Assumption: The beat times of a target music audio are estimated in advance by a beat tracking method [28].
Input: Beat-level chroma features $\mathbf{X}^c \triangleq \mathbf{x}_{1:T}^c$ ($\mathbf{x}_t^c \in \mathbb{R}^{12}$), MFCCs $\mathbf{X}^m \triangleq \mathbf{x}_{1:T}^m$ ($\mathbf{x}_t^m \in \mathbb{R}^{12}$), and mel spectra $\mathbf{X}^s \triangleq \mathbf{x}_{1:T}^s$ ($\mathbf{x}_t^s \in \mathbb{R}^{128}$) obtained from the target music signal, where T is the number of beats (quarter notes).
Output: Section labels $\mathbf{Z} \triangleq z_{1:N}$ ($z_n \in \{1, \ldots, K\}$) with durations $\mathbf{D} \triangleq d_{1:N}$ ($d_n \in \{1, \ldots, L\}$), where N is the number of sections, K is the number of distinct section labels, and L is the maximum number of beats in a section.

The notation $i{:}j$ represents a set of indices from i to j. Let $\mathbf{X}$ be $\{\mathbf{X}^c, \mathbf{X}^m, \mathbf{X}^s\}$, and $\mathbf{x}$ be $\{\mathbf{x}^c, \mathbf{x}^m, \mathbf{x}^s\}$.

3.2 Model Formulation

As shown in Fig. 2, we formulate a hierarchical HSMM of observed features $\mathbf{X}$ with latent sequences of section labels and abstract chord labels. Let $\mathbf{S} \triangleq \mathbf{S}_{1:N}$ be a sequence of chord sequences, where $\mathbf{S}_n \triangleq s_{n,1:d_n}$ ($s_{n,\tau} \in \{1, \ldots, M\}$) is a chord sequence in section n and M is the maximum number of chords in a section. The full probabilistic model $p(\mathbf{X}, \mathbf{Z}, \mathbf{D}, \mathbf{S})$ is defined as

$$p(\mathbf{X}, \mathbf{Z}, \mathbf{D}, \mathbf{S}) = p(\mathbf{X}|\mathbf{Z}, \mathbf{D}, \mathbf{S})p(\mathbf{S}|\mathbf{Z}, \mathbf{D})p(\mathbf{Z}, \mathbf{D}), \quad (1)$$

where $p(\mathbf{X}|\mathbf{Z}, \mathbf{D}, \mathbf{S})$ is an acoustic model of observed features $\mathbf{X}$, $p(\mathbf{S}|\mathbf{Z}, \mathbf{D})$ is a left-to-right Markov model of chord

labels $\mathbf{S}$ and $p(\mathbf{Z}, \mathbf{D})$ is an ergodic semi-Markov model of section labels $\mathbf{Z}$ with durations $\mathbf{D}$.

3.2.1 Semi-Markov Chain of Section Labels

The ergodic semi-Markov model $p(\mathbf{Z}, \mathbf{D})$ in Eq. (1) represents the generative process of section labels $\mathbf{Z}$ and their durations $\mathbf{D}$ as follows:

$$p(\mathbf{Z}, \mathbf{D}) = p(z_1, d_1) \prod_{n=2}^{N} p(z_n, d_n | z_{n-1}, d_{n-1}), \quad (2)$$

where the individual terms are given by

$$p(z_1, d_1) = \rho_{z_1} \psi_{d_1}, \quad (3)$$

$$p(z_n, d_n | z_{n-1}, d_{n-1}) = \pi_{z_{n-1} z_n} \psi_{d_n}, \quad (4)$$

$$p(z_N, d_N | z_{N-1}, d_{N-1}) = \pi_{z_{N-1} z_N} \psi_{d_N} \upsilon_{z_N}, \quad (5)$$

where ρ_z, $\pi_{zz'}$, and υ_z are the initial, transition, and terminal probabilities of section labels and ψ_d is the duration probability.

3.2.2 Left-to-Right Markov Chain of Chord Labels

The left-to-right Markov model $p(\mathbf{S}|\mathbf{Z}, \mathbf{D})$ in Eq. (1) represents the generative process of chord labels $\mathbf{S}$ as follows:

$$p(\mathbf{S}|\mathbf{Z}, \mathbf{D}) = \prod_{n=1}^{N} p(s_{n,1}) \prod_{\tau=2}^{d_n} p(s_{n,\tau} | s_{n,\tau-1}, z_n), \quad (6)$$

where the individual terms are given by

$$p(s_{n,1} = 1) = 1, \quad (7)$$

$$p(s_{n,\tau} | s_{n,\tau-1}, z_n) = \phi_{s_{n,\tau-1} s_{n,\tau}}^{(z_n)}, \quad (8)$$

where z_n is the corresponding section label and $\phi_{ss'}^{(z)}$ is the transition probability from state s to state s'. The left-to-right Markov model meets a condition that the initial state has $s_{n,1} = 1$ and $s_{n,\tau_1} \leq s_{n,\tau_2}$ for $\tau_1 < \tau_2$. We introduce a hyperparameter σ that describes the maximum number of states that may be skipped in a transition; a transition from state s to state $s + \sigma$ is allowed. In this way, the model allows chord labels to be repeated with some variations in sections of the same label.
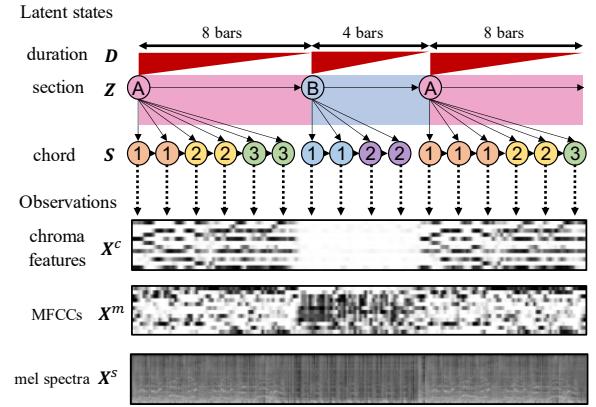
3.2.3 Emission of Acoustic Features

Given that the chroma features $\mathbf{X}^c$, the MFCCs $\mathbf{X}^m$, and the mel spectra $\mathbf{X}^s$ are conditionally and temporally independent, the acoustic model $p(\mathbf{X}|\mathbf{Z}, \mathbf{D}, \mathbf{S})$ in Eq. (1) is factorized as follows:

$$p(\mathbf{X}|\mathbf{Z}, \mathbf{D}, \mathbf{S}) = \prod_{t=1}^{T} \chi_{z_t, s_t}^{c}(\mathbf{x}_t^c) \chi_{z_t}^{m}(\mathbf{x}_t^m) \chi_{z_t}^{s}(\mathbf{x}_t^s), \quad (9)$$

where z_t and s_t are the section and chord labels at beat t, respectively, determined by the section-level latent variables $\mathbf{Z}$, $\mathbf{D}$, and $\mathbf{S}$, and $\chi_{z,s}^c$, χ_z^m, and χ_z^s are the emission probabilities of chroma features $\mathbf{x}^c$, MFCCs $\mathbf{x}^m$, and mel spectra $\mathbf{x}^s$, respectively.

The chroma features $\mathbf{x}^c \in \mathbb{R}^{12}$ are generated depending on both the section label z and the chord label s having the left-to-right property. The chord/chroma repetitiveness is thus represented by applying the same set of emission probabilities to all sections of the same label. The emission

Figure 2. Proposed LSTM-HSMM hybrid model.

probability $\chi_{z,s}^c(\mathbf{x}^c)$ in Eq. (9) is given by a multivariate Gaussian distribution as follows:

$$\chi_{z,s}^c(\mathbf{x}^c) = \mathcal{N}(\mathbf{x}^c | \boldsymbol{\mu}_{z,s}^c, (\mathbf{\Lambda}_{z,s}^c)^{-1}), \quad (10)$$

where $\boldsymbol{\mu}_{z,s}^c$ and $\mathbf{\Lambda}_{z,s}^c$ are a mean vector and a precision matrix, respectively.

The MFCCs $\mathbf{x}^m \in \mathbb{R}^{12}$ are generated depending on the section label z. This allows the model to capture the homogeneity of the timbral characteristics of each section. The emission probability $\chi_z^m(\mathbf{x}^m)$ in Eq. (9) is also given by a multivariate Gaussian distribution as follows:

$$\chi_z^m(\mathbf{x}^m) = \mathcal{N}(\mathbf{x}^m | \boldsymbol{\mu}_z^m, (\mathbf{\Lambda}_z^m)^{-1}), \quad (11)$$

where $\boldsymbol{\mu}_z^m$ and $\mathbf{\Lambda}_z^m$ are a mean vector and a precision matrix, respectively.

The mel spectra $\mathbf{x}^s \in \mathbb{R}^{128}$ are generated depending on the section label z. The emission probability $\chi_z^s(\mathbf{x}^s)$ in Eq. (9) is computed as follows:

$$\chi_z^s(\mathbf{x}^s) = p(\mathbf{x}^s | z) \propto \frac{p(z|\mathbf{x}^s)}{p(z)}, \quad (12)$$

where $p(z)$ is a unigram probability of section labels, and the probability $p(z|\mathbf{x}^s)$ is estimated by a labeling network (BLSTM) that infers section labels from mel spectra at the frame level. Let $p(z|\mathbf{x}^s)$ be the average of the frame-level outputs of the network in beat units.

3.2.4 Prior Distributions Based on Musical Knowledge

To use prior knowledge about musical sections, we formulate a Bayesian HSMM by putting conjugate prior distributions on the model parameters $\mathbf{\Theta} \triangleq \{\boldsymbol{\rho}, \boldsymbol{\psi}, \boldsymbol{\pi}, \boldsymbol{\upsilon}, \boldsymbol{\phi}, \boldsymbol{\mu}, \mathbf{\Lambda}\}$ [26]. We put Gaussian-Wishart prior distributions on the multivariate Gaussian parameters as follows:

$$\boldsymbol{\mu}_{z,s}^c, \mathbf{\Lambda}_{z,s}^c \sim \mathcal{N}(\boldsymbol{\mu}_{z,s}^c | \mathbf{m}_0^c, (\beta_0^c \mathbf{\Lambda}_{z,s}^c)^{-1}) \, \mathcal{W}(\mathbf{\Lambda}_{z,s}^c | \mathbf{W}_0^c, \nu_0^c),$$

$$\boldsymbol{\mu}_z^m, \mathbf{\Lambda}_z^m \sim \mathcal{N}(\boldsymbol{\mu}_z^m | \mathbf{m}_0^m, (\beta_0^m \mathbf{\Lambda}_z^m)^{-1}) \, \mathcal{W}(\mathbf{\Lambda}_z^m | \mathbf{W}_0^m, \nu_0^m),$$

where $\mathbf{m}_0^c$, β_0^c, $\mathbf{W}_0^c$, ν_0^c, $\mathbf{m}_0^m$, β_0^m, $\mathbf{W}_0^m$, and ν_0^m are hyperparameters. We then put Dirichlet prior distributions on the categorical parameters as follows:

$$\boldsymbol{\rho} \triangleq \rho_{1:K} \sim \text{Dirichlet}(\mathbf{a}^{\boldsymbol{\rho}}), \quad (13)$$

$$\boldsymbol{\psi} \triangleq \psi_{1:L} \sim \text{Dirichlet}(\mathbf{a}^{\boldsymbol{\psi}}), \quad (14)$$

$$\boldsymbol{\pi}_z \triangleq \pi_{z(1:K)} \sim \text{Dirichlet}(\mathbf{a}^{\boldsymbol{\pi}_z}), \quad (15)$$

$$\boldsymbol{v} \triangleq v_{1:K} \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{v}}), \tag{16}$$

$$\boldsymbol{\phi}_s^{(z)} \triangleq \phi_{s(1:M)}^{(z)} \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{\phi}}), \tag{17}$$

where $\mathbf{a}^{\boldsymbol{\rho}}$, $\mathbf{a}^{\boldsymbol{\psi}}$, $\mathbf{a}^{\boldsymbol{\pi}_z}$, $\mathbf{a}^{\boldsymbol{v}}$, and $\mathbf{a}^{\boldsymbol{\phi}}$ are hyperparameters. The key advantage of Bayesian inference is that unnecessary sections can be automatically removed by controlling these sparseness-related hyperparameters.

Because "verse B" tends to come after "verse A", and section durations tend to be the integer multiples of the four measures in popular music, such a statistical tendency can be incorporated in the prior distribution. Specifically, we set $\mathbf{a}^{\boldsymbol{\rho}}$ to the empirical initial section probabilities $\mathbf{a}_{\mathrm{emp}}^{\boldsymbol{\rho}}$, $\mathbf{a}^{\boldsymbol{\pi}_z}$ to the empirical section transition probabilities $\mathbf{a}_{\mathrm{emp}}^{\boldsymbol{\pi}_z}$, $\mathbf{a}^{\boldsymbol{v}}$ to the empirical terminal section probabilities $\mathbf{a}_{\mathrm{emp}}^{\boldsymbol{v}}$, and $\mathbf{a}^{\boldsymbol{\psi}}$ to the empirical section duration probabilities $\mathbf{a}_{\mathrm{emp}}^{\boldsymbol{\psi}}$. These probabilities are multiplied by a constant factor.

3.3 Bayesian Inference

Because the posterior distribution $p(\mathbf{Z}, \mathbf{D}, \mathbf{S}, \boldsymbol{\Theta}|\mathbf{X})$ is analytically intractable, we use the Gibbs sampling method. We first sample the latent variables $\mathbf{Z}$, $\mathbf{D}$, and $\mathbf{S}$ from the distribution $p(\mathbf{Z}, \mathbf{D}, \mathbf{S}|\boldsymbol{\Theta}, \mathbf{X})$ and then sample the model parameters $\boldsymbol{\Theta}$ from the distribution $p(\boldsymbol{\Theta}|\mathbf{Z}, \mathbf{D}, \mathbf{S}, \mathbf{X})$.

3.3.1 Pretraining

We compute the empirical distributions $\mathbf{a}_{\mathrm{emp}}^{\boldsymbol{\rho}}$, $\mathbf{a}_{\mathrm{emp}}^{\boldsymbol{\psi}}$, $\mathbf{a}_{\mathrm{emp}}^{\boldsymbol{\pi}_z}$, and $\mathbf{a}_{\mathrm{emp}}^{\boldsymbol{v}}$ from training data. $\left(a_{\mathrm{emp}}^{\boldsymbol{\rho}}\right)_z$ is the number of times that the sequence of section labels starts from a section z. $\left(a_{\mathrm{emp}}^{\boldsymbol{\psi}}\right)_d$ is the number of times that section labels have a duration d. $\left(a_{\mathrm{emp}}^{\boldsymbol{\pi}_z}\right)_{z'}$ is the number of transitions from a state z to a state z'. $\left(a_{\mathrm{emp}}^{\boldsymbol{v}}\right)_z$ is the number of times that the sequence of section labels ends with a section z.

3.3.2 Initialization of Latent Variables

To avoid bad local optima, we initialize the section labels $\mathbf{Z}$ and durations $\mathbf{D}$ with the labeling network. The frame-level posterior probabilities of section labels estimated by the network are averaged in each beat t. A section label of each beat is estimated to be one that achieves the maximum value of the posterior probability at the beat. Consecutive section labels are considered as a single section; however, when the duration length of an integrated section is shorter than four beats (one bar), the section is further merged into a left or right section depending on the posterior probabilities. Because it is inefficient to deal with sections that are too long, we divide sections with a duration length longer than 32 beats into 32-beat units. After that, we perform the sampling of chord sequences $\mathbf{S}$ and model parameters $\boldsymbol{\Theta}$.

3.3.3 Sampling Latent Variables

We use the forward filtering-backward sampling algorithm for sampling $\mathbf{Z}$, $\mathbf{D}$, and $\mathbf{S}$. We introduce variables z_t and d_t that denote the section label and duration starting at beat $t - d_t + 1$ and ending at beat t. We also define the marginalized emission probability for this section $\omega_{z_t}(\mathbf{x}_{t-d_t+1:t})$, which can be calculated by the forward algorithm for the Markov model of chord labels.

In the forward filtering step for the Markov model of section labels, we initialize and update the forward variables $\alpha_t(z_t, d_t) = p(z_t, d_t, \mathbf{x}_{1:t})$ as follows:

$$\alpha_t(z_t, d_t = t) = \rho_{z_t} \psi_{d_t} \omega_{z_t}(\mathbf{x}_{1:t}), \tag{18}$$

$$\alpha_t(z_t, d_t) \\ = \sum_{z', d'} \alpha_{t-d_t}(z', d') \pi_{z' z_t} \psi_{d_t} \omega_{z_t}(\mathbf{x}_{t-d_t+1:t}). \tag{19}$$

In the backward sampling step, the section labels $\mathbf{Z}$ and durations $\mathbf{D}$ are sequentially sampled in the reverse order:

$$p(z_T, d_T|\mathbf{X}) \propto \alpha_T(z_T, d_T). \tag{20}$$

When variables z_t and d_t are already sampled, the variables $z_{t'}$ and $d_{t'}$ at beat $t' = t - d_t$ are sampled according to the probability

$$p(z_{t'}, d_{t'}|z_{t:T}, d_{t:T}, \mathbf{X}) \propto \alpha_{t'}(z_{t'}, d_{t'}) \pi_{z_{t'} z_t}. \tag{21}$$

Next, the chord labels $\mathbf{S}$ are sampled using the sampled $\mathbf{Z}$ and $\mathbf{D}$. Each chord sequence $\mathbf{S}_n$ is sampled by forward filtering-backward sampling for the Markov model of chord labels in section n. In the forward filtering step, we calculate the probabilities $\zeta_{n, s_{n,\tau}}$ recursively as follows:

$$\zeta_{n, s_{n,1}} = p(s_{n,1}, \mathbf{x}_1|z_n, d_n) \\ = \delta_{s_{n,1}1} \chi_{z_n, 1}(\mathbf{x}_1), \tag{22}$$

$$\zeta_{n, s_{n,\tau}} = p(s_{n,\tau}, \mathbf{x}_{1:\tau}|z_n, d_n) \tag{23}$$

$$= \left(\sum_{s_{n,\tau-1}} \zeta_{n, s_{n,\tau-1}} \phi_{s_{n,\tau-1} s_{n,\tau}}^{(z_n)} \right) \chi_{z_n, s_{n,\tau}}(\mathbf{x}_\tau),$$

where $\mathbf{x}_\tau$ is a vector of observed features at the beat $\tau \in \{1, \ldots, d_n\}$ considered in relation to the section boundary, and $\chi_{z,s}(\mathbf{x})$ is a merged emission probability $p(\mathbf{x}|z, s)$. In the backward sampling step, the chord sequence $\mathbf{S}_n$ is sampled in the reverse order as follows:

$$p(s_{n,d_n}|z_n, d_n, \mathbf{x}_{1:d_n}) \propto \zeta_{n, s_{n,d_n}}, \tag{24}$$

$$p(s_{n,\tau}|z_n, d_n, s_{n,\tau+1:d_n}, \mathbf{x}_{1:d_n}) \propto \zeta_{n, s_{n,\tau}} \phi_{s_{n,\tau} s_{n,\tau+1}}^{(z_n)}. \tag{25}$$

3.3.4 Sampling Model Parameters

We use the Gibbs sampling method for updating the model parameters as follows:

$$\boldsymbol{\rho} \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{\rho}} + \mathbf{b}^{\boldsymbol{\rho}}), \tag{26}$$

$$\boldsymbol{\pi}_z \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{\pi}_z} + \mathbf{b}^{\boldsymbol{\pi}_z}), \tag{27}$$

$$\boldsymbol{\psi} \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{\psi}} + \mathbf{b}^{\boldsymbol{\psi}}), \tag{28}$$

$$\boldsymbol{v} \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{v}} + \mathbf{b}^{\boldsymbol{v}}), \tag{29}$$

$$\boldsymbol{\phi}_s^{(z)} \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{\phi}} + \mathbf{b}^{\phi_s^{(z)}}), \tag{30}$$

$$\boldsymbol{\Lambda}_{z,s}^c \sim \mathcal{W}(\mathbf{W}_{z,s}^c, \nu_{z,s}^c), \tag{31}$$

$$\boldsymbol{\mu}_{z,s}^c|\boldsymbol{\Lambda}_{z,s}^c \sim \mathcal{N}(\mathbf{m}_{z,s}^c, (\beta_{z,s}^c \boldsymbol{\Lambda}_{z,s}^c)^{-1}), \tag{32}$$

$$\boldsymbol{\Lambda}_z^m \sim \mathcal{W}(\mathbf{W}_z^m, \nu_z^m), \tag{33}$$

$$\boldsymbol{\mu}_z^m|\boldsymbol{\Lambda}_z^m \sim \mathcal{N}(\mathbf{m}_z^m, (\beta_z^m \boldsymbol{\Lambda}_z^m)^{-1}), \tag{34}$$

where $\mathbf{b}^{\boldsymbol{\rho}} \in \mathbb{R}^K$, $\mathbf{b}^{\boldsymbol{\pi}_z} \in \mathbb{R}^K$, $\mathbf{b}^{\boldsymbol{\psi}} \in \mathbb{R}^L$, $\mathbf{b}^{\boldsymbol{v}} \in \mathbb{R}^K$, and $\mathbf{b}^{\phi_s^{(z)}} \in \mathbb{R}^M$ are vectors that count the sampled data. $b_z^{\boldsymbol{\rho}}$ is 1 if $z = z_1$ and 0 otherwise, $b_{z'}^{\boldsymbol{\pi}_z}$ is the number of transitions from state z to state z', $b_d^{\boldsymbol{\psi}}$ is the number of times that sampled sections have a duration of d, $b_z^{\boldsymbol{v}}$ is 1

if $z = z_N$ and 0 otherwise, and $b_{s'}^{\phi_s^{(z)}}$ is the number of transitions from state s to state s' in the Markov model of chord labels in section z. The parameters $\mathbf{m}_{z,s}^c$, $\beta_{z,s}^c$, $\mathbf{W}_{z,s}^c$, and $\nu_{z,s}^c$ are calculated as follows:

$$\beta_{z,s}^c = \beta_0^c + N_{z,s}, \quad \nu_{z,s}^c = \nu_0^c + N_{z,s}, \tag{35}$$

$$\mathbf{m}_{z,s}^c = \frac{1}{\beta_{z,s}^c}(\beta_0^c \mathbf{m}_0^c + N_{z,s}\overline{\mathbf{x}}_{z,s}^c), \tag{36}$$

$$(\mathbf{W}_{z,s}^c)^{-1} = (\mathbf{W}_0^c)^{-1} + N_{z,s}\mathbf{U}_{z,s}^c$$
$$+ \frac{\beta_0^c N_{z,s}}{\beta_0^c + N_{z,s}}(\overline{\mathbf{x}}_{z,s}^c - \mathbf{m}_0^c)(\overline{\mathbf{x}}_{z,s}^c - \mathbf{m}_0^c)^{\mathrm{T}}, \tag{37}$$

where we have defined

$$N_{z,s} = \sum_{t=1}^{T} \delta_{z_t z}\delta_{s_t s}, \tag{38}$$

$$\overline{\mathbf{x}}_{z,s}^c = \frac{1}{N_{z,s}}\sum_{t=1}^{T} \delta_{z_t z}\delta_{s_t s}\mathbf{x}_t^c, \tag{39}$$

$$\mathbf{U}_{z,s}^c = \frac{1}{N_{z,s}}\sum_{t=1}^{T} \delta_{z_t z}\delta_{s_t s}(\mathbf{x}_t^c - \overline{\mathbf{x}}_{z,s}^c)(\mathbf{x}_t^c - \overline{\mathbf{x}}_{z,s}^c)^{\mathrm{T}}. \tag{40}$$

The parameters $\mathbf{m}_z^m$, β_z^m, $\mathbf{W}_z^m$, and ν_z^m can be calculated similarly.

3.3.5 Viterbi Training

Since the samples from the Gibbs sampler are not necessarily local optima of the posterior distribution, we apply Viterbi training in the last step of the parameter estimation. Specifically, we apply the Viterbi algorithm (instead of the forward filtering-backward sampling algorithm) to estimate the latent variables and update the model parameters to the expectation values of the posterior probabilities (instead of samples from those probabilities). It is known that Viterbi training is generally efficient for finding an approximate local minimum [29].

3.3.6 Refinements

We introduce a weighting factor $w_{\mathrm{dur}}(\geq 1)$ for the duration probability to enhance its effect. Specifically, we replace the probability factor ψ_d in the forward algorithm (18) and (19) with $(\psi_d)^{w_{\mathrm{dur}}}$. Similar replacements are applied to the Viterbi training step and the final estimation step of the latent states explained in Section 3.4. We also introduce a weighting factor w_{label} that balances the emission probabilities for mel spectra with the other emission probabilities. We replace the emission probability $\chi_z^s(\mathbf{x}^s)$ in (9) with $(\chi_z^s(\mathbf{x}^s))^{w_{\mathrm{label}}}$.

3.4 Estimation of Musical Sections

After training the model parameters $\mathbf{\Theta}$, we compute the maximum a posteriori (MAP) estimate of the musical sections. Specifically, we maximize the posterior probability $p(\mathbf{Z}, \mathbf{D}|\mathbf{\Theta}, \mathbf{X})$ with respect to the section labels $\mathbf{Z}$ and durations $\mathbf{D}$. This can be solved by integrating out the chord labels $\mathbf{S}$ and applying the Viterbi algorithm for HSMMs [30] to the Markov model of section labels.

4. EVALUATION

Experiments were conducted to investigate the performance of the proposed method.

4.1 Experimental Conditions

To evaluate our model, we used the 100 pieces from the RWC Popular Music Database [31] with structure annotations [32] for evaluation. We extracted chroma features using the deep feature extractor [33] and MFCCs and mel spectrograms using the librosa library [34]. Beat information was obtained using the madmom library [28]. The labeling network consisted of a single-layer BLSTM with 2048×2 cells and a fully-connected layer with output dimension K. The network was trained with 10-fold cross validation, and the empirical distributions $\mathbf{a}_{\mathrm{emp}}^{\rho}$, $\mathbf{a}_{\mathrm{emp}}^{\psi}$, $\mathbf{a}_{\mathrm{emp}}^{\pi_z}$, and $\mathbf{a}_{\mathrm{emp}}^{v}$ were trained with piece-wise cross validation for the 100 pieces. For parameter estimation, we iterated the Gibbs sampling 15 times and the Viterbi training 3 times, which took around five times longer than the duration of an input signal with a standard CPU.

The hyperparameters of the proposed model were set as follows: $K = 10$, $L = 40$, $M = 16$, $\sigma = 1$, $w_{\mathrm{dur}} = 4$, $w_{\mathrm{label}} = 0.5$, $\mathbf{a}^{\rho} = 1 \cdot \mathbf{a}_{\mathrm{emp}}^{\rho}$, $\mathbf{a}^{\pi} = 1 \cdot \mathbf{a}_{\mathrm{emp}}^{\pi}$, $\mathbf{a}^{\psi} = 64 \cdot \mathbf{a}_{\mathrm{emp}}^{\psi}$, $\mathbf{a}^{v} = 1 \cdot \mathbf{a}_{\mathrm{emp}}^{v}$, $\mathbf{a}^{\phi} = \mathbb{I}$, $\mathbf{m}_0^c = \mathbb{E}[\mathbf{X}^c]$, $\beta_0^c = 64$, $\mathbf{W}_0^c = (\nu_0^c \mathrm{cov}[\mathbf{X}^c])^{-1}$ with $\nu_0^c = 512$, $\mathbf{m}_0^m = \mathbb{E}[\mathbf{X}^m]$, $\beta_0^m = 2$, and $\mathbf{W}_0^m = (\nu_0^m \mathrm{cov}[\mathbf{X}^m])^{-1}$ with $\nu_0^m = 16$, where $\mathbb{I}$ denotes a vector with all entries equal to 1. The first parameter K was determined according to the number of labels used in [24], as shown in the legend in Fig. 3. The next two parameters L and M were determined by consulting the statistics of the annotated data. In the data, most sections have a length of 40 beats or less. If we assume a section length of 32 beats (8 measures) and a chord duration of 2 beats, the expected number of chords in each section is 16. The value of σ was set to 1 for simplicity. The other parameters were determined by a coarse optimization w.r.t. the evaluation measures explained below. Each parameter was optimized by a grid search, fixing the other parameters. Further optimization of the parameters is left for future work.

We evaluated the estimation results in terms of segmentation, clustering, and labeling. The qualities of segmentation and clustering were evaluated in the same way as MIREX [35]. The quality of segmentation was evaluated by the F-measures of section boundaries denoted by $F_{0.5}$ and $F_{3.0}$ [36]. Specifically, an estimated boundary is accepted as correct if it is within $\pm 0.5/3.0$ seconds from the ground-truth boundary. The precision rate is the percentage of correct boundaries in estimated boundaries, the recall rate is the percentage of true boundaries that are correctly estimated, and the F-measures $F_{0.5}$ and $F_{3.0}$ are defined as the harmonic means of the precision and recall rates.

The quality of clustering was evaluated by the pairwise F-measure denoted by F_{pair} [37] defined as follows. We compared pairs of frames (with a length of 100 ms) that are labeled with the same class in an estimation result with those in the ground truth. The precision rate, recall rate,

Method	Segmentation		Clustering	Labeling
	$F_{0.5}$ (%)	$F_{3.0}$ (%)	F_{pair} (%)	(%)
GS3 [39]	**52.3**	73.5	54.2	n/a
SUG2 [40]	25.8	**73.7**	37.3	n/a
FK2 [41]	30.0	65.7	63.4	n/a
Paulus'09 [24]	n/a	63.0	**63.7**	34.4
Cascade model	38.3	63.9	54.9	38.8
Baseline model	30.6	53.5	43.3	39.5
Proposed model	43.3	66.5	54.6	**45.3**

Table 1. Evaluation results of a comparative experiment.

and F-measure are defined as follows:

$$P_{\text{pair}} = \frac{|P_E \cap P_A|}{|P_E|}, \quad R_{\text{pair}} = \frac{|P_E \cap P_A|}{|P_A|}, \quad (41)$$
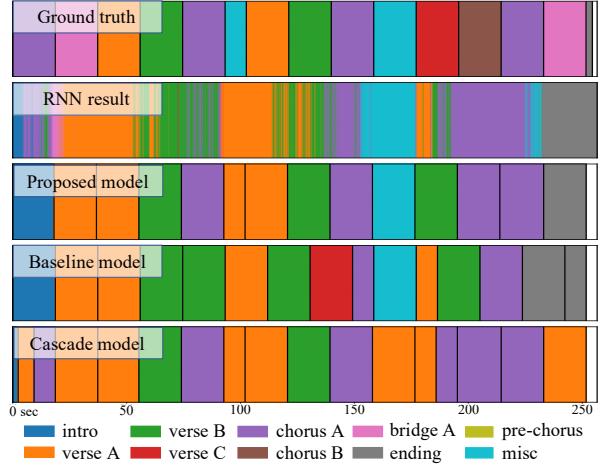
$$F_{\text{pair}} = \frac{2 P_{\text{pair}} R_{\text{pair}}}{P_{\text{pair}} + R_{\text{pair}}}, \quad (42)$$

where P_E denotes the set of similarly labeled frame pairs in the estimation and P_A denotes that in the ground truth. These values are calculated using the `mir_eval` library [38]. The quality of labeling was evaluated by the accuracy in frame units, as in [24]. This is calculated by comparing a label assigned to each frame in the result and the ground truth.

For comparison in the segmentation and the clustering, we refer to GS3 [39], SUG2 [40], and FK2 [41], published in MIREX. In addition, for comparison in all three viewpoints, we quoted the result of [24] and ran two models, a cascade model and a baseline model. In the cascade model, the frame-level labels obtained by the BLSTM were counted for each cluster obtained by the HSMM [26]. The most frequently occurring label in a cluster was estimated to be the label in that cluster. Although the baseline model is similar to the proposed model, it outputs neither chroma features nor MFCCs and only uses the Viterbi decoding to obtain results.

4.2 Experimental Results

Table 1 shows the evaluation results. In the labeling accuracy, the proposed method outperformed the other methods that have the labeling ability. Compared with the cascade model using the labeling results of the BLSTM in the post-processing, the proposed model had better performance in segmentation and labeling. This indicates the effectiveness of joint segmentation and labeling in the unified probabilistic framework. In addition, compared with the baseline model using the Viterbi decoding only, the proposed model achieved better performance in all metrics. This revealed the effectiveness of piece-specific Bayesian learning based on the prior distributions. In contrast, the proposed method did not always achieve the state-of-the-art performance except for labeling. It may be because the proposed method tended to yield unnatural repetitions of the same label with various lengths. In general, sections of the same label have approximately the same length. Such a constraint could be incorporated by introducing a duration probability distri-

Figure 3. Example results by proposed, baseline, and cascade model (RWC-MDB-P-2001 No. 25)

bution specific for each label.

Example results are shown in Fig. 3. The cascade model yielded some mislabeled sections with correctly estimated boundaries because the clustering errors of the HSMM were propagated. In contrast, because the proposed method performs segmentation and labeling simultaneously, such errors were reduced effectively. While the baseline model yielded errors originating from the errors of the BLSTM, such errors were corrected in the result of the proposed method. This suggested that the proposed method has the ability to prevent such errors by focusing on the homogeneity of MFCCs and repetitiveness of chroma features. We found that the proposed method erroneously estimated "chorus A" at the beginning of this song as "intro". Such errors could be avoided by adjusting the weights of the initial and emission probabilities or training the BLSTM with the connectionist temporal classification (CTC) loss function [42] to remove frequent musically unnatural label switching.

5. CONCLUSION

We have presented a deep generative approach to music structure analysis based on a Bayesian LSTM-HSMM hybrid model. The model represents the essential characteristics of sections, homogeneity, repetitiveness, and regularity, with MFCCs, chroma features, and mel spectra. Music segmentation and section labeling are performed jointly by unsupervised Bayesian learning of the model. The experimental results showed that the proposed method is effective for musical structure analysis.

The proposed method considers homogeneity, repetitiveness, and regularity, but not novelty, which has been emphasized in conventional research [1]. Exploiting this aspect remains an avenue for future work. It is also important to deal with further hierarchies [17], as music has a hierarchical structure moving from motives and phrases to sections and section groups [43].

7. REFERENCES

[1] J. Paulus, M. Müller, and A. Klapuri, "State of the art report: Audio-based music structure analysis," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 625–636.

[2] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2000, pp. 452–455.

[3] K. Jensen, "Multiple scale music segmentation using rhythm, timbre, and harmony," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 159–159, 2007.

[4] M. Goto, "A chorus section detection method for musical audio signals and its application to a music listening station," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 14, no. 5, pp. 1783–1794, 2006.

[5] J. Serrà, M. Müller, P. Grosche, and J. Arcos, "Unsupervised detection of music boundaries by time series structure features," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 1613–1619.

[6] G. Peeters and V. Bisot, "Improving music structure segmentation using lag-priors," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 337–342.

[7] J. Serrà, M. Müller, P. Grosche, and J. Arcos, "Unsupervised music structure annotation by time series structure features and segment similarity," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1229–1240, 2014.

[8] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 417–422.

[9] T. Grill and J. Schlüter, "Music boundary detection using neural networks on combined features and two-level annotations," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 531–537.

[10] J. B. L. Smith and M. Goto, "Using priors to improve estimates of music structure," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 554–560.

[11] G. Sargent, F. Bimbot, and E. Vincent, "Estimating the structural segmentation of popular music pieces under regularity constraints," *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 2, pp. 344–358, 2017.

[12] A. Maezawa, "Music boundary detection based on a hybrid deep model of novelty, homogeneity, repetition and duration," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 206–210.

[13] M. Cooper and J. Foote, "Summarizing popular music via structural similarity analysis," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2003, pp. 127–130.

[14] M. M. Goodwin and J. Laroche, "A dynamic programming approach to audio segmentation and speech/music discrimination," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004, pp. 309–312.

[15] H. Grohganz, M. Clausen, N. Jiang, and M. Müller, "Converting path structures into block structures using eigenvalue decompositions of self-similarity matrices," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 209–214.

[16] O. Nieto and T. Jehan, "Convex non-negative matrix factorization for automatic music structure identification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 236–240.

[17] B. McFee and D. P. W. Ellis, "Analyzing song structure with spectral clustering," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 405–410.

[18] T. Cheng, J. B. L. Smith, and M. Goto, "Music structure boundary detection and labelling by a deconvolution of path-enhanced self-similarity matrix," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 106–110.

[19] J.-J. Aucouturier and M. Sandler, "Segmentation of musical signals using hidden Markov models," in *Audio Engineering Society (AES) Convention*, 2001, pp. 1–8.

[20] M. Levy and M. Sandler, "New methods in structural segmentation of musical audio," in *European Signal Processing Conference (EUSIPCO)*, 2006, pp. 1–5.

[21] L. Ren, D. Dunson, S. Lindroth, and L. Carin, "Dynamic nonparametric Bayesian models for analysis of music," *Journal of the American Statistical Association (JASA)*, vol. 105, no. 490, pp. 458–472, 2008.

[22] L. Barrington, A. B. Chan, and G. Lanckriet, "Modeling music as a dynamic texture," *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 18, no. 3, pp. 602–612, 2010.

[23] F. Kaiser and G. Peeters, "A simple fusion method of state and sequence segmentation for music structure discovery," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 257–262.

[24] J. Paulus and A. Klapuri, "Music structure analysis using a probabilistic fitness measure and a greedy search algorithm," *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 17, no. 6, pp. 1159–1170, 2009.

[25] ——, "Labelling the structural parts of a music piece with markov models," in *Computer Music Modeling and Retrieval (CMMR)*, 2008, pp. 166–176.

[26] G. Shibata, R. Nishikimi, E. Nakamura, and K. Yoshii, "Statistical music structure analysis based on a homogeneity-, repetitiveness-, and regularity-aware hierarchical hidden semi-markov model," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 268–275.

[27] N. C. Maddage, C. Xu, M. S. Kankanhalli, and X. Shao, "Content-based music structure analysis with applications to music semantics understanding," in *ACM International Conference on Multimedia (ACMMM)*, 2004, pp. 112–119.

[28] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: A new python audio and music signal processing library," in *ACM International Conference on Multimedia (ACMMM)*, 2016, pp. 1174–1178.

[29] A. Allahverdyan and A. Galstyan, "Comparative analysis of Viterbi training and maximum likelihood estimation for HMMs," in *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 1674–1682.

[30] S.-Z. Yu, "Hidden semi-Markov models," *Artificial Intelligence*, vol. 174, no. 2, pp. 215–243, 2010.

[31] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical and jazz music databases," in *International Conference on Music Information Retrieval (ISMIR)*, 2002, pp. 287–288.

[32] M. Goto, "AIST annotation for the RWC music database," in *International Conference on Music Information Retrieval (ISMIR)*, 2006, pp. 359–360.

[33] Y. Wu and W. Li, "Automatic audio chord recognition with MIDI-trained deep feature and BLSTM-CRF sequence decoding model," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 27, no. 2, pp. 355–366, 2019.

[34] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Python in Science Conference*, 2015, pp. 18–24.

[35] A. F. Ehmann, M. Bay, J. S. Downie, I. Fujinaga, and D. D. Roure, "Music structure segmentation algorithm evaluation: Expanding on MIREX 2010 analyses and datasets," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 561–566.

[36] D. Turnbull, G. Lanckriet, E. Pampalk, and M. Goto, "A supervised approach for detecting boundaries in music using difference features and boosting," in *International Conference on Music Information Retrieval (ISMIR)*, 2007, pp. 51–54.

[37] M. Levy and M. Sandler, "Structural segmentation of musical audio by constrained clustering," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 16, no. 2, pp. 318–326, 2008.

[38] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mir_eval: A transparent implementation of common MIR metrics," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[39] T. Grill and J. Schlüter, "Structural segmentation with convolutional neural networks mirex submission," in *Music Information Retrieval Evaluation eXchange (MIREX)*, 2015.

[40] J. Schlüter, K. Ullrich, and T. Grill, "Structural segmentation with convolutional neural networks mirex submission," in *Music Information Retrieval Evaluation eXchange (MIREX)*, 2014.

[41] F. Kaiser and G. Peeters, "Music structural segmentation task: Ircamstructure submission," in *Music Information Retrieval Evaluation eXchange (MIREX)*, 2013.

[42] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2006, pp. 369–376.

[43] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. MIT Press, 1983.

PERCEPTUAL VS. AUTOMATED JUDGEMENTS OF MUSIC COPYRIGHT INFRINGEMENT

Yuchen Yuan[1], Sho Oishi[1], Charles Cronin[2], Daniel Müllensiefen[3], Quentin Atkinson[4], Shinya Fujii[1], Patrick E. Savage[1]*

[1]Keio University, Japan; [2]George Washington University Law School, USA; [3]Goldsmiths, University of London, UK; [4]University of Auckland, New Zealand

*psavage@sfc.keio.ac.jp

ABSTRACT

Music copyright lawsuits often result in multimillion dollar damage awards or settlements, yet there are few objective guidelines for applying copyright law in infringement claims involving musical works. Recent research has attempted to develop objective methods based on automated similarity algorithms, but there remains almost no data on the role of perceived similarity in music copyright decisions despite its crucial role in copyright law. We collected perceptual data from 20 participants for 17 adjudicated copyright cases from the USA and Japan after editing the disputed sections to contain either full audio, melody only, or lyrics only. Due to the historical emphasis in legal opinions on melody as the key criterion for deciding infringement, we predicted that listening to melody-only versions would result in perceptual judgements that more closely matched actual past legal decisions. Surprisingly, however, we found no significant differences between the three conditions, with participants matching past decisions in between 50-60% of cases in all three conditions. Automated algorithms designed to calculate melodic and audio similarity produced comparable results: both algorithms were able to match past decisions with identical accuracy of 71% (12/17 cases). Analysis of cases that were difficult to classify suggests that melody, lyrics, and other factors sometimes interact in complex ways difficult to capture using quantitative metrics. We propose directions for further investigation of the role of similarity in music copyright law using larger and more diverse samples of cases and enhanced methods, and adapting our perceptual experiment method to avoid relying for ground truth data only on court decisions (which may be subject to selection bias). Our results contribute to important practical debates, such as whether jury members should be allowed to listen to full audio recordings during copyright cases.

1. INTRODUCTION

Music copyright law protects the lawful rights and interests of music creators and performers, but in some music copyright infringement cases, its application has caused bitter controversy. As litigation becomes more frequent,

inappropriate music copyright lawsuits not only inhibit music creativity but also waste millions of taxpayer dollars annually to cover the adjudication of these disputes. The legal system and music industry could both benefit from automated methods that could reduce subjectivity in music copyright decisions, and several recent studies have proposed such automated methods [1-4]. While the accuracy of some algorithms have been tested against previous court decisions, they have not yet been tested against perceptual data to determine how different musical and extra-musical factors interact in copyright law.

"Substantial similarity" and "protectable expression" are central concepts in US copyright law, the understanding of which could potentially be supplemented through automated and/or perceptual analyses. The concept of "substantial similarity" requires not only that the defendant can be shown to have copied musical material, but that this copying of protected musical expression was so extensive that the two works are substantially similar [5]. Data on degrees of computed and/or perceived similarity can help to determine objective standards for how much copying is required to be considered "substantial".

Evaluating what is considered "protectable expression" is more qualitative and complex. Many musical aspects such as scales, certain rhythmic patterns, and timbres are considered to be such basic and commonplace musical ideas as not to be copyrightable. For example, many blues songs all use very similar blues scales, 12-bar harmonic progressions, vocal styles and instrumentation, but copying these aspects is not considered copyright infringement. Instead, melody (i.e., the sequence of pitches) and lyrics have traditionally played predominant roles against other musical factors [6-7]. However, it has been disputed whether jury members should be allowed to listen to full-audio or melody-only versions of musical works because people may perceive and judge differently when comparing pairs of melodies or other musical features [8]. For example, a core issue in the recently concluded case involving the Blurred Lines [9] was whether the jury should be allowed to listen to a full audio recording including lyrics and background instrumentation of the complaining work, or whether it should only be exposed to the sheet music that was deposited with the US Copyright Office [6].

To quantitatively compare the effects of melody, lyrics, and other factors, we designed a controlled experiment where we constructed versions of a disputed musical work containing the full audio (including lyrics, melody, and other factors such as instrumentation), melody only (pitches and rhythms in MIDI representation), and lyrics

only (text representation). Because of the historical dominance of melody, we predicted that participants would most accurately match past decisions when presented with melody-only versions, and that automated algorithms based on melodic data would more accurately match past decisions than ones based on full-audio data.

Section 2 discusses related research. Section 3 discusses the data selection for our study. In Section 4, we demonstrate the design and details of the perceptual experiment. In Section 5, we show how the melodic and audio similarity are calculated by automated algorithms (PMI and Musly, respectively). In Sections 6, 7, and 8, we discuss the performance of the two automated methods, compare the automated and perceptual methods, summarize current results, and discuss future directions for improvement.

2. RELATED WORK

2.1 Perceptual Experiments

In one previous experimental study, Lund used two past court cases (*Swirsky v. Carey* and *Gasté v. Kaiserman*) and manipulated MIDI representations of the works to change aspects such as tempo, rhythm, and instrumentation [8]. Lund found that such manipulations reduced the accuracy of participants' judgements of copyright infringements even though it was assumed that such non-melodic features should not play a role in decisions. Lund argued that this demonstrated that the "lay listener test" was flawed because it relies on subjective listening to audio recordings that may differ in non-melodic aspects. However, Lund did not compare full audio recordings with these MIDI representations, so it remains unknown whether listeners are in fact more accurate when listening to MIDI representations than when listening to full audio recordings.

2.2 Automatic Analysis

Müllensiefen and Pendzich developed an algorithm for judging melodic similarity that compares the profile of successive pitch intervals in two disputed songs against each other, while weighting them against a database of comparable profiles from 14,063 pop songs using a weighting formula for estimating perceptual salience [1]. When they applied this algorithm to a database of 20 past music copyright decisions focused on melodic similarity, they found the best-performing version of their algorithm was able to accurately identify 90% (18/20) of past cases.

Savage et al. later developed a Percent Melodic Identity (PMI) method for quantifying melodic evolution based on automatic sequence alignment algorithms used in molecular genetics to measure melodic similarity [10]. When they applied this method to the same set of cases as Müllensiefen & Pendzich, it accurately predicted 80% (16/20) of cases, despite being a simpler method that didn't require calibration to an existing database of popular songs [3].

While the related task of cover song detection has a long history of study in music information retrieval [11-12], to our knowledge no audio similarity algorithms have yet been tested for their ability to evaluate copyright infringement. However, many general audio similarity algorithms have been evaluated through the Music Information Retrieval Exchange (MIREX) competition. We thus chose the audio similarity algorithm implemented in Musly, an open-source library of audio music similarity algorithms, because it has consistently performed at or near the top of audio similarity algorithms as evaluated in MIREX [13].

3. DATASET OF MUSIC COPYRIGHT INFRINGEMENT CASES

We chose a set of 17 court decisions whose main copyright issue focused on substantial similarity of the melodies (Table 1). 14 of these 17 cases are from the US, and these 14 represent a subset of 20 cases from the Music Copyright Infringement Resource [14] that were previously analyzed [1, 3] for which full audio recordings were available for both of the disputed musical works (the remaining 6 cases were not included because one or both musical works were represented only by sheet music and/or MIDI files). We also included 3 court decisions from Japan in order to increase cultural diversity in the dataset for further study on adaptability to music other than Western music. Of the 17 cases courts found no infringement in 8 cases, and infringement in 9.

4. PERCEPTUAL EXPERIMENT

4.1 Experiment Design

We conducted an online perceptual experiment where participants were each asked to judge substantial similarity for the 17 cases. The disputed segments of the musical works (mean length: 22s; range: 3-55s) were presented in one of three different versions: full-audio (the recorded versions including all instrumental and/or vocal parts), melody-only (MIDI rendition of the pitches and rhythms of the main melody), and lyrics-only (lyrics shown as visual text, without any accompanying audio). For the melody condition, in order to control for all non-melodic factors including instrumentation, key, and tempo, transcribed melodies from the original audio recordings were edited as necessary to exactly correspond to the audio recordings[1]: these transcribed melodies were transposed to have a tonic of C, and were then recorded using the MIDI piano in MuseScore played back at a tempo that was the average of the tempi from the plaintiff and defendant recordings. For the lyrics condition, the three instrumental works without lyrics (cf. Table 1) simply showed "[no lyrics]". These three types of presentations were repeated twice: once using the originally disputed pair of musical works, and once using the original defendant work but comparing it against a randomly selected plaintiff work from the other 16 cases.

---

[1] While preparing the audio files for experiments we noticed several minor inconsistencies between the audio files and the transcriptions provided by the authors of [1]. In some cases, these were small errors in pitch/rhythm; in others, only one half of a larger section was transcribed. The original transcriptions were not initially published but have now been uploaded to https://github.com/pesavage/copyright/tree/master/MIDIs_plagiarismcases_MullensiefenPendzich2009 to allow comparison as necessary. The corrected transcriptions are available at https://github.com/compmusiclab/music-copyright.

No.	Cou ntry	Case	Complaining Work	Length (seconds)	Defending Work	Length (seconds)	Court Decision	PMI (cutoff = 46.8%)	Musly-calculated Similarity (cutoff = 32.8%)	Perceptual Accuracy - Full audio	Perceptual Accuracy - Melody only	Perceptual Accuracy - Lyrics only	Perceptual Similarity - Full audio	Perceptual Similarity - Melody only	Perceptual Similarity - Lyrics only
1	JP	Harry vs. Suzuki	"Boulevard of Broken Dreams"	33	"ワン・レイニーナイト・イン・トーキョー" (One Rainy Night in Tokyo)	23	0	25%	25%	65%	80%	100%	3.15	2.9	1.4
2	US	Cottrill vs. Spears	"What You See is What You Get"	22	"What U See is What U Get"	24	0	35%	41%	70%	95%	65%	2.75	1.85	3
3	US	Baxter vs. MCA	"Joy"	7	"Theme from 'E.T.'"	19	0	37%	12%	85%	90%	N/A	2.7	2	N/A
4	US	Swirsky vs. Carey	"One of Those Love Songs"	29	"Thank God I Found You"	32	1	45%	76%	60%	35%	0%	3.45	3	1.4
5	US	Repp vs. Lloyd-Webber	"Till You"	27	"Phantom Song"	38	1	45%	15%	50%	35%	100%	3.15	4.35	1.25
6	JP	Kobayashi vs. Hattori	"どこまでも行こう" (Dokomademoikō)	23	"記念樹" (Kinenju)	40	1	47%	10%	55%	45%	10%	3.6	3.2	1.55
7	US	Three Boys Music vs. Michael Bolton	"Love Is A Wonderful Thing"	10	"Love Is A Wonderful Thing"	17	1	47%	63%	70%	30%	50%	3.65	3.25	3.7
8	US	Herald Square Music vs. Living Music	"Day By Day"	32	"Theme N.B.C.'s 'Today Show'"	30	1	51%	5%	45%	40%	N/A	3.6	2.85	N/A
9	US	Grand Upright vs. Warner	"Alone Again (Naturally)"	5	"Alone Again"	6	1	53%	25%	70%	30%	50%	4.2	2.9	4
10	US	Bright Tunes Music vs. Harrisongs Music	"He's So Fine"	27	"My Sweet Lord"	55	1	58%	35%	25%	45%	5%	2.5	3.25	1.3
11	US	Selle vs. Gibb	"Let It End"	21	"How Deep Is Your Love"	19	0	63%	11%	55%	40%	95%	3.25	3.65	1.65
12	US	Louis Gaste vs. Morris Kaiserman	"Pour Toi"	17	"Feelings"	21	1	65%	33%	50%	50%	0%	3.4	3.8	1.35
13	US	Granite Music vs. United Artists	"Tiny Bubbles"	18	"Hiding The Wine"	11	0	67%	4%	60%	40%	N/A	3.3	3.8	N/A
14	US	Fantasy vs. Fogerty	"Run Through The Jungle"	21	"The Old Man Down The Road"	21	0	67%	62%	40%	45%	100%	3.45	3.3	1.4
15	US	Jean et al. vs. Bug Music	"Hand Clapping Song"	3	"My Love Is Your Love"	4	0	71%	20%	45%	80%	90%	3.75	2.6	2.8
16	US	Levine vs. McDonald's	"Life Is A Rock (But The Radio Rolled Me)"	22	"McDonald's Menu Song"	26	1	80%	63%	65%	45%	10%	4	3.6	1.8
17	JP	HarumakiGohan vs. Mori	"八月のレイニー" (Hachigatsu no reinī)	21	"M.A.K.E"	22	1	100%	54%	75%	85%	10%	4.25	4.35	2

Table 1. The 17 music copyright infringement cases analyzed and respective melodic similarity (PMI), audio music similarity (Musly), and perceptual experiment results. Cases are ordered by increasing PMI values. In "Court Decision" column, "0" represents no infringement, and "1" represents infringement. Cases in italics are those PMI failed to accurately classify, and bold indicates those Musly failed to classify. Columns highlighted in light blue are the accuracy of perceptual judgement for the 17 court cases judged by the 20 participants for full-audio, melody-only, and lyrics-only, corresponding to the data in Figure 1. Columns highlighted in light green are perceptual similarity values used for comparison between automatic methods and perceptual judgement in Section 6, corresponding to the data in Figures S2 and S3. Three defending works marked in orange text are instrumental.

This gave a total of 102 different pairs of musical works to evaluate (17 cases × 3 presentations [full, melody, lyrics] × 2 pairings [original plaintiff vs. random plaintiff]), presented in fully random order (without separate blocks for different conditions; i.e., any given sample might be full-audio, melody-only, or lyrics-only; original case or not). Each experiment took approximately 2 hours for one participant to complete evaluations for these 102 pairs.

For each pair, the participant is given a pair of music excerpts, "A" and "B". "A" is always a plaintiff's work while "B" is always a defendant's work. After listening to the full-audio or MIDI or reading the lyrics of the two music works, the participant needs to answer two questions: 1) How similar are A and B? (5-point Likert scale: "not at all similar", "a little similar", "somewhat similar", "very similar", and "extremely similar"). 2) Do you think the second music work ("B") infringed the copyright of the first one's ("A")? (Yes/no answer.) The following criteria for infringement were provided, taken from [8] (which was in turn adapted from real instructions given to juries [details of the adaptation were not provided]):

To find music copyright infringement between plaintiff's and defendant's songs, you must find that the songs are substantially similar. Two works are substantially similar if the original expression of ideas in the plaintiff's (Song #1) copyrighted work and the expression of ideas in the defendant's work (Song #2) that are shared are substantially similar. Original expression are those unique aspects of plaintiff's song that are not common or ordinary to the genre or to music generally. The amount of similarity must be both quantitatively

and qualitatively significant, that is the defendant's song copied either a substantial portion of the original expression of the plaintiff's song, or copied a smaller but qualitatively important portion of the plaintiff's song.

In short, this investigation imitates the lay listener test used to see whether an ordinary observer recognizes that the defendant appropriated something belonging to the plaintiff [8, 15].
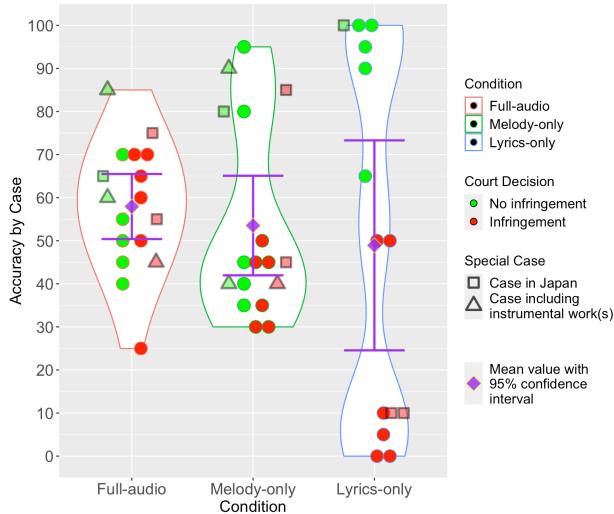
4.2 Results

We collected perceptual data from 20 participants from our institution. 9 were male, and 11 were female. 17 were between 17-28 years old, 1 was between 29-50, and 2 were over 50. The native languages of participants were Chinese (13 participants), Japanese (6) and English (1). 11 reported substantial music experience while 9 did not. Table 1 summarizes all results for perceptual and automated experiments.

Figure 1 and S1 show how accurately the 20 participants' judgement of infringement matched the official court decisions when they were given full-audio, melody-only, or lyrics-only versions of music pieces from the 17 court cases. Note that accuracy is measured as how likely participants were to match court decisions, whether that decision was of infringement or no infringement. Although the perceptual data were collected for the three cases including instrumental works, these cases were omitted from the lyrics-only analyses because infringement of lyrics is clearly impossible for instrumental works. In Figure S1,

individual data points represent mean accuracy for individual *participants* (n = 20) across the 17 cases, while in Figure 1 individual data points represent mean accuracy for individual *cases* (n = 17) across the 20 participants.

Surprisingly, the accuracy numbers by participants of the three condition groups distributed quite closely, with mean accuracy of 58%, 54%, and 49% for full-audio, melody-only, and lyrics-only groups respectively. Not only was our predicted difference between melody-only and full-audio not significant (paired t = 1.7, df = 19, one-tailed p = 0.95), but what small difference there was between full-audio and melody-only was in the opposite direction from our predictions (participants were slightly *more* accurate when presented with full-audio than with melody alone). However, randomized control pairs had modal accuracies of 100% for all three conditions (cf. Figure S1), confirming that participants were able to perform all three tasks much more accurately than by chance. In addition, participants who self-reported as musicians showed no significant differences in accuracy compared to non-musicians (full: t = 0.63, df = 11, 1-sided p-value = 0.27; melody: t = 1.20, df = 17, 1-sided p-value = 0.12; lyrics: t = 0.68, df = 14, 1-sided p-value = 0.25).

Figure 1. Accuracy of perceptual judgement for each of the 17 court cases, as measured by the percentage of the 20 participants whose judgements of music copyright infringement matched court decisions.

Figure 1 plots the accuracy of perceptual judgement for the 17 court cases judged by the 20 participants. As in Fig. S1, the means of the three conditions are similar. Interestingly, however, the accuracy values are approximately normally distributed for the full-audio condition, while the melody-only and lyrics-only conditions have bimodal, hourglass-shaped distributions. Furthermore, full-audio cases show no major differences in the distribution of infringing vs. non-infringing cases, while melodic cases show some skewing toward higher accuracy for non-infringing cases and lyrics show a strong dichotomy between high accuracy for non-infringing cases and low accuracy

for infringing cases. No clear differences are notable for the small subsets of cases from Japan or those involving instrumental works.

5. AUTOMATIC ANALYSIS

We performed automatic similarity analysis of these cases using two different automated algorithms focused on melodic and audio similarity, respectively.

5.1 Melodic Similarity (Percent Melodic Identity [PMI])

We chose the PMI (Percent Melodic Identity) method to calculate melodic similarity because it has been validated in previous research using a similar sample of copyright cases. Like Judge Learned Hand's "comparative method" [6] to test musical similarity, the PMI method begins by transposing two melodies transcribed in staff notation into a same key, eliminating rhythmic information by assigning all notes equal time values, and then aligning and counting the confluence of notes[1]. Following the procedure, we prepared note sequences of disputed melodies all transposed to a C tonic for consistency (just as was done when preparing MIDI files). The PMI algorithm then automatically aligns each sequences pair, and counts the number of identical notes (*ID*). The percentage of identical notes shared between the pair of melodies, named percent melodic identity (*PMI*) [3], is calculated by dividing *ID* by the average length of the melodies pair (L_1 and L_2), as follows:

$$PMI = 100\left(\frac{ID}{\frac{L_1 + L_2}{2}}\right)$$

5.1.1 Melodic Similarity Results

The PMI values computed for all 17 music copyright infringement cases are shown in Table 1. Receiver Operating Characteristic (ROC) analysis was used to assess the prediction given by PMI values. The area under the ROC curve (AUC) is 0.61. The optimal cutoff PMI value is 46.8% with sensitivity = 0.89 and specificity = 0.50. Using this cutoff, PMI method was able to accurately classify 12 out of the 17 cases (71%) to match their court decisions. The five cases highlighted by italic font in Table 1 are those that the PMI method failed to classify correctly, discussed further below.

5.2 Audio Similarity (Musly)

Musly currently implements two music similarity algorithms. One implements Mandel-Ellis audio similarity algorithm [16]. The other one, which is the default one, improves Mandel-Ellis algorithm to compute audio similarity for best results. Specifically, it computes a representation of each song's audio signal based on 25 Mel-Frequency Cepstral Coefficients (MFCCs) to estimate a Gaussian model and finally a single timbre model to be compared, computes similarity between each pair of timbre models using Jensen-Shannon approximation, and normalizes the

---

[1] Note that rhythms are not eliminated for the perceptual stimuli, only for the PMI calculation (see [10] for discussion of treatment of rhythm in the PMI method).

similarities with Mutual Proximity [17-18]. We used the default algorithm because it has been found to have higher accuracy [17].

We prepared the full-audio version of the music excerpts from the dataset of court cases and fed them to the default algorithm of Musly to compute similarity. The output of the Musly algorithm is a distance matrix where distances, i.e. differences, between every two songs are listed. Because the Musly default algorithm normalizes the results, all the distances range between 0 and 1. Consequently, we calculated the audio music similarity by subtracting distance values from 1 and multiplying by 100 to convert the results into percentage terms for consistency with our other methods.

5.2.1 Audio Similarity Results

The results of Musly-calculated audio music similarity values for all 17 tested cases are shown in Table 1, appended next to the column of PMI values. The area under the ROC curve (AUC) is 0.69. The optimal cutoff threshold of Musly-calculated similarity is 32.8% with sensitivity = 0.67 and specificity = 0.75. Using this cutoff, Musly algorithm was also able to accurately classify 12 out of the 17 cases (71%) to match the court's decisions. The five failure cases are highlighted by bold font in Table 1 and briefly analyzed below.

6. AUTOMATED VS. PERCEPTUAL JUDGEMENTS

6.1 PMI vs. Perceptual Data

Mean perceptual similarity of each court case was calculated by averaging participants' individual ratings of similarity. The perceptual similarity values for the 17 court cases are listed in Table 1 and highlighted by light green. Figure S2 shows the relationship between PMI values and perceptual similarity under the three different conditions. Regression analyses show that the PMI melodic similarity is significantly correlated with perceptual similarity for both full-audio and melody-only conditions (full: R = 0.58, p = 0.014; melody: R = 0.59, p = 0.012), but not for the lyrics-only condition (R = -0.058, p = 0.84).

6.2 Musly vs. Perceptual Data

We also compared the Musly-calculated audio music similarity with the perceptual data collected. Figure S3 shows the correlation between Musly similarity and perceptual similarity of the 17 tested court cases under three different conditions for perceptual judgement. Regression analyses indicate that the Musly audio similarity has no significant correlations with perceptual similarity for all three condition groups of "full-audio", "melody-only", and "lyrics-only" (full: R = 0.26, p = 0.32; melody: R = 0.082, p = 0.76; lyrics: R = 0.059, p = 0.84).

7. DISCUSSION

Overall, our analyses showed moderate agreement between automated and perceptual judgements of music copyright infringement. Both automated similarity algorithms – PMI for symbolic data and Musly for audio data – matched past court decisions with relatively high accuracy (both 71%). The fact that PMI was significantly correlated with perceptual similarity for both melody-only and full-audio provides validation for PMI as a perceptually relevant measure of melodic similarity and is consistent with the idea that melodic similarity plays a role in judgements of overall musical similarity [19].

The lack of correlation between Musly's audio similarity algorithm and perceptual similarity was surprising given that Musly's algorithm has previously performed well in evaluations of general musical similarity. This may be partly explained by Musly's reliance on MFCCs to capture timbral and rhythmic similarity, not melodic similarity. Previous studies have shown that limited inter-rater reliability in judgements of musical similarity can limit the performance of automated algorithms [13]. Future analyses using supervised learning or other algorithms for capturing melodic similarity [1] may be able to improve performance, although the subjective nature of musical similarity will still place limits on the ability of any algorithm to match human judgements.

Surprisingly, both automated methods had higher accuracy than that of perceptual judgement, with both automated methods able to accurately predict 71% (12/17) of previous court decisions while perceptual accuracy were 58% and 54% under full-audio and melody-only conditions respectively. We suspect that allowing the algorithms to optimize the similarity threshold via the ROC analysis helped to improve - and probably overfit - the automated analyses[1]. Future analyses with larger data samples should consider calibrating parameters on a training subset before evaluating them on a separate test subset.

There are several possibilities for the low levels of perceptual accuracy. The fact that participants showed very high levels of accuracy (almost 100%) for randomized plaintiff samples suggests that the results were not merely random, but the inclusion of such samples might conceivably have skewed judgements by including levels of dissimilarity rarely included in real court cases. The fact that musicians performed similarly to non-musicians suggests that lack of musical expertise is also unlikely to explain the low performance. Although we cannot rule out effects of participants' familiarity because we failed to collect such data, any familiarity effects when participants were aware of the cases would be predicted to increase, rather than decrease, accuracy.

Instead, some past court decisions (e.g., the cases involving "He's So Fine" and "Blurred Lines") have been so controversial as to be debatable whether they were in fact "correct" [6]. Indeed, it seems likely that the dynamics of copyright lawsuits create a type of selection bias in which cases where infringement or lack of infringement are obvious are more likely to be resolved out of court[2] without a final court decision, while only the most ambiguous

---

[1] The chance of getting an accuracy of 12 or more correct by chance is actually 26%.

[2] One case (HarumakiGohan v. Mori) was settled out of court, and this case displayed some of the highest levels of participant accuracy.

cases require the court to make a final adjudication. In the future, rather than relying for ground truth only on court decisions and the selection bias they may create, our perceptual experiment may provide an alternative source of ground truth for disputes that were resolved out of court and thus tend to lack objective legal documentation.

Our prediction that listening to melody-only would provide superior accuracy than listening to full-audio was not supported. The fact that our prediction was not only not significant but was in the wrong direction suggests that limited statistical power cannot explain this result. Instead, despite legal arguments suggesting that non-melodic factors should generally be ignored and the sample having been selected based on the criteria of melodic similarity [1], individual cases are always complex and factors such as lyrics, instrumentation, and other non-melodic factors did in fact play roles in past decisions [14]. Overall, participants tended to judge melody-only versions as less similar than full-audio, with accuracy tending to be lower for cases judged as infringing. This suggests that participants have more difficulty detecting infringement using melody only. Since including non-melodic information appears to help (or at least not hurt) improve accuracy even for this sample emphasizing melodic similarity, this may suggest that allowing juries to hear full audio recordings without restricting them to sheet music depositions could actually help improve accuracy in legal cases. However, this hypothesis remains speculative until it can be more rigorously tested at larger scales (and the issue discussed above of determining "correct" decisions more thoroughly addressed).

The average results for each case shown in Figure 1 displayed a normal distribution for full-audio but were hourglass-shaped/bimodal for melody-only and lyrics-only. For the lyrics-only condition, this distribution reflects that most participants judged non-infringement for most cases, which is consistent with the fact that this sample was not selected to include many example of lyrics infringement. The melody-only condition led to higher accuracy for some cases (as predicted), but lower accuracy for others (contra predictions).

The accuracy of the PMI algorithm for the current study of 71% (12/17 cases) was slightly lower than the value of 80% (16/20 cases) reported in a previous study using a similar dataset. There are two reasons for this: 1) The sample was different – this study excluded 6 cases without matching full audio recordings and added 3 new Japanese cases (the new Japanese cases were not selected based on PMI values or any quantitative criteria, but they all were correctly classified by the PMI algorithm). 2) In the process of preparing controlled audio files for the experiment that were exactly matched, we noticed that several of the transcriptions used in [1] and [3] either did not exactly match the audio recordings, or had mismatched lengths.

Compared with the previous published study on PMI [3], the current PMI method successfully classified two case that were not accurately classified in the 2018 testing (*Three Boys Music v. Michael Bolton* and *Grand Upright v. Warner*), but three cases previously classified successfully now failed to be successfully classified (*Swirsky v. Carey*, *Granite Music v. United Artists*, and *Jean et al. v. Bug Music*). Two cases (*Selle v. Gibb* and *Fantasy v.*

Fogerty) remained failures in both studies, but these two exceptions were not due primarily to a failure of the melodic similarity algorithm but rather to the complex nature of musical copyright law [3]. These discrepancies show how results from the PMI method can be affected by errors and uncertainties in the transcription process.

While the Musly algorithm resulted in the same overall accuracy as the PMI method (71%), 4 of the 5 mis-classified cases were different between the two methods. Both methods mis-classified *Fantasy v. Fogerty* as infringing when the court decision was non-infringement (see [3] for discussion of legal details). The four cases uniquely misclassified by the Musly but not PMI method largely seemed to be of the type predicted by the melody-centric view of copyright in which non-melodic similarities or differences interfered with assessment of melodic similarity. For instance, *Herald Square Music v. Living Music* showed low audio similarity via Musly despite high melodic similarity and a finding of infringement. In this case, the different timbres where one melody is performed by a saxophone with background noise while the other is sung by a vocalist with piano accompaniment seem to obscure similarities in the two melodies.

The fact that both algorithms failed for different sets of cases, and the fact that participants who made judgements only based on audio similarity without information about the historical/legal context performed even lower than the algorithms, suggests that the complexities of copyright law are difficult to fully capture through objective measurement of similarity alone. The relative emphasis on melody, lyrics, other musical aspects, and extra-musical legal factors changes from case to case, limiting the power of any single objective method. This supports previous caveats that, while objective quantitative methods may help supplement traditional qualitative analysis, "Trial by algorithm will never replace trial by jury, nor should it." [3].

8. FUTURE DIRECTIONS

The primary limitation of our study at present is its limited size and scope, with a dataset of only 17 court decisions (17 from USA) and perceptual ratings from only 20 participants. Furthermore, some of the cases include non-musical aspects that make it difficult for current automated methods focusing on musical similarity to identify those exceptions. Thus, we plan to expand the testing data by including more usable cases which have court decisions and have no non-musical factors that have affected the court decisions. Preliminary screening of the 238 cases at the Music Copyright Infringement Resource [14], we found 50 potentially usable court cases we plan to investigate in future studies. To increase diversity and cross-cultural generalizability, we also plan to identify more non-US cases, particularly from Japan and China where music industry revenues are substantial.

One promising direction may be to expand from a focus purely on music copyright infringement to also include the related domain of cover-song detection. Because there are larger databases and more sophisticated algorithms being developed for cover-song detection, these may provide more powerful methods that could be adapted to copyright infringement in future research [11-12].

9. DATA/CODE AVAILABILITY

Musical stimuli, data and analysis code are available at https://github.com/compmusiclab/music-copyright. The full experiment can be accessed at https://music.keio.moe/experiments/copyright/full.

10. AUTHOR CONTRIBUTIONS

Conceptualization: PES, CC, QDA, DM, SF, SO, YY; Methodology/ Analysis/ Investigation/ Visualization: YY, SO, PES; Project administration/ Supervision/ Funding acquisition: PES; Writing – original draft: YY, PES, SO; Writing – review & editing: CC, DM, QDA, SF.

11. ACKNOWLEDGMENTS

12. REFERENCES

[1] D. Müllensiefen, and M. Pendzich, "Court decisions on music plagiarism and the predictive value of similarity algorithms," *Musicae Scientiae*, 13(1 Suppl), pp. 257-295, 2009.

[2] M. Robine, P. Hanna, P. Ferraro, and J. Allali, "Adaptation of string matching algorithms for identification of near-duplicate music documents," in *Proc. of the International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN 2007)*, Amsterdam, Netherlands, 2007, pp. 37-43.

[3] P. E. Savage, C. Cronin, D. Müllensiefen, Q. D. Atkinson, "Quantitative evaluation of music copyright infringement," in *Proc. of the 8th International Workshop on Folk Music Analysis (FMA 2018)*, Thessaloniki, Greece, 2018, pp. 61-66.

[4] E. Selfridge-Field, "Substantial Musical Similarity in Sound and Notation: Perspectives from Digital Musicology," *Colorado Technology Law Journal*, vol. 16, pp. 249-284, 2018.

[5] *Lotus Development Corp. v. Borland Intern., Inc.*, 49 F.3d 807, 813 (1st Cir. 1995).

[6] J. P. Fishman, "Music as a Matter of Law," *Harvard Law Review*, vol. 131, no. 7, pp. 1861-1923, 2018.

[7] E. Selfridge-Field, "Conceptual and Representational Issues in Melodic Comparison," in *Melodic Similarity: Concepts, Procedures, and Applications*, Cambridge, MA, USA: MIT Press, 1998.

[8] J. Lund, "An empirical examination of the lay listener test in music composition copyright infringement," *Virginia Sports & Entertainment Law Journal*, vol. 11, pp. 137-177, 2011.

[9] *Williams v. Gaye*, 885 F.3d 1150, 1160 (9th Cir. 2018).

[10] P. E. Savage and Q. D. Atkinson, "Automatic tune family identification by musical sequence alignment," in *Proc. of the 16th International Society for Music Information Retrieval Conf. (ISMIR 2015)*, Malaga, Spain, 2015, pp. 162-168.

[11] J. Serrà, E. Gómez, and P. Herrera, "Audio cover song identification and similarity: background, approaches, evaluation, and beyond," in *Advances in Music Information Retrieval (Studies in Computational Intelligence, vol. 274)*, Z. W. Raś and A. A. Wieczorkowska (eds), Springer-Verlag Berlin Heidelberg, 2010, pp. 307-332.

[12] F. Yesiler, C. Tralie, A. A. Correya, D. F. Silva, P. Tovstogan, E. G. Gutiérrez, and X. Serra, "Da-TACOS: A dataset for cover song identification and understanding," in *Proc. of the 20th International Society for Music Information Retrieval Conf. (ISMIR 2019)*, Delft, Netherlands, 2019, pp. 327-334.

[13] A. Flexer and T. Grill, "The problem of limited inter-rater agreement in modelling music similarity," *Journal of New Music Research*, vol. 45, no. 3, pp. 239-251, 2016.

[14] C. Cronin, *Music Copyright Infringement Resource*, Retrieved from https://blogs.law.gwu.edu/mcir/, 2020.

[15] *Arnstein v. Porter*, 154 F.2d 473 (2d Cir. 1946).

[16] M. Mandel and D. Ellis, "Song-level features and support vector machines for music classification," in *Proc. of the 6th International Society for Music Information Retrieval Conf. (ISMIR 2005)*, London, UK, 2005.

[17] D. Schnitzer, *Musly - an open-source audio music similarity library*, https://www.musly.org, 2014.

[18] D. Schnitzer, A. Flexer, M. Schedl, and G. Widmer, "Using Mutual Proximity to Improve Content-Based Audio Similarity," in *Proc. of the 12th International Society for Music Information Retrieval Conf. (ISMIR 2011)*, Miami, Florida, USA, vol. 11, 2011, pp. 79-84.

[19] H. Allan, D. Müllensiefen, and G. A. Wiggins, "Methodological Considerations in Studies of Musical Similarity," in *Proc. of the 8th International Society for Music Information Retrieval Conf. (ISMIR 2007)*, Vienna, Austria, vol. 6, no. 1, 2007, pp. 463-466.

MEASURING DISRUPTION IN SONG SIMILARITY NETWORKS

Felipe Falcão[1]   **Nazareno Andrade**[1]   **Flávio Figueiredo**[2]
Diego Silva[3]   **Fabio Morais**[1]
[1] Universidade Federal de Campina Grande, Brazil
[2] Universidade Federal de Minas Gerais, Brazil
[3] Universidade Federal de São Carlos, Brazil

`felipevf@copin.ufcg.edu.br, nazareno@computacao.ufcg.edu.br`

ABSTRACT

Investigating music with a focus on the similarity relations between songs, albums, and artists plays an important role when trying to understand trends in the history of music genres. In particular, representing these relations as a similarity network allows us to investigate the innovation presented by these entities in a multitude of points-of-view, including disruption. A disruptive object is one that creates a new stream of events, changing the traditional way of how a context usually works. The proper measurement of disruption remains as a task with large room for improvement, and these gaps are even more evident in the music domain, where the topic has not received much attention so far. This work builds on preliminary studies focused on the analysis of music disruption derived from metadata-based similarity networks, demonstrating that the raw audio can augment similarity information. We developed a case study based on a collection of a Brazilian local music tradition called Forró, that emphasizes the analytical and musicological potential of the musical disruption metric to describe and explain a genre trajectory over time.

1. INTRODUCTION

Inflections on creative threads are prevalent events that can be observed multiple times throughout music history [1]. The emergence of punk rock in the early seventies, for example, changed the traditional rock and roll in many aspects to create a unique music expression [2]. The music branch of the punk culture brought heavy lyrics, aggressive looks, and even deep acoustic changes to the songs, which were more aroused and noisier than songs from previous decades. Such music aspects were replicated over time, as evidenced for example in the expert-curated influences credited in the AllMusic guide [3] to artists from the early stages of the punk rock (e.g. *Ramones*, *Bad Religion*, *Sex Pistols*). The guide attributes to these bands influence over more recent ones, such as *Green Day* and *The Offspring*.
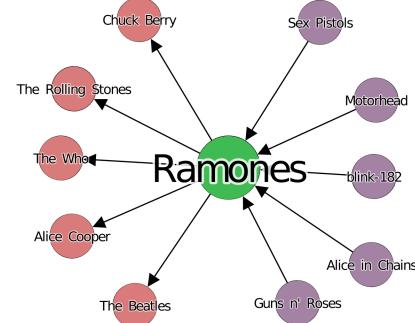
Figure 1. Network topology for a disruptive artist.
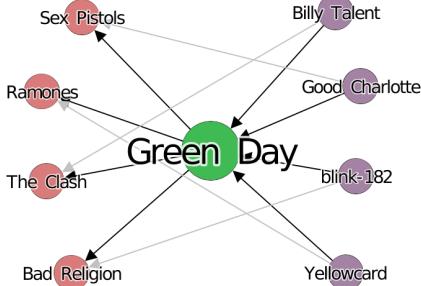
Regarding the different creative roles played by artists during the genre trajectories, the AllMusic guide defines the *Ramones* as *"inarguably the most relevant band in punk history, creating the stylistic prototype that would be followed by countless bands who emerged in their wake"* [1] . That points out to a particular innovative case where an artist had a significant and primary influence over the rupture of some well-established guidelines. Thus, an artist can be considered *disruptive* when your music contribution is developed in a self-sufficient way, abruptly shifting the present creative thread.

Conversely, AllMusic suggests a different nature of innovation when describes *Green Day* as *"influenced by the late-'70s punk predecessors, they went on to introduce a new, younger generation to the genre"* [2] . On the opposite of the disruptive movement by the *Ramones*, this excerpt allows describing *Green Day*'s creative potential as a consolidation of the previous practices, including their particular musical signature in the meantime.

Both musical creative natures can be represented by a network model, where the nodes represent artists and the edges symbolize the influence relation of one artist over another. Figure 1 shows a disruptive innovation by the *Ramones* based on influences metadata extracted from AllMusic, where edges indicate an "influenced by" relation. Predecessors (i.e., one that chronologically preceded another) of a focal node (in green) are represented by red

---

[1] `https://www.allmusic.com/artist/ramones-mn0000490004/biography`
[2] `https://www.allmusic.com/artist/green-day-mn0000154544/biography`

Figure 2. Network topology for a consolidator artist.

Figure 3. Demonstration for different types of influence.

nodes, whereas purple nodes represent the successors.

Due to its self-sufficient nature, one may expect that most of the artists (nodes) that succeed and are influenced by a disruptive artist connect to this artist, but not to its predecessors. A similar explanation can be given to describe the network topology for artists that consolidate the genres over time (Figure 2 for *Green Day*): these artists reaffirm a thread of influences, as their successors are usually influenced by both them and their predecessors.

Despite its simple semantics and noticeable potential in enriching musicological analyses about the history of genres, there is limited research that measures the disruption of songs over time. In particular, this work is based on the disruption quantification using a metric derived from audio similarity networks. Specifically, we model a network of similarity among songs and use their temporal precedence to explore patterns of similarity that reveal creative aspects and disruption over time.
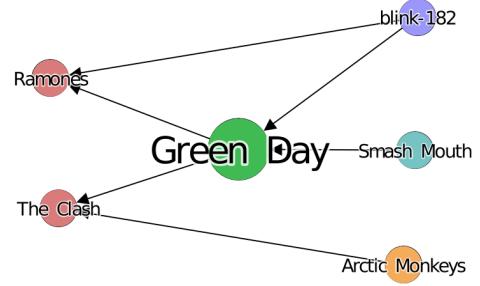
Features extracted from raw audio are reportedly a rich source of similarity information [4], as they cover many music aspects, such as timbre [5, 6], harmony [7, 8], and rhythm [9]. Therefore, consider such types of data to construct similarity networks can be valuable in understanding how songs of the same genre are acoustically related. In this work, a musical disruption analysis is proposed over this similarity network, allowing to unveil some interesting findings of the disruption of songs over time. To promote a better interpretation of results, we collected a new audio dataset comprising songs of a definite style, called Forró.

In this analysis we represent songs as Mel-Frequency Cepstral Coefficients (MFCCs), using these representations to build a similarity network that connects songs with similar acoustics. Next, we process this network's topology to calculate the disruption metric for all songs, summarizing the most disruptive music pieces. This analysis allows us to validate the disruption metric in the music context. Both the dataset and the network representations generated during the experiments are made available for further studies.

2. BACKGROUND

2.1 Music Innovation & Corpora

Music innovation is not a popular main research topic, being usually mentioned in studies focused on modeling mu-

sic influence [10–12]. In particular, Shalit et al. [13] proposed a dynamic topic model to represent music influence over time using metadata and audio. In their work a song is considered influential if its language gets replicated by subsequent works, while innovation is modeled as the extent of which the model accounts for a song when trained only with data from the past. Their findings leveraged the Million Song Dataset [14] to point to correlations between influence and innovation only during some short periods in the early 70's and mid-90's.

Associations between music influence and innovation were also investigated by Noyer and his collaborators [15]. Using a network to represent influences between artists, the authors tried to find topological differences between innovative and non-innovative artists. Their approach analysed artists data from 1951 to 2008, measuring innovation as the number of Grammy awards won by each artist. Conclusions identify that innovation in fact impacts network topology, showing that artists with more awards presented considerably more structural holes on their sub-networks.

Corpora plays a major role when representing the influence relations between artists, albums and songs. Both the Million Song Dataset [13,16] and the information available on the AllMusic music catalogue [15, 17, 18] have been used as audio and metadata source for many MIR tasks, including influence modeling.

2.2 Disruption Index

This present work builds on a network metric proposed by Funk & Owen-Smith [19] to measure destabilizing and consolidating influences of inventions over existing technology streams. Their CD_t index assumes that the degree of destabilizing influence (disruption) of an invention within an influence network should be measured in terms of "how future inventions make use of the technological predecessors cited by a focal patent". Given that notion, Figure 3 depicts an example of the three types of music influences for a focal node a (*Green Day* in the example) used for measuring disruption according with CD_t: Let n_i be the number of nodes i that reference only a and none of its predecessors (*e.g., Smash Mouth*), n_j be the number of nodes j that reference both a and at least one of its predecessors (*e.g., blink-182*), while n_k accounts for the number of nodes k that do not reference a but reference at least one of its predecessors (*e.g., Arctic Monkeys*). Disruption

(henceforth referenced as D) is then measured as:

$$D = \frac{n_i - n_j}{n_i + n_j + n_k} \tag{1}$$

D ranges from -1 to 1, where values close to 1 indicate nodes with highly disruptive potential whereas measures around the negative extreme denote nodes that mostly consolidated influences over time and therefore were cited in parallel with their predecessors.

The original case study for D was developed over a database of patents granted in the US between 1977 and 2005. Their findings indicate that disruptive inventions are usually boosted by federal research funding initiatives, while commercial ties are more related to the consolidation of the *status quo*. Such validation was later expanded by Wu et al. [20], that enriched the dataset with scientific papers and software repositories, accounting now for a total of 65 million observations. The study concluded that disruptive products are associated with smaller teams, while larger groups mostly consolidated knowledge.
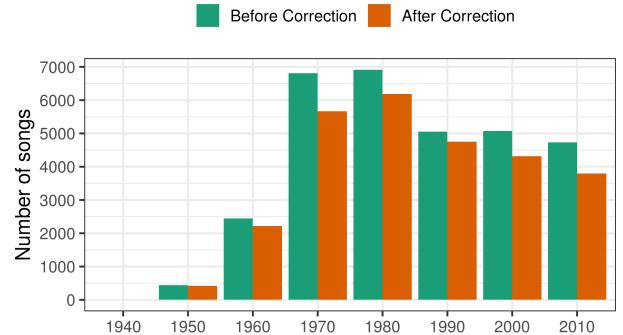
The disruption metric was recently experimented in the music context by Figueiredo and Andrade [17]. They leveraged influence metadata from AllMusic to create a network linking 32,568 artists according with their influence relations (i.e. a "link from artist a to b denotes that a has been influenced by b"). Disruptions are then extracted for all the network components, triggering discussions regarding disruptive and consolidator potential. In particular, they confirm the results of [13] about the lack of correlation between influence and disruption, also concluding that D translates structural insights that are not derived from any existing network metrics.

3. COLLECTED MUSIC DATA
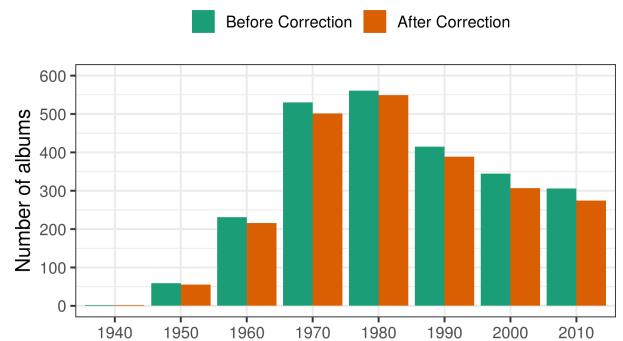
Our musical disruption analysis uses audio data from a Brazilian cultural manifestation called Forró (composed by music, dance, and festivities), native from the northeast of Brazil during the second half of the 19th century. The Forró music genre is composed of three preponderant instruments: accordion, triangle, and a percussive drum called *zabumba*. *Luiz Gonzaga* is the most prominent representative of this genre and is responsible for spreading his music to other regions of Brazil.

The audio data was obtained from the collaborative site *Forró em Vinil* [21], which organize and publish contents that register the history of Forró (e.g. albums, books, and pictures). The audio collection is maintained by media collectors that own long play records and CD's that are no longer produced by record labels. These collectors digitize their media and provide the audio files to the site's administrators, responsible for curating the collection. We built a dataset covering Forró songs ranging from the years 1945 to 2016, by scraping the site via a web crawler. Overall, 2,449 distinct albums were collected, grouping a total of 31,485 songs, each one annotated with artist, album, and release year.

To ensure that the collected data is only comprised of Forró songs, we excluded other genres found on a descrip-

Figure 4. Histogram for number of songs over decades.

Figure 5. Histogram for number of albums over decades.

tive analysis phase. Moreover, to guarantee a chronological information required by the nature of this study, we filtered out albums without release year informed. These data corrections were necessary to satisfy genre-specific and time constraints requirements. Figures 4 and 5 show the song and album distribution over decades before and after dataset correction, respectively. Out of the original 2,449 albums, 2,293 satisfied the constraints, accounting for 27,352 songs which we considered.

4. MUSIC & SIMILARITY REPRESENTATIONS

We leverage Mel-Frequency Cepstral Coefficients (MFCCs) as feature for audio similarity estimation. MFCCs are robust music representations often used in many music information retrieval tasks [22], including genre classification [23], music recommendation [24, 25] and audio similarity [26, 27]. Moreover, given its reported ability to model timbre information [28], we expect that this feature will also capture relevant audio events when iterating over our data. In particular, we look for disruptive episodes when an artist included new instruments to the basic setup discussed on Section 3, which is something that actually happened during the history of Forró.

Similarly to what is proposed by Choi et al. [22] for their baseline feature, we employ as our audio feature the means and standard deviations for 20 MFFCs and their first order derivatives over the entire song. Audio processing techniques are aided by the Librosa package [29]. The result of this feature extraction methodology is a collection

32

Task #	# of classes (n in top-n)	Max. items per class	Sample size	Precision
1	20 classes	500 items	6572 items	**0.79**
2	50 classes	25 items	1125 items	**0.88**

Table 1. Sampling settings and reported precisions for artist (#1) and album (#2) classification tasks.

of 27,352 vectors (henceforth referenced as *feature vectors*), each one containing 80 elements that represent audio information.

An extra validation step is also conducted to confirm those feature vectors encode enough audio information to generate comprehensible music similarities. Two multiclass audio classification tasks are designed to measure the precision of machine learning classifiers when trained with feature vectors from the *Forró em Vinil* Dataset:

- Task 1: Artist classification: classification of artists among the *top-n* (those with more songs);

- Task 2: Album classification: classification of albums among the *top-n* (those with more songs);
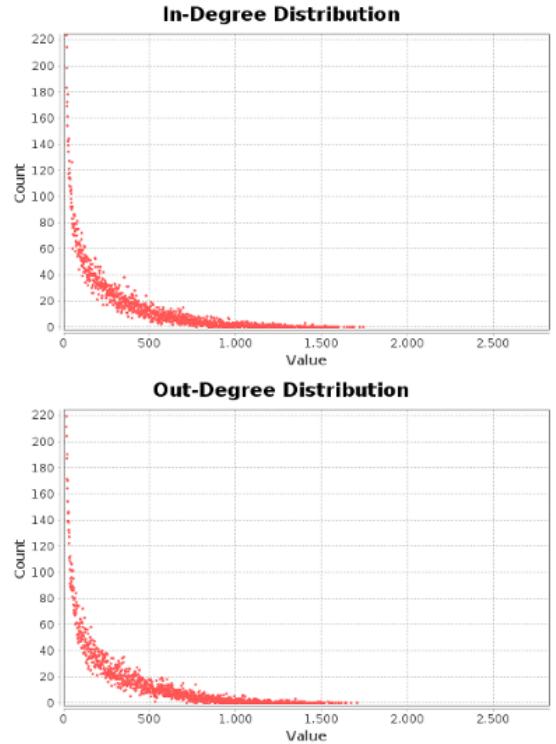
SVM classifiers are used in both cases, given their efficiency in tasks with small training sets. Model training was done using scikit-learn [30] and experiments are run with 10-fold cross-validation using stratified splits. Models have their parameters optimized upon the use of gridsearch on the validation phase and reported precision values are related to the best classifier after all splits are done. Table 1 summarizes both the dataset sampling strategies and scores for artist and album classifiers, indicating high precisions for both cases (79% and 88%, respectively).

These partial results present two interesting findings that support the next steps. First we can now fairly assume that our vector representations encode enough audio information to derive similarity measures. The second conclusion refers to the best performing kernel function considered by the grid-search routine for both classifiers: the Radial Basis Function (RBF) kernel. The RBF kernel models vector distance, and its mathematical definition [31] assigns to itself a similarity interpretation [32] (i.e., values ranging from 0 to 1, inversely proportional to the vector distance). Given its potential on providing similarity insights for sequential data, we opt for using the RBF kernel as similarity measure for pairs of feature vectors.

5. SIMILARITY NETWORK

To measure disruption we first need to construct a directed network connecting similar objects. When it comes to song similarity networks, the nodes are the songs and an edge between any pair of songs represent a similarity relationship. We now describe how our network was built.

Each song from the *Forró em Vinil* Dataset is a single node in this network. As for the edges, although the RBF kernel allows us to quantify the similarity between any pair of songs, the binary choice about whether or not we create

Figure 6. Distribution of in and out degrees.
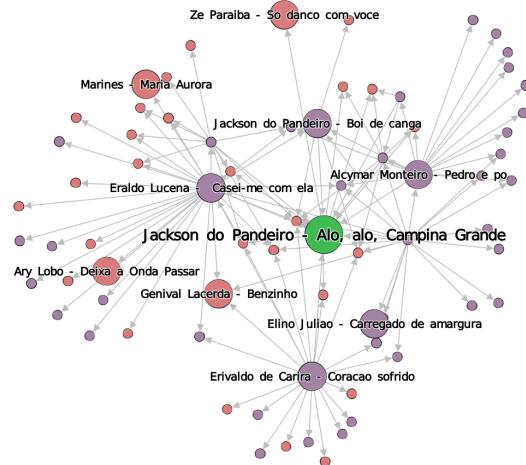
an edge between two nodes depends on the definition of a similarity threshold above which we can safely ensure that a similarity edge exists. In order to empirically select this threshold, we leverage the fact that songs from the same album are arguably a fair ground truth for noticeable similarity (i.e., these songs usually share the same instrument and voice settings). Thus, we iterate over the whole dataset checking the average similarities between each song s from album a and every other song s' from a. This analysis informs an average similarity of ≈ 0.90, which from now on is used as threshold when creating edges.

To create our network, for each song s we query the similarity matrix among all songs and create an edge from s to a predecessor s' if their similarity is greater than or equal the threshold. Additionally, to limit our analysis to a timeframe where stylistic movements are observable, we enforce a time window within which two songs must fall to in order to enable connections between them. Here the size of this time window is 10 years, as it seems reasonable in this context and was also used in [20] when deriving influences between scientific papers.

Out of the 27,352 original songs from the dataset, 26,452 are included in the resulting graph, connected by 5,728,466 directed edges. This minimal difference from the original songs count is explained by the removal of disconnected nodes (i.e. songs that do not sound similar to any other). 98% of all nodes are densely connected to the same giant component and in and out-degrees distribution can be observed on Figure 6. Moreover, Figure 7 illustrates an ego-network extracted from the original structure.

Song	Artist	Album (year)	Disruption Index & Comments
Padrinho Cícero do Juazeiro	*Trio Juazeiro*	*Pedaço de fulô (1982)*	$D = 1$ ($n_i = 37, n_j = 0, n_k = 0$). A fast song (140bpm) with a clear and complex accordion arrangement. The sub-network focused around its node evidences connections with multiple songs from same albums, what might indicate the emergence of a new (disruptive) acoustic setting that was adopted by following artists, like *Clemilda* and *Roberto do Acordeon*.
Namorada de João	*Coroné Narcisinho*	*Forró do Serrado (1969)*	$D = 1$ ($n_i = 28, n_j = 0, n_k = 0$). The song brings a very audible triangle as part of its percussive setup, what can't be perceived in most of the songs from the same epoch. Dominant triangles can also be heard in many of the songs that succeded *Namorada de João*, as in *Esse forró eu danço* (*Abdias* - 1977).
Sem vergonheira	*Marinês*	*Canção da fé (1972)*	$D = 1$ ($n_i = 24, n_j = 0, n_k = 0$). Marinês is one of the first female Forró singers. *Sem vergonheira*, as many of her songs, presents a combination of a strong lead singing voice and effective backing vocals, an unusual practice back then. Similar strategy is used by some of its succeeding songs, like *Quebra-cabeça* (*Trio Nordestino - 1981*).
Derramaro o gai	*Luiz Gonzaga*	*O nordeste na voz de Luiz Gonzaga (1962)*	$D = 1$ ($n_i = 22, n_j = 0, n_k = 0$). The refined accordion melodies are undoubtedly the strongest aspect of *Luiz Gonzaga*'s work, and this song is proof of that. *Derramaro o gai* has multiple disruptive connections with other songs from its very same album, as well as similarities with songs from *Severino Januário*, his brother.
Lembranças	*Flávio José*	*Só confio em tu (1977)*	$D = 1$ ($n_i = 19, n_j = 0, n_k = 0$). The song empowers the acoustic guitar among the original Forró instrumentation, what was rare back in the late seventies. Similar songs by *Flávio José* solidify this new creative branch, imitated by artists like *Marinês* and *Genival Lacerda*.

Table 2. Top-5 of disruptive songs according to the D measure.

Figure 7. Ego-network for *Alô, alô, minha Campina Grande* by *Jackson do Pandeiro* ($D = -0.07$). Red nodes preceded and purple succeeded the focal, green, node.

6. DISRUPTION ANALYSIS
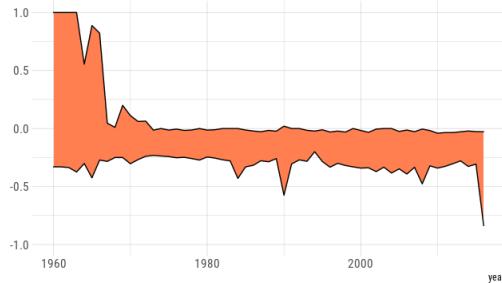
We can now combine the disruption metric D with the similarity network proposed in the previous section to trigger discussions regarding the disruptive potential of songs over the history of Forró. Since disruption as modeled by Equation 1 depends on preceding data (i.e., n_j and n_k nodes), we decide to use songs prior to 1960 only as comparison data for the following decades, hence no disruptions for these are reported. In other words, the songs from the forties and fifties are a part of the graph (they impact the disruption of future songs), we just do not report their disruption. All the other songs have their disruption indexes

derived according to the i, j and k as described on Section 2.2.

Table 2 depicts data from the disruption ranking and summarizes the five most disruptive songs of the *Forró em Vinil* Dataset, trying to support these findings with specifics related to the songs acoustics and their similarity relations. Although artist influence is not a mandatory requirement when determining disruption, it's meaningful to evidence that music pieces from representative artists such as *Luiz Gonzaga*, *Marinês* and *Flávio José* are considered disruptive according to our analysis.

We draw special attention to songs from *Sivuca* that are included among the most disruptive ones (eight songs with $D \geq 0.5$). This musician, widely acclaimed for his work both in Brazil and the United States, was a multi-instrumentalist with strong accordion and acoustic guitar skills. Many of his songs with high disruption in the network combine elements from a variety of genres other than Forró, like *Choro*, *Frevo*, Jazz and Blues. The acoustic richness assigned to his work as well as the uniqueness of the music performed by *Sivuca* generate a lot of internal similarity relations between his own songs, causing the high disruptions. To put it another way, when it comes to Forró, *Sivuca* was disruptive in the sense that his work was mostly influenced by himself, and himself only. This peculiar finding is a representative example of how the disruption metric can actually help to identify meaningful events hidden inside the history of genres.

Next, we leverage the disruption information to model how the creative thread for Forró was developed during the past seven decades. With this analysis we aim at finding exactly when the genre presented creative inflections and how often these events happen during its history. Due
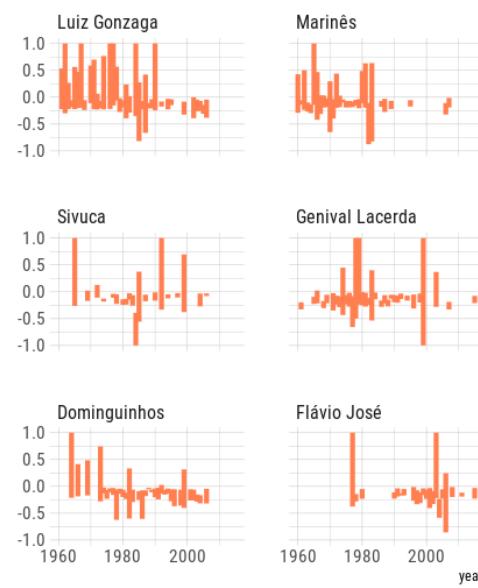
Figure 8. 5 to 95 percentile range of D over the years.

to the large number of songs with D around zero, caused by the dense network, we opt to summarize this disruption distribution in Figure 8 using 5 to 95 percentile ranges over the years. In overall, the higher disruptions of Forró are mostly concentrated on its first years, specially in the interval between 1960 and 1970. While at a first glance this may look like a natural consequence of these being the first songs in the dataset, recall that we omit an entire decade from Figure 8 (i.e., to filter out biases due to first mover advantage, songs from the 1950's impact the disruption of future songs but are not present in our analysis).

We further queried the ranking to understand what happened in the 1960's. Firstly, we see that this high creative load is guided by multiple disruptive songs from pioneers of the history of Forró, like *Luiz Gonzaga*, *Jackson do Pandeiro* and *Marinês*. When we investigate the biographies of these artists (from the AllMusic Guide), we point out facts such as: *Jackson do Pandeiro*[3] is considered: *"one of the most inventive and influential Brazilian musicians"*, *Luiz Gonzaga*[4] is cited as '*"one of the most influential figures of Brazilian popular music in the twentieth century"*. Finally, *Marinês* was the first woman to have a Forró group[5]. Biographies were last accessed in August 2020.

Nevertheless, we do point out that the following decades were also presented with disruptive songs. In particular, we propose an artist by year investigation to unveil some insights regarding artists who have unsettled the creative structure of Forró. Figure 9 uses the same percentile approach as Figure 8 to summarize the disruption information for the six artists with higher averaged D for aggregated data (i.e., all songs from the artist in the network). Again we see *Luiz Gonzaga* and *Marinês* figuring as very disruptive artists, with a high creative production specially until 1980, when their careers came to an end (*Luiz Gonzaga* died in 1989 and *Marinês* reduced her production after late 1980). Their creative legacy seems to have been inherited by *Genival Lacerda* and *Flávio José*, other disruptive artists that have been active since the seventies and which often perform disruptive songs since then. These other artists provide further evidence that our ranking is not entirely explained by first mover advantage.

Figure 9. 5 to 95 percentile range of D for disruptive artists over the years.

7. FUTURE WORK & CONCLUSIONS

The present study proposed an audio-based approach to extend the experimentation of a disruption metric in the music context. A new dataset comprised of songs from a Brazilian music tradition was collected to allow for a specific case study. The data supported the generation of an audio similarity network that models the creative flow of songs over time. Results derived from the disruption index underline the semantic potential attached to it, by triggering discussions about specific times when the genre had creative inflections and even which artists were responsible for these events. We argue in favor of applying similar approaches to different contexts, as this may unveil interesting findings about the history of many music genres.

A complementary research direction encourages some enhancements on Equation 1. In particular, we advocate that this formula should also account for the nodes similarities encoded on the edges, instead of simply dealing with creative relations in a binary fashion. That would prevent future studies from having to define a similarity threshold to choose whether or not similarity edges are included in the network, as suggested by this work.

Reproducibility: Both the MFCCs for the *Forró em Vinil* Dataset and the generated similarity network (Graph Exchange XML Format)[6], as well as the code used during the experiments[7] are publicly available.

---

[3] https://www.allmusic.com/artist/mn0000109367
[4] https://www.allmusic.com/artist/mn0000316340
[5] https://www.allmusic.com/artist/mn0000371916

[6] https://zenodo.org/record/3820920
[7] https://github.com/nazareno/forro-disruption

8. REFERENCES

[1] P. Tschmuck, *Creativity and innovation in the music industry.* Springer, 2006.

[2] D. Simonelli, "Anarchy, pop and violence: Punk rock subculture and the rhetoric of class, 1976-78," *Contemporary British History*, vol. 16, no. 2, 2002.

[3] AllMusic, "Allmusic - record reviews, streaming songs, genres and bands," (accessed August, 2020), "https://www.allmusic.com".

[4] M. Müller, *Information retrieval for music and motion.* Springer, 2007, vol. 2.

[5] M. Muller and S. Ewert, "Towards timbre-invariant audio features for harmony-based music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, 2010.

[6] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra, "Timbre analysis of music audio signals with convolutional neural networks," in *EUSIPCO.* IEEE, 2017.

[7] M. Müller and S. Ewert, "Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features," in *ISMIR*, 2011.

[8] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," in *ISMIR*, 2016.

[9] G. T. Toussaint, L. Matthews, M. Campbell, and N. Brown, "Measuring musical rhythm similarity: Transformation versus feature-based methods." *Journal of Interdisciplinary Music Studies*, vol. 6, no. 1, 2012.

[10] J. Atherton and B. Kaneshiro, "I said it first: Topological analysis of lyrical influence networks." in *ISMIR*, 2016.

[11] N. J. Bryan and G. Wang, "Musical influence network analysis and rank of sample-based music." in *ISMIR*, 2011.

[12] N. Collins, "Influence in early electronic dance music: An audio content analysis investigation." in *ISMIR*, 2012.

[13] U. Shalit, D. Weinshall, and G. Chechik, "Modeling musical influence with topic models," in *ICML*, 2013.

[14] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," 2011.

[15] E. Noyes, I. E. Allen, and S. Parise, "Artistic influences and innovation in the popular music industry," *Frontiers of Entrepreneurship Research*, vol. 30, no. 15, 2010.

[16] J. Serrà, Á. Corral, M. Boguñá, M. Haro, and J. L. Arcos, "Measuring the evolution of contemporary western popular music," *Scientific reports*, vol. 2, 2012.

[17] F. Figueiredo and N. Andrade, "Quantifying disruptive influence in the allmusic guide," in *ISMIR*, 2019.

[18] N. Collins, "Computational analysis of musical influence: A musicological case study using MIR tools." in *ISMIR*, 2010.

[19] R. J. Funk and J. Owen-Smith, "A dynamic network measure of technological change," *Management Science*, vol. 63, no. 3, 2017.

[20] L. Wu, D. Wang, and J. A. Evans, "Large teams develop and small teams disrupt science and technology," *Nature*, vol. 566, no. 7744, 2019.

[21] F. em Vinil, "Forró em vinil - um pequeno apanhado da música nordestina em vinil!" (accessed August 2nd, 2020), "https://www.forroemvinil.com".

[22] K. Choi, G. Fazekas, M. B. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," *CoRR*, vol. abs/1703.09179, 2017. [Online]. Available: http://arxiv.org/abs/1703.09179

[23] G. Kour and N. Mehan, "Music genre classification using MFCC, SVM and BPNN," *International Journal of Computer Applications*, vol. 112, no. 6, 2015.

[24] D. Bogdanov, M. Haro, F. Fuhrmann, E. Gómez, and P. Herrera, "Content-based music recommendation based on user preference examples," in *WOMRAD Workshop*, 2010.

[25] B. NIU, L.-z. KONG, S.-l. LUO, L.-m. PAN, and L. GUO, "Individuality music recommendation model based on MFCC and gmm [j]," *Transactions of Beijing Institute of Technology*, vol. 4, 2009.

[26] J. H. Jensen, M. G. Christensen, D. P. Ellis, and S. H. Jensen, "Quantitative analysis of a common audio similarity measure," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 4, 2009.

[27] A. Flexer, D. Schnitzer, M. Gasser, and T. Pohle, "Combining features reduces hubness in audio similarity," *Children*, vol. 15, no. 15.95, 2010.

[28] F. De Leon and K. Martinez, "Enhancing timbre model using MFCC and its time derivatives for music similarity estimation," in *EUSIPCO.* IEEE, 2012.

[29] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *SciPy Conf.*, 2015.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, 2011.

[31] K.-M. Chung, W.-C. Kao, C.-L. Sun, L.-L. Wang, and C.-J. Lin, "Radius margin bounds for support vector machines with the rbf kernel," *Neural computation*, vol. 15, no. 11, pp. 2643–2681, 2003.

[32] J.-P. Vert, K. Tsuda, and B. Schölkopf, "A primer on kernel methods," *Kernel methods in computational biology*, vol. 47, 2004.

POP909: A POP-SONG DATASET FOR MUSIC ARRANGEMENT GENERATION

Ziyu Wang[1]   **Ke Chen**[2]   **Junyan Jiang**[1]   **Yiyi Zhang**[3]
Maoran Xu[4]   **Shuqi Dai**[5]   **Xianbin Gu**[1]   **Gus Xia**[1]
[1] Music X Lab, Computer Science Department, NYU Shanghai
[2] CREL, Music Department, UC San Diego
[3] Center for Data Science, New York University
[4] Department of Statistics, University of Florida
[5] Computer Science Department, Carnegie Mellon University

`{ziyu.wang, jj2731, yz2092, xianbin.gu, gxia}@nyu.edu,`
`knutchen@ucsd.edu, maoranxu@ufl.edu, shuqid@cs.cmu.edu`
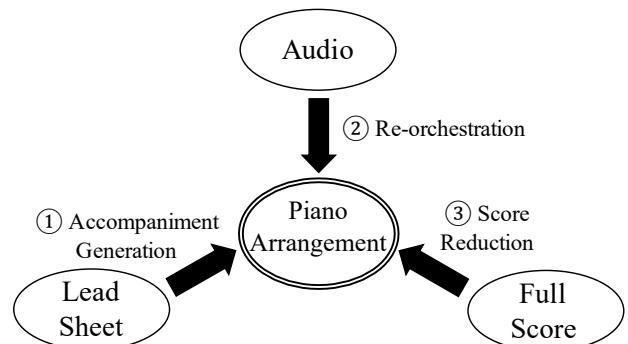
ABSTRACT

Music arrangement generation is a subtask of automatic music generation, which involves reconstructing and re-conceptualizing a piece with new compositional techniques. Such a generation process inevitably requires reference from the original melody, chord progression, or other structural information. Despite some promising models for arrangement, they lack more refined data to achieve better evaluations and more practical results. In this paper, we propose POP909, a dataset which contains multiple versions of the piano arrangements of 909 popular songs created by professional musicians. The main body of the dataset contains the vocal melody, the lead instrument melody, and the piano accompaniment for each song in MIDI format, which are aligned to the original audio files. Furthermore, we provide the annotations of tempo, beat, key, and chords, where the tempo curves are hand-labeled and others are done by MIR algorithms. Finally, we conduct several baseline experiments with this dataset using standard deep music generation algorithms.

1. INTRODUCTION

Music arrangement, the process of reconstructing and re-conceptualizing a piece, can refer to various *conditional* music generation tasks, which includes *accompaniment generation* conditioned on a lead sheet (the lead melody with a chord progression) [1–4], transcription and *re-orchestration* conditioned on the original audio [5–7], and *reduction* of a full score so that the piece can be performed by a single (or fewer) instrument(s) [8,9]. As shown in Figure 1, arrangement acts as a bridge, which connects lead sheet, audio and full score. In particular, *piano arrange-*

Figure 1: Illustration of the role of piano arrangement in the three forms of music composition, where ① and ② are covered by our POP909 dataset.

ment is one of the most favored form of music arrangement due to its rich musical expression. With the emergence of player pianos [10] and expressive performance techniques [11, 12], we expect the study of piano arrangement to be more meaningful in the future, towards the full automation of piano composition and performance.

In the computer music community, despite several promising generative models for arrangement, the lack of suitable datasets becomes one of the main bottlenecks of this research area (as pointed by [19, 20].) A desired arrangement dataset should have three features. First, the arrangement should be a style-consistent re-orchestration, instead of an arbitrary selection of tracks from the original orchestration. Second, the arrangement should be paired with an original form of music (audio, lead sheet, or full score) with precise time alignment, which serves as a supervision for the learning algorithms. Third, the dataset should provide external labels (e.g., chords, downbeat labels), which are commonly used to improve the controllability of the generation process [21]. Until now, we have not seen such a qualified dataset. Although most existing high-quality datasets (e.g., [13, 15]) contain at least one form of audio, lead melody or full score data, they have less focus on arrangement, lacking accurate alignment and

Dataset	Size	Paired Property				Annotation			Modality
		Polyphony	Lead Melody	Audio	Time-alignment	Beat	Key	Chord	
Lakh MIDI [13]	170k	✓		✓	Δ	✓	Δ	Δ	score, perf
JSB Chorales [14]	350+	✓			N/A	✓	✓		score
Maestro [15]	1k	✓		✓					perf
CrestMuse [16]	100	✓		✓	✓	✓	✓		score, perf
RWC-POP [17]	100	✓	✓	✓	Δ	✓	✓	✓	score, perf
Nottingham [18]	1k		N/A		N/A	✓	✓	✓	score
POP909	1k	✓	✓	✓	✓	✓	✓	✓	score, perf

Table 1: A summary of existing datasets.

labels.

To this end, we propose POP909 dataset.[1] It contains 909 popular songs, each with multiple versions of piano arrangements created by professional musicians. The arrangements are in MIDI format, aligned to the lead melody (also in MIDI format) and the original audios. Furthermore, each song are provided with manually labeled tempo curves and machine-extracted beat, key and chord labels using music information retrieval algorithms. We hope our dataset can help with future research in automated music arrangement, especially task ① and ② indicated in Figure 1:

Task 1: Piano accompaniment generation conditioned on paired melody and auxiliary annotation. This task involves learning the intrinsic relations between melody and accompaniment, including the selection of accompaniment figure, the creation of counterparts and secondary melody, etc.

Task 2: Re-orchestration from audio, i.e., the generation of piano accompaniment based on the audio of a full orchestra.

Besides those main tasks, our dataset can also be used for unconditional symbolic music generation, expressive performance rendering, etc.

2. RELATED WORK

In this section, we begin with a discussion of different modalities of music data in Section 2.1. We then review some existing composition-related datasets in Section 2.2 and summarize the requirements of a qualified arrangement dataset in Section 2.3. Again, our focus is piano arrangement and this dataset is designed for task ① and ② indicated in Figure 1, i.e., *piano accompaniment generation* based on the lead melody or the original audio.

2.1 Modalities of Music Generation

As discussed in [22], music data is intrinsically multi-modal and most generative models focus on one modality. In specific, music generation can refer to: 1) *score generation* [23–26], which deals with the very abstract symbolic representation, 2) *performance rendering* [4, 19, 20], which regards music as a sequence of controls and usually involves timing and dynamics nuances, and 3) *audio synthesis* [27, 28], which considers music as a waveform or

---
[1] The dataset is available at https://github.com/music-x-lab/POP909-Dataset

spectrogram. The POP909 dataset is targeted for arrangement generation in the modality of score and performance.

2.2 Existing Datasets

Table 1 summarizes the existing music datasets which are the potential resources for the piano arrangement generation tasks. The first column shows the dataset name, and the other columns show some important properties of each dataset.

Lakh MIDI [13] is one of the most popular datasets in symbolic format, containing 176,581 songs in MIDI format from a broad range of genres. Most songs have multiple tracks, most of which are aligned to the original audio. However, the dataset does not mark the lead melody track or the piano accompaniment track and therefore cannot be directly used for piano arrangement.

Maestro [15] and E-piano [29] contains classical piano performances in time-aligned MIDI and audio formats. However, the boundary between the melody and accompaniment is usually ambiguous for classical compositions. Consequently, the dataset is not suitable for the arrangement task ①. Moreover, the MIDI files are transcription rather than re-harmonization of the audio, which makes it inappropriate for the arrangement task ② either.

Nottingham Database [18] is a high-quality resource of British and Irish folk songs. The database contains MIDI files and ABC notations. One drawback of the dataset is that it only contains monophonic melody without polyphonic texture.

RWC-POP [17], CrestMuse [16], and JSB-Chorale [14] all contain polyphonic music pieces with rich annotations. However, the sizes of these three datasets are relatively small for training most deep generative models.

2.3 Requirements of Datasets for Piano Arrangement

We list the requirements of a music dataset suitable for the study of piano arrangement. The design objective of POP909 is to create a reliable, rich dataset that satisfies the following requirements.

- **A style-consistent piano track**: The piano track can either be an re-orchestration of the original audio or an accompaniment of the lead melody.

- **Lead melody or audio**: the necessary information for the arrangement task ① and ②, respectively.

- **Sufficient annotations** including key, beat, and chord labels. The annotations not only provide structured information for more controllable music generation, but also offer a flexible conversion between score and expressive performance.

- **Time alignment** among the piano accompaniment tracks, the lead melody or audio, and the annotations.

- **A considerable size**: while traditional machine learning models can be trained on a relatively small dataset, deep learning models usually require a larger sample size (expected 50 hours in total duration).

3. DATASET DESCRIPTION

POP909 consists of piano arrangements of 909 popular songs. The arrangements are time-aligned to the corresponding audios and maintain the original style and texture. Extra annotation includes beat, chord, and key information.

3.1 Data Collection Process

We hire professional musicians to create piano arrangements. In order to maintain a high-quality standard of the arrangements, we divide the musicians into two teams: the *arranger team* and the *reviewer team*. The collection is finalized through an iterative procedure between two teams. For each song, each iteration goes through three steps:

1. Arrangement: the arranger team creates an arrangement from scratch, or revise the previous version of arrangement.

2. Review: the reviewer team decides whether the current version is qualified and comments on how to improve the arrangement in case further revisions are required.

3. Discussion: musicians from both teams catch up with the progress, discuss and improve details of arrangement standards.
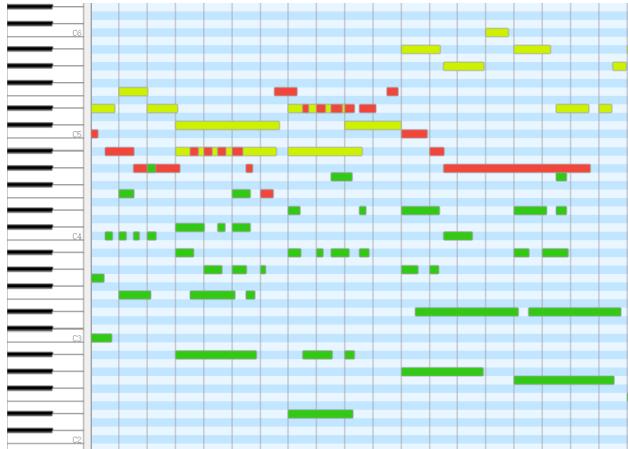
We start the process from a list of 1000 popular songs and finally select 909 songs with high arrangement quality. We not only present the last revision (i.e., the qualified version) of each song but also provide the unqualified versions of each song created during the iterative process. This multi-version feature may potentially offer a broader application scenario of the dataset.

3.2 Data Content and Format

In POP909, the total duration of 909 arrangements is about 60 hours. The songs are composed by 462 artists. The release of all songs spans around 60 years (from the earliest in 1950s to the latest around 2010).

Each piano arrangement is stored in MIDI format with three tracks. Figure 2 shows an example of a three-track MIDI file, in which different tracks are labeled with different colors. The three tracks are:

- MELODY: the lead (vocal) melody transcription.

Figure 2: An example of the MIDI file in a piano roll view. Different colors denote different tracks (red for MELODY, yellow for BRIDGE, and green for PIANO).

- BRIDGE: the arrangement of secondary melodies or lead instruments.

- PIANO: the arrangement of main body of the accompaniment, including broken chords, arpeggios, and many other textures.

Here, the combination of BRIDGE and PIANO track forms the piano accompaniment arrangement of the original song. Each MIDI file is aligned with the original audio by manually labeled tempo curve. Moreover, each note event contains expressive dynamics (i.e., detailed velocity control) based on the original audio.

Beat, chord, and key annotations are provided in five separate text files for each song. Annotations for beat and chord have both MIDI and audio versions while key changes annotations are merely extracted from audios.[2] The relevant music information retrieval algorithms are discussed in Section 4.
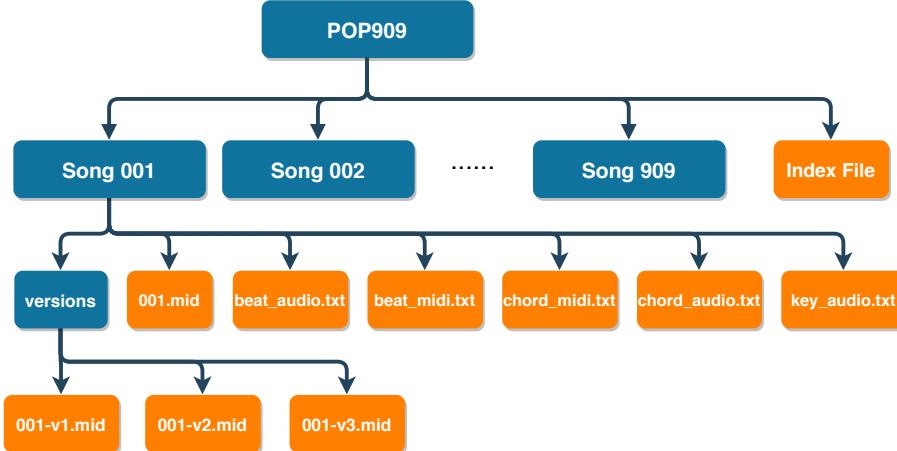
3.3 Data Folder Structure

Figure 3 demonstrates the folder structure of POP909. In the root directory, there are 909 folders, corresponding to 909 songs. In each folder, we provide the MIDI format arrangement, text format annotations, and a folder of all arrangement versions produced during the iterative processes.

The annotation files contain beat, chord and key annotations in plain text format. Table 2 shows the partial annotations of the song `003` in table format for better illustration purposes. For the beat annotation, `beat_audio` and `beat_midi` are the annotation files extracted from audio and MIDI, respectively. The source of chord and key annotations are indicated in a similar way.

Finally, we provide an index file in the root directory containing the song name, artist name, number of modified times and other useful metadata of the dataset.

---

[2] For annotations from MIDI files, the qualified (final) version of arrangements is used.

Figure 3: The folder structure of POP909. The blue boxes denote the folder and the orange boxes denote the file.

4. ANNOTATION METHODS

In this section, we discuss how we annotate the beat, chord and key information. For each of the three tasks, different algorithms are applied to extract information from MIDI or audio.

4.1 Beat & Downbeat Estimation

We first extract beat information from MIDI files by taking advantage of two features of the MIDI performance: (1) human-annotated tempo curves, and (2) the accompaniment figure of arrangements which shows a significant sign of beat and downbeat attacks.

Our method can be seen as a modification of the beat-tracking algorithms used in [30, 31]. First, we estimate the initial beat position and use the tempo curve to deduce subsequent beat positions. Second, we estimate the number of beats in a measure by calculating the auto-correlation of the extracted beat features (MIDI onset and velocity), assuming time signature is in general consistent within one song except for some infrequent phase changes. Finally, we search among all the possible phase shifts and find the optimal beat track that has the highest correlation with the extracted features.

We also provide the beat and downbeat annotations extracted from the audio using the algorithm introduced in [32] and compare them with the annotations extracted from MIDI.

For beat position estimation, the two algorithms have more than 90% consistency when the maximum error tolerance is 100 ms, which is acceptable in the data collection process. For downbeat estimation, the two algorithms have 80% agreement. We provide both extraction results in our annotation files.

4.2 Chord Label Extraction

We also provide the chord labels extracted from both MIDI and audio files. For the audio chord recognition, we adopt a large-vocabulary chord transcription algorithm by [33]. As chord changes in popular music are most likely to happen

at beat positions, we post-process the chord boundaries by aligning them to beats to produce the final chord labels.

file	beat time	downbeat_1	downbeat_2
	0.02	1.0	0.0
	0.75	0.0	0.0
	1.49	1.0	1.0
beat_midi	2.22	0.0	0.0
	2.95	1.0	0.0
	3.68	0.0	0.0
	...	...	...

file	beat time	beat order
	1.46	1.0
	2.18	2.0
beat_audio	2.92	3.0
	3.66	4.0
	...	...

file	start time	end time	chord
	0.02	0.75	N
	0.75	1.49	N
chord_midi	1.49	4.41	G:min7
	4.41	7.34	Eb:sus2
	...	...	...

file	start time	end time	chord
	0.00	1.46	N
	2.46	4.39	G:min7
chord_audio	4.39	7.31	Eb:maj(9)
	...	...	...

file	start time	end time	key
key_audio	1.46	226.00	Bb:maj

Table 2: The first several lines of the annotation files for song `003`. "downbeat_1" and "downbeat_2" in `beat_midi` are the two downbeat extractors under simple meter and compound meter assumptions, respectively.

For MIDI chord recognition, we adopt a method similar to the one proposed in [34]. We made two minor changes based on the original algorithm. First, the chord segmentation is performed on the beat level. Second, we alter the chord templates to include more chord qualities

used by pop songs: (1) triads (`maj`, `min`, `dim`, `aug`) with inversions, (2) basic sevenths (`maj7`, `min7`, `7`, `dim7`, `hdim7`) with inversions, (3) suspended chords (`sus2`, `sus4`, `sus4(b7)`), and (4) sixth chords (`maj6`, `min6`).

Note that the arrangement and its original audio may have different chord progressions. For example, a `C:maj` chord may be arranged into `C:sus2`, if necessary. Therefore, both annotations are reasonable and they are not necessarily consistent with each other. To compare the extraction accuracy, we compute the matching rate of the root notes of the chords extracted from both methods. Results show that the matching degree of more than 800 songs in POP909 are above 75%. On the other hand, there are still a few songs whose matching degrees are below 40%. The main reasons are: (1) some of these audio recordings are slightly out of tune, and (2) some parts of the audio have complicated sound effects, in which case our teams decide to re-arrange the chord progression.

4.3 Key Signature Extraction

We also provide key signature annotation from the audio files. We adopt an algorithm very similar to [35]. The original algorithm performs the key classification for a whole song based on the averaged frame-wise feature. In our modified algorithm, we also allow key changes in the middle of the song using a median filter to post-process the frame-level labels.

5. EXPERIMENTS

In this section, we conduct two baseline experiments on music (score-modality) generation with the POP909 dataset: 1) polyphonic music generation (without melody condition), and 2) piano arrangement generation conditioned on melody. For both tasks, we use the Transformer architecture [36] for its advantages in capturing long-term dependencies on time-series data.

5.1 Polyphonic Music Generation

We use a transformer encoder with relative positional encoding [19, 37] to model the distribution of polyphonic music. We adopt a MIDI-like event-based representation slightly modified from [19, 38] to encode the polyphonic music. Each piece of music is represented as a series of events, including note onsets, offsets, velocity changes, and time shifts. We further quantize time shifts tokens under the resolution of $\frac{1}{4}$ beat. In total, we use 16 time-shift events, ranging from $\frac{1}{4}$ beat to 4 beats. Longer notes or rests can be represented by multiple time-shift tokens in a sequence. Table 3 shows the details of our data representation.

We split the dataset into 3 subsets: 90% for training, 5% for validation, and 5% for testing. We set the maximum sequence length $L = 2048$, transformer hidden size $H = 512$, the number of attention heads $h = 6$, and the number of attention layers $N = 6$. Cross Entropy loss is used as the loss function and early stopping is applied.

Event type	Tokenization
Note-On	0-127 (MELODY & BRIDGE track) 256-383 (PIANO track)
Note-Off	128-255 (MELODY & BRIDGE) 384-511 (PIANO track)
Time-Shift	512-527
Velocity	528-560

Table 3: The tokenization of the modified MIDI-like event sequence representation.

GPT-2-based transformer in POP909			
Train Loss	Train Acc.	Test Loss	Test Acc.
2.08978	0.62021	2.38122	0.54529

Table 4: The report of training and test loss and prediction accuracy of MIDI event tokens.

We use Adam optimizer [39] with hyperparameters $\beta_1 = 0.9, \beta_2 = 0.998$. We further adopt the warm-up schedule to control the learning rate. Formally, at the i-th warm-up step, the learning rate

$$lr = \frac{1}{\sqrt{H}} \times \min(\frac{1}{\sqrt{i}}, \frac{i}{S\sqrt{S}}), \tag{1}$$

where $S = 4000$ is a hyperparameter controlling the number of warm-up steps. The training result is presented in Table 4.
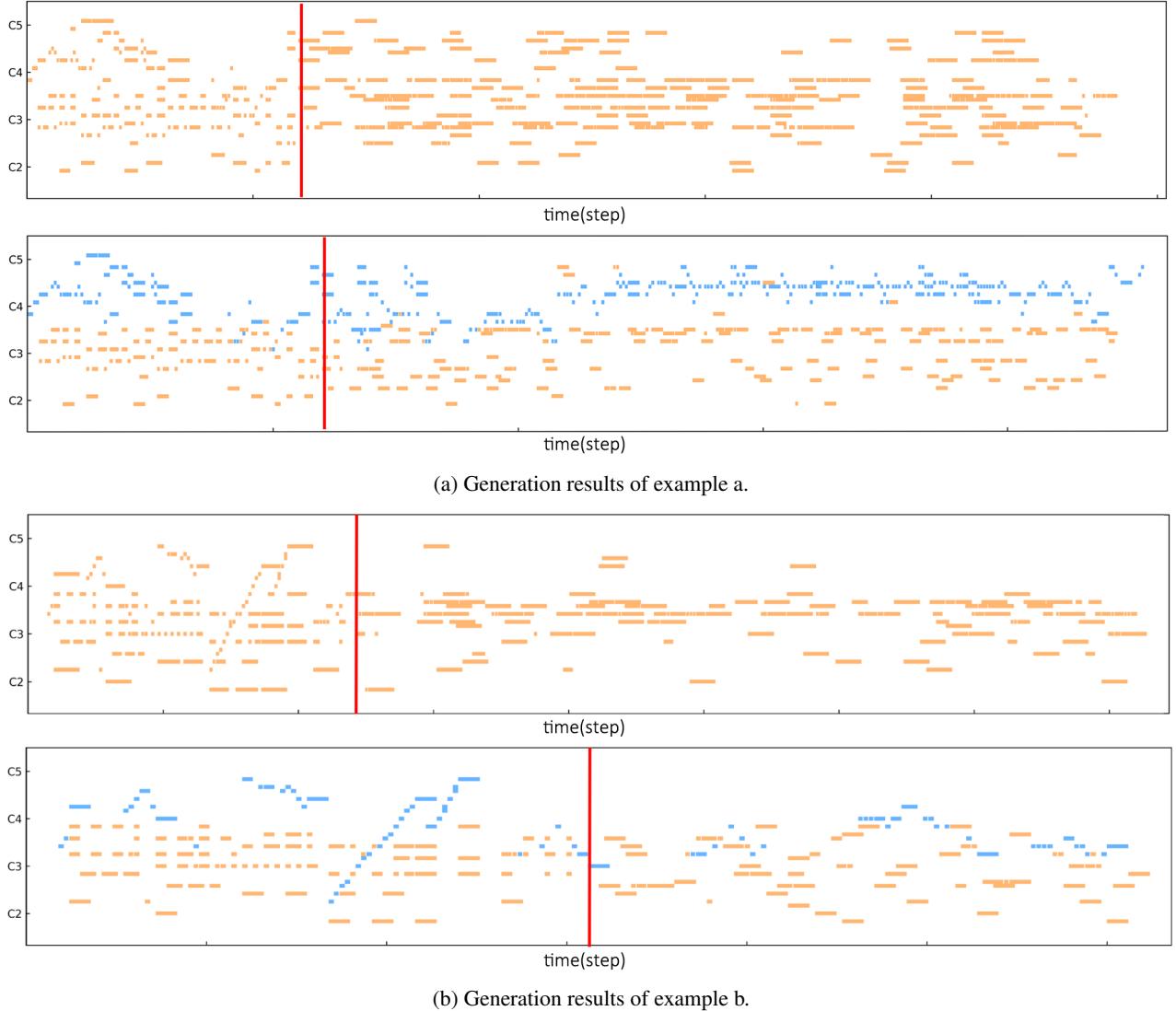
5.2 Piano Arrangement Generation

In the second experiment, we design an automatic piano arrangement task: piano accompaniment generation conditioned on the melody. In the data processing step, we first merge the MELODY track and the BRIDGE track into the *main melody* and regard PIANO track the *piano accompaniment*.

We use the same (trained) model in Section 5.1 to model the joint distribution of the main melody and piano accompaniment. During the inference, we force the generated melody to match the given melody condition, generating the most likely accompaniment conditioned on the melody. (A similar conditional generation method has been used in [20].)

5.3 Experiment Results

Figure 4 shows several examples generated by the trained model. In each subfigure, the top piano roll shows the polyphonic music generation (introduced in Section 5.1) result and the bottom piano roll shows the piano arrangement generation (introduced in Section 5.2) result conditioned on the main melody (the blue track). In both cases, the first 500 MIDI-event tokens are given as the context; the red line separates the given context and the generated outputs. We see that the generated pieces capture basic harmonic relationships between the melody and accompaniment and contain consistent rhythmic patterns. Although the quality is still far from the music generated by state-of-the-art

(a) Generation results of example a.

(b) Generation results of example b.

Figure 4: Generation examples with POP909 dataset. Unconditioned polyphonic music generation and piano arrangement generation (blue for the melody, orange for the accompaniment) of the two selected examples are displayed.

algorithms [19, 40], they serve as a baseline to illustrate our dataset usage. We believe that the model can produce better and more structured results with the development of deep generative models.

6. CONCLUSION

In conclusion, we contributed POP909, a tailored dataset for music arrangement. It contains multiple versions of professional piano arrangements in MIDI format of 909 popular songs, together with precise tempo curve aligned to the original audio recordings. We also provide annotations of tempo, beat, downbeat, key, and chord labels. To guarantee a high data quality, the dataset was collected via the collaboration of two groups of professional musicians, arrangers and reviewers, in an interactive process. Apart from the arrangement problem, the POP909 dataset serves as a high-quality resource for structural music generation and cross-modal music generation.

7. ACKNOWLEDGEMENT

8. REFERENCES

[1] I. Simon, D. Morris, and S. Basu, "Mysong: automatic accompaniment generation for vocal melodies," in *Proceedings of the 2008 Conference on Human Factors in Computing Systems (CHI)*. Florence, Italy: ACM, 2008, pp. 725–734.

[2] A. Elowsson and A. Friberg, "Algorithmic composition of popular music," in *the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music*, 2012, pp. 276–285.

[3] Z. Wang and G. Xia, "A framework for automated pop-song melody generation with piano accompaniment arrangement," *arXiv preprint arXiv:1812.10906*, 2018.

[4] H. Dong, W. Hsiao, L. Yang, and Y. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 34–41.

[5] H. Takamori, T. Nakatsuka, S. Fukayama, M. Goto, and S. Morishima, "Audio-based automatic generation of a piano reduction score by considering the musical structure," in *International Conference on Multimedia Modeling (ICMM)*. Springer, 2019, pp. 169–181.

[6] G. Percival, S. Fukayama, and M. Goto, "Song2quartet: A system for generating string quartet cover songs from polyphonic audio of popular music." in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 114–120.

[7] Y.-N. Hung, I. Chiang, Y.-A. Chen, Y.-H. Yang *et al.*, "Musical composition style transfer via disentangled timbre representations," *arXiv preprint arXiv:1905.13567*, 2019.

[8] E. Nakamura and S. Sagayama, "Automatic piano reduction from ensemble scores based on merged-output hidden markov model," in *Proceedings of the 41st International Computer Music Conference (ICMC)*, 2015.

[9] J.-L. Huang, S.-C. Chiu, and M.-K. Shan, "Towards an automatic music arrangement framework using score reduction," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 8, no. 1, pp. 1–23, 2012.

[10] M. Xu, Z. Wang, and G. Xia, "Transferring piano performance control across environments," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, United Kingdom: IEEE, 2019, pp. 221–225.

[11] G. Xia, "Expressive collaborative music performance via machine learning," Ph.D. dissertation, Carnegie Mellon University.

[12] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Graph neural network for music score data and modeling expressive piano performance," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 3060–3070.

[13] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," *PhD thesis, Columbia University*, 2016.

[14] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*. icml.cc / Omnipress, 2012.

[15] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019.

[16] M. Hashida, T. Matsui, and H. Katayose, "A new music database describing deviation information of performance expressions," in *Proceedings of 9th International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, PA, USA, 2008, pp. 489–494.

[17] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical and jazz music databases," in *Proceedings of 3rd International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 2002.

[18] E. Foxley, "Nottingham database," https://ifdo.ca/~seymour/nottingham/nottingham.html, 2011.

[19] C. A. H. et al., "Music transformer: Generating music with long-term structure," in *7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019.

[20] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 685–692.

[21] K. Chen, W. Zhang, S. Dubnov, G. Xia, and W. Li, "The effect of explicit structure encoding of deep neural networks for symbolic music generation," in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*. IEEE, 2019, pp. 77–84.

[22] S. Dai, Z. Zhang, and G. G. Xia, "Music style transfer: A position paper," *Proceeding of International Workshop on Musical Metacreation (MUME)*, 2018.

[23] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *Proceedings of the 34th International Conference on Machine Learning, (ICML)*. Sydney, NSW, Australia: PMLR, 2017, pp. 1362–1371.

[24] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 343–351.

[25] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *Proceedings of 1st Conference on Computer Simulation of Musical Creativity (CSMC)*, 2016.

[26] K. Chen, G. Xia, and S. Dubnov, "Continuous melody generation via disentangled short-term representations and structural conditions," in *14th International Conference on Semantic Computing (ICSC)*. San Diego, CA, USA: IEEE, 2020, pp. 128–135.

[27] A. van den Oord et al., "Wavenet: A generative model for raw audio," in *The 9t Speech Synthesis Workshop*. Sunnyvale, CA, USA: ISCA, 2016, p. 125.

[28] P. D. et al., "Jukebox: A generative model for music," *CoRR*, vol. abs/2005.00341, 2020.

[29] "International piano-e-competition," http://www.piano-e-competition.com/.

[30] C. Raffel and D. P. Ellis, "Intuitive analysis, creation and manipulation of midi data with pretty midi," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR), Late Breaking and Demo Papers*, Taipei, Taiwan, 2014, pp. 84–93.

[31] H. Grohganz, M. Clausen, and M. Müller, "Estimating musical time information from performed MIDI files," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014, pp. 35–40.

[32] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, United States, 2016, pp. 255–261.

[33] J. Jiang, K. Chen, W. Li, and G. Xia, "Large-vocabulary chord transcription via chord structure decomposition," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 644–651.

[34] B. Pardo and W. P. Birmingham, "Algorithms for chordal analysis," *Computer Music Journal*, vol. 26, no. 2, pp. 27–49, 2002.

[35] F. Korzeniowski and G. Widmer, "End-to-end musical key estimation using a convolutional neural network," in *25th European Signal Processing Conference (EUSIPCO)*. Kos, Greece: IEEE, 2017, pp. 966–970.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[37] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT (NAACL)(Short Papers)*. New Orleans, Louisiana, USA: Association for Computational Linguistics, 2018, pp. 464–468.

[38] I. Simon and S. Oore, "Performance rnn: Generating music with expressive timing and dynamics," https://magenta.tensorflow.org/performance-rnn, 2017.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceeding of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[40] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *Proceedings of 21st International Conference on Music Information Retrieval (ISMIR), virtual conference*, 2020.

CONNECTIVE FUSION:
LEARNING TRANSFORMATIONAL JOINING OF SEQUENCES
WITH APPLICATION TO MELODY CREATION

Taketo Akama

Sony Computer Science Laboratories, Tokyo, Japan

`taketo.akama@sony.com`
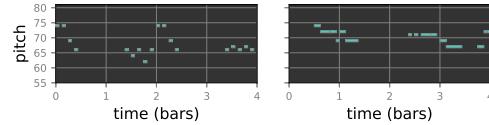
ABSTRACT

We present *Connective Fusion*, a music generation scheme by transformational joining of two musical sequences for creative purposes. Given two shorter sequences as inputs, our model transforms each of them such that their concatenation is more coherent to form a longer sequence, while each of the transformed shorter sequences retains meaningful similarity with the corresponding input sequence. In short, our model connects and fuses two contextually unrelated sequences in a coherent way. This transformation can be applied iteratively to gradually fuse the input sequences. The *style latent space* is simultaneously learned, allowing users to control how the two sequences are merged. Our approach comprises two steps of unsupervised learning: a deep generative model with a latent space is learned, followed by adversarial learning of the transformation function in the latent space. We demonstrate the usefulness of our method through the task of melody creation using a symbolic music dataset.
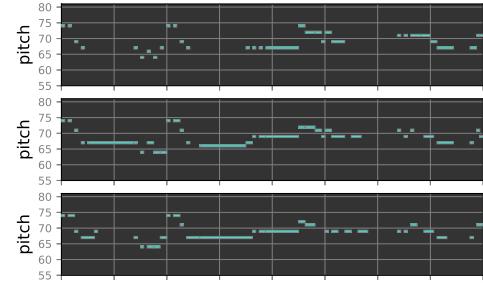
1. INTRODUCTION

Spurred by the progress of deep neural networks, symbolic music generation systems, especially the ones with user controllability have gathered renewed interests these days. Most of the systems with controllability can be categorized into generating continuation [1], regenerating arbitrary positions [2, 3], unsupervised conditional generation [4, 7], or transforming musical sequences such that musical attributes, concepts, or styles are altered [4–8].

In this paper, we seek another category, generation by fusing input musical sequences as ideas. Specifically, we propose *Connective Fusion*, a generative transformation scheme which allows two input musical sequences to be transformed such that the concatenated musical sequence is musically plausible (See Fig.1 for the illustration). After the transformation, each of the two sequences has meaningful similarity with the one before the transformation. Two input musical sequences can also be transformed it-
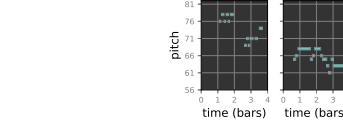
(a) Given inputs: random combination of two sequences
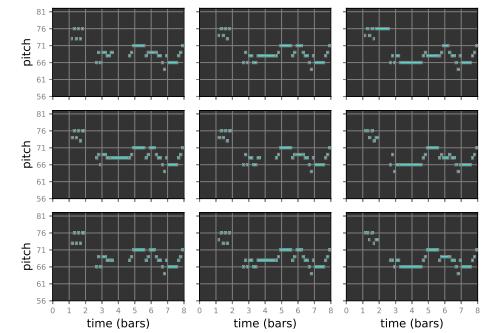
(b) Transformed outputs of iteration 1, 2, and 3

Figure 1: Iterated application of transformation.

(a) Given inputs: random combination of two sequences
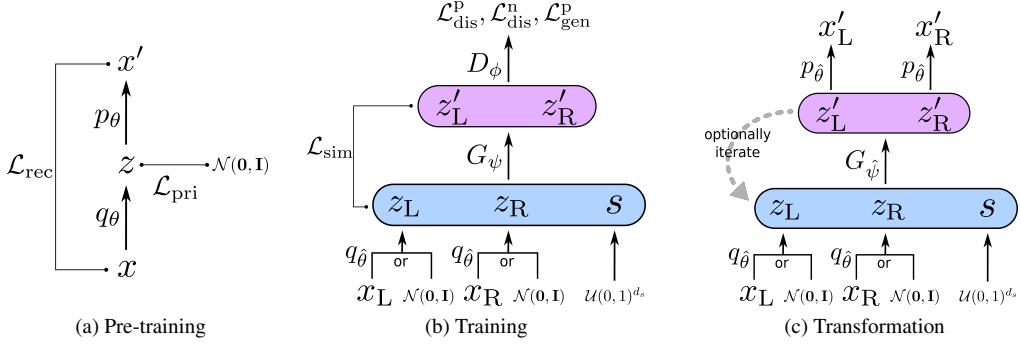
(b) Transformed outputs

Figure 2: Transformation with *style space* exploration.

eratively to gradually increase the coherency of joining (Fig.1). Our generative transformation differs from pure transformation in that it permits users to explore or sample from the *style space* of how the two sequences are combined (Fig.2). The application of Connective Fusion includes i) providing users with musical ideas based on not

Figure 3: Connective Fusion schematic diagrams.

necessarily polished ideas on hand, and ii) creating novel musical flows by combining and fusing musical fragments of different characteristics in a coherent way.

Technically, our approach is training a transformation function in an unsupervised learning manner, consisting of two steps: the pre-training step and the training step. In the pre-training step, a Variational Auto-Encoder (VAE) [9] is trained to obtain mappings between the *data space* and the *latent space*. A given pair of musical sequences are fed into the encoder of the VAE to be mapped to the corresponding latent vectors in the latent space. In the following training step, models are trained upon the representation of the VAE latent space. The transformation function, the *generator* is trained adversarially [10] such that transformed results are indistinguishable from the human-made musical sequences. Together with the adversarial loss, the similarity loss, consisting of the latent space distance before and after the transformation, is simultaneously taken into account in order to train models with desired amount of transformation.

In experiments, we empirically demonstrate that our method outperforms a baseline method with various parameter settings in terms of reality and five musical statistics at each transformation amount. We also quantitatively show that iterated application of our transformation allows gradual transformation while having comparable to or better performance than single (non-iterative) transformation.

Our contributions of this paper are: i) proposing a new problem setting/task and its solution, ii) presenting a model and procedure for learning style space, iii) introducing iterated application of transformation for gradual transformation, and iv) demonstrating the performance and the application for melody creation.

2. METHODOLOGY

2.1 Problem Scenario

Suppose we have a dataset of sequences $\mathcal{D}_y = \{y^{(i)} = x_{\mathrm{L}}^{(i)} \oplus x_{\mathrm{R}}^{(i)}\}_{i=1}^{N}$, where $y^{(i)} \in \mathcal{Y}$ is the concatenation of two subsequences $x_{\mathrm{L}}^{(i)} \in \mathcal{X}$ and $x_{\mathrm{R}}^{(i)} \in \mathcal{X}$. We can consider $\mathcal{D}_y$ as a sequence pair dataset $\mathcal{D} = \{(x_{\mathrm{L}}^{(i)}, x_{\mathrm{R}}^{(i)})\}_{i=1}^{N}$. For instance, $y^{(i)}$ is a musical sequence of 8 bars, whereas $x_{\mathrm{L}}^{(i)}$

and $x_{\mathrm{R}}^{(i)}$ are musical sequences of 4 bars.

Using the dataset $\mathcal{D}$, the scheme of Connective Fusion basically solves the following task: given two sequences $x_{\mathrm{L}}, x_{\mathrm{R}} \in \mathcal{X}$, transforming x_{L} to x_{L}' and x_{R} to x_{R}' such that the concatenated sequence $x_{\mathrm{L}}' \oplus x_{\mathrm{R}}'$ becomes more probable to be a sample in $\mathcal{Y}$ than $x_{\mathrm{L}} \oplus x_{\mathrm{R}}$, while the transformed sequences $x_{\mathrm{L}}', x_{\mathrm{R}}'$ retain meaningful similarity to the given sequences $x_{\mathrm{L}}, x_{\mathrm{R}}$, respectively. For example, this serves as a solution for generating longer musical sequences given shorter musical sequence pairs of any combinations as inspiration, which is useful for composing new music based on unpolished musical ideas on hand.

2.2 Approach

Schematic diagrams of pre-training, training, and transformation of Connective Fusion are depicted in Fig.3.

2.2.1 Pre-training

Our approach is first training a Variational Auto-Encoder (VAE) model for obtaining bidirectional mappings between each sequence $x \in \mathcal{X}$ and its latent vector $z \in \mathcal{Z} \subset \mathbb{R}^{d_z}$, which has the compressed information of x.

2.2.2 Training

Then we train a generator G that transforms any pair $(z_{\mathrm{L}}, z_{\mathrm{R}}) \in \mathcal{Z} \times \mathcal{Z}$ to $(z_{\mathrm{L}}', z_{\mathrm{R}}') \in \mathcal{Z} \times \mathcal{Z}$ that are indistinguishable from the pairs in the dataset. The generator also takes a style vector $s \in \mathcal{S}$ from the style space $\mathcal{S} = [0, 1]^{d_s}$ as an additional input. Formally, the generator can be written as $(z_{\mathrm{L}}', z_{\mathrm{R}}') = G(z_{\mathrm{L}}, z_{\mathrm{R}}, s)$ and $G \colon \mathcal{Z} \times \mathcal{Z} \times \mathcal{S} \to \mathcal{Z} \times \mathcal{Z}$. The generator G is trained together with the discriminator $D \colon \mathcal{Z} \times \mathcal{Z} \to [0, 1]$ with the adversarial learning framework [10]. In addition to the adversarial loss, we add the similarity loss in order to adjust the degree of similarity before and after the transformation.

2.2.3 Transformation

Given two sequences x_{L} and x_{R}, these sequences are first fed to the encoder of the VAE to obtain z_{L} and z_{R}, respectively. The latent vectors z_{L} and z_{R} together with a style vector $s \in \mathcal{S}$ sampled from $\mathcal{S} = [0, 1]^{d_s}$ are then fed to the

generator G to transform z_L, z_R to z'_L, z'_R, respectively. Finally, the transformed latent vectors z'_L, z'_R are fed to the decoder to generate x'_L, x'_R, respectively. The concatenation $x'_L \oplus x'_R$ is the generated sequence. Instead, z_L and z_R can also be sampled from the prior distribution $p(z)$, providing users with scratch generation functionality followed by the transformation functionality.

The style space $\mathcal{S}$ allows us to control *how the two sequences are connectively fused*. For example, as illustrated in Fig.2 and explained in the experiments Sec.3.3, users can choose preferred styles of fusion among interpolated styles on a 2D plane.

We also propose iterated transformation with G, which consists in applying G repeatedly, drawing two interrelated trajectories in the latent space. The higher the number of iterations becomes, the farther the latent vectors tend to move from the original position. This is useful for users to control the degree of similarity in transformation.

2.3 Advantage of Approach Using Latent Space

Using the representation in $\mathcal{Z}$ but not the one in $\mathcal{X}$ i) is useful for defining similarity before and after the transformation, ii) is computationally inexpensive, and iii) bypasses the difficult problem of back-propagating through discrete sampling of sequences.

2.4 Pre-training Detail: Encoder and Decoder Training with Auto-Encoding VB algorithm

To train the VAE, another dataset $\mathcal{D}' = \{x^{(j)}\}_{j=1}^{2N}$ is derived from the dataset $\mathcal{D} = \{(x_L^{(i)}, x_R^{(i)})\}_{i=1}^{N}$, where $x^{(2i-1)} = x_L^{(i)}$ and $x^{(2i)} = x_R^{(i)}$. The encoder model $q_\theta(x|z)$ and the decoder model $p_\theta(z|x)$ in the VAE are trained with the following optimization problem:

$$\max_\theta \mathbb{E}_{x \sim \mathcal{D}'} \left[\mathbb{E}_{z \sim q_\theta(z|x)} \left[\log p_\theta(x|z) \right] - KL\left(q_\theta(z|x) || p(z) \right) \right],$$
(1)

which is the maximization of the variational lower bound [9], the lower bound of the marginal log likelihood. Here, KL denotes the Kullback-Leibler (KL) divergence. Note that the sampling operation $z \sim q_\theta(z|x)$ is differentiable using the reparametrization trick. For the purpose of visualization in Fig.3, we name the two terms in the optimization problem $\mathcal{L}_{rec} = -\mathbb{E}_{x \sim \mathcal{D}'} \left[\mathbb{E}_{z \sim q_\theta(z|x)} \left[\log p_\theta(x|z) \right] \right]$ and $\mathcal{L}_{pri} = \mathbb{E}_{x \sim \mathcal{D}'} \left[KL\left(q_\theta(z|x) || p(z) \right) \right]$. For the rest of this paper, $\hat{\theta}$ denotes the estimated model parameter after the optimization of Eq.1. We use normal distribution for the encoder model $q_\theta(z|x)$.

2.5 Training Detail: Generator and Discriminator Training with Adversarial Learning

For notational simplicity, we introduce datasets of latent vectors $\mathcal{D}_z = \{(z_L^{(i)}, z_R^{(i)})\}_{i=1}^{N}$ and $\mathcal{D}'_z = \{z^{(j)}\}_{j=1}^{2N}$ corresponding to $\mathcal{D} = \{(x_L^{(i)}, x_R^{(i)})\}_{i=1}^{N}$ and $\mathcal{D}' = \{x^{(j)}\}_{j=1}^{2N}$ respectively, where $z_L^{(i)} = \text{argmax}_z q_{\hat\theta}(z|x_L^{(i)})$, $z_R^{(i)} =$

$\text{argmax}_z q_{\hat\theta}(z|x_R^{(i)})$, and $z^{(j)} = \text{argmax}_z q_{\hat\theta}(z|x^{(j)})$. The generator G and the discriminator D are parametrized by ψ and ϕ, respectively. In the following, we use short hand $\mathcal{L}_{D_\phi}^p(z_L, z_R) = -\log D_\phi(z_L, z_R)$, $\mathcal{L}_{D_\phi}^n(z_L, z_R) = -(1 - \log D_\phi(z_L, z_R))$, and $\mathcal{U} = \mathcal{U}(0,1)^{d_s}$.

2.5.1 Discriminator Loss

In the adversarial learning framework, the discriminator classifies two sets of samples: the *real class* and the *fake class*. In Connective Fusion, samples in the real class are latent vector pairs (z_L, z_R) sampled from the dataset $\mathcal{D}_z$. Formally, the loss function for the real class of the discriminator is

$$\mathcal{L}_{dis}^p = \mathbb{E}_{(z_L, z_R) \sim \mathcal{D}_z} \left[\mathcal{L}_{D_\phi}^p(z_L, z_R) \right].$$
(2)

The fake class for the discriminator are latent vector pairs (z_L, z_R) which are obtained by i) sampling independently with the uniform distribution over the dataset $\mathcal{D}'_z$ (1st term of Eq.3), ii) sampling independently from the prior $p(z)$ (2nd term of Eq.3), iii) the generator G transforming samples (z_L, z_R, s), where z_L, z_R are sampled in the same way as i, while s are sampled from $\mathcal{U}$, the uniform distribution over $\mathcal{S} = [0,1]^{d_s}$ (3rd term of Eq.3), and iv) the generator G transforming samples (z_L, z_R, s), where z_L, z_R are sampled in the same way as ii, while s are sampled from $\mathcal{U}$ (4th term of Eq.3). Formally, the loss function is

$$\begin{aligned}
\mathcal{L}_{dis}^n = {} & \mathbb{E}_{z_L \sim \mathcal{D}'_z} \mathbb{E}_{z_R \sim \mathcal{D}'_z} \left[\mathcal{L}_{D_\phi}^n(z_L, z_R) \right] \\
& + \mathbb{E}_{z_L \sim p(z)} \mathbb{E}_{z_R \sim p(z)} \left[\mathcal{L}_{D_\phi}^n(z_L, z_R) \right] \\
& + \mathbb{E}_{z_L \sim \mathcal{D}'_z} \mathbb{E}_{z_R \sim \mathcal{D}'_z} \mathbb{E}_{s \sim \mathcal{U}} \left[\mathcal{L}_{D_\phi}^n(G_\psi(z_L, z_R, s)) \right] \\
& + \mathbb{E}_{z_L \sim p(z)} \mathbb{E}_{z_R \sim p(z)} \mathbb{E}_{s \sim \mathcal{U}} \left[\mathcal{L}_{D_\phi}^n(G_\psi(z_L, z_R, s)) \right].
\end{aligned}$$
(3)

Finally, the overall loss function for the discriminator becomes

$$\mathcal{L}_{dis} = \mathcal{L}_{dis}^p + \mathcal{L}_{dis}^n.$$
(4)
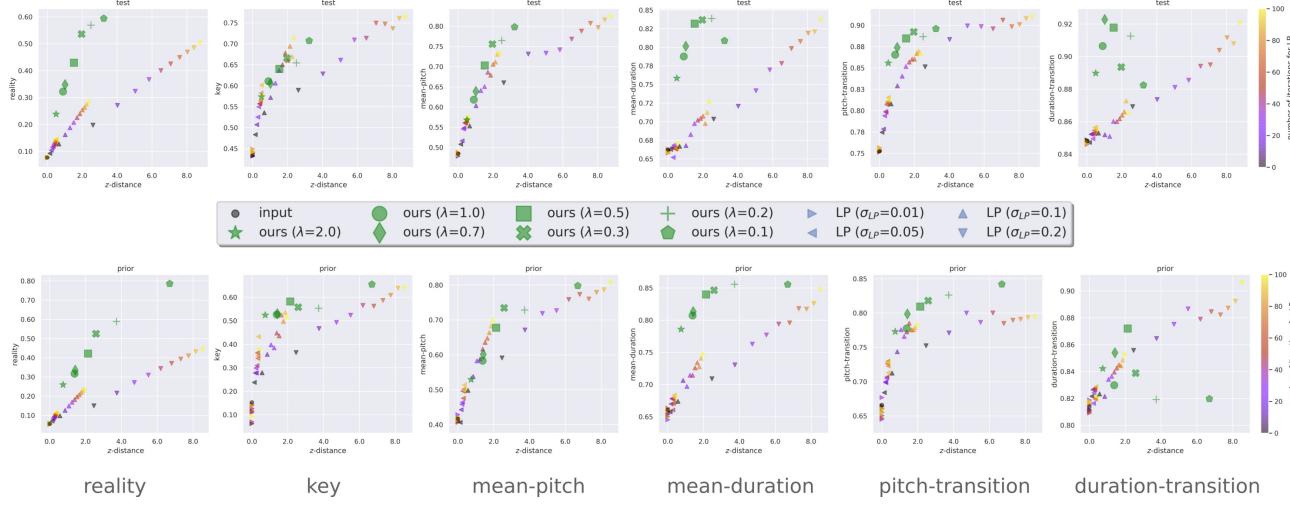
2.5.2 Generator Loss

Based on the discriminator, a generator is trained such that the generated samples become more like the real samples rather than the fake samples. We consider two kinds of generated samples which are sampled in the same way as iii and iv in the previous section 2.5.1. Formally, the loss function is
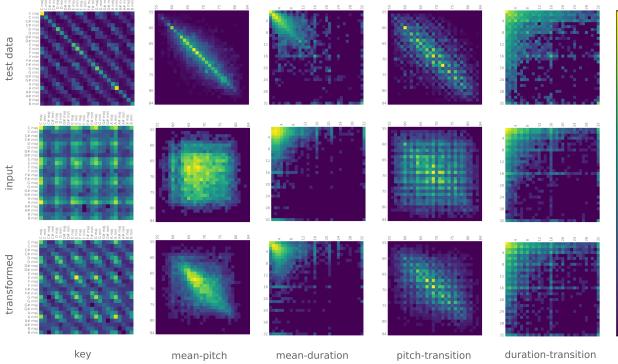
$$\begin{aligned}
\mathcal{L}_{gen}^p = {} & \mathbb{E}_{z_L \sim \mathcal{D}'_z} \mathbb{E}_{z_R \sim \mathcal{D}'_z} \mathbb{E}_{s \sim \mathcal{U}} \left[\mathcal{L}_{D_\phi}^p(G_\psi(z_L, z_R, s)) \right] \\
& + \mathbb{E}_{z_L \sim p(z)} \mathbb{E}_{z_R \sim p(z)} \mathbb{E}_{s \sim \mathcal{U}} \left[\mathcal{L}_{D_\phi}^p(G_\psi(z_L, z_R, s)) \right].
\end{aligned}$$
(5)

Additionally, we introduce the similarity loss which is the latent space distance between samples before and after the transformation:
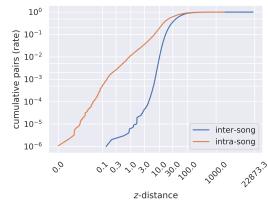
$$\begin{aligned}
\mathcal{L}_{sim} = {} & \mathbb{E}_{z_L \sim \mathcal{D}'_z} \mathbb{E}_{z_R \sim \mathcal{D}'_z} \mathbb{E}_{s \sim \mathcal{U}} \left[\mathcal{L}_{dist}(z_L, z_R, s) \right] \\
& + \mathbb{E}_{z_L \sim p(z)} \mathbb{E}_{z_R \sim p(z)} \mathbb{E}_{s \sim \mathcal{U}} \left[\mathcal{L}_{dist}(z_L, z_R, s) \right],
\end{aligned}$$
(6)

Figure 4: Evaluation of our model on six metrics vs z-distance. The upper/lower rows are transformation results where inputs are randomly created pairs of test/generated (sampled from prior) data. The color gradient corresponds to the number of iterations for LP. For all metrics, higher values are better. See Sec.3.5.

Figure 5: Musical statistics are corrected after our transformation. Brighter color indicates higher probability. See the second paragraph of 3.4.

Figure 6: Analysis of z-distance in the latent space $\mathcal{Z}$. See the last paragraph of 3.4.

where

$$\mathcal{L}_{\text{dist}}(z_{\text{L}}, z_{\text{R}}, s) = \frac{1}{d_z} \left\| \frac{1}{\bar{\sigma}_z^2} \log\left(1 + (z'_{\text{L}} - z_{\text{L}})^2\right) \right\|_1$$
$$+ \frac{1}{d_z} \left\| \frac{1}{\bar{\sigma}_z^2} \log\left(1 + (z'_{\text{R}} - z_{\text{R}})^2\right) \right\|_1$$
$$\text{with } (z'_{\text{L}}, z'_{\text{R}}) = G_\psi(z_{\text{L}}, z_{\text{R}}, s). \tag{7}$$

Following the latent constraint paper [6], $\bar{\sigma}_z \in \mathbb{R}^{d_z}$ is chosen to be the standard deviation $\sigma(x) \in \mathbb{R}^{d_z}$ of the encoder model $q_{\hat{\theta}}(z|x) = \mathcal{N}\left(\mu(x), \text{diag}\left(\sigma^2(x)\right)\right)$ av-

eraged over all the training dataset. Precisely, $\bar{\sigma}_z = 1/|\mathcal{D}'| \sum_{x \in \mathcal{D}'} \sigma(x)$. Finally, the overall loss function for the generator becomes
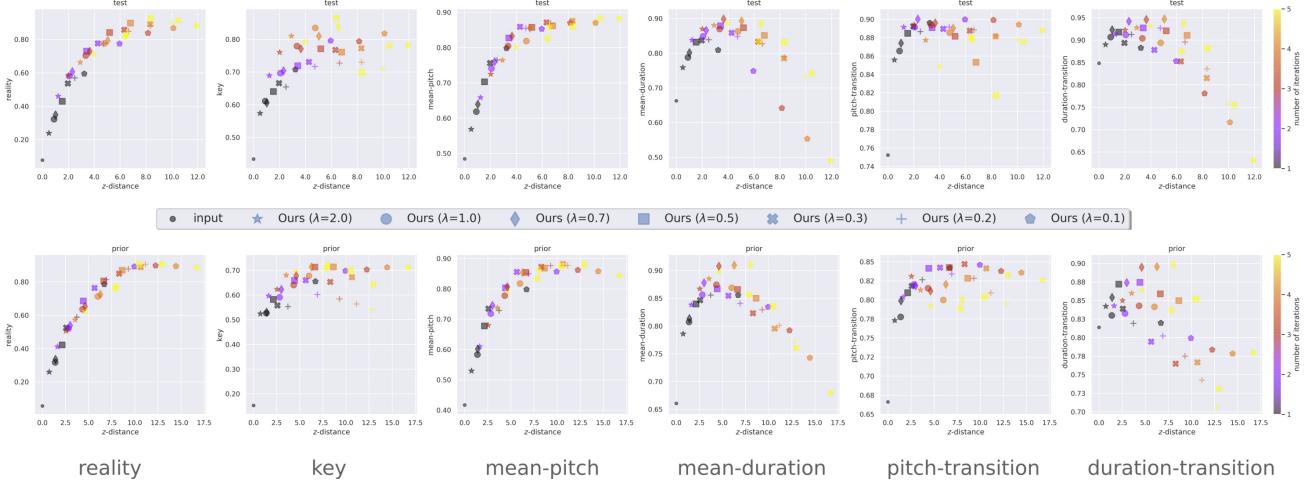
$$\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{gen}}^{\text{p}} + \lambda \mathcal{L}_{\text{sim}}. \tag{8}$$

3. EXPERIMENTS

3.1 Dataset

We create datasets from LMD-matched of Lakh MIDI dataset [11], comprising 45,129 files matched to the song identity entries in the Million Song Dataset [12]. Each song has one or several different versions of MIDI files. We first extract files with 4/4 time signature, use accompanying tempo information to determine beat locations, and quantize each beat into 4. We then split the song identities into the proportion of 11:1:6:1:1 to create train-1, validation-1, train-2, validation-2, and test dataset, respectively. Train-1 and validation-1 datasets are for training proposed and baseline models, whereas train-2 and validation-2 datasets are for training evaluation models. We filter out non-monophonic tracks, bass or drum tracks, and the tracks outside the pitch range of [55, 84]. We conduct data augmentation by transposing tracks to all possible keys if the transposed tracks stay within the pitch range of [55, 84]. We retrieve 8-bar sliding windows (with a stride of 1 bar) from each track followed by filtering out windows that have more than one bar consecutive rests. Finally, for each split of train-1, validation-1, train-2, validation-2, and test dataset, we create a dataset $\mathcal{D} = \{(x_{\text{L}}^{(i)}, x_{\text{R}}^{(i)})\}_{i=1}^N$ by assigning the first 4-bars and the last 4-bars of each 8-bar window to $x_{\text{L}}^{(i)}$ and $x_{\text{R}}^{(i)}$, respectively. For encoding musical sequences, we use Melodico-rhythmic encoding proposed in [3].

Figure 7: Evaluation of iterated application of our model. Iterated application of models with larger λ tends to compare favorably with others. The upper/lower rows are transformation results where inputs are randomly created pairs of test/generated (sampled from prior) data. The color gradient corresponds to the number of iterations for our method. For all metrics, higher values are better. See Sec.3.6.

3.2 Implementation Details

2-layered long short term memory (LSTM) [13] is used for the encoder and the decoder of the VAE, with the number of latent dimensionality $d_z = 64$. 5-layered multilayer perceptron (MLP) is used for the generator and the discriminator, where input vectors are simply concatenated and ReLU activation is employed. Following the latent constraint paper [6], we use the gating mechanism for each of the outputs z'_L and z'_R. For the decoder to generate sequences, argmax operation is utilized at each time step greedily for sampling each token. In the adversarial learning, the parameter updates of the model are alternating between updating the discriminator 10 times and updating the generator once. Adam optimizer [14] is used for training with the parameters $(\alpha, \beta_1, \beta_2) = (0.000005, 0, 0.9)$.

3.3 Transformation Examples

Fig.1 and Fig.2 are the transformation examples for $\lambda = 2.0$ and $\lambda = 0.3$, respectively. In Fig.1b at iteration 1, the used pitch set of x_L becomes aligned to that of x_R, and the note at time around 4 transforms to the one with longer duration to join two input sequences. As the number of iterations increases, the note durations become more uniform, while retaining similarity to the input sequences in terms of pitch contours or rhythmic properties. In Fig.2b, four corners are generated by feeding s randomly sampled from $\{0, 1\}$ for each dimension and the others are interpolations. Interestingly, the time 4 to 8 of the bottom left sequence is the exact transposition of the right input of Fig.2a, whereas other sequences has variations and transpositions.

3.4 Evaluation Metrics

We use *reality*, correlations of five musical statistics evaluated against the test data, and *z-distance* as evaluation metrics. Reality is defined as the probability that a generated sequence is classified as human-made. Two-layered transformer binary classification model is trained on train-2 and validation-2 datasets, where the pair sequences from the dataset $\mathcal{D}$ are labeled as positive, while the randomly shuffled pair sequences from the same dataset are labeled as negative. The mean of output values of this classification model and its accuracy are 0.959 and 0.953 on the test dataset. Reality is a holistic metric of quality which quantifies how likely a sequence is human-made.

As musical statistics, we choose to use key, mean-pitch (MP), mean-duration (MD) for evaluating whether the first and the last 4-bars are more musically consistent after transformation. We also employ pitch-transition (PT) and duration-transition (DT) for evaluating if transitions between the first and the last 4-bars are smooth and natural. Fig.5 illustrates these five musical statistics. For key, MP, and MD, these values are estimated for the first and the last 4-bars, which are interpreted as Markov transitions of musical states, creating matrices as in Fig.5. For PT and DT, we extract two or three consecutive notes around the boundary of the first and the last 4-bars, and again compute Markov transition matrices of each statistics as states. Finally, as a quantitative metric, we compute Pearson correlation coefficient between the matrix made from the test dataset and the one from the samples outputted by models.

z-distance is a scaled squared Mahalanobis distance $dist(z', z) = 1/d_z(z' - z)\Sigma^{-1}(z' - z)^\intercal$, where $\Sigma = \mathrm{diag}(\bar{\sigma}_z^2)$. We analyze z-distance in the latent space $\mathcal{Z}$ using song id annotations of the dataset. We randomly sampled 1,000,000 pairs of 4-bar sequences from the entire dataset, where pairs are sampled either from the same or different song id, corresponding to intra- and inter-song in Fig.6. Pairs with identical sequences are filtered out to analyze the similarity rather than the identicality. Fig.6 shows

that sequences from different song ids are several orders of magnitude less likely to have smaller (say, less than 1.0 or 3.0) z-distances, compared with those from the same ids. The difference of z-distances probably comes from the fact that a lot of variations can be seen within each song id and similar themes are rarely seen among different song ids especially for melody, suggesting that smaller z-distances includes variations and that z-distance encodes musically meaningful similarity.

Algorithm 1 Latent Perturb (baseline method).

Input: z_L, z_R, and σ_LP

1: $S_\text{max} \leftarrow \text{argmax}_x\, p_{\hat{\theta}}(x|z_\text{L}) \oplus \text{argmax}_x\, p_{\hat{\theta}}(x|z_\text{R})$
2: $R_\text{max} \leftarrow$ estimate reality of S_max
3: **for** $i = 1$ to $NumIter$ **do**
4: $\quad \epsilon_\text{L}, \epsilon_\text{R} \sim \mathcal{N}(\mathbf{0}, \text{diag}(\sigma_\text{LP}^2))$
5: $\quad S_\text{L} \leftarrow \text{argmax}_x\, p_{\hat{\theta}}(x|z_\text{L} + \epsilon_\text{L})$
6: $\quad S_\text{R} \leftarrow \text{argmax}_x\, p_{\hat{\theta}}(x|z_\text{R} + \epsilon_\text{R})$
7: $\quad R_\text{tmp} \leftarrow$ estimate reality of $S_\text{tmp} = S_\text{L} \oplus S_\text{R}$
8: $\quad$ **if** $R_\text{tmp} > R_\text{max}$ **then**
9: $\quad\quad S_\text{max} \leftarrow S_\text{tmp}$; $R_\text{max} \leftarrow R_\text{tmp}$
10: $\quad\quad z_\text{L} \leftarrow z_\text{L} + \epsilon_\text{L}$; $z_\text{R} \leftarrow z_\text{R} + \epsilon_\text{R}$
11: $\quad$ **end if**
12: **end for**
13: **return** S_max

3.5 Comparisons of Methods

We introduce a naive but strong baseline method called *Latent Perturb (LP)*, which is summarized in Algorithm 1. Note that the reality estimation model is trained with the train-1 dataset, and the argmax operation is approximated with the greedy sampling scheme. Compared to data space, perturbation in the latent space can efficiently search similar samples with high reality.

Fig.4 shows the comparisons of baseline methods and ours evaluated on z-distance vs the other evaluation metrics, where z-distance is the average of the mean of z-distances for z_L and z_R. The reason for choosing LP with σ_LP in $[0.01, 0.2]$ is to make sure their resulting evaluation metrics span the range that the proposed technique "ours" spans. For samples from the prior as well as from the test dataset, our methods, especially with larger λ tend to outperform in all the metrics, even though LP methods use abundant computational budgets—100 forward computations of the decoder model and the classification model.

3.6 Evaluation of Iterated Transformation

Fig.7 illustrates the performance of iterated application of our transformation. Interestingly, performance of models with larger λ after several iterations are often comparable to or better than that of models with smaller λ. This means that a model with larger λ can be used for transformation with different distances, without sacrificing the performance. For example, as illustrated in Fig.1, the model with $\lambda = 2.0$, if iteratively applied, can be used for gradual transformations with several different distances. Their

quality is as satisfactory as single (non-iterative) transformation with models of smaller λ.

4. RELATED WORK

Concatenative Synthesis (CS) methods [15, 16] typically use a large database of source audio segmented into units, and search units that match each segment of the target audio by unit selection algorithms. Mashup methods [17, 18] combine two or more songs to create entertaining musical results. Typically tracks from different songs are superimposed and combined. Our Connective Fusion differs from CS and Mashup in that i) ours is generative rather than searching and using existing segments of music, ii) ours tries to combine any segments, whereas CS/Mashup typically combine some searched segments, iii) ours is unsupervised learning which does not need any human annotation in the dataset or musical expertise for training models, and iv) ours is symbolic which is often not the case in CS/Mashup.

Learning transformation in the latent space has been studied for image, texts, and music [6, 19–23]. Notably, Mueller et al. proposed to transform texts to have specified attributes based on optimization in the latent space [21]. Engel et al. proposed to transform image or music to become more realistic or to have specified attributes in the latent space by using adversarial learning [6]. Lore et al. and more recently Jahanian et al. instead proposed to transform image by learning latent space directions [22, 23].

Learning transformation with adversarial learning without paired data has been extensively studied. The input and output domain of transformation in these studies can be categorized into the same data domain with different classes [24] or attributes [6, 25], the semantic domain to the data domain [24], and semantically similar domain such as simulation to real [26, 27] and unsupervised language translation [28, 29]. On the other hand, the input and output domains of transformation in this paper are essentially the same except that they have different lengths/counts.

5. CONCLUSION

We introduced Connective Fusion, a new scheme of interactively generating sequences for creative purposes. Given two sequences of random combination, our model transforms each of them to similar one such that their concatenation is more indistinguishable from human-made sequences. The transformation model is adversarially learned in the latent space. Our model equips with user control—choosing the amount of transformation as well as exploring in the style space. In experiments of melody creation on a symbolic music dataset, we empirically demonstrated that our method outperforms a baseline method of various parameter settings, and that iterative gradual transformation not just introduced new functionality but also works as satisfactory as or sometimes better than non-iterative transformation.

6. REFERENCES

[1] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *International Conference on Learning Representations*, 2019.

[2] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a steerable model for Bach chorales generation," in *Proc. of the 34th International Conference on Machine Learning*, 2017.

[3] G. Hadjeres and F. Nielsen, "Anticipation-rnn: enforcing unary constraints in sequence generation, with application to interactive music generation," *Neural Computing and Applications*, vol. 32, no. 4, pp. 995–1005, 2020. [Online]. Available: https://doi.org/10.1007/s00521-018-3868-4

[4] J. Gillick, A. Roberts, J. H. Engel, D. Eck, and D. Bamman, "Learning to groove with inverse sequence transformations," in *Proceedings of the 36th International Conference on Machine Learning, ICML*, 2019.

[5] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. of the 35th International Conference on Machine Learning*, 2018.

[6] J. Engel, M. Hoffman, and A. Roberts, "Latent constraints: Learning to generate conditionally from unconditional generative models," in *International Conference on Learning Representations*, 2018.

[7] T. Akama, "Controlling symbolic music generation based on concept learning from domain knowledge," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019.

[8] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: modeling dynamics and instrumentation of music with applications to style transfer," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, E. Gómez, X. Hu, E. Humphrey, and E. Benetos, Eds., 2018.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR*, 2014.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, 2014.

[11] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, 2016.

[12] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[15] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, 1996, pp. 373–376 vol. 1.

[16] A. Zils and F. Pachet, "Musical Mosaicing," in *Proceedings of the International Conference on Digital Audio Effects*, 2001.

[17] N. Tokui, "Massh! - a web-based collective music mashup system," in *Proceedings - 3rd International Conference on Digital Interactive Media in Entertainment and Arts, DIMEA 2008*, 2008.

[18] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "Automashupper: Automatic creation of multi-song music mashups," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 2014.

[19] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Advances in Neural Information Processing Systems 29*, 2016.

[20] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] J. Mueller, D. Gifford, and T. Jaakkola, "Sequence to better sequence: Continuous revision of combinatorial structures," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[22] L. Goetschalckx, A. Andonian, A. Oliva, and P. Isola, "Ganalyze: Toward visual definitions of cognitive image properties," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[23] A. Jahanian*, L. Chai*, and P. Isola, "On the "steerability" of generative adversarial networks," in *International Conference on Learning Representations*, 2020.

[24] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision, ICCV*, 2017.

[25] G. Lample, S. Subramanian, E. Smith, L. Denoyer, M. Ranzato, and Y.-L. Boureau, "Multiple-attribute text rewriting," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=H1g2NhC5KQ

[26] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017.

[27] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017.

[28] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato, "Unsupervised machine translation using monolingual corpora only," in *International Conference on Learning Representations*, 2018.

[29] M. Artetxe, G. Labaka, E. Agirre, and K. Cho, "Unsupervised neural machine translation," in *International Conference on Learning Representations*, 2018.

TOWARDS MULTIMODAL MIR: PREDICTING INDIVIDUAL DIFFERENCES FROM MUSIC-INDUCED MOVEMENT

Yudhik Agrawal[1]   **Samyak Jain**[1]   **Emily Carlson**[2]
Petri Toiviainen[2]   **Vinoo Alluri**[1]

[1] Cognitive Science Lab, International Institute of Information Technology, Hyderabad, India
[2] Department of Music, Art and Culture Studies, University of Jyväskylä, Finland

{yudhik.agrawal, samyak.j}@research.iiit.ac.in, {emily.j.carlson, petri.toiviainen}@jyu.fi
vinoo.alluri@iiit.ac.in

ABSTRACT

As the field of Music Information Retrieval grows, it is important to take into consideration the multi-modality of music and how aspects of musical engagement such as movement and gesture might be taken into account. Bodily movement is universally associated with music and reflective of important individual features related to music preference such as personality, mood, and empathy. Future multimodal MIR systems may benefit from taking these aspects into account. The current study addresses this by identifying individual differences, specifically Big Five personality traits, and scores on the Empathy and Systemizing Quotients (EQ/SQ) from participants' free dance movements. Our model successfully explored the unseen space for personality as well as EQ, SQ, which has not previously been accomplished for the latter. R^2 scores for personality, EQ, and SQ were 76.3%, 77.1%, and 86.7% respectively. As a follow-up, we investigated which bodily joints were most important in defining these traits. We discuss how further research may explore how the mapping of these traits to movement patterns can be used to build a more personalized, multi-modal recommendation system, as well as potential therapeutic applications.

1. INTRODUCTION

From the perspective of most computational analysis, music can be defined as sound, its important features yielding to the decomposition of waveforms. However, for the vast majority of history, musical sound could not be separated from its source; to whatever degree it may have evolved biologically to serve various human functions, music must be regarded as an embodied and socially embedded phenomenon [1–3]. Research has shown intimate links between musical features and human movement, including the reflection of hierarchical rhythmic structures in embodied eigen movements [4], the reflection of higher-level musical structures in group movement to Electronic Dance Movement [5], and reflection of spectral and timbral features of music in dance [6]. Bodily movement is one of the most commonly reported responses to music [7], and movement to music is one of very few universal features of music across cultures [8].

This paper towards Multimodal MIR takes into consideration the multi-modality of music, and takes into account one of the primary aspect of musical engagement, i.e, movement. It is therefore insufficient to consider music only in terms of sound when trying to understand human digital use and interaction with music. This may be especially true in terms of user experience and personalization; human movement in response to music reflects not only the music itself but characteristics of the individual, such as personality [9] and emotion [10]. Indeed, research has shown that music-induced movement is so individual that its features can be used in person-identification with a high degree of accuracy [11]. This is in line with previous research, such as that of Cutting et al. [12] demonstrating that friends can recognise each other from their walk with only point-light (stick figure) displays of movement, without the need for other distinguishing features. This paradoxical balance between universality and individuality in human motoric responsiveness to music poses a challenge for the creation of digital music interfaces which take music-induced movement into account in providing personalized music experiences. Although the concept of an interactive music system has long been proposed that allow music playback to be controlled and altered via human gestures [13], human-movement based interaction techniques and devices are fast gaining importance in the field of HCI [14]. In this context, it makes decoding aspects of a user/individual via human movements a key and useful endeavor, which would then aid in the design of more personalized experiences.

2. RELATED WORK

The specific features used in previous work associated movement with individual differences are quite varied. Satchell et al. [15] examined speed, relative and absolute rotation of the body and found relationships between relative movement of the upper and lower body during walk-

ing in both FFM personality traits and gait, while Micha-lak et al. [16] were able to associate low mood with lateral body sway and posture. In dance, relevant features have in-cluded amount of movement of the whole body relative to itself and to the environment, responsiveness to music fea-tures such as tempo [4, 17]. Another area for exploration of individual differences in movement patterns has been in the context of disorders that have altered or impaired movement [18–20]. These links allow us to postulate that movement patterns should give us information related to individual traits and tendencies which can be then linked to music preferences, mood or emotion in relation to mu-sic experiences and could have implications for music ther-apy as well as for music information retrieval. However, as an initial step, there exist no studies that predict per-sonality and empathy as a function of movement patterns. The current study focuses on identifying FFM personality traits, as well as scores on the Empathy Quotient (EQ) and Systemizing Quotient (SQ), from participants' free dance movements to various genres of music. The EQ measures participants' tendency to empathize with others [21], while the SQ measures the tendency to think in terms of sys-tems [22]. These two measures were originally developed to increase understanding of people with ASD, as in this population trait systemizing tends to be very high while empathy tends to be low. However previous work has also used the EQ/SQ to determine how these traits are dis-tributed in the general population. Although previous work has found relationships between empathy and responsive-ness to changes in heard music or in dance partner [23, 24], and between EQ/SQ scores and music preferences [25, 26], general movement patterns associated with empathy have not, to the knowledge of the authors, been explored using dance movement, nor have patterns related to systemizing tendencies.

3. METHOD
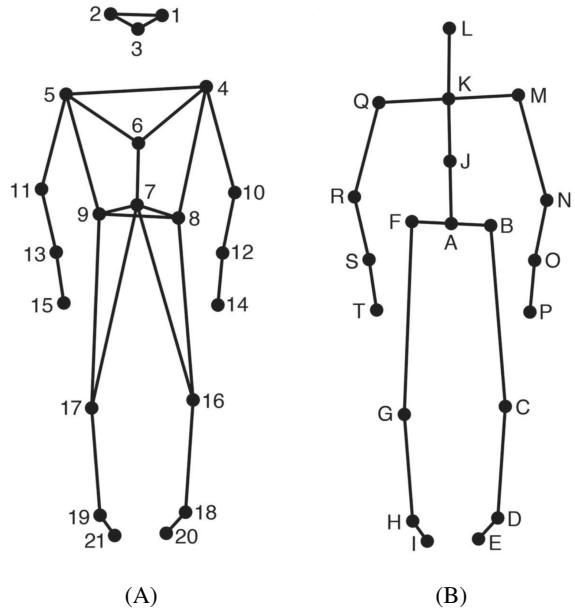
3.1 Participants

Data acquired was from a previous study [27] comprising data from 73 university students (54 females, mean age = 25.74 years, std = 4.72 years). Thirty-three reported hav-ing received formal musical training; five reported one to three years, ten reported seven to ten years, while sixteen reported ten or more years of training. Seventeen partici-pants reported having received formal dance training; ten reported one to three years, five reported four to six years, while two reported seven to ten years. Participants were of 24 different nationalities, with Finland, the United States, and Vietnam being the most frequently represented. For attending the experiment, participants received two movie ticket vouchers each. All participants spoke and received instructions in English. Fifteen participants were excluded from further analysis due to incomplete data. They were asked to listen to the music and move as freely as they de-sired, but staying within the marked capture space. The aim of these instructions was to create a naturalistic set-ting, such that participants would feel free to behave as

they might in a real-world situation.

3.2 Apparatus, Stimuli, and Procedure
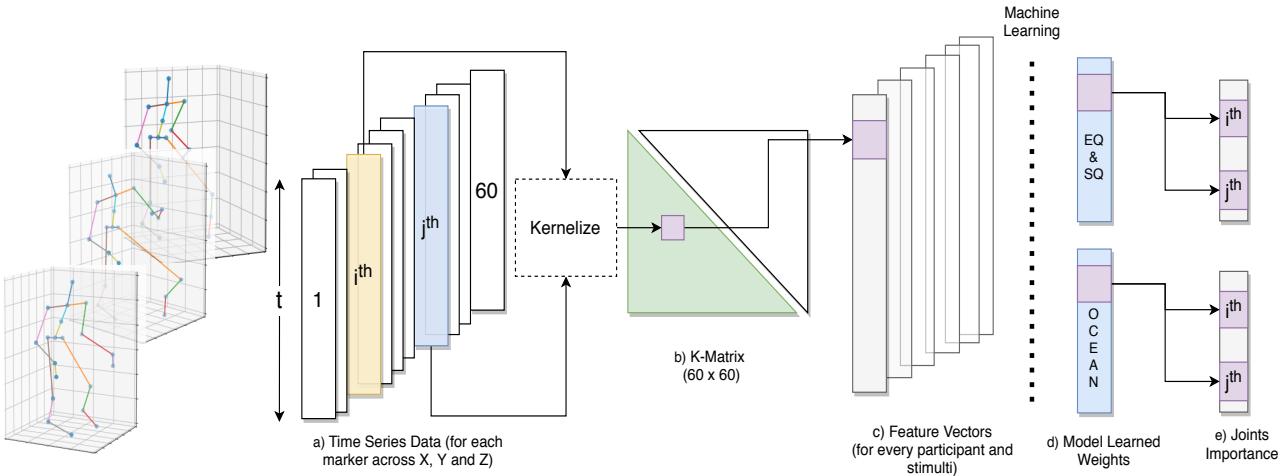
Participants' movements were recorded using a twelve-camera optical motion-capture system (Qualisys Oqus 5+), tracking at a frame rate of 120 Hz, the three-dimensional position of 21 reflective markers attached to each partic-ipant. Markers were located as follows (L=left, R=right, F=front, B=back) 1: LF head; 2: RF head; 3: B head; 4: L shoulder; 5: R shoulder; 6: sternum; 7: stomach; 8: LB hip; 9: RB hip; 10: L elbow; 11: R elbow; 12: L wrist; 13: R wrist; 14: L middle finger; 15: R middle finger; 16: L knee; 17: R knee; 18: L ankle; 19: R ankle; 20: L toe; 21: R toe. The stimuli comprised sixteen 35-second excerpts from eight genres, in randomized order: Blues, Country, Dance, Jazz, Metal, Pop, Rap, and Reggae. The stimuli for the experiment were selected using a computa-tional process based on social-tagging and acoustic data. The selection pipeline was designed to select naturalistic stimuli that were uncontroversially representative of their respective genres, which would also be appropriate to use in a dance setting. Moreover, investigating movements to multiple genres of music further adds to the generalizabil-ity of our findings.

Figure 1: Marker and joint locations (A) Anterior view of the marker locations a stick figure illustration; (B) Anterior view of the locations of the secondary markers/joints used in animation and analysis of the data

3.2.1 Personality and Trait Empathy Measures

The Big Five Inventory (BFI) was used to capture the five predominant personality dimensions, namely, Openness, Conscientiousness, Extraversion, Agreeableness, and Neu-roticism [28]. Trait Empathy was measured using the EQ- and SQ-short form version, developed and validated by Wakabayashi et al. [29], as a result giving an EQ and SQ score per participant.

Figure 2: Overview of our Pipeline. Given the position of joints across time frames in 3D Euclidean space(a), we apply pairwise correntropy between time series x_i and x_j and calculate the K-matrix (b). Then, taking the lower triangular part of the symmetric covariance matrix, we get the feature vectors (c). After training the regression model on the feature vectors, we get the weight vector(d). Finally, corresponding weight values from the learned weight vector are mapped to the corresponding joints to get the per-joint importance (e).

3.3 Feature Extraction

The analysis and prediction pipeline is illustrated in Figure 2. To facilitate extraction of kinematic features using the MATLAB Motion Capture (MoCap) Toolbox [30], a set of 20 secondary markers, subsequently referred to as joints, was derived from the original 21 markers. The locations of these 20 joints are depicted in Figure 1. The locations of joints B, C, D, E, F, G, H, I, M, N, O, P, Q, R, S, and T are identical to the locations of one of the original markers, while the locations of the remaining joints were obtained by averaging the locations of two or more markers; Joint A: midpoint of the two back hip markers; J: midpoint the shoulder and hip markers; K: midpoint of shoulder markers; and L: midpoint of the three head markers. The instantaneous velocity of each marker in each direction was calculated. Instantaneous velocity was estimated by time differentiation followed by the application of a 2nd-order Butterworth filter with a cutoff frequency of 24Hz [30].

The features used in our analysis is the co-variances of position and velocity. The co-variances between all marker time series in each direction (X, Y and Z) within each participant for each stimulus. We used a non-linear measure to calculate covariance between the markers. This method, referred to as correntropy between time series x_i and x_j [31], is given by:

$$K(x_i, x_j) = e^{\frac{-||x_i - x_j||_2^2}{2\sigma^2 T^2}} \qquad (1)$$

where $||x_i - x_j||_2$ is L2-norm between x_i and x_j, σ is a constant, 12.0 in our case and T is the length of the time-series. The L2-norm is divided by T to normalize according to time series length since it has different lengths with varying stimuli. Since the number of joints are 20 and each joint has three coordinates, the dimension of K

would be 60×60. The lower triangular part excluding the diagonal elements of the symmetric covariance matrix was vectorised to produce a feature vector of length 1770 for each participant and for each stimuli.

We also run our experiments using the Normalized feature vectors calculated by using Position and Velocity, we employed standard Gaussian Normalization technique:

$$\hat{X} = \frac{X - \mu(X)}{\sigma(X)} \qquad (2)$$

where $\hat{X}$ is the feature vector, $\mu(X)$ is the mean and $\sigma(X)$ is standard deviation.

3.4 Model Regression

The most common regression model for value prediction tasks used is Linear Regression. The goal here is to find an optimal line that minimizes the total prediction error. But this model treats its parameters as unknown constants whose values must be derived. Moreover, the weights become sensitive when the dataset size is large. So to prevent the model from overfitting, we took principal components of the features to train the model (For the result sections, we will be considering 243 components for position data and 137 components for velocity data which gave us the best results). We also approached this problem by using Bayesian Regression other than Principal Component Regression (PCR) [1] .

In Bayesian Regression the parameters are treated as random variables belonging to an underlying distribution. Depending on the dataset, we can be more or less certain about the weights. Since, the parameters of the model belong to a distribution, the predictions of the model also belong to a distribution. So we have confidence bounds on

---

[1] Detailed analysis of Principal Component Regression (PCR) and Bayesian Regression are discussed in the supplementary.

our predictions. Therefore, they are better at representing the uncertainty of a model's predictions.

3.4.1 Personality and Trait Empathy Prediction

The features extracted are used to train five different Bayesian Regression models to predict each of the five personality traits - Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism (OCEAN). The features extracted are used to train both regression models to predict EQ and SQ respectively. The model is trained and evaluated on the described dataset.

3.5 Visualizing the Weight Vector

To interpret the coefficients (also known as the weights or model parameters) of the regression models, we add the value of the feature vector to the corresponding joints. In our algorithm, we first find the index in the 60×60 matrix and then add the absolute value to those joints.

The sign of the coefficient indicates the direction of the relationship but the magnitude preserves the importance. After that, Min-Max Normalization is applied to bring the values in the range (0, 1) for better visualizing the same variable across similar tasks.

$$\overline{JI}[i] = \left(\frac{JI[i] - min(JI)}{max(JI) - min(JI)} \right) \forall JI[i] \qquad (3)$$

where $\overline{JI}$ represent the normalized Joint Importance Vector. Algorithm 1 describes the pseudo-code to get the importance of joints from the weights of the trained regression model.

Algorithm 1 Joint Importance

Result: Calculate a vector J of 20 dimensions representing importance of each joint.

W is the weight vector; J is the importance vector initialised with 0; S contains lower triangular indices excluding diagonal indices; 0-indexing is followed;

1: $S \leftarrow LowerTriangularIndices(60 \times 60)$
2: $N \leftarrow S.length()$
3: **for** $k = 0 : N - 1$ **do**
4:     $(i, j) := S(k)$
5:     $(\hat{i}, \hat{j}) \leftarrow IndexToJoint(i, j)$
6:     $J(\hat{i}) := J(\hat{i}) + |W(k)|$
7:     $J(\hat{j}) := J(\hat{j}) + |W(k)|$
8: **end for**
9: **return** J

After getting a vector of 20 dimension, we reduce it to 12 before visualizing joint importance. We did this by taking the average of joints which occur in pairs eg. (L shoulder, R shoulder), (L wrist, R wrist).

3.5.1 Evaluation Metric

(a) Root Mean Square Error (RMSE): It computes a risk metric corresponding to the expected value of the root of squared (quadratic) error or loss.

(b) R^2 Score: It represents the proportion of the variance(of y) that has been explained by the independent variables in the model. [2]

As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable and at the same time the R^2 helps us evaluate the goodness of fit in capturing the variance in training data. We calculate RMSE and R^2 on multiple splits so that we get an average estimate of the accuracy.

3.6 Results

3.6.1 EQ and SQ

The results for EQ prediction are in Table 1 and SQ prediction are in Table 2. The results are calculated using 5-fold cross validation. The range of EQ and SQ is 0-80. The boldface values represent the best score. The 'N' in the tables denote that Gaussian Normalization was applied on the features. We trained and evaluated two different models for each of the aforementioned tasks. We can see that using position data, instead of velocity data, to generate the feature vectors, gave us the best results. Also, we can see that the Bayesian Regression gave better results than Principal Component Regression on both metrics. So from here on, we will be using Bayesian Regression for other prediction and analysis tasks.

Input	PCR		Bayesian Ridge	
	RMSE	R^2	RMSE	R^2
Position	3.071	0.708	**2.722**	**0.771**
Position(N)	3.201	0.684	2.733	0.765
Velocity	4.938	0.249	4.343	0.423
Velocity(N)	4.583	0.353	4.015	0.503

Table 1: Prediction Results for Empathizing Quotient

Input	PCR		Bayesian Ridge	
	RMSE	R^2	RMSE	R^2
Position	2.398	0.781	**2.161**	**0.867**
Position(N)	2.363	0.786	2.502	0.838
Velocity	4.448	0.252	3.832	0.469
Velocity(N)	4.211	0.329	3.714	0.552

Table 2: Prediction Results for Systemizing Quotient

3.6.2 Personality Regression

The results for OCEAN value prediction on Dataset can be found in Table 3. The results are calculated using 5-fold cross validation. The range of the personality values is 1.0-5.0. We can see that using position data to extract features gave the best results on predicting all five personality traits on the dataset. We can concur that using position data instead of velocity data in the kernelized space is better for these regression tasks.

---

[2] Detailed explanation of metrics is provided in the supplementary.

Input	Openness		Conscientiousness		Extraversion		Agreeableness		Neuroticism	
	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2
Position	**0.197**	**0.776**	**0.317**	**0.760**	**0.384**	0.743	**0.252**	**0.776**	**0.384**	**0.758**
Position(N)	0.227	0.740	0.332	0.690	0.414	**0.756**	0.273	0.716	0.390	0.739
Velocity	0.332	0.464	0.487	0.415	0.556	0.523	0.440	0.335	0.557	0.483
Velocity(N)	0.304	0.527	0.426	0.543	0.501	0.623	0.408	0.442	0.461	0.654

Table 3: Prediction Results for Five Personality Traits using Bayesian Regression

3.6.3 Joints' Importance

For evaluating joint importance we used learned weights of the model using position data across different tasks. For the purpose of analyzing the importance of the joints, we reduced them to 12 by taking the average for those joints which occur in pairs eg. (L shoulder, R shoulder). This was also done for hips, knee, ankle, toe, elbow, wrist, and finger. Altogether the results in characterizing an individual trait is dominated by the limbs than the core of the body.

From the relative joint importance depicted in Figure 3, we observe that 'Ankle', Elbow' and 'Shoulder' play an important role in determining EQ and SQ of an individual, whereas 'Neck' and 'Torso' have a negligible contribution. We also infer that 'Finger', 'Hip', and 'Knee' are more crucial joints for predicting EQ than for SQ whereas 'Elbow' holds significantly higher importance for predicting SQ than for EQ.

Figure 4 displays the relative joint importance of personality along with the mean plotted in each sub-figure. The farther away from the mean the joint importance value for an individual joint is, the more important it is in characterizing that trait. Some similarities in the joint importance profiles across the personality traits can be attributed to the inherent correlation that exists among them [3]. We observe that it is the 'Finger', 'Elbow', and 'Knees' that contribute to Feature Importance whereas 'Root', 'Neck' and 'Torso' have negligible contribution. For characterizing Conscientiousness, 'Shoulders', 'Knees' and 'Neck' play a crucial role while 'Head' and 'Toe' plays an important role for Extraversion. For Agreeableness, 'Neck' and 'Wrists' have relatively less importance as compared to other joints whereas, 'Wrists' play an important role in Openness. Finally, there are no significant defining features for Neuroticism, which indicates that their expression in Dance Movements through Music-Induced Movements are very limited.

4. DISCUSSION
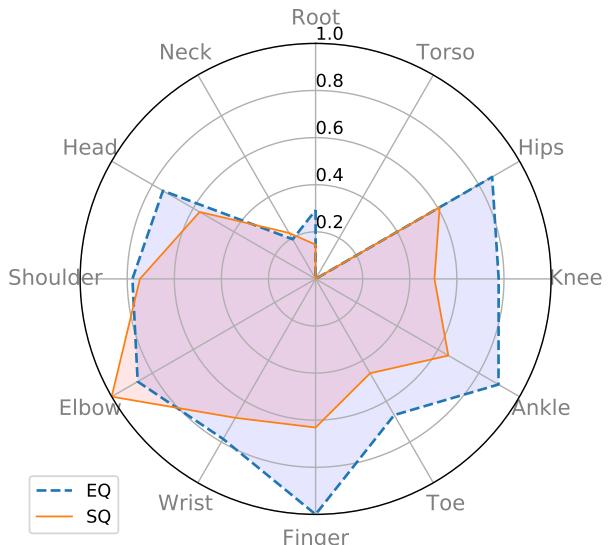
Music experiences are highly embodied, making it necessary to consider individual embodied responses to music in developing more advanced personalized user experiences. The current study is among the first to the authors' knowledge to use participants' free dance movements to predict personality traits, and both the Empathizing and Systemizing Quotients (EQ/SQ).

Co-variance between joint velocities has previously been used to identify an individual from their free dance movements with a high degree of accuracy [11]. The results of the current analysis show co-variance to be a useful feature in predicting individual differences. However, we achieved considerably better prediction accuracy by using position data than velocity data.

Overall, the limbs of the body seemed to have more importance in predicting individual traits than the core body. This is in line with the fact that gesture plays an important role in communication [32], and as specifically regards the EQ/SQ, as these tests were originally developed in conjunction with studies of ASD, in which gesture and imitative movement appear to be compromised [33]. Although the sample used in the current study comprised typically functioning (non-ASD) participants, the accuracy of prediction of EQ/SQ scores in this analysis is worth highlighting in light of recent work suggesting the existence of motor signatures unique to ASD, detectable from whole body movements as well as data drawn from participants' interaction with tablets [19, 34].

The specific markers that were important in the prediction of individual traits in some cases corroborates previous work, and in some cases contradicts it. Luck et al. [9] found correlations between Extraversion and speed of head movement, which supports the current finding that the head is of particular importance in identifying Extraver-

Figure 3: Relative importance of Joints in EQ and SQ Tasks using the Position Data.

---

[3] The table for Spearman Correlation between the personality traits is provided in the supplementary material.

Figure 4: Relative importance of Joints of the five personality traits(Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism) using the Position Data. The black line indicates the mean importance of the corresponding joint marker. The red dotted line in the top left sub-figure indicates the standard deviation about the mean.

sion. Carlson et al. [17] found that, compared to Conscientiousness, the core body was more important in responsiveness to musical tempo in relation to Extraversion, which is partly supported by the slightly greater importance of the finger and wrist markers to Extraversion in our study, but partly contradicted by the importance of the shoulder marker in Extraversion. The difference between findings may relate to the use in the current study of position rather than velocity or acceleration data; that is, core body posture while moving to music may be more indicative of Conscientiousness than core body movement. EQ scores were more related to head, finger, hips and lower limb joints than SQ scores, which may be partly attributed to gender-typical movement patterns as females tend to score higher on the EQ than males [21, 35].

Several limitations of the current study should be noted. First, the majority of participants were from European or North American countries, and all eight music stimuli were of Western origin, limiting the degree to which results can be generalized cross-culturally. Secondly, There may exist potential bias due to gender imbalance. Future work could include separate analysis performed within gender categories. And lastly, participants' preferences for heard stimuli were not included in our model. This would be an important feature to focus on in future work, as preference

and enjoyment are highly relevant to personalized MIR.

Further extension of this work could help to make music recommendation systems more robust. Previous work has considered the relationship between personality and music preference [25, 36], while Greenberg et al. [26] explored the relationship between music preference and empathizing-systematizing theory, suggesting even that music may play a role in increasing empathy in people with empathy-related disorders, such as ASD. However, the relationship between embodiment, personality and musical experiences requires further exploration.

To conclude, this study represents an early step towards multimodal MIR. To make this approach applicable to personalized gesture-based retrieval systems, it can be extended to monocular video captured by accessible devices such as a mobile phone camera. This approach would be feasible due to recent progress in the area of 3D human pose estimation in predicting the body joint coordinates from a monocular video [37–39]. This would then allow future recommendation systems to take embodied processes into account, resulting in better and more responsive personalized experiences.

5. REFERENCES

[1] J. C. Bispham, "The human faculty for music: What's special about it?" Ph.D. dissertation, University of Cambridge, 2018.

[2] I. Cross, "The evolutionary nature of musical meaning," *Musicae scientiae*, vol. 13, no. 2_suppl, pp. 179–200, 2009.

[3] J. Richter and R. Ostovar, ""it don't mean a thing if it ain't got that swing"–an alternative concept for understanding the evolution of dance and music in human beings," *Frontiers in human neuroscience*, vol. 10, p. 485, 2016.

[4] P. Toiviainen, G. Luck, and M. R. Thompson, "Embodied meter: hierarchical eigenmodes in music-induced movement," *Music Perception*, vol. 28, no. 1, pp. 59–70, 2010.

[5] R. T. Solberg and A. R. Jensenius, "Pleasurable and intersubjectively embodied experiences of electronic dance music," *Empirical Musicology Review*, vol. 11, no. 3-4, pp. 301–318, 2017.

[6] B. Burger and P. Toiviainen, "Embodiment in electronic dance music: Effects of musical content and structure on body movement," *Musicae Scientiae*, p. 1029864918792594, 2018.

[7] M. Lesaffre, L. D. Voogdt, M. Leman, B. D. Baets, H. D. Meyer, and J.-P. Martens, "How potential users of music search and retrieval systems describe the semantic quality of music," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 5, pp. 695–707, 2008.

[8] B. Nettl, "An ethnomusicologist contemplates universals in musical sound and musical culture," *The origins of music*, vol. 3, no. 2, pp. 463–472, 2000.

[9] G. Luck, S. Saarikallio, B. Burger, M. R. Thompson, and P. Toiviainen, "Effects of the big five and musical genre on music-induced movement," *Journal of Research in Personality*, vol. 44, no. 6, pp. 714–720, 2010.

[10] G. Luck, S. Saarikallio, B. Burger, M. Thompson, and P. Toiviainen, "Emotion-driven encoding of music preference and personality in dance," *Musicae Scientiae*, vol. 18, no. 3, pp. 307–323, 2014.

[11] E. Carlson, P. Saari, B. Burger, and P. Toiviainen, "Dance to your own drum: Identification of musical genre and individual dancer from motion capture using machine learning," *Journal of New Music Research*, pp. 1–16, 2020.

[12] J. E. Cutting and L. T. Kozlowski, "Recognizing friends by their walk: Gait perception without familiarity cues," *Bulletin of the psychonomic society*, vol. 9, no. 5, pp. 353–356, 1977.

[13] M. Subotnick, "Interactive music playback system utilizing gestures," Nov. 1 2001, uS Patent App. 09/835,840.

[14] M. Gillies, "Understanding the role of interactive machine learning in movement interaction design," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 26, no. 1, pp. 1–34, 2019.

[15] L. Satchell, P. Morris, C. Mills, L. O'Reilly, P. Marshman, and L. Akehurst, "Evidence of big five and aggressive personalities in gait biomechanics," *Journal of nonverbal behavior*, vol. 41, no. 1, pp. 35–44, 2017.

[16] J. Michalak, N. F. Troje, J. Fischer, P. Vollmar, T. Heidenreich, and D. Schulte, "Embodiment of sadness and depression—gait patterns associated with dysphoric mood," *Psychosomatic medicine*, vol. 71, no. 5, pp. 580–587, 2009.

[17] E. Carlson, B. Burger, J. London, M. R. Thompson, and P. Toiviainen, "Conscientiousness and extraversion relate to responsiveness to tempo in dance," *Human movement science*, vol. 49, pp. 315–325, 2016.

[18] M. J. de Dreu, A. Van Der Wilk, E. Poppe, G. Kwakkel, and E. E. van Wegen, "Rehabilitation, exercise therapy and music in patients with parkinson's disease: a meta-analysis of the effects of music-based movement therapy on walking ability, balance and quality of life," *Parkinsonism & related disorders*, vol. 18, pp. S114–S119, 2012.

[19] A. Anzulewicz, K. Sobota, and J. T. Delafield-Butt, "Toward the autism motor signature: Gesture patterns during smart tablet gameplay identify children with autism," *Scientific reports*, vol. 6, no. 1, pp. 1–13, 2016.

[20] E. B. Torres, M. Brincker, R. W. Isenhower III, P. Yanovich, K. A. Stigler, J. I. Nurnberger Jr, D. N. Metaxas, and J. V. José, "Autism: the micro-movement perspective," *Frontiers in integrative neuroscience*, vol. 7, p. 32, 2013.

[21] S. Baron-Cohen and S. Wheelwright, "The empathy quotient: an investigation of adults with asperger syndrome or high functioning autism, and normal sex differences," *Journal of autism and developmental disorders*, vol. 34, no. 2, pp. 163–175, 2004.

[22] S. Baron-Cohen, J. Richler, D. Bisarya, N. Gurunathan, and S. Wheelwright, "The systemizing quotient: an investigation of adults with asperger syndrome or high–functioning autism, and normal sex differences," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 358, no. 1430, pp. 361–374, 2003.

[23] J. M. S. Bamford and J. W. Davidson, "Trait empathy associated with agreeableness and rhythmic entrainment in a spontaneous movement to music task:

Preliminary exploratory investigations," *Musicae Scientiae*, vol. 23, no. 1, pp. 5–24, 2019.

[24] E. Carlson, B. Burger, and P. Toiviainen, "Dance like someone is watching: A social relations model study of music-induced movement," *Music & Science*, vol. 1, p. 2059204318807846, 2018.

[25] E. Carlson, P. Saari, B. Burger, and P. Toiviainen, "Personality and musical preference using social-tagging in excerpt-selection." *Psychomusicology: Music, Mind, and Brain*, vol. 27, no. 3, p. 203, 2017.

[26] D. M. Greenberg, S. Baron-Cohen, D. J. Stillwell, M. Kosinski, and P. J. Rentfrow, "Musical preferences are linked to cognitive styles," *PloS one*, vol. 10, no. 7, 2015.

[27] E. Carlson, B. Burger, and P. Toiviainen, "Empathy, entrainment, and perceived interaction in complex dyadic dance movement," *Music Perception: An Interdisciplinary Journal*, vol. 36, no. 4, pp. 390–405, 2019.

[28] O. P. John, S. Srivastava *et al.*, "The big five trait taxonomy: History, measurement, and theoretical perspectives," *Handbook of personality: Theory and research*, vol. 2, no. 1999, pp. 102–138, 1999.

[29] A. Wakabayashi, S. Baron-Cohen, S. Wheelwright, N. Goldenfeld, J. Delaney, D. Fine, R. Smith, and L. Weil, "Development of short forms of the empathy quotient (eq-short) and the systemizing quotient (sq-short)," *Personality and individual differences*, vol. 41, no. 5, pp. 929–940, 2006.

[30] B. Burger and P. Toiviainen, "Mocap toolbox-a matlab toolbox for computational analysis of movement data," in *10th Sound and Music Computing Conference, SMC 2013, Stockholm, Sweden*. Logos Verlag Berlin, 2013.

[31] W. Liu, P. P. Pokharel, and J. C. Príncipe, "Correntropy: Properties and applications in non-gaussian signal processing," *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5286–5298, 2007.

[32] S. Goldin-Meadow, "Talking and thinking with our hands," *Current directions in psychological science*, vol. 15, no. 1, pp. 34–39, 2006.

[33] A. F. d. C. Hamilton, R. M. Brindley, and U. Frith, "Imitation and action understanding in autistic spectrum disorders: how valid is the hypothesis of a deficit in the mirror neuron system?" *Neuropsychologia*, vol. 45, no. 8, pp. 1859–1868, 2007.

[34] D. Wu, J. V. José, J. I. Nurnberger, and E. B. Torres, "A biomarker characterizing neurodevelopment with applications in autism," *Scientific reports*, vol. 8, no. 1, pp. 1–14, 2018.

[35] N. F. Troje, "Decomposing biological motion: A framework for analysis and synthesis of human gait patterns," *Journal of vision*, vol. 2, no. 5, pp. 2–2, 2002.

[36] P. J. Rentfrow and S. D. Gosling, "The do re mi's of everyday life: the structure and personality correlates of music preferences." *Journal of personality and social psychology*, vol. 84, no. 6, p. 1236, 2003.

[37] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3d human pose estimation in video with temporal convolutions and semi-supervised training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7753–7762.

[38] A. Venkat, C. Patel, Y. Agrawal, and A. Sharma, "Humanmeshnet: Polygonal mesh recovery of humans," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[39] Y. Cheng, B. Yang, B. Wang, and R. T. Tan, "3d human pose estimation using spatio-temporal networks with explicit occlusion training," *arXiv preprint arXiv:2004.11822*, 2020.

IMPROVED HANDLING OF REPEATS AND JUMPS IN AUDIO–SHEET IMAGE SYNCHRONIZATION

Mengyi Shan
Harvey Mudd College
mshan@g.hmc.edu

TJ Tsai
Harvey Mudd College
ttsai@g.hmc.eud

ABSTRACT

This paper studies the problem of automatically generating piano score following videos given an audio recording and raw sheet music images. Whereas previous works focus on synthetic sheet music where the data has been cleaned and preprocessed, we instead focus on developing a system that can cope with the messiness of raw, unprocessed sheet music PDFs from IMSLP. We investigate how well existing systems cope with real scanned sheet music, filler pages and unrelated pieces or movements, and discontinuities due to jumps and repeats. We find that a significant bottleneck in system performance is handling jumps and repeats correctly. In particular, we find that a previously proposed Jump DTW algorithm does not perform robustly when jump locations are unknown a priori. We propose a novel alignment algorithm called Hierarchical DTW that can handle jumps and repeats even when jump locations are not known. It first performs alignment at the feature level on each sheet music line, and then performs a second alignment at the segment level. By operating at the segment level, it is able to encode domain knowledge about how likely a particular jump is. Through carefully controlled experiments on unprocessed sheet music PDFs from IMSLP, we show that Hierarachical DTW significantly outperforms Jump DTW in handling various types of jumps.

1. INTRODUCTION
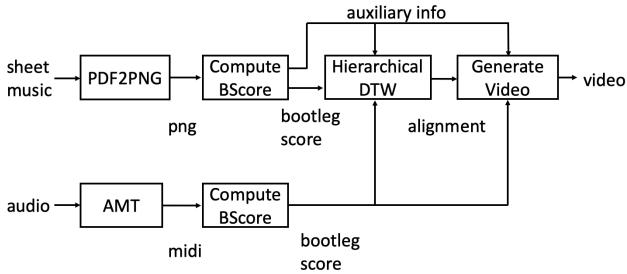
This paper tackles the problem of generating piano score following videos in a fully automated manner. Given an audio recording of a piano performance, our long-term goal is to build a system that can (a) identify the piece and automatically download the corresponding sheet music PDF from the International Music Score Library Project (IMSLP) website, and (b) generate a video showing the corresponding line of sheet music at each time instant in the audio recording. In this work, we focus exclusively on task (b), assuming that the correct sheet music PDF has been identified. This task requires us to perform audio–sheet music alignment on completely unprocessed PDF

files from IMSLP. This paper describes the key insights we have gained in building such a system, along with a novel alignment algorithm developed in the process.

Many previous works have studied cross-modal alignment between sheet music images and audio. Two general categories of approaches have been proposed. The first approach is to convert the sheet music images to a symbolic representation using optical music recognition (OMR), to collapse the pitch information across octaves to get a chroma representation, and then to compare this representation to chroma features extracted from the audio. This approach has been applied to synchronizing audio and sheet music [1] [2] [3], identifying audio recordings that correspond to a given sheet music representation [4], and finding the corresponding audio segment given a short segment of sheet music [5]. The second approach is to convert both sheet music and audio into a learned feature space that directly encodes semantic similarity. This has been done using convolutional neural networks combined with canonical correlation analysis [6] [7], pairwise ranking loss [8] [9], or some other suitable loss metric. This approach has been explored in the context of online sheet music score following [10], sheet music retrieval given an audio query [11] [8] [9], and offline alignment of sheet music and audio [8]. Recent works [12] [13] have also shown promising results formulating the score following problem as a reinforcement learning game. See [14] for an overview of work in this area.

The main difference between our current task and previous work is that we are working with totally unprocessed data "in the wild." All of the above works make one or more of the following assumptions, which are untrue in our task. First, many works focus primarily on training and testing with synthetic sheet music. In our case, we are primarily working with digital scans of physical sheet music. Second, most works assume that the data has been cleaned and preprocessed in various ways. For example, it is commonly assumed that unrelated pages of sheet music have been removed. Many works further assume that each page has been segmented into lines, so that the data is presented as a sequence of image strips each containing a single line of sheet music. In our task, the raw PDF from IMSLP may contain unrelated movements, pieces, or filler pages like the title page or table of contents. We also obviously cannot assume that each page has already been segmented perfectly. Third, all of the above works assume that the music does not have any jumps or repeats. In our

Figure 1. Architecture of proposed system. The sheet music and audio are both converted into bootleg scores, and then aligned with the Hierarchical DTW algorithm.

task, we have to be able to handle common discontinuities like repeats, D.C. al coda, D.S. al fine, etc.

In attempting to build a system that can handle messy, real-world data, we discovered two things. First, we found that most of the above issues can be resolved to a reasonable degree by suitably combining existing tools in the MIR literature. However, we also discovered that a significant bottleneck in system performance was handling jumps and repeats. In particular, we found that a previously proposed Jump DTW alignment algorithm [15] does not yield satisfactory performance when jump locations are unknown a priori.

There are several existing offline algorithms for aligning two feature sequences in the presence of jumps or repeats. Jump DTW [15] is a variant of dynamic time warping (DTW) where additional long-range transitions are allowed in the cost matrix at potential jump locations. Mueller and Appelt [16] and Grachten et al. [17] also propose variants of DTW for partial alignment in the presence of structural differences. One limitation of these latter two works is that repeated sections are handled by simply skipping or deleting sections of features, so that the actual alignment of the repeated section is not known. Joder et al. [18] frame the alignment problem as a conditional random field with additional transitions inserted at known jump locations. Jiang et al. [19] use a modified Markov model that allows arbitrary jumps to follow a musician during a practice session with lots of do-overs and jumps. There are also several real-time score following algorithms that handle various types of jumps [20] [21] [22] [23], though our focus in this work is on the offline context. In this study, we primarily focus on Jump DTW as the closest match to our target scenario: it is an offline algorithm, targeted at performances rather than practice sessions, and it provides a complete estimated alignment in the presence of jumps.

The main conceptual contribution of this paper is a novel alignment algorithm called Hierarchical DTW. Unlike Jump DTW, it does not require knowledge of jump locations a priori, but instead considers every line transition as a potential jump location. The algorithm is called Hierarchical DTW because it first performs an alignment at the feature level with each sheet music line, and then uses the results to perform a second alignment at the segment level. By performing an alignment at the segment level, we can encode domain knowledge about which types of jumps

are likely. The algorithm is very simple and only has two hyperparameters, which both have very clear and intuitive interpretations. Through carefully controlled experiments on unprocessed PDFs from IMSLP, we show that Hierarchical DTW significantly outperforms Jump DTW on the piano score following video generation task. [1]

2. SYSTEM DESCRIPTION

Figure 1 shows a high-level overview of our proposed system. We will explain its design in three parts: feature extraction, alignment, and video generation.

2.1 Feature Extraction

The first step is to convert both the sheet music and audio into bootleg score representations. The bootleg score [24] is a recently proposed feature representation for aligning piano sheet music images and MIDI. For sheet music, it encodes the position of filled noteheads relative to the staff lines. The bootleg score itself is a $62 \times N$ binary matrix, where 62 indicates the total number of possible staff line positions in both the left and right hands, and where N indicates the total estimated number of simultaneous note events. For MIDI files, each note onset can be projected onto the bootleg score using the rules of Western musical notation. Ambiguities due to enharmonic representations or left-right hand attribution are handled by simply setting all possible positions to 1.
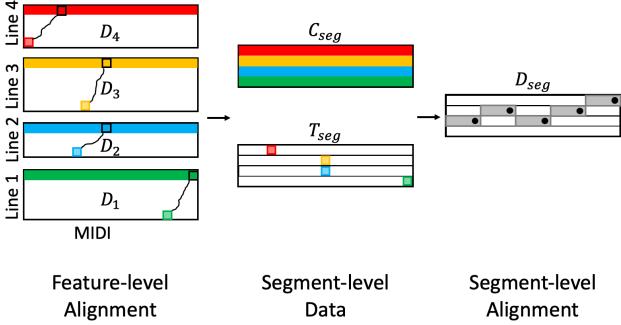
We computed the bootleg score representations in the following manner. We convert each PDF into a sequence of PNG images at 300 dpi, compute a bootleg score for each page, and then represent the entire PDF as a sequence of bootleg score fragments, where each fragment corresponds to a single line of music. Note that these fragments may include lines of music from other unrelated movements or pieces in the same PDF, or may even represent nonsense features coming from filler pages. Next, we transcribe the audio recording using the Onsets and Frames [25] automatic music transcription system, and then convert the estimated MIDI into its corresponding bootleg score. In this work, we treat the bootleg score computation and music transcription as fixed feature extractors.

2.2 Alignment

The second main step is to align the bootleg score representations. We propose a novel alignment algorithm called Hierarchical DTW to accomplish this task. Figure 2 shows an overview of the algorithm, which consists of three stages.

The first stage is to perform feature-level alignment. We do this using a variant of DTW called subsequence DTW, which finds the optimal alignment between a short query sequence and any subsequence within a reference sequence. We perform subsequence DTW between each sheet music bootleg score fragment (each corresponding to one line of music) and the entire MIDI bootleg score, as

---

[1] Code, data, and example score following videos can be found at https://github.com/HMC-MIR/YoutubeScoreFollowing.

Figure 2. Overview of Hierarchical DTW. Subsequence DTW is performed at the feature level on each sheet music line. The results are used to generate the segment-level data matrices, and then a second alignment is performed at the segment level. Only a few selected elements of T_{seg} are shown for illustration.

shown on the left side of Figure 2. [2] We use the normalized negative inner product distance metric proposed in [24] along with allowable transitions $\{(1,1),(1,2),(2,1)\}$ with weights $\{1,1,2\}$. For a more detailed explanation of subsequence DTW, we refer the reader to [26].

The second stage is to construct the segment-level data matrices. There are two matrices that need to be constructed. The first matrix is formed by taking the last row in every cumulative cost matrix D_i from stage 1 and stacking them into a matrix of size $L \times M$, where L indicates the total number of lines of music in the PDF and M indicates the total number of features in the MIDI bootleg score. This matrix contains subsequence path scores and is denoted as C_{seg} in Figure 2. It will play a role analogous to the pairwise cost matrix when we do dynamic programming at the segment level. The second matrix T_{seg} is the same size as C_{seg} and indicates allowable transitions at the segment level. Each element $T_{seg}[i,j]$ is computed by identifying the jth element in the last row of D_i, and then backtracking from this element to determine the beginning location of the matching path. $T_{seg}[i,j]$ thus indicates the starting location of the best matching path in the ith line of sheet music ending at position j in the MIDI bootleg score. In Figure 2, a few selected elements in T_{seg} are shown as colored boxes to illustrate this process. Note that, in order to construct T_{seg}, we need to backtrace from every possible location for every line of sheet music.

The third stage is to perform segment-level alignment. Here, we use dynamic programming to find the optimal path through C_{seg} using transitions in T_{seg}. We construct a segment-level cumulative cost matrix D_{seg} by filling out its entries column-by-column using dynamic programming. The first column of D_{seg} is initialized to all zeros, which ensures that the matching path can start on any line of music without penalty. Note that, unlike regular DTW where the set of allowable transitions and weights is the same at every location, here the set of allowable transitions and weights is different for each element of D_{seg}.

Since the transitions are all unique, we simply encode the previous location rather than the transition type (e.g. the previous location $(i-1, j-1)$ instead of the transition $(1,1)$). When computing $D_{seg}[i,j]$, there are two types of allowable transitions. The first type of transition is skipping elements. This means transitioning from $(i, j-1)$ and moving directly to the right by one position without accumulating any score. Here, the candidate path score is $D_{seg}[i,j] = D_{seg}[i, j-1]$. The second type of transition is matching the ith line of music (ending) at this position. In this case, we can transition from the end of any line of music immediately before the matching segment begins. If we let $k \triangleq T_{seg}[i,j]$ be the beginning of the matching subsequence path, then there are L different possible transitions from $(n, k-1), n = 0, \ldots, L-1$ where n indicates the line of music. Here, the candidate path scores are $D_{seg}[i,j] = D_{seg}[n, k-1] + w_{n,i} \cdot C_{seg}[i,j] + p_{n,i}$, where $w_{n,i}$ is a multiplicative weight and $p_{n,i}$ is an additive penalty for jumps. We can summarize the dynamic programming rules for the segment-level alignment as

$$k \triangleq T_{seg}[i,j] \tag{1}$$

$$D_{seg}[i,j] = min \begin{cases} D_{seg}[i, j-1] \\ D_{seg}[0, k-1] + w_{0,i} \cdot C_{seg}[i,j] + p_{0,i} \\ D_{seg}[1, k-1] + w_{1,i} \cdot C_{seg}[i,j] + p_{1,i} \\ \ldots \end{cases}$$
$$\tag{2}$$

where the minimum is calculated over all sheet music lines $n = 0, \ldots, L-1$. When filling out the entries of D_{seg} using dynamic programming, we also keep track of backtrace information in a separate matrix. Once D_{seg} has been constructed, we identify the element in the last column of D_{seg} with the lowest path score, and then backtrace from that position to determine the optimal alignment path. Figure 2 shows the optimal alignment path as a series of black dots and the induced segmentation of the MIDI bootleg score as gray rectangles.

The real power of Hierarchical DTW comes from setting $w_{n,i}$ and $p_{n,i}$ in an intelligent way that encodes musical domain knowledge. These values can be adapted to allow no jumps, allow arbitrary jumps, or anything in between. For example, disallowing jumps means setting $p_{n,i} = \infty \cdot \mathbb{1}(i \neq n+1)$. The system described below is one possible instantiation based on three assumptions: (a) the performed lines of music will form a contiguous block (e.g. we will not go from page 13 to 34 to 19), (b) backwards jumps (from repeats) are to lines of music we have seen before, and (c) forward jumps (from D.S. al fine) are to one line past the furthest line of music that has been seen before (which we refer to as the "leading edge"). For the allowed jump transitions, multiplicative weights are set to 1 and additive penalties are set to $-\gamma \cdot p_{avg}$, where γ is a hyperparameter and p_{avg} is the result of calculating the best subsequence path score for each line of sheet music and averaging the scores across all lines. So, if $\gamma = 1$, the jump penalty approximately offsets 1 line of matching music. Note that we can keep track of which lines have been seen

before by defining two matrices R_{lower} and R_{upper} which are the same size as C_{seg} and keep track of the range of lines that have been seen in the optimal path ending at any position (i, j). R_{lower} and R_{upper} can be updated along with D_{seg} and the backtrace matrix during the dynamic programming stage. For regular forward transitions, we allow moving to the next line, staying on the current line (slowing down), or skipping one line (speeding up). These three transitions have multiplicative weights 1, α, and α and additive penalties of 0 (all), respectively. We found that allowing additional time warping at the segment level with multiplicative penalty $\alpha = 0.5$ allows the algorithm to recover from large mistakes more easily.

Hierarchical DTW is simple yet flexible. The version described above only has two hyperparameters that correspond to a multiplicative penalty for speeding up/slowing down (α) and an additive penalty for jumps (γ). Yet, the framework of Hierarchical DTW makes it possible to selectively allow very specific types of jumps that obey common musical conventions.

2.3 Video Generation
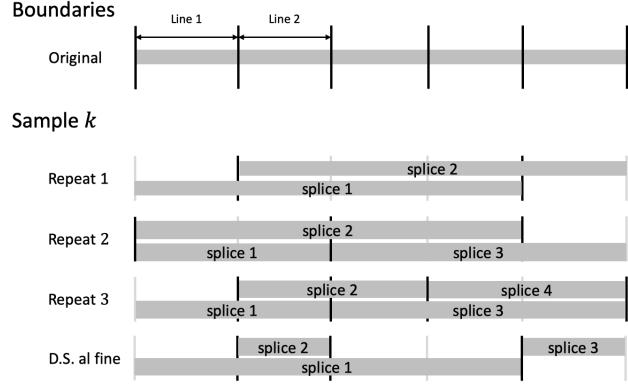
The third main step is to generate the score following video. In order to translate the predicted segment-level alignment into a score following video, we need additional auxiliary information from the bootleg score feature computation. For the audio recording, we need to keep track of the correspondence between each MIDI bootleg score feature column and its corresponding time in the audio recording. For the sheet music, we need to keep track of the correspondence between each sheet music bootleg score feature column and its corresponding page and pixel range in the sheet music images. We modified the original code provided in [24] to return this information, in addition to the bootleg score features. Given this auxiliary information and the predicted segment-level alignment, we can generate the score following video in a very straightforward manner: we simply show the predicted line of sheet music at every time instant in the audio recording.

3. EXPERIMENTAL SETUP

In this section, we explain the datasets and metrics used to evaluate our proposed system.

Our data is a derivative of the Sheet MIDI Retrieval dataset [24]. We will first describe the original dataset, and then explain how we used it to generate the data for this current work. The original dataset contains scanned sheet music from IMSLP for 200 solo piano pieces across 25 composers. The sheet music comes with manual annotations of how many lines of music are on each page, and how many measures are on each line. For each of the 200 pieces, there is a corresponding MIDI file and ground truth annotations of measure-level timestamps.
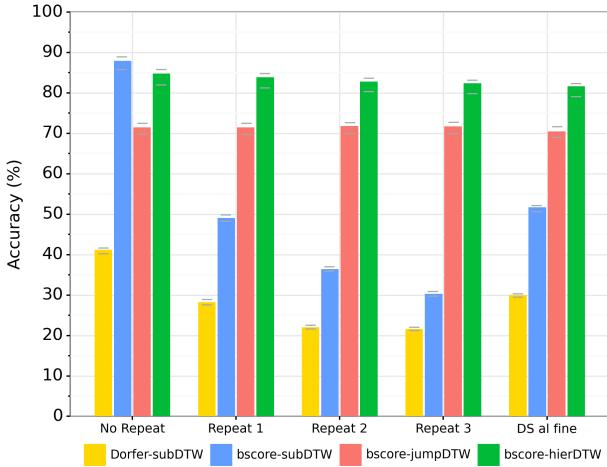
We derived our dataset in the following manner. We synthesize the MIDI files to audio using the FluidSynth library. By combining the sheet music and MIDI annotations, we determine the time intervals in the audio record-

Figure 3. Generating audio with repeats. The original audio recording is segmented by lines of sheet music. We sample k boundary points without replacement, and then splice and concatenate audio segments to generate the data with repeats.

ing that correspond to each line of sheet music. For each sheet music PDF in the Sheet MIDI Retrieval dataset, we retrieved the original PDF from the IMSLP website. The only difference between these two files is that the original IMSLP PDF contains other unrelated movements, pieces, and filler pages that were removed during the preparation of the Sheet MIDI Retrieval dataset. For example, one PDF in the test set contains 127 pages, of which only 17 correspond to the piece of interest. Because we want to test how well our system handles this type of noise, we use the original PDF with no preprocessing or data cleaning whatsoever. We augmented the sheet music annotations by converting the original IMSLP PDFs into PNG files at 300 dpi and manually annotating the vertical pixel range for every line of sheet music played in the audio recording. This required annotating a total of 1090 pages with $11,556$ pixel positions. By combining all of our annotations together, we can determine the page and pixel range of the line of sheet music that is currently being played at every point in the audio recording. In total, there are 13.0 hours of annotated audio. Because there are no repeats or jumps in the sheet music, we call this data the "No Repeat" dataset.

We also generate several synthetic datasets to test how well our system handles jumps and repeats. The process of generating a synthetic dataset consists of three steps, as shown in Figure 3. The first step is to identify the $L + 1$ boundary positions of the L lines of sheet music that are played in the audio recording. The second step is to randomly sample k boundary points without replacement. The value of k depends on the types of jumps we want to simulate. In this work, we consider four schemas: 1 repeat ($k = 2$), 2 repeats ($k = 3$), 3 repeats ($k = 4$), and D.S. al fine ($k = 3$). The third step is to splice and concatenate the audio to generate a modified audio recording as shown in Figure 3. Note that all of the synthetic datasets have the exact same sheet music, but their audio recordings have been spliced to reflect the desired schema. Since the process of sampling is random, we generate five different samples for every audio recording. The four syn-

Figure 4. Comparison of system performance on benchmarks with various types of jumps. The bar levels indicate accuracy with a scoring collar of 0.5 sec. The short gray lines indicate accuracy with scoring collars of 0 and 1.0 seconds.

thetic datasets described above have 84, 94, 100, and 81 hours of audio, respectively. The ground truth annotations are modified accordingly.

We evaluate system performance using a simple accuracy metric. Because our goal is to generate score following videos, we want to use an evaluation metric that correlates with user experience. The accuracy simply indicates the percentage of time that the correct line of music is being shown to the user. When calculating accuracy, we use a scoring collar, in which small intervals $(t_i - \Delta t, t_i + \Delta t)$ around the ground truth transition timestamps t_i are ignored during scoring. This is a standard practice in evaluating time-based segmentation tasks like speech activity detection [27]. By using a range of scoring collar values, we can also gain insight into what fraction of our errors occur very close to the transition boundaries.

For all experiments, we use (the same) 40 pieces for training and 160 pieces for testing. This results in 160 test queries for the No Repeat benchmark (10.6 hours of audio) and $160 \times 5 = 800$ test queries for the benchmarks with jumps (69.2, 76.9, 81.8, and 66.1 hours). Since we treat the bootleg score computation and automatic music transcription as fixed feature extractors, our system has no trainable weights and only 2 hyperparameters (α, γ). So, we only use a small fraction of the data for developing the algorithm, and we reserve most of the data for testing.

4. RESULTS

In this section, we present our experimental results on the piano score following video generation task.

We compare our proposed system to three other baseline systems. The first baseline system ('bscore-subDTW') is identical to our proposed system in Figure 1 except that it replaces the Hierarchical DTW with a simple subsequence DTW. The second baseline system ('bscore-jumpDTW') is also identical to our proposed system except that it replaces

the Hierarchical DTW with Jump DTW [15]. Because Jump DTW was designed to handle jumps and repeats, we expect this system to provide the most competitive baseline results. The third baseline system ('Dorfer-subDTW') is based on Dorfer et. al [9]. This system approaches the audio–sheet music alignment task by training a multimodal CNN to project chunks of sheet music and chunks of audio spectrogram into the same feature space where similarity can be computed directly. We used the pretrained CNN provided in [9] as a feature extractor, and then apply subsequence DTW. Finally, our proposed Hierarchical DTW system is indicated as 'bscore-hierDTW.'

Figure 4 shows the results of these four systems. The histogram bars indicate the accuracies with a scoring collar of $\Delta t = .5$ sec. There are four things to notice about these results. First, the Dorfer-subDTW system performs poorly on all benchmarks. This indicates that this system does not generalize well to the scanned sheet music from IMSLP. Second, the bscore-subDTW system performs well on the No Repeat benchmark (87.9% accuracy), but performs poorly on all other benchmarks (e.g. 30.3% on the Repeat 3 benchmark). This is to be expected, since subsequence DTW cannot handle jumps and repeats. Third, Jump DTW is significantly worse than subsequence DTW on the No Repeat benchmark (71.5% vs. 87.9%), but it has consistent performance across benchmarks with repeats and jumps (71.5%, 71.8%, 71.7%, and 70.5%). This indicates that Jump DTW is able to cope with discontinuities, but with a significant cost in performance. Fourth, the Hierarchical DTW system is only slightly worse than subsequence DTW on the No Repeat benchmark (84.8% vs. 87.9%), and its performance decreases only slightly on the other benchmarks (83.9%, 82.8%, 82.4%, 81.6%). We can see that the Hierarchical DTW system consistently outperforms Jump DTW by 10-13% across all benchmarks. These results indicate that Hierarchical DTW is able to handle repeats and jumps reasonably well, and with a much smaller performance cost than Jump DTW.

5. ANALYSIS

In this section, we conduct two different analyses to gain more insight into system behavior.

5.1 Failure Modes

The first analysis answers the question, "What are the failure modes for each system?" To answer this question, we identified the individual queries that had the poorest accuracy, and then investigated the reasons for the errors.

The Dorfer system has two primary failure modes. The first failure mode is that the system is not designed to handle jumps, so it performs very poorly on any datasets with jumps or repeats. Note, however, that this system also performs poorly on the No Repeat benchmark. When we investigated the reasons for this, we discovered the second major failure mode: page segmentation. The sub-system for segmenting each page into lines of music performed very poorly on many pages in the dataset. This is perhaps
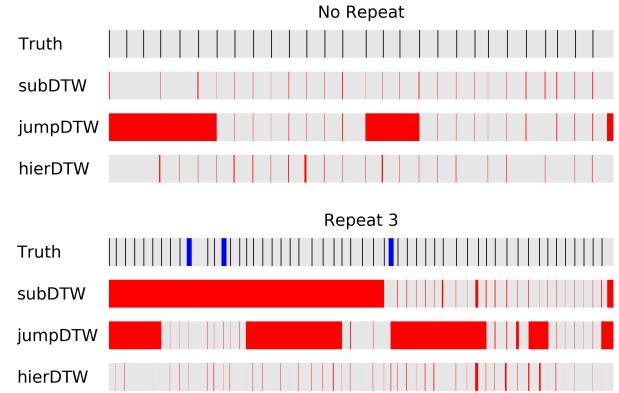
not surprising, since the original system was developed and trained on synthetic sheet music, where staff lines are perfectly horizontal. In this case, the assumptions in this work do not translate well to our task of working with IMSLP scanned sheet music.

The subsequence DTW system also has two primary failure modes. The first is (again) that the system cannot handle jumps or repeats. When we investigated the reasons for major errors on the No Repeat benchmark, we find that the failures primarily come from mistakes in the bootleg score representation. The bootleg score does not account for octave markings or clef changes, and it does not detect non-filled noteheads (e.g. half or whole notes). When there are long stretches of sheet music that contain several of these elements at the same time, the bootleg score is a poor representation of the sheet music. For example, three of the pieces in the test set are Erik Satie's Gymnopedies, where the sheet music is almost entirely non-filled noteheads. These pieces had close to 0% accuracy and caused a decrease of several percentage points on the aggregate accuracy score.

The JumpDTW system has one primary failure mode: it often jumps to incorrect lines of music. This occurs when either (a) there are similar lines of music in multiple places (e.g. the recapitulation of a theme), or (b) significant bootleg score errors cause the system to match random lines of music elsewhere in the piece. This problem is most clearly seen in the No Repeat benchmark, where it often takes jumps when none are present.

The Hierarchical DTW system has two primary failure modes. The first failure mode is prolonged bootleg score failures, which cause the algorithm to insert spurious small jumps. Once the bootleg score becomes an accurate representation again, the system is usually able to recover. The second failure mode is when the sheet music contains very repetitive measures and lines. This problem is particularly bad when the sheet music is very short (e.g. 2-3 pages long) and has jumps or repeats.

Figure 5 shows a visualization tool for diagnosing failure modes. The top half of Figure 5 shows four gray strips, each representing the duration of a single audio recording in the No Repeat benchmark. The topmost strip contains black vertical lines indicating the location of the ground truth sheet music line transitions. The three strips below it show the predictions of the subsequence DTW, Jump DTW, and Hierarchical DTW systems, where errors are shown in red. The bottom half of Figure 5 shows the same information for a query in the Repeat 3 benchmark. The location of the jumps are indicated with blue vertical lines. We can see many of the failure modes described above. For example, Jump DTW has spurious jumps in both queries but is able to follow two of the repeats in the bottom query. Subsequence DTW is unable to handle the jumps in the bottom query, but matches well after the last jump occurs. Finally, we can see that the Hierarchical DTW system is able to follow the correct sequence of sheet music lines, and its errors primarily occur close to line transitions.

Figure 5. Visualizing system predictions for a query with no repeats (top half) and a query with three repeats (bottom half). Black lines show ground truth line transitions, red regions indicate errors, and blue lines show repeats.

5.2 Error Locations

The second analysis answers the question, "Where are the errors located?" One way we can answer this question is to calculate system performance across a range of values for the scoring collars. This can tell us how close the errors are to line transition boundaries. Figure 4 shows the results of each system with various scoring collar values. The histogram bar level indicates the default scoring collar $\Delta t = .5$ sec, and the results with Δt set to 0 sec and 1.0 sec are shown as short horizontal gray lines directly below and above the histogram bar level, respectively. Note that as Δt increases, the accuracy will increase monotonically.

There are two things to notice about the results with various scoring collars. First, we see that even with a generous scoring collar of $\Delta t = 1$ sec, the accuracies of all systems only increase about 1-2%. This indicates that most of the errors are not slight misalignments at the line transitions, but are instead large errors due to total alignment failures. Second, we observe that the results with Hierarchical DTW on benchmarks with jumps is only marginally worse than the No Repeat benchmark. This indicates that Hierarchical DTW is able to handle discontinuities reasonably well. Combining these two observations, the failures in the bscore-hierDTW system seem to primarily come from large misalignments due to prolonged bootleg score failures. This strongly suggests that the performance bottleneck is the bootleg score representation, not the Hierarchical DTW alignment.

6. CONCLUSION

We present a method for audio-sheet image alignment that combines a bootleg score representation with a novel alignment algorithm called Hierarchical DTW, which performs alignment at both the feature-level and the segment-level in order to handle repeats, jumps, and unknown offset in the sheet music. We show that Hierarchical DTW significantly outperforms Jump DTW in handling jumps and repeats on unprocessed sheet music.

7. REFERENCES

[1] D. Damm, C. Fremerey, F. Kurth, M. Müller, and M. Clausen, "Multimodal presentation and browsing of music," in *Proc. of the International Conference on Multimodal Interfaces (ICMI)*, 2008, pp. 205–208.

[2] F. Kurth, M. Müller, C. Fremerey, Y. Chang, and M. Clausen, "Automated synchronization of scanned sheet music with audio recordings," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2007, pp. 261–266.

[3] V. Thomas, C. Fremerey, M. Müller, and M. Clausen, "Linking sheet music and audio – challenges and new approaches," in *Multimodal Music Processing*, 2012, vol. 3, pp. 1–22.

[4] C. Fremerey, M. Müller, F. Kurth, and M. Clausen, "Automatic mapping of scanned sheet music to audio recordings," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2008, pp. 413–418.

[5] C. Fremerey, M. Clausen, S. Ewert, and M. Müller, "Sheet music-audio identification," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2009, pp. 645–650.

[6] M. Dorfer, A. Arzt, and G. Widmer, "Towards end-to-end audio-sheet-music retrieval," in *Neural Information Processing Systems (NIPS) End-to-End Learning for Speech and Audio Processing Workshop*, 2016.

[7] M. Dorfer, J. Schlüter, A. Vall, F. Korzeniowski, and G. Widmer, "End-to-end cross-modality retrieval with cca projections and pairwise ranking loss," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 117–128, 2018.

[8] M. Dorfer, A. Arzt, and G. Widmer, "Learning audio-sheet music correspondences for score identification and offline alignment," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2017, pp. 115–122.

[9] M. Dorfer, J. Hajič, A. Arzt, H. Frostel, and G. Widmer, "Learning audio-sheet music correspondences for cross-modal retrieval and piece identification," *Trans. of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 22–33, 2018.

[10] M. Dorfer, A. Arzt, S. Böck, A. Durand, and G. Widmer, "Live score following on sheet music images," in *Late Breaking Demos at the International Conference on Music Information Retrieval (ISMIR)*, 2016.

[11] M. Dorfer, A. Arzt, and G. Widmer, "Towards score following in sheet music images," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2016, pp. 789–795.

[12] M. Dorfer, F. Henkel, and G. Widmer, "Learning to listen, read, and follow: Score following as a reinforcement learning game," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2018, pp. 784–791.

[13] F. Henkel, S. Balke, M. Dorfer, and G. Widmer, "Score following as a multi-modal reinforcement learning problem," *Trans. of the International Society for Music Information Retrieval*, vol. 2, no. 1, pp. 67–81, 2019.

[14] M. Müller, A. Arzt, S. Balke, M. Dorfer, and G. Widmer, "Cross-modal music retrieval and applications: An overview of key methodologies," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 52–62, 2019.

[15] C. Fremerey, M. Müller, and M. Clausen, "Handling repeats and jumps in score-performance synchronization," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2010, pp. 243–248.

[16] M. Müller and D. Appelt, "Path-constrained partial music synchronization," in *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2008, pp. 65–68.

[17] M. Grachten, M. Gasser, A. Arzt, and G. Widmer, "Automatic alignment of music performances with structural differences," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 607–612.

[18] C. Joder, S. Essid, and G. Richard, "A conditional random field framework for robust and scalable audio-to-score matching," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2385–2397, 2011.

[19] Y. Jiang, F. Ryan, D. Cartledge, and C. Raphael, "Offline score alignment for realistic music practice," in *Sound and Music Computing Conference*, 2019.

[20] T. Nakamura, E. Nakamura, and S. Sagayama, "Real-time audio-to-score alignment of music performances containing errors and arbitrary repeats and skips," *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 24, no. 2, pp. 329–339, 2015.

[21] A. Arzt and G. Widmer, "Towards effective 'any-time' music tracking," in *Proc. of the Starting AI Researchers' Symposium*, 2010.

[22] A. Arzt, G. Widmer, and S. Dixon, "Automatic page turning for musicians via real-time machine listening," in *Proc. of the European Conference on Artificial Intelligence (ECAI)*, 2008, pp. 241–245.

[23] B. Pardo and W. Birmingham, "Modeling form for on-line following of musical performances," in *Proc. of the National Conference on Artificial Intelligence*, vol. 20, no. 2, 2005, pp. 1018–1023.

[24] D. Yang, T. Tanprasert, T. Jenrungrot, M. Shan, and T. Tsai, "Midi passage retrieval using cell phone pictures of sheet music," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 916–923.

[25] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2018, pp. 50–57.

[26] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[27] *NIST Open Speech-Activity-Detection Evaluation Plan*, 2016 (accessed May 6, 2020), https://www.nist.gov/system/files/documents/itl/iad/mig/Open_SAD_Eval_Plan_v10.pdf.

ZERO-SHOT SINGING VOICE CONVERSION

Shahan Nercessian

iZotope, Inc.

`shahan@izotope.com`

ABSTRACT

In this paper, we propose the use of speaker embedding networks to perform zero-shot singing voice conversion, and suggest two architectures for its realization. The use of speaker embedding networks not only enables the capability to adapt to new voices on-the-fly, but also allows for model training on unlabeled data. This not only facilitates the collection of suitable singing voice data, but also allows networks to be pretrained on large speech corpora before being refined on singing voice datasets, improving network generalization. We illustrate the effectiveness of the proposed zero-shot singing voice conversion algorithms by both qualitative and quantitative means.

1. INTRODUCTION

Singing voice conversion (SVC) is the transformation of a singing performance from one vocalist to that of another. It can be used for creative manipulations of the voice that go far beyond traditional time stretching and pitch/formant shifting [1]. SVC methods must learn to disentangle speaker content from acoustic features [2], while accurately preserving input phonetic and pitch information in the converted output. Relative to similar methods applied to speech, the singing voice exhibits a larger pitch range and generally slower transitions between phonetic units, which conversion networks must be able to accommodate for [3, 4].

Most approaches to SVC rely on some form of vocoder which synthesizes vocal waveforms. The SVC task then becomes one of transforming vocoder features from a performance of a source singer to that of some target voice. Unlike approaches to voice conversion on speech, which usually leverage neural vocoders such as WaveNet [5] or WaveRNN [6] as their back-end speech synthesizer, SVC and singing synthesis algorithms tend to use hand-designed vocoders such as WORLD [7] for acoustic modeling and synthesis of the voice (with some exceptions, as in [8]). This is because they explicitly separate pitch from timbral components [3, 4, 9]. Accordingly, it is possible to learn timbral transformations while preserving pitch, which is not usually guaranteed when using neural vocoder [2].

This may come at the expense of reduced expressivity relative to neural vocoders, but is considered to be acceptable given its pitch-preserving characteristics [4].

Generative Adversarial Networks (GANs) [3, 9, 10] and Variational Autoencoders (VAEs) [11] have become popular choices for learning transformations of vocoder features for both SVC and singing synthesis. Different strategies have been investigated to model several target voices, and specifically, to adapt systems for new voices not seen during model training. One such strategy involves assigning a random embedding to the unseen voice, and resuming model training on data of the unseen voice so as to update this embedding and perform any necessary refinements to the model [12, 13]. More recently, conversion algorithms in the speech domain have used pretrained speaker embedding networks designed for speaker verification tasks [14] in order to encode speaker identity [15]. These approaches have the advantage that, upon having trained the speaker embedding network on many speakers, conversion algorithms can be adapted to new voices in a zero-shot manner, requiring no further training of the model and with as few as one sample of an unseen voice.

In this paper, we adapt zero-shot voice conversion methodologies [15] utilizing speaker embedding networks for the application of SVC. We use the WORLD vocoder and suggest two architectures for carrying out zero-shot SVC. We show that the zero-shot nature of the algorithm allows for SVC on unlabeled data. Moreover, we posit that SVC systems are amenable to initial training on large speech datasets which are more widely available, followed by model adaptation on smaller singing voice datasets. To the best of our knowledge, this is the first work to tackle zero-shot SVC. Unlike singing synthesis algorithms, such as [4, 10, 13], it does not require predefined annotations of phonetic transitions or pitch, as this information is extracted from acoustic features of the source performance.
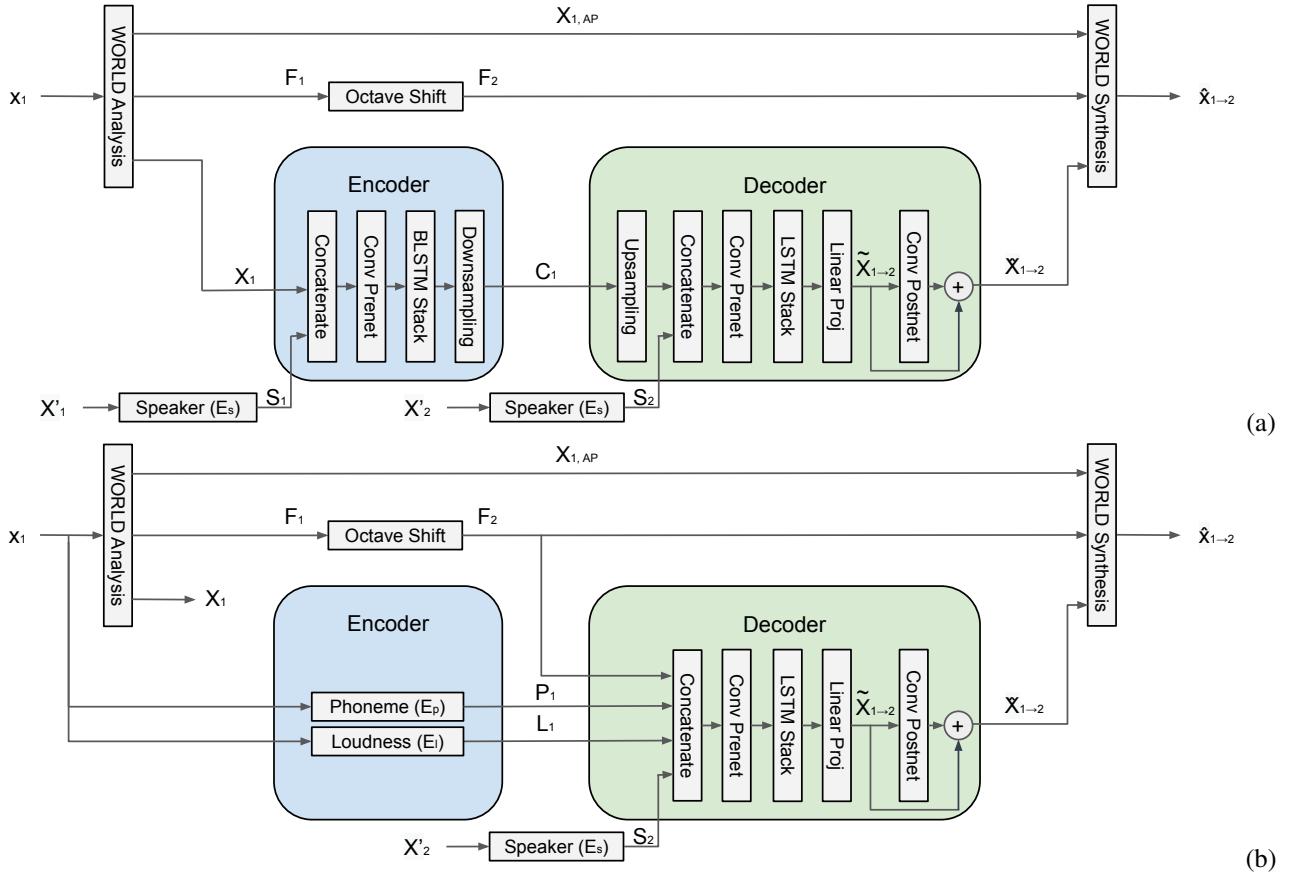
The remainder of this paper is structured as follows: We suggest architectures for zero-shot SVC in Section 2. We evaluate model performance via qualitative and quantitative means in Section 3. Lastly, we draw conclusions and allude to future work in Section 4.

2. SVC ALGORITHMS

We use the WORLD vocoder for analysis and synthesis of singing voices due to its ability to separate timbral and pitch components. Specifically, the system decomposes a vocal signal into a harmonic spectral envelope and an aperiodicity envelope, based on a tonality-gated estimate

Figure 1. (a) Adapted AutoVC and (b) fixed encoder network architectures for zero-shot SVC.

of fundamental frequency. The conversion task primarily involves transformation of the harmonic spectral envelope, leaving the aperiodicity envelope unchanged. As in [4], we reduce the dimensionality of harmonic spectral envelopes to 60 coefficients at each time step, using truncated frequency warping in the cepstral domain with an allpole warping coefficient $\alpha = 0.45$ [16]. We consider two different architectures for SVC, as illustrated in Figure 1, drawing inspiration from [2, 15, 17, 18].

2.1 AutoVC

The first architecture is an adaptation of the AutoVC architecture [15] for singing voice, which operates on harmonic spectral envelopes extracted from WORLD (instead of Mel spectrograms which are ultimately fed into a WaveNet vocoder as in the original work). It is comprised of a speaker embedding network $E_s(\cdot)$ which takes as input a Mel spectogram and generates a single fixed-dimensional speaker embedding, a content encoder $E(\cdot)$ which takes as input the harmonic spectral envelope and speaker embedding from a source utterance and generates a latent encoding, and a decoder network $D(\cdot)$ which constructs the converted harmonic spectral envelope from a latent encoding and target speaker embedding.

The input to the encoder is the harmonic spectral envelope X_1 computed from a source utterance x_1. This is concatenated with a source speaker embedding $S_1 = E_s(X'_1)$ at each time step, where X'_1 is a Mel spectrogram of the

same or potentially different utterance x'_1 from the same source speaker. The encoder consists of a convolutional prenet, comprised of three 1D convolutional layers with 512 output channels and kernel size 5, each followed by batch normalization and ReLU activation. This result is passed through two bidirectional LSTM layers with forward and backward cell dimensions of 32, yielding an encoding of dimension 64. This is temporally downsampled by 32, yielding the content encoding C_1. The inclusion of S_1 in the encoder network helps the encoder to more easily learn a speaker-independent encoding.

The decoder begins by upsampling the latent encoding C_1 to its original temporal resolution. Given the Mel spectrogram X'_2 of some utterance x'_2 from the same target speaker as the target utterance x_2, the speaker embedding $S_2 = E_s(X'_2)$ is concatenated with the upsampled encoding. The concatenated features pass through a convolutional prenet similar to that in the encoder, followed by three LSTM layers with cell dimension 1024. The outputs of the LSTM layer are linearly projected to dimension 60, serving as an initial estimate $\tilde{X}_{1\rightarrow 2}$ of the converted harmonic spectral envelope. This initial estimate is refined by means of a convolutional postnet consisting of five 1D convolutional layers of kernel size 5. Batch normalization and Tanh are applied to the first four layers, and they each output 512 channels. The final layer applies no activation and outputs 60 channels. The converted harmonic spectral envelope $\hat{X}_{1\rightarrow 2}$ is produced by adding the output of the

postnet to $\tilde{X}_{1\rightarrow2}$.

During training, we set $x_1 = x_2$, $S_1 = S_2$, and accordingly, $X_1 = X_2$, $\tilde{X}_{1\rightarrow1} = \tilde{X}_{1\rightarrow2}$, and $\hat{X}_{1\rightarrow1} = \hat{X}_{1\rightarrow2}$. The objective function used for training AutoVC is

$$\begin{aligned}\mathcal{L} =& \mathrm{E}[|X_1 - \hat{X}_{1\rightarrow1}|_2^2] + \\ & \mu\mathrm{E}[|X_1 - \tilde{X}_{1\rightarrow1}|_2^2] + \\ & \lambda\mathrm{E}[|E(X_1, S_1) - E(\hat{X}_{1\rightarrow1}, S_1)|_1]\end{aligned} \qquad (1)$$

The first term is the reconstruction loss between the original and reconstructed harmonic spectral envelopes. The second term is a reconstruction loss between the original and initially estimated harmonic spectral envelopes, which empirically helps model convergence. The third term is a latent regressor loss [19] penalizing differences in encodings between the original and converted harmonic spectral envelopes. In practice, hyperparameters μ and λ can be set to 1 [15]. The model is trained as an autoencoder, with the hope that its bottleneck will be small enough to disentangle speaker identity but large enough to allow for an accurate reconstruction.

During inference, S_2 can be set to the speaker embedding of some target singer to perform a conversion. Given a source pitch contour F_1 extracted during WORLD analysis, the target pitch contour F_2 should be adjusted to accommodate the register of the target singer, and therefore, $F_2 = F_1 + F_{\Delta1\rightarrow2}$. The pitch shift $F_{\Delta1\rightarrow2}$ can be determined automatically by measuring the median pitches of source and target performances, and taking their difference rounded to the nearest octave. The aperiodicity spectral envelope of the source performance $X_{1,AP}$ is used as is. The converted audio waveform $\hat{x}_{1\rightarrow2}$ is computed by feeding the transformed harmonic spectral envelope, source aperiodicity spectral envelope, and target pitch contour F_2 as input to the WORLD synthesis engine.

2.2 Fixed encoder model

The second architecture is similar to AutoVC, but replaces the encoder $E(\cdot)$ with a number of conditioning signals, such as those found in [2]. By design, these conditioning signals encode the input in a speaker-independent way using explicit features, akin to the timbre transfer networks in [18]. We capture linguistic content using phonetic posteriorgrams (PPGs) extracted from a phoneme classifier $E_p(\cdot)$, as in [17]. The classifier passes 40 Mel frequency cepstral coefficients (MFCCs) per frame through two bidirectional LSTMs with 128 units per direction. A final dense layer with softmax activation yields the classifier output, which is compared to ground truth labels using a categorical cross-entropy loss during training. We trained the network on the TIMIT dataset [20], using its prescribed training and test sets. The dataset consists of audio and sample level timestamps of phonetic transitions from one of 61 classes (including a silence class). The output of the phoneme classifier is, therefore, a 61-dimensional vector at each time frame. The classification accuracy on the test set is 65%, which is found to be sufficient to act as a speaker-independent representation of linguistic content.

We extract loudness information (L) using the computational steps $E_l(\cdot)$ as in [21]. We compute an A-weighted power spectrum, which puts greater emphasis on higher frequencies. The result is aggregated across all frequencies and converted to decibels to produce a loudness value (in dbA) at each time step. Lastly, we include the target pitch contour F_2.

The decoder concatenates the target pitch contour F_2, $P_1 = E_p(x_1)$, $L_1 = E_l(x_1)$ with the target speaker embedding S_2. The inclusion of these different conditioning signals attempts to account for timbral changes which may vary as a function of the pitch and dynamics of a particular performance, while instructing the decoder of its underlying linguistic content. The decoder network is almost identical to that in AutoVC, except that we remove the upsampling operation as we no longer need to construct an information bottleneck for speaker disentanglement. We refer to this architecture as the fixed encoder model, because all conditioning signals are either computed without a neural network, or using a pretrained neural network whose weights are frozen during the training of the decoder network. The training objective is similar to that in Eqn. (1), except that the third term is no longer applicable and is therefore removed. Note that in this case, the source harmonic spectral envelope X_1 is never actually passed as input to the network, but is used as a target for reconstruction during training.

2.3 Architecture comparisons

We notionally discuss the potential advantages and disadvantages associated with the architectures proposed here. The main advantage of the AutoVC architecture is that it does not rely on a dedicated training set for extracting phonetic information. This information is learned by the encoder itself during model training. This could potentially have better implications for cross-lingual applications, in the case that the set of phoneme labels of a dataset itself introduces a language bias [22]. It does, however, incur some risk, as the encoder is solely responsible for learning all timbral variations in vocalization. It also requires a temporal downsampling/upsampling of its encoding to create an information bottleneck for speaker disentanglement, which has some additional latency implications in the decoder. The fixed encoder architecture is computationally less intensive, as the phoneme classifier is substantially smaller than the encoder network in AutoVC. It also avoids the need for temporal downsampling/upsampling. The main disadvantages of this architecture is the reliance on data to train a phoneme classifier, as well as the fact that its expressivity is limited to that provided by its conditioning signals.

2.4 Universal Background Model (UBM)

While we could simply train SVC networks "from scratch" on singing voice datasets, we consider leveraging the interesting fact that the use of speaker embeddings for encoding vocal identity (instead of one-hot labels) allows the system to be trained on unlabeled data. Arguably, any "clean"

speech or singing voice clip could now be used for training SVC systems. It is generally understood that there is significantly more speech data than their is singing voice data for research purposes. Borrowing nomenclature from the speech recognition community, an initial pretraining on large speech corpora is like training a UBM [23] from which other networks can be adapted for the more specific SVC task. We would hope that such a model would serve as a better initial condition for training a SVC network than random weights, and that the resulting system would at the very least generalize to more voices.

3. EXPERIMENTAL RESULTS

3.1 Experimental setup

Two datasets are used for training conversion networks in this work. We use the VCTK corpus, which consists of over 40 hours of speech from 109 speakers [24]. This corpus serves as both a supervised speaker dataset to compare performance between supervised and (unsupervised) zero-shot networks, as well as a sufficiently large dataset for training a UBM for further model fine tuning. As in [15], we retain 90% of the data of each speaker for training, and save the remainder as a test set. Additionally, we use a proprietary and unlabeled dataset consisting of 7 hours of singing voice data, which we simply call the SVC dataset. Again, we retain 90% of the data for training, and save the remainder as a test set. Note that the lack of labels in this dataset poses no problem for zero-shot network training.

We make use of an open-source speaker embedding network [1] pretrained to minimize the Generalized End-to-End Loss [14]. This speaker embedding network generates a 256-dimensional speaker embedding from a 40-band Mel spectrogram using an LSTM architecture and retaining only the output from the final time step. During training, we use an entire utterance for x'_1, whereas x_1 is a 2 second cut from the same utterance. The speaker embedding network and the phoneme classifier are pretrained and frozen during the training of the conversion networks.

All models operate at 16 kHz with a frame rate of 12.5 ms, and were trained with a batch size of 2 using the ADAM optimizer and a learning rate of 10^{-3}. We train four configurations for each model architecture described here. The first configuration, VCTK (one-hot), is trained on the VCTK corpus using the labels provided by the dataset, which are converted to a one-hot representation and fed as S_1 to the network. This configuration serves as a baseline to compare against its zero-shot counterpart. The second configuration, VCTK (zero-shot), is trained on the VCTK corpus using speaker embeddings for S_1. The first two configurations are each trained for 150,000 steps. In the third configuration, SVC (zero-shot), we train zero-shot architectures on the SVC dataset for 500,000 steps. In the final configuration, VCTK→SVC (zero-shot), the second configuration is used as an initial state, and training is resumed for 350,000 steps on the SVC dataset (for a total of 500,000 steps). For audio examples, please visit the

demo site [2] associated with this paper.

3.2 Performance assessment

We assess networks by both qualitative and quantitative means. The main goal of this paper is to illustrate that speaker embeddings networks can indeed be utilized for training zero-shot SVC networks. Since we are unaware of any other published methods for zero-shot SVC such as the ones introduced here, and in order to provide some form of comparative analysis, we focus our attention to analyzing differences in results between the training configurations outlined here. For our quantitative evaluation, we report the reconstruction loss for each network (the first term in Eqn. (1)), which when computed on harmonic spectral envelopes, effectively serves as a Mel cepstral distortion metric. For our qualitative evaluation, we conducted surveys with 15 participants within our organization who have some critical listening experience, and tabulated mean opinion scores (MOS). We conduct separate surveys for overall conversion quality and on similarity to the target voice. While we provide examples from both architectures in the supplementary material of this work, we restrict ourselves to samples generated from training variants of the fixed encoder architecture for subjective evaluations. The first reason for this restriction is simply to minimize the number of listening options so as not to overwhelm participants taking part in the survey. The second reason is because the inclusion of one-hot speaker labels for S_1 in the encoder network of AutoVC would require that input source samples come from its closed speaker set. Therefore, it is not practically possible to use the VCTK (one-hot) training configuration on AutoVC on singing voice examples without removing S_1 from the network, leading to a potentially unfair comparison.

The results of our quantitative analysis assessed on both the VCTK and SVC datasets are shown in Tables 1 and 2, respectively. Across both architectures, we can confirm that the replacement of one-hot labels with speaker embeddings does not dramatically hurt conversion performance.

	AutoVC	Fixed Encoder
VCTK (one-hot)	0.1837	**0.1882**
VCTK (zero-shot)	**0.1634**	0.1891
SVC (zero-shot)	0.2930	0.3590
VCTK→SVC (zero-shot)	0.2557	0.3232

Table 1. Reconstruction loss on the VCTK test set.

	AutoVC	Fixed Encoder
VCTK (one-hot)	N/A	N/A
VCTK (zero-shot)	0.3007	0.4314
SVC (zero-shot)	0.1650	0.1959
VCTK→SVC (zero-shot)	**0.1439**	**0.1850**

Table 2. Reconstruction loss on the SVC test set.

---

[1] https://github.com/CorentinJ/Real-Time-Voice-Cloning.

[2] https://sites.google.com/izotope.com/ismir2020-audio-demo

In fact, we see that for the AutoVC architecture, VCTK (zero-shot) actually outperforms VCTK (one-hot), while offering the added functionality of zero-shot adaptation to new unseen voices. This result is consistent with the findings in [15]. We note that there is a significant degradation in performance assessed quantitatively when applying either VCTK (zero-shot) directly to singing voice samples, or when applying SVC networks directly to VCTK samples, highlighting that there is indeed a mismatch between the speech and singing voice domains. There is a consistent improvement when using our proposed adaptation strategy, with VCTK→SVC (zero-shot) outperforming SVC (zero-shot), both in the speech domain and more importantly, in the singing voice domain of interest. Overall, the best performing approach for singing voice based on this quantitative evaluation is AutoVC trained using VCTK→SVC (zero-shot) training configuration, though the computationally lighter, fixed encoder model does perform comparably well. It is worth noting that the VCTK (one-hot) configuration is not applicable for evaluation on the SVC dataset as it does not have the immediate ability to adapt to new voices.

The results of our qualitative analysis, converting singing voice performances using target voices from both the VCTK and SVC test sets are shown in Tables 3 and 4, respectively. First and foremost, we observe that speaker embeddings networks can, in general, be used for zero-shot SVC. We note that conversion networks trained on speech can be used on singing voice, but they have some trouble maintaining consistent spectral envelopes over prolonged vowels. Lastly, while not formally a part of the subjective evaluation, we informally observe comparable performance between architectures, with a preference towards one architecture over the other on a per-case basis.

With target voices from VCTK, there is no remarkable difference between networks trained using one-hot speaker labels or using zero-shot speaker embeddings, but the latter naturally allows adaptation to new voices. While SVC (zero-shot) is trained to be adapted to properties of singing voice, it is trained on less data and has been exposed to fewer voices. Though it was able to generate voices resembling the VCTK target voices due to its zero-shot nature, and worked comparably to other methods, it understandably received the lowest MOS in this case. The networks trained on the SVC dataset are more successful when using target voices from the SVC test set (and again, are better adapted to the time scale of phonetic transitions in singing). In this case, there is some degradation in quality for the system trained using the VCTK (zero-shot) configuration, and the VCTK (one-hot) configuration is not even applicable. We again see an improvement for networks trained using VCTK→SVC (zero-shot) relative to SVC (zero-shot) in this scenario. In fact, the VCTK→SVC (zero-shot) training configuration outperforms other methods in terms of overall quality for both VCTK and SVC target voices. The VCTK (zero-shot) and VCTK→SVC (zero-shot) training configurations are the top performers in terms of voice similarity for VCTK and SVC target

	Quality	Similarity
VCTK (one-hot)	2.377	2.828
VCTK (zero-shot)	2.447	**3.051**
SVC (zero-shot)	2.289	2.549
VCTK→SVC (zero-shot)	**2.476**	2.664

Table 3. Mean opinion scores on singing voice with target voices from the VCTK test set with fixed encoder model.

	Quality	Similarity
VCTK (one-hot)	N/A	N/A
VCTK (zero-shot)	2.154	2.610
SVC (zero-shot)	2.477	2.772
VCTK→SVC (zero-shot)	**2.674**	**2.937**

Table 4. Mean opinion scores on singing voice with target voices from the SVC test set with fixed encoder model.

voices, respectively.

Lastly, we further exemplify the zero-shot nature of our proposed method by subjecting our system to target voices outside of the VCTK and SVC datasets. These examples were generated without any further training of models and using just 1-2 seconds of audio from a target voice in order to compute speaker embeddings. While quality and voice similarity could obviously be improved by further model fine tuning on more data from target voices, it is apparent that the system can generate reasonable conversions resembling the voices from the reference material in a zero-shot manner.

4. CONCLUSION

In this paper, we propose the application of speaker embedding networks for zero-shot SVC. We suggest two architectures for carrying out zero-shot SVC using the WORLD vocoder for modeling singing voice. Overall, we find that speaker embeddings can indeed be used directly for zero-shot SVC. Moreover, zero-shot networks replacing one-hot speaker labels with speaker embeddings perform as well as (or even better than) their supervised closed set counterparts, with the invaluable added benefits that they can be trained on unlabeled data and can potentially adapt to new voices without requiring further training. Furthermore, we show that there is some benefit to training zero-shot SVC networks by adapting an initial model trained on large amounts of speech data. In future work, we will investigate learning latent factors which can allow for further expressive manipulation of conversion results. While some initial progress to this end has been made using Gaussian Mixture VAEs (GMVAEs) [11], they have largely been limited to sung vowels. We can likely generalize this to more practical singing voice by utilizing the conditioning signals used in this work. We are also interested in replacing the WORLD vocoder with learned vocoders based on differentiable digital signal processing, as in [18, 25], in order to enable lightweight end-to-end training.

5. ACKNOWLEDGEMENTS

6. REFERENCES

[1] K. Lent, "An efficient method for pitch shifting digitally sampled sounds," *Computer Music Journal*, vol. 13, no. 4, pp. 65–71, 1989.

[2] S. Nercessian, "Improved zero-shot voice conversion using explicit conditioning signals," in *Proc. of Interspeech 2020*, 2020, accepted.

[3] W. Zhao, W. Wang, Y. Sun, and T. Tang, "Singing voice conversion based on WD-GAN algorithm," in *Proc. of the 2019 IEEE 4th Advanced Information Tech., Electronic and Automation Control Conference (IAEAC)*, 2019, pp. 950–954.

[4] M. Blaauw and J. Bonada, "A neural parametric singing synthesizer," in *Proc. of Interspeech 2017*, 2017.

[5] A. van den Oord *et al.*, "WaveNet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[6] N. Kalchbrenner *et al.*, "Efficient neural audio synthesis," *arXiv:1802.08435*, 2018.

[7] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems*, vol. E99-D, no. 7, pp. 1877–1884, 2016.

[8] E. Nachmani and L. Wolf, "Unsupervised singing voice conversion," in *Proc. of Interspeech 2019*, 2019, pp. 2583–2587.

[9] B. Sisman, K. Vijayan, M. Dong, and H. Li, "SINGAN: Singing voice conversion with generative adversarial networks," in *Proc. of the 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2019, pp. 112–118.

[10] P. Chandna, M. Blaauw, J. Bonada, and E. Gomez, "WGANSing: A multi-voice singing voice synthesizer based on the wasserstein-gan," in *Proc. of the 27th European Signal Processing Conference*, 2019.

[11] Y. Luo, C. Hsu, K. Agres, and D. Herremans, "Singing voice conversion with disentangled representations of singer and vocal technique using variational autoencoders," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020.

[12] S. O. Arik *et al.*, "Deep voice 2: Multispeaker neural text-to-speech," *Advances in Neural Information Processing Systems*, vol. 30, pp. 2962–2970, 2017.

[13] M. Blaauw, J. Bonada, and R. Daido, "Data efficient voice cloning for neural singing synthesis," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[14] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 4879–4883.

[15] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, "AutoVC: Zero-shot voice style transfer with only autoencoder loss," in *Proc. of the International Conference on Machine Learning*, 2019.

[16] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai, "Mel-generalized cepstral analysis - a unified approach to speech spectral estimation," in *Proc. of the International Conference on Spoken Language Processing*, 1994.

[17] L. Sun, K. Li, H. Wang, S. Kang, and H. Meng, "Phonetic posteriorgrams for many-to-one voice conversion without parallel data training," in *Proc. of the IEEE International Conference on Multimedia and Expo*, 2016, pp. 1–6.

[18] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Proc. of the International Conference on Learning Representations*, 2020, pp. 26–30.

[19] J. H. Lee, H. S. Choi, and K. Lee, "Audio query-based music source separation," in *Proc. of the International Society for Music Information Retrieval Conference*, 2019.

[20] J. S. Garapolo *et al.*, *TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1*. Philadelphia: Linguistic Data Consortium, 1993.

[21] L. Hantrakul, J. Engel, A. Roberts, and C. Gu, "Fast and flexible neural audio synthesis," in *Proc. of the International Society for Music Information Retrieval Conference*, 2019.

[22] Y. Zhou, X. Tian, H. Xu, R. K. Das, and H. Li, "Cross-lingual voice conversion with bilingual phonetic posteriorgram and average modeling," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 6790—6794.

[23] T. Hasan and J. H. L. Hansen, "A study on universal background model training in speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 1890–1899, 2011.

[24] C. Veaux, J. Yamagishi, and K. MacDonald, *CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit*. Edinburgh: The Centre for Speech Technology Research (CSTR), University of Edinburgh, 2016.

[25] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter-based waveform model for statistical parametric speech synthesis," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 5916–5920.

MUSIC SKETCHNET: CONTROLLABLE MUSIC GENERATION VIA FACTORIZED REPRESENTATIONS OF PITCH AND RHYTHM

Ke Chen[1]   **Cheng-i Wang**[3]   **Taylor Berg-Kirkpatrick**[2]   **Shlomo Dubnov**[1]

[1] CREL, Music Department, [2] UC San Diego   [3] Smule, Inc
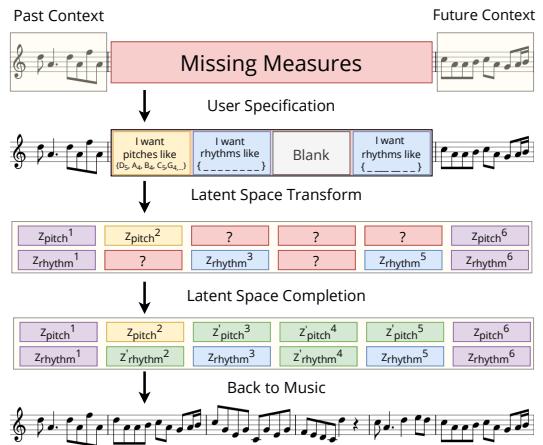
[1,2]{knutchen, tberg, sdubnov}@ucsd.edu, [3]cheng-i.wang@smule.com

ABSTRACT

Drawing an analogy with automatic image completion systems, we propose Music SketchNet, a neural network framework that allows users to specify partial musical ideas guiding automatic music generation. We focus on generating the missing measures in incomplete monophonic musical pieces, conditioned on surrounding context, and optionally guided by user-specified pitch and rhythm snippets. First, we introduce SketchVAE, a novel variational autoencoder that explicitly factorizes rhythm and pitch contour to form the basis of our proposed model. Then we introduce two discriminative architectures, SketchInpainter and SketchConnector, that in conjunction perform the guided music completion, filling in representations for the missing measures conditioned on surrounding context and user-specified snippets. We evaluate SketchNet on a standard dataset of Irish folk music and compare with models from recent works. When used for music completion, our approach outperforms the state-of-the-art both in terms of objective metrics and subjective listening tests. Finally, we demonstrate that our model can successfully incorporate user-specified snippets during the generation process.

1. INTRODUCTION

As a research area, automatic music generation has a long history of studying and expanding human expression/creativity [1]. The use of neural network techniques in automatic music generation tasks has shown promising results in recent years [2]. In this paper, we focus on a specific facet of the automatic music generation problem on how to allow users to flexibly and intuitively control the outcome of automatic music generation. Prior work supports various forms of conditional music generation. MuseGan [3] allows users to condition generated results on full-length multi-track music. DeepBach [4] provides a constraint mechanism that allows users to limit the generated results to match composer styles. Music Trans-

Figure 1. The music sketch scenario. The model is designed to fill the missing part based on the known context and user's own specification.
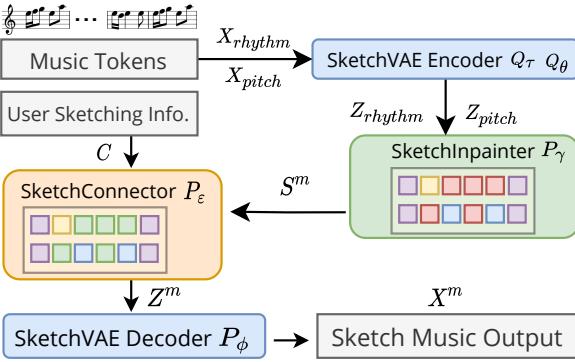
former [5] supports a accompaniment arrangement from an existing melody track in classical music. However, all these approaches require the user preference to be defined in terms of complete musical tracks.

Inspired by the sketching and patching work from computer vision [6–10], we propose Music SketchNet [1] which allows users to specify partial musical ideas in terms of incomplete and distinct pitch and rhythm representations. More specifically, we generalize the concept of sketching and patching – wherein a user roughly sketches content for a missing portion of an image – to music, as depicted in Figure 1. The proposed framework will complete the missing parts given the known context and user input. To the best of our knowledge, there has been limited work on sketching in music generation. Some work [4,11] has used Markov Chain Monte Carlo (MCMC) to generate music with given contexts or generate music conditioned on simple starting and ending notes [12]. The most related task is music inpainting: completing a musical piece by generating a sequence of missing measures given the surrounding context, but without conditioning on any form of user preferences. Music InpaintNet, [13] completes musical pieces by predicting vector representations for missing measures, then the vector representations are decoded to output symbolic music through the use of a variational autoencoder (VAE) [14].

Our proposed music sketching scenario takes music in-

---

[1] https://github.com/RetroCirce/Music-SketchNet.

Figure 2. The Music SketchNet pipeline. The color patterns inside Inpainter and Connector correspond to the latent space transform and completion process in Figure 1.

painting a step further. We let users specify musical ideas by controlling pitch contours or rhythm patterns, not by complete musical tracks. The user input is optional: users can choose to specify musical ideas, or let the system fill in predictions without conditioning on user preferences.

Music SketchNet consists of three component, as depicted in Figure 2: (1) SketchVAE is a novel variational autoencoder that converts music measures into high dimensional latent variables. By the use of a factorized inference network, SketchVAE decouples latent variables into two parts: pitch contour and rhythm, which serve as the control parameters for users. (2) SketchInpainter contains stacked recurrent networks to handle the element-level inpainting prediction in the latent space. (3) SketchConnector receives users' sketches of pitch, rhythm, or both, combines them with the prediction from SketchInpainter, and finalizes the generation.

In this paper, we show that the proposed Sketch-VAE is capable of factorizing music input into latent variables meaningfully, and the proposed SketchInpainter/SketchConnector allows users to control the generative process. The novel training and evaluation methodologies of the SketchConnector are also presented.

2. MUSIC SKETCHING

We formalize the music sketching task as solving the following three problems: (1) how to represent music ideas or elements, (2) how to generate new materials given the past and future musical context and (3) how to process users' input and integrate it with the system. A visualization of the sketching scenario is depicted in Figure 1.

We propose three neural network components to tackle the three problems. The SketchVAE encodes/decodes the music between external music measures and the learned factorized latent representations. The SketchInpainter predicts musical ideas in the form of the latent variables given known context. And the SketchConnector combines the predictions from SketchInpainter and users' sketching to generate the final latent variables which are sent into the SketchVAE decoder to generate music output. A diagram

showing the proposed pipeline is shown in Figure 2.

2.1 Problem Definition

More formally, the proposed sketch framework can be described as a joint probability model of the missing musical content, X^m, conditioned on the past, future, and user sketching input. The joint probability breaks down into a product of conditional probabilities corresponding to subcomponents of the framework:

$$P_{\phi,\varepsilon,\gamma,\theta,\tau}(X^m, Z, S|X^p, X^f, C) =$$

$$P_\phi(X^m|Z^m) \qquad \text{(SketchVAE Decoder)}$$

$$* P_\varepsilon(Z^m|S^m, C) \qquad \text{(SketchConnector)}$$

$$* P_\gamma(S^m_{pitch}|Z^p_{pitch}, Z^f_{pitch}) \qquad \text{(SketchInpainter)}$$

$$* P_\gamma(S^m_{rhythm}|Z^p_{rhythm}, Z^f_{rhythm}) \quad \text{(SketchInpainter)}$$

$$* Q_\theta(Z^p_{pitch}, Z^f_{pitch}|X^p_{pitch}, X^f_{pitch})$$

$$* Q_\tau(Z^p_{rhythm}, Z^f_{rhythm}|X^p_{rhythm}, X^f_{rhythm})$$

$$\text{(SketchVAE Encoders)}$$

X indicates the input/output music sequence, Z is the sequence for $\{z\}$ the latent variable , S is the SketchInpainter's predicted sequence, C is users' sketching input. The superscripts, p, m, f indicate the past, missing and future context. The subscripts, $pitch$ and $rhythm$ indicate the pitch and rhythm latent variables. Q_θ, Q_τ, P_ϕ are the SketchVAE pitch/rhythm encoders and decoder parameters, P_γ represents the SketchInpainter, and P_ε is the SketchConnector.
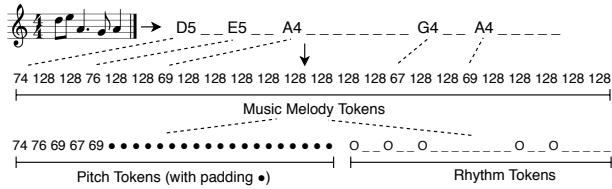
2.2 SketchVAE for Representation

MusicVAE [15] is one of the first works applying the variational auto-encoder [14] to music. MeasureVAE [13] further focuses on representing isolated measures and utilizes a hierarchical decoder to handle ticks and beats. EC^2-VAE [16] factorizes music measures with separate vectors representing pitch and rhythm by a single encoder and two decoders. Our proposed SketchVAE aims to factorize representations by introducing a factorized encoder that considers pitch and rhythm information separately in the encoder channels. Different from EC^2-VAE, it could allow users to enter parts of the information (rhythm and/or pitch) optionally.

SketchVAE aims to represent a single music measure as a latent variable z that encodes rhythm and pitch contour information in separate dimensions (z_{pitch}, z_{rhythm}). It contains (1) a pitch encoder $Q_\theta(z_{pitch}|x_{pitch})$, (2) a rhythm encoder $Q_\tau(z_{rhythm}|x_{rhythm})$, and (3) a hierarchical decoder $P_\phi(x|z_{pitch}, z_{rhythm})$ as shown in Figure 4.

2.2.1 Music Score Encoding

Similar to [15], we encode the monophonic midi melody by using [0, 127] for the note onsets, 128 for holding state, and 129 for the rest state. We cut each measure into 24 frames to correctly quantize eighth-note triplets like [13], and encode the midi as described in the previous sentence.

Figure 3. An example of the encoding of a monophonic melody.

As Figure 3 shows, we further process the encoded 24-frame sequence x into x_{pitch} and x_{rhythm}, the pitch and rhythm token sequences respectively. The pitch token sequence x_{pitch} is obtained by picking all note onsets in x with padding (shown by "•" in Figure 3) to fill 24 frames. The rhythm token sequence x_{rhythm} is obtained by replacing all pitch onsets with the same token (shown by "O" and "_" in Figure 3). A similar splitting strategy is also used in [17]. Our motivation is to provide users with two intuitive music dimensions to control, and to help enforce better factorization in the latent representation for later prediction and control.

2.2.2 The Pitch Encoder and Rhythm Encoder

After pre-processing x, x_{pitch} only contains the note value sequence, while x_{rhythm} only has the duration and onset information. x_{pitch} and x_{rhythm} are then fed into two different GRU [18] encoders for variational approximation. The outputs of each encoder are concatenated into $z = [z_{pitch}, z_{rhythm}]$.

2.2.3 The Hierarchical Decoder
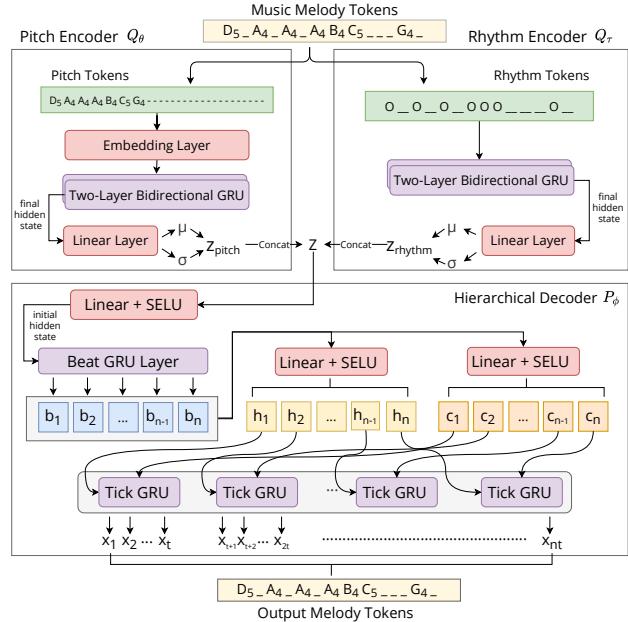
After we obtain the latent variable z, we feed it into the hierarchical decoder. This decoder is similar to the decoder used in MeasureVAE [13]. As shown in the bottom part in Figure 4, it contains an upper "beat" GRU layer and a lower "tick" GRU layer. This division's motivation is to decode z into n beats first and then decode each beat into t ticks. As a result, the note information in each measure will be decoded in a musically intuitive way. For the tick GRU, we use the teacher forcing [19, 20] and auto-regressive techniques to train the network efficiently. The output is conditioned frame-by-frame not only on the beat token but also on the last tick token.

2.2.4 Encoding the Past, Missing and Future Musical Context

The latent variable sequences Z^p, Z^m, and Z^f are then obtained by processing the music input in measure sequences X^p, X^m, and X^f. Both X^m and Z^m are masked during training. This encoding part is shown in the left block of Figure 5.

2.3 SketchInpainter for Initial Prediction

Next, we describe the model component that performs the music inpainting to predict latent representations for the missing measures. The SketchInpainter accepts Z_{pitch} and Z_{rhythm} as two independent inputs from SketchVAE.

Figure 4. SketchVAE structure: pitch encoder, rhythm encoder and hierarchical decoder. Rhythm tokens: the upper dashes denote the onsets of note, and the bottom dashes denote the hold/duration state. We use pitch symbols to represent the tokens numbers for better illustration.
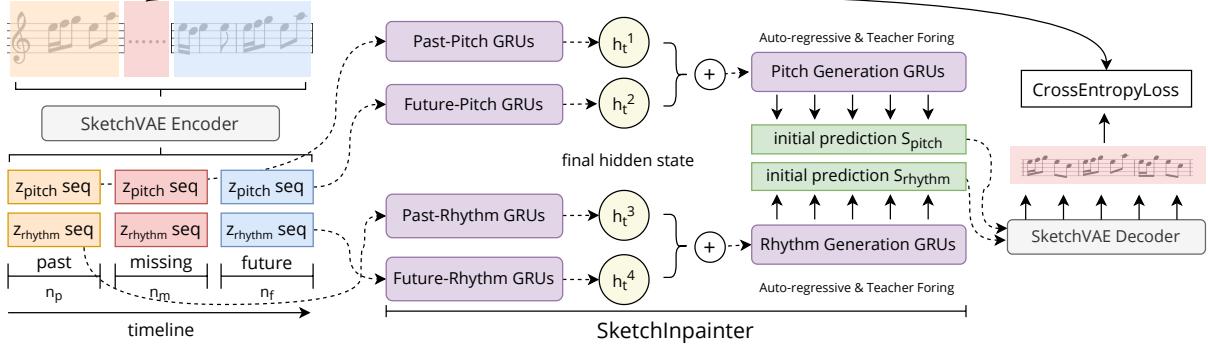
Then only the past and future Z_{pitch} and Z_{rhythm} are fed into the pitch/rhythm GRU groups respectively. The output from each GRU group is the hidden state h, as shown in the middle of Figure 5.

Then we combine the past/future hidden states h from both the pitch and rhythm GRU groups and use them as the initial states for the pitch/rhythm generation GRUs. The generation GRUs then predict the missing latent variables by $S^m = (S_{pitch}, S_{rhythm})$, as shown in the green box in Figure 5. Each generation GRU is trained with the teach forcing and auto-regressive techniques.

Each output vector s^m from S^m has the same dimension as the latent variable z from Z. We first build a model with only SketchVAE and SketchInpainter that directly predicts the missing music material, X^m. As the right block of Figure 5 shows, S^m is sent into the SketchVAE decoder and we compute the cross entropy loss between the predicted music output and the ground truth. This is the stage I training in our model, detailed in Section 3.3.

2.4 SketchConnector for Finalization

The predicted S^m from SketchInpainter can already serve as a good latent representation for the missing part X^m. We continue by devising the SketchConnector, $P_\varepsilon(Z^m|S^m, C)$, to modify the prediction with user control. To make up for the lack of correlation between pitch and rhythm in current predictions, we introduce the SketchConnector as a way to intervene/control the generative process, that also leads to a wider musical expressivity of the proposed system.

Figure 5. SketchInpainter structure. We feed the music tokens into the SketchVAE and obtain the latent variable sequences. And we feed the sequences into the pitch GRU and the rhythm GRU groups to generate the initial prediction S.

2.4.1 Random Unmasking

With S^m obtained from SketchInpainter, we concatenate it with Z^p and Z^f again. However, before we feed it back into the network, we randomly unmask some of the missing parts to be the ground-truth (simulating user providing partial musical context). The masked S^m are shown by the red boxes in Figure 6. We replace some s from S^m to be the real answer in Z^m, denoted as C. We observe that this optimization is very similar to BERT [21] training. The difference is that BERT randomly masks the ground truth labels to be unknown, but SketchConnector randomly unmasks the predictions to be truths. The unmasking rate is set to 0.3.

Intuitively, this allows the model to learn a more close relation among current rhythm, pitch tokens, and the nearest neighbour tokens. In the sketch inference scenario, the randomly unmasked measures will be replaced by the user sketching information, which allows a natural transition between the training and testing process.

2.4.2 Transformer-based Connector
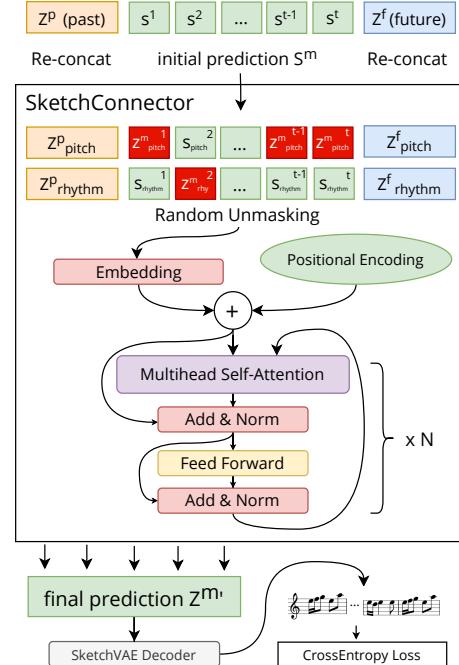
Then with S^m and the random unmasking data C, we feed them into a transformer encoder with absolute positional encoding. In contrast to [5], we do not use relative positional encoding because our inputs are vectors representing individual measures, whose length is far shorter than midi-event sequences.

The output of the SketchConnector, Z^m, will be the final prediction for the missing part. We feed it into the SketchVAE decoder, and compute the cross entropy loss of the output with the ground-truth.

3. EXPERIMENT

3.1 Dataset and Baseline

To evaluate the SketchVAE independently, we compare our model with two related systems: MeasureVAE [13] and EC^2-VAE [16]. For SketchNet, we compare our generation results with Music InpaintNet [13], which has shown better results than the earlier baseline [12]. Similar to [13], we use the Irish and Scottish monophonic music

Figure 6. The SketchConnector: the output of SketchInpainter is randomly unmasked and fed into a transformer encoder to get the final output.

Dataset [22] and select the melodies with a 4/4 time signature. About 16000 melodies are used for training and 2000 melodies for testing.

3.2 SketchVAE Measurements

3.2.1 Reconstruction

For SketchVAE, MeasureVAE and EC^2-VAE, the dimension of latent variable $|z|$ is set to 256, half for the pitch contour, and the other half for the rhythm. We set the learning rate to 1e-4 and use Adam Optimization with $\beta_1 = 0.9$ and $\beta_2 = 0.998$. Three models achieve the accuracy (the reconstruction rate of melodies) 98.8%, 98.7%, 99.0% respectively. We can clearly conclude that all VAE models are capable of converting melodies to latent variables by achieving the accuracy around 99%. SketchVAE is capable of encoding/decoding musical materials in SketchNet.

Model	Irish-Test			Irish-Test-R			Irish-Test-NR		
	loss ↓	pAcc ↑	rAcc ↑	loss ↓	pAcc ↑	rAcc ↑	loss ↓	pAcc ↑	rAcc ↑
Music InpaintNet	0.662	0.511	0.972	0.312	0.636	0.975	0.997	0.354	0.959
SketchVAE + InpaintRNN	0.714	0.510	0.975	0.473	0.619	0.981	1.075	0.374	0.964
SketchVAE + SketchInpainter	0.693	0.552	0.985	0.295	0.692	0.991	1.002	0.389	0.977
SketchNet	**0.516**	**0.651**	**0.985**	**0.206**	**0.799**	**0.991**	**0.783**	**0.461**	**0.977**

Table 1. The generation performance of different models in Irish and Scottish monophonic music dataset. The InpaintRNN is the generative network in Music InpaintNet.

Model	Complexity↑	Structure↑	Musicality↑
Original	3.22	3.47	3.56
InpaintNet	2.98	3.01	3.09
SketchNet	3.04	3.29	3.26

Table 2. Results of the subjective listening test.

3.2.2 Comparison with EC^2-VAE

EC^2-VAE [16] is also capable of decoupling the latent variable into rhythm and pitch contour dimensions. However, SketchVAE's encoders can accept pitch contour/rhythm inputs separately. Rhythm and pitch controls can be manipulated independently in the sketching scenario where the user might not specify an entire musical measure (e.g., just a rhythm pattern). By contrast, EC^2-VAE requires a completed measure before encoding. If users want to specify either rhythm or pitch controls, the model must first fill in the other half part before inputting it, which prohibits the possibility of the separate control.

3.3 Generation Performance

3.3.1 Training Results

The SketchNet's training is separated into stage I and II. In stage I, after training the SketchVAE, we freeze its parameters and train the SketchInpainter as shown in the right block of Figure 5. In stage II, with the trained SketchVAE and SketchInpainter, we freeze both, concatenate S^m with the past/future latent variables, and feed them to the SketchConnector for training.

We compare four models by using 6 measures of past and future contexts to predict 4 measures in the middle (i.e. $n_p = n_f = 6$, and $n_m = 4$). Music InpaintNet [13] is used as the baseline, along with several variations. Early stopping is used for all systems.

We compute three metrics: loss, pitch accuracy, and rhythm accuracy to evaluate the model's performance. The pitch accuracy is calculated by comparing only the pitch tokens between each generation and the ground truth (whether the model generates the correct pitch in the correct position). And the rhythm accuracy is calculated by comparing the duration and onset (regardless of what pitches it generates). The overall accuracy and loss are

negatively correlated.

For this part of the experiment, we also use two special test subsets. We compute the similarities between the past and future contexts of each song in the Irish test set, pick the top 10% similar pairs (past and future contexts are almost the same) and bottom 10% pairs (almost different), and create the Irish-Test-R (repetition) and Irish-Test-NR (non-repetition) subsets.
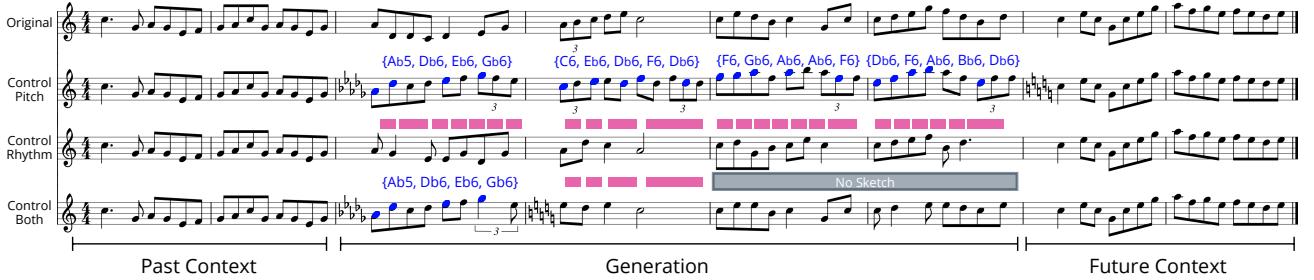
From Table 1, we can see that SketchNet beats all other models for all test sets. The performance improved more for pitch then for rhythm. The accuracy is almost the same between the 1st and 2nd model. Accuracy is slightly better if we use SketchInpainter to treat rhythm and pitch independently during generation. Lastly, with the power of transformer encoder and random unmasking process done in SketchConnector, we can achieve the best performance by using SketchNet (bottom row in Table 1). We further follow [23] to use the Bootstrap significance test to verify the difference between each pair's overall accuracy for models in the whole Irish-Test set (Four models, i.e. six pairs in total). The sample time is set to 10000. After calculation, all p-values except the fist and second model pair (p-value = 0.402) are less than 0.05, which proves that SketchNet is different from the left three models.

In the repetition test subsets, the loss of Music Inpaint-Net is 0.312, which is lower enough to capture repetitions in the musical context and fill in the missing part by copying. In most cases, copying is the correct behaviour because the original melody has repetitive pattern structures. The loss is a measurement to evaluate if the model can learn the repetitive pattern and copy mechanism from the data. the SketchNet slightly outperforms InpaintNet.

3.3.2 Subjective Listening Test

However, the more interesting result is the generation with non-repetition subset. In this case, models cannot merely copy because original melodies do not repeat its content. We see higher losses in all models in this subset compared to the repetition subset. Intuitively, it means that repetitive patterns are essential to the reconstruction task, not necessarily the expressivity of the generated output would be less.

To further evaluate the proposed SketchNet, we conduct an online subjective listening test to let subjects judge the generated melodies from the non-repetition subset. Each subject will listen to three 32-second piano-rendered

Figure 7. An example of sketch generation. From top to bottom: original, pitch/rhythm/mixture control. The blue pitch texts denote pitch controls, and the pink segments denote rhythm controls.

Control Info.	Rhythm	Pitch
Pitch Acc.	0.189	**0.881**
Rhythm Acc.	**0.973**	0.848

Table 3. The accuracy of the virtual control experiment.

melodies: the original, the Music InpaintNet's generation, and the SketchNet's generation. Songs are randomly picked from the Irish-Test-NR set. The beginning and ending (past & future) are the same for the three melodies. Since the subjective feeling of music is complicated to quantify, we chose three criteria: the number of notes (**complexity**), the repetitiveness between musical structures (**structure**), and the degree of harmony of the music (**overall musicality**). In this way, subjects with different levels of music skills can all give reasonable answers.

Before rating the songs, subjects will see three criteria descriptions as we introduced below. The rating is ranged from 1.0 to 5.0 with a 0.5 step. We collected 318 surveyed results from 106 subjects (each subject listens to three groups, nine melodies in total). The average rating of each criteria for all models are shown in Table 2. The subjective evaluations of all three criteria in SketchNet are better for those of Music InpaintNet. Similar to section 3.3.1, we also conduct a pairwise significance test via Bootstrap in three criteria. All p-values except the <complexity: InpaintNet, SketchNet> (p-value = 0.364) are less than 0.05. It proves that three models (including original songs) are significantly different in structure and overall musicality (subjective feeling to a person). As for the complexity, we believe that the results generated by the two models are similar in terms of the richness of notes, and our model does not significantly increase the number of notes generated.

3.4 Sketch Scenario Usage

The contribution of Music SketchNet is not only shown in the performance of the generation in section 3.3, but can also be shown in the interactive scenario where users can control the generated output by specifying the rhythm or pitch contour in each measure.

Figure 7 shows an example of a non-repetition subset melody, where the first and last two measures are given, and the middle parts is generated. The first track is the

original melody, the second track is generated with the pitch contour control, the third track is generated with the rhythm control, and the fourth track is controlled with both pitch and rhythm. We can see that each generated melody follows the control from users and develops music phrases accordingly in the missing part. Moreover, each measure is in line with the past and future measures even in the case of scale shift.

We also provide a "virtual control experiment" to statistically show that users' control did influence the model's generation process. We randomly collect 3000 sample pairs (A, B) from the Irish-Test set. And we use the pitch/rhythm of Sample B to be the sketch information in the same missing position of Sample A. Then we let the model make the generation. We then compute the pitch/rhythm accuracy [2] in the missing position between the generation and Song B. From 3 we can see if we sketch song B's rhythm into the model, the generation will follow the rhythm with 97.3% accuracy but has different (18.9%) pitches. However, when we sketch pitches, the pitches in the generation will be highly (88.1%) in line with the sketching. This proves that the user's control has a relatively high guiding effect on the result of the model generated at the specified position.

4. CONCLUSION & FUTURE WORK

In this paper, we propose a new framework to explore decoupling latent variables in music generation. We further convert this decoupling into controllable parameters that can be specified by the user. The proposed Music SketchNet achieves the best results in the objective and subjective evaluations. Practically, we show the framework's application for the music sketching scenario where users can control the pitch contour and/or rhythm of the generated results. There are several possible extensions for this work. Music elements other than pitch and rhythm can be applied into the music sketching scenario by the latent variable decoupling. Also, how to represent a polyphonic music piece in the latent space is another pressing issue. Both are future works that can generalize this model to more applied scenarios.

---

[2] The metric to calculate the pitch accuracy is different from section 3.3.1, because the generated pitches in the new song might have different onset positions. We leverage the Longest Common Sequence to calculate the accuracy. The implementation is presented in the code archive.

5. ACKNOWLEDGEMENT

6. REFERENCES

[1] G. Loy, *Composing with Computers - a Survey of Some Compositional Formalisms and Music Programming Languages*. MIT Press, 1990.

[2] J. Briot, G. Hadjeres, and F. Pachet, *Deep Learning Techniques for Music Generation*. Springer, 2020.

[3] H. Dong, W. Hsiao, L. Yang, and Y. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 34–41.

[4] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *Proceedings of the 34th International Conference on Machine Learning, ICML*, 2017, pp. 1362–1371.

[5] C. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *7th International Conference on Learning Representations, ICLR*, New Orleans, LA, USA.

[6] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: a randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, 2009.

[7] Y. Güçlütürk, U. Güçlü, R. van Lier, and M. A. J. van Gerven, "Convolutional sketch inversion," in *Computer Vision ECCV Workshops*. Amsterdam, The Netherlands: Springer, 2016, pp. 810–824.

[8] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. Honolulu, HI, USA: IEEE Computer Society, 2017, pp. 6836–6845.

[9] Q. Yu, F. Liu, Y. Song, T. Xiang, T. M. Hospedales, and C. C. Loy, "Sketch me that shoe," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. Las Vegas, NV, USA: IEEE Computer Society, 2016, pp. 799–807.

[10] K. Xu, K. Chen, H. Fu, W. Sun, and S. Hu, "Sketch2scene: sketch-based co-retrieval and co-placement of 3d models," *ACM Trans. Graph.*, 2013.

[11] J. Sakellariou, , F. Tria, L. Vittorio, and F. Pachet, "Maximum entropy model for melodic patterns," in *ICML Workshop on Constructive Machine Learning*, 2015.

[12] G. Hadjeres and F. Nielsen, "Anticipation-rnn: enforcing unary constraints in sequence generation, with application to interactive music generation," *Neural Computing and Applications*, 2018.

[13] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, Delft, The Netherlands, 2019, pp. 343–351.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR*, Banff, AB, Canada, 2014.

[15] A. Roberts, J. H. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proceedings of the 35th International Conference on Machine Learning, ICML*. Stockholm, Sweden: PMLR, 2018, pp. 4361–4370.

[16] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, Delft, The Netherlands, 2019, pp. 596–603.

[17] B. Genchel, A. Pati, and A. Lerch, "Explicitly conditioned melody generation: A case study with interdependent rnns," in *Proceedings of the 7th International Workshop on Musical Meta-creation, MUME*, 2019.

[18] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*. Doha, Qatar: ACL, 2014, pp. 1724–1734.

[19] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, Montreal, Quebec, Canada, 2015, pp. 1171–1179.

[20] A. Goyal, A. Lamb, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 4601–4609.

[21] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, NAACL-HLT.   Minneapolis, MN, USA: Association for Computational Linguistics, 2019, pp. 4171–4186.

[22] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *Conference on Computer Simulation of Musical Creativity, CSMC*, 2016.

[23] T. Berg-Kirkpatrick, D. Burkett, and D. Klein, "An empirical investigation of statistical significance in NLP," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL*.   ACL, 2012, pp. 995–1005.

JOINT ANALYSIS OF MODE AND PLAYING TECHNIQUE IN GUQIN PERFORMANCE WITH MACHINE LEARNING

Yu-Fen Huang[1]   **Jeng-I Liang**[2]   **I-Chieh Wei**[1]   **Li Su**[1]

[1]Institute of Information Science, Academia Sinica, Taiwan
[2]Department of Traditional Music, Taipei National University of the Arts, Taiwan

{yfhuang, sma1033, lisu}@iis.sinica.edu.tw, liangjengi@gmail.com

ABSTRACT

Music is hierarchically structured, in which the global attributes (e.g., the determined tonal structure, musical form) dominate the distribution of local elements (e.g., pitch, playing technique arrangement). Existing methods for instrumental playing technique detection mostly focus on the local features extracted from audio. However, we argue that structural information is critical for both global and local tasks, particularly considering the characteristics of guqin music. Incorporating mode and playing technique analysis, this study demonstrates that the structural relationship between notes is crucial for detecting mode, and such information also provides extra guidance for the playing technique detection in local-level. In this study, a new dataset is compiled for guqin performance analysis. The mode detection is achieved by pattern matching, and the predicted results are conjoined with audio features to be inputted into a neural network for playing technique detection. Advanced techniques are developed to optimize the extracted pitch contour from the audio. It is manifest in the results that the global and local features are inter-connected in guqin music. Our analysis identifies key components affecting the recognition of mode and playing technique, and challenging cases resulting from unique properties of guqin audio signal are discussed for further research.

1. INTRODUCTION

The guqin (古琴) is a plucked seven-string Chinese musical instrument existing for over 3,000 years, and has been selected as UNESCO World Cultural Heritage.[1] In guqin performance, it is an intrinsic convention for musicians to implement diverse playing techniques, in order to communicate their interpretations of the performed music. In the theory of guqin performance, such configuration of playing technique in local-level is considered to be connected and reflect the higher-level, hierarchical tonal structure in

---

[1] https://ich.unesco.org/en/RL/guqin-and-its-music-00061

pentatonic modes [1]. However, existing research does not appear to provide sufficient empirical evidence to support the theory. In particular, in the research area of Music Information Retrieval (MIR), key/mode detection has been regarded as the recognition of the overall, global construction of music piece, whereas playing technique detection is usually taken as a classification task relying on local features extracted from the audio. This study aims to solve this discontinuity, and argue that the global tonal structure and the local configuration of playing technique are interconnected in guqin performance. This work takes MIR and machine learning frameworks as the means for empirical music analysis, and contribute to several aspects including:

1) to design and compile a new dataset, GQ39, featured by representative historical guqin recordings and note-by-note annotations;
2) to demonstrate the importance of tonal structural information, and to identify the crucial components contributing to both the mode detection and playing technique detection;
3) to bridge the knowledge gap between the theory and empirical observations, as well as to highlight the connection between the high-level structure and local elements in music.

In the subsequent section, related research regarding the tonal structure investigation, playing technique classification, and guqin performance theories will be reviewed. The theoretical basis and the procedure to construct the GQ39 dataset will be described in Section 3. The mode detection is performed on the dataset, and the results are analyzed in Section 4. The playing technique is further investigated using a neural network, and decisive components playing important roles in the task are discussed in Section 5, followed by the concluding remarks in Section 6.

2. RELATED WORK

For the high-level tonal structure in music, key detection has been one of the core issues to be explored in Western tonal music. Methods based on chord progression rules are applied to detect key modulation in audio [2]. Neural networks are utilized for key classification, and it has been found that the global harmonic structure in the whole piece plays an important role to identify local keys of short segments [3]. In addition to the connection between the local and global tonal structures, it has also been proven that

different music styles are more easily to be classified by informing the global key and key-related pitch classes [4]. In Indian art music, the usage of raga achieves the structural coherence in music, and different types of raga can be identified according to their pitch distribution [5]. The implicit patterns of melodic and timing features in raga has also been explored [6]. In Arab-Andalusian music, its centonization is analyzed using a high-order n-gram model [7], and different music patterns, nawba, can be classified using template matching [8]. In Georgian music, its unique tuning system has been examined through melodic and harmonic aspects, and the structural relationship between pitch intervals has been found [9].

In previous studies of playing technique classification, diverse features are extracted from audio according to the traits of individual instruments. In particular, strings appear to be high-profile instruments in this research area. For electric guitar, note-level timbral and pitch features are considered to be major components for identifying different types of playing techniques [10], whereas for guitar, each playing technique possesses distinguishable cepstral and phase features [11]. In a study to classify plucking styles of electric bass guitar, intra-note attributes corresponding to the attack and decay parts of the notes are further examined using spectral and statistical descriptors [12]. In violin performance, note-level features including dynamics, vibrato rate and vibrato extent are proven to be connected with score-informed expressive schemes [13], and idiosyncratic performing styles of individual violinists can be characterized by their articulation, energy, and vibrato attributes [14].

In guqin performance, the fingering and playing techniques for strings pressing in the left hand have been regarded as a primary component to affect the expression and aesthetic perception of performance, especially since the late Ming dynasty (around the late 17th century) [15–17]. Furthermore, the left-hand techniques are not merely local ornaments attached to single musical events, and should be contemplated within the global context of complete music composition. As stated in conventional guqin performance theory, the hierarchical tonal structure in the mode can vastly affect the selection of playing technique in the practice of guqin performance [1, 18, 19]. The connection between the global tonal structure and local technique elements is manifest in guqin music theories, and such association between the mode and playing technique is also a prevailing, shared character in many Asian music cultures [20,21]. However, only few efforts have been devoted to explore empirical evidence regarding how the global and local aspects conjoin with one another in the actual performance practice [22, 23]. This study therefore aims to bridge the knowledge gap between the guqin music theories and the empirical observation, and explore the connection between the mode structure and the playing technique implement using statistics and machine learning.

3. GQ39 DATASET

Guqin music is constructed from diverse modes of pentatonic scale, and the performance carries distinctive ex-

Rank	Mode 1	Mode 2	Mode 3	Mode 5	Mode 6
1st	1	2	3	5	6
2nd	5	6	-	2	3
3rd	2	3	-	6	-

Table 1. The hierarchical structure in pentatonic scale (the top row): the 3 importance level for note degrees (the bottom row) in 5 modes (the middle row).

pressions in playing techniques. We therefore design and compile a new dataset for music performance analysis according to such properties.

3.1 The guqin performance and pentatonic scale

Guqin performers pluck strings by their right hands and press strings using left hands for performing. Guqin performance is characterized by its flexibility, which render plenty of freedom for musicians to choose a wide selection of playing techniques to perform the same piece of music. In particular, the usage of vibrato and portamento is central among all playing techniques to carry the representative traits of individual music pieces and musicians.

The pentatonic scale (see the top row of Table 1 as an example) forms the main construction of guqin music. While other altered chromatic tones and microtones can be included, the pentatonic scale retain the central position in guqin compositions. The pentatonic scale can be transformed into 5 different modes, each of which has a different order of notes according to its initial degree: mode 1 (mode Gong (宫); degree 1, 2, 3, 5, 6), mode 2 (mode Shang (商); degree 2, 3, 5, 6, 1), mode 3 (mode Jue (角); degree 3, 5, 6, 1, 2), mode 5 (mode Zhi (徵); degree 5, 6, 1, 2, 3), and mode 6 (mode Yu (羽); degree 6, 1, 2, 3, 5). Each mode has its own hierarchical structure, which is achieved by assigning different importance levels to note degrees based on the Circle of Fifths. In each mode, at most 3 notes can be considered as prominent component (equivalent concept to the Tonic, Dominant, and Subdominant in Western major and minor scales, see the lower two rows in Table 1) [1, 19]. It should be noted that the pentatonic scale only denotes the relative intervals between notes, but not the absolute pitches, which means that the degree 1 in the scale can be assigned to any pitch in 12 semitones.

3.2 Data collection and labelling

The GQ39 dataset consists of 39 audio recordings of prevalent guqin solo compositions and corresponding event-by-event annotations. The audio data are extracted from a massive collection of guqin historical recording, with the recording years ranging from 1960 to 1990 [24]. 39 excerpts played by five different guqin performers are se-

lected according to a professional guqin musician's opinion (with the performing experience of 18 years). The 39 selected audio excerpts cover representative composing and performing styles and contain diverse playing techniques. The GQ39 dataset only includes guqin music in mode 1, 2, and 5, on account of the facts that: 1) the 2 modes with less than 3 important notes (mode 3 and mode 6) are considered as having less stable structure; 2) mode 3 is only occasionally applied to guqin composition; 3) guqin music composed by mode 6 is relatively complicated, which is not applicable for this exploratory study. The collected audio recordings are roughly 2,000 seconds long in total.
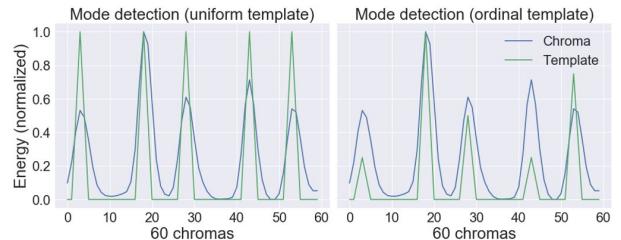
Considering the typical feature of guqin playing, we define an event as a plucking movement in the right hand, which may correspond to either one single note, or a series of note pitch variations and sliding movements in the left hand (ranging up to 5 semitones). Each event is then annotated with 13 types of label by professional guqin musicians, including 2 music-level features (tuning, mode), and 11 event-level features regarding the event context (note degree, pitch range), playing techniques in the right hand for plucking (plucked string, plucking finger, plucking technique type), playing techniques in the left hand for string pressing (pressed position, pressing finger, technique type for pitch variation, technique type for timbre variation), and performing matters (event onset, event duration). As the result, the GQ39 dataset comprises 2,303 annotated events in total (mean # of event = 59.1, SD = 29 per excerpt). The annotations are available on the website, together with details regarding the annotation procedure and dataset descriptions. [2]

4. MODE DETECTION AND ANALYSIS

In this section, we examine mode, the global tonal structure in guqin performance. Chromas are derived from audio recordings. Two types of template representing the tonal structure are designed for pattern matching. The statistical analysis reveals that the inherent characters of mode construction are reflected in the performance. And the results of mode detection indicate that the hierarchical configuration is a crucial element to identify different modes.

4.1 Data representation and mode matching

In mode detection, three types of data are obtained from audio data: the chroma representation of constant-Q transform (CQT), pitch salience function, and pitch contour. For the first type of data, the chromagrams of audio recordings are derived using CQT [25]. For the second and the third types of data, the pitch estimation network, Crepe, is applied to estimate the pitch salience function and the pitch contour [26]. For all types of data, we obtain 60 chromas instead of 12 chromas per octave for higher pitch resolution, considering the facts that: 1) the tuning in guqin is not equal temperament, and 2) the dataset contains large amounts of protamento and transitions between semitones.

Figure 1. The mode detection with 2 template types: the uniform template (left) and the ordinal template (right). This instance shows the mode 5 template, with degree 5 being shifted to the position of 18 in x-axis.

For each recording, the energy of each chroma is accumulated and is normalized to the range between 0 and 1.
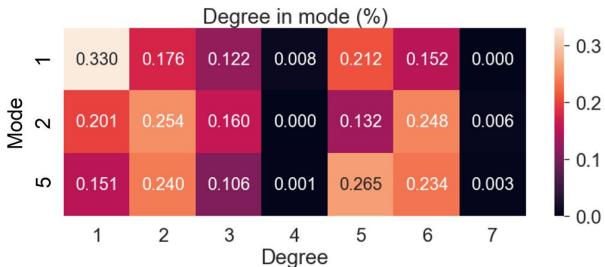
We design two types of mode template, i.e., the uniform template and the ordinal template (see Figure 1 as examples), to be matched up with the chroma representation from audio. For each mode, the *uniform mode template* is constructed by denoting the positions of note degrees as 1 and otherwise 0 in 12 semitones (e.g., mode 1 = (1,0,1,0,1,0,0,1,0,1,0,0)). In addition to the degree positions, the *ordinal mode template* includes the information regarding the importance level of degrees. The degree positions are represented as 4, 3, 2 individually following the order of importance ranking in Table 1, and the two trivial degrees are marked as 1, otherwise 0 in 12 semitones (e.g., mode 1 = (4,0,2,0,1,0,0,3,0,1,0,0)). The two 12-D templates are extended to 60-D for the subsequent matching with 60-chroma representation (each chroma unit represents 20 cents). Complying the treatment procedure for audio chroma representation, all mode templates are normalized to the range between 0 and 1.

The chroma representation of each music excerpt is then matched with mode templates. The built mode templates are shifted to the position of 60 chromas in turn, and are then compared with the audio chroma representation using Pearson correlation. This procedure results in 360 matching pairs for per music excerpt (2 template types (uniform/ordinal) x 3 modes (mode1/2/5) x 60 chroma shifts). The mode is determined by the matching with the highest Pearson correlation coefficient.

4.2 Results and discussion

The statistics of the annotated note degrees reveal the structural configuration of modes, and such observation provides insights regarding how the music theory is practiced in actual performances. As stated in conventional music theories, mode 1 holds the most solid construction among all modes [18]. As can be observed in Figure 2, the note degrees with higher importance ranking usually occur more frequently in the composition (such as degree 1, 5, 2 in mode 1; degree 2, 6, 3 in mode 2; degree 5, 2, 6 in mode 5). Furthermore, in mode 1, the prominent roles of the three principal notes are settled by obvious differences of occurrence ratios, compared to two other trivial degrees. In the mode detection task, we can take a step

Figure 2. The statistics of the annotated degrees (x-axis) in different modes (y-axis) (in %).

Type	CQT		Salience		Contour	
Result	uni	ord	uni	ord	uni	ord
Correct #	14	**34**	19	29	17	30
Fifth #	4	**4**	5	8	6	8
Miss #	21	**1**	15	2	16	1
Accuracy	0.41	**0.92**	0.55	0.84	0.51	0.87

Table 2. The results of mode detection task (3 data types (CQT/ salience function/ pitch contour) x 2 template types (uniform/ ordinal)).

further to evaluate that how the structural configuration of mode affects the task outcome.

For the evaluation of mode detection, we follow the weighted accuracy used in the audio key detection campaign in MIREX.[3] The instance is marked as 1 point when the prediction is the same mode as the ground truth, whereas 0.5 points is marked when the predicted mode is in perfect fifth above the ground truth (i.e. mode 1 is predicted as mode 5). The results are shown in Table 2. It is evident that all the tasks with ordinal mode template outperform their counterparts applying uniform template. In fact, the *t*-test for the prediction accuracy using uniform/ ordinal template yields a two-tailed *p*-value of 0.0013, which indicates a significant difference between the tasks adopting/ without the structural configuration of mode. To incorporate the statistics of annotated notes with the results of mode detection, above analyses lead to the finding that the information regarding the hierarchical configuration in mode contributes vastly in the mode detection.

5. PLAYING TECHNIQUE DETECTION AND ANALYSIS

In this section, we investigate the classification of left-hand playing technique in guqin performance, and also examine the interaction between global and local features during the classification. Frame-level together with mid-level and high-level features are obtained from audio data and annotations. Dynamic programming is applied to optimize the extracted pitch contour. Six types of playing techniques are then classified using a neural network. The statistical analysis indicates that the hierarchical tonal configuration is embedded in the distribution of portamento types. The classification results suggest that mid-level and high-level features can facilitate the recognition of technique type in local-level. Major components to improve the classification are identified. Challenging cases are further discussed.

5.1 Data extraction

In playing technique detection, three types of data are derived from audio recordings. For the first and second types of data, we acquire the spectrogram using CQT, and obtain the pitch salience function using Crepe following the procedure stated in Section 4.1. For the third type of data, we further apply dynamic programming to eliminate the

---

[3] https://www.music-ir.org/mirex/wiki/2019:Audio_Key_Detection

spikes in the estimated pitch contour. Given the output of Crepe $X \in \mathbb{R}^{K \times N}$, the pitch salience function at time s_i is $X[:, s_i]$, K is the number of frequency bins, and N is the number of frames, the pitch contour $\mathcal{S} := \{s_i\}_{i=1}^N$ is extracted with the following objective function:

$$\mathcal{S}^* = \arg\max_{\mathcal{S}} \sum_{i=1}^N X[:, s_i] - \lambda \sum_{i=2}^N |s_{i+1} - s_i|. \quad (1)$$

Equation (1) can be solved with dynamic programming [27]. The seconds term of (1) is to enhance the smoothness of the extracted pitch contours. The parameter λ controls the smoothness of the pitch contour, and in this work we set $\lambda = 10^{-3}$. This facilitates the processing of those guqin historical recordings with relatively low audio quality.

Six types of playing techniques are labelled for the pitch variation movement in the left hand: *none* (such as 直按 and 散音), *vibrato* (such as 吟 and 猱), *upward portamento* (such as 綽 and 上), *downward protamento* (such as 注 and 下), *inverted-U protamento* (such as 進復 and 撞), and *U protamento* (such as 退復 and 豆). The three types of data mentioned above are then divided into event segments according to the onset and duration annotations, and each segment corresponds to one of the six technique types. Figure 3 illustrates the featured pitch contour and examples of event segments in 6 technique types. The segments are then treated with padding, resulting the the input segment size of cqt (156, 60), pitch salience function (361, 60), and pitch contour (361,).

In order to investigate that how the meta-, structural music features connect with technique implements in local-level, we further extract 62 high-level and mid-level features including 2 *music-level features* (mode, # of event in music), 12 *event-level features* (event duration, event index, note degree, importance ranking of degree, also 8 descriptive statistical indicators of pitch contour (the mean, maximum, minimum, range, standard deviation, skewness, kurtosis, the time difference between the maximum and minimum)). The first six features (the two features in music-level and the first four features in event-level) are obtained from the prediction of our mode detection model as described in section 4. The eight descriptive indicators are extracted from dynamic programming pitch contour. Since the critical traits of each technique type may appear in different parts of the overall pitch contour, we also extract *intra-event-level features* by computing the eight descriptive statistical indicators for six intra-event pitch con-

Figure 3. The CQT (middle) and pitch salience segments (right) corresponding to the six technique types of guqin.

Data type	CQT	Salience	Contour
Frame-level only	0.743	0.845	0.842
With mid-, high-level	0.840	0.839	0.842

Table 3. The average accuracy of the playing technique detection task using various local and high-level features.

tour per event. To define the range of intra-event pitch contour, each segment is resampled to the mean length of all segments (0.7 seconds), and the resampled contour is divided by six hop windows (window size = 0.2 seconds, hop = 0.1 seconds), resulting 48 intra-event-level features for per musical event (8 indicators x 6 windows).

5.2 Playing technique classification

The playing technique is classified by a 10-layer neural network, which is composed of 2 convolutional layers (kernel = 3 x 3, stride = 1, fmap = 32), 1 self-attention layer (kernel = 1 x 1), and then followed by 3 convolutional layers (kernel = 3 x 3, stride = 1, fmap = 32), 3 fully-connected layers (neuron = 512), and a softmax output layer. The attention layer is constructed based on [28] to further investigate which high- and mid- level elements affect the results of technique classification in local level. The framework is implemented using Tensorflow. The training process is carried out by minimizing the cross-entropy between the model output and the one-hot label using Adam Optimizer with the learning rate of 10^{-5}. The dataset and the code to implement the model will be released afterward.

We design six experimental settings to examine the interaction between the global and local features. Three types of frame-level data: CQT, pitch salience function, and pitch contour with dynamic programming processing are inputted into the network respectively. They are then
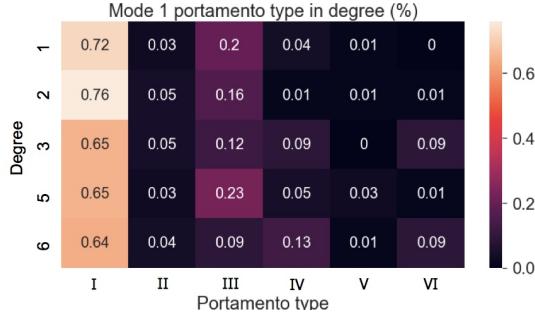
associated with high-level and mid-level features derived from the prediction from the previous mode detection and labelled annotations, resulting six experimental settings in total (as shown in Table 3). For each setting, 10-fold cross-validation is performed (roughly 1800 seconds of audio recordings for training, and 200 seconds for testing).

5.3 Results and discussion
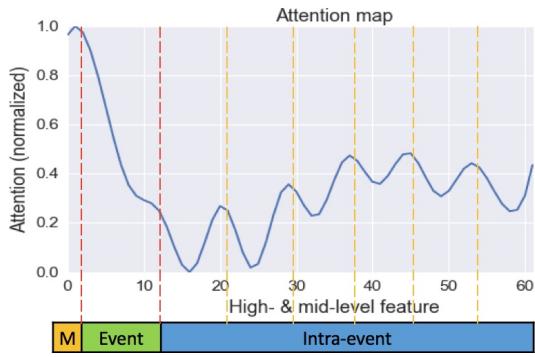
The statistical analysis shows that the distribution of portamento types reflects the high-level mode structure. As shown in Figure 4, except the type 1 technique without any pitch variation, type 3 portamento possesses highest occurrence ratio compared to other portamento types for important degrees in mode 1 (degree 1, 5, 2), and similar tendency can also be observed in other modes. This provides empirical evidence for the theoretical basis of guqin performing convention, where musicians frequently apply type 3 (upward) portamento to emphasize highlighted notes, and on the contrary, they tend to implement type 4 (downward) portamento to decorate trivial notes [16].

For the the results of technique classification, as can be seen in Table 3, the mid-level and high-level features contribute to the improvement of CQT data, but not for other two data types. This outcome may owing to the fact that the statistical descriptors of pitch contour are already contained in mid-level and high-level features, and such information of pitch contour may provide extra guidance for technique classification.

In order to further investigate that which are the decisive high-level and mid-level features to affect the technique classification, we follow the procedure in [28] and plot the self-attention with the feature map outputted from the neural network for 62 high- and mid- level features. All the values are normalized to the range between 0 and 1 for comparison. Figure 5 presents the attention map for the
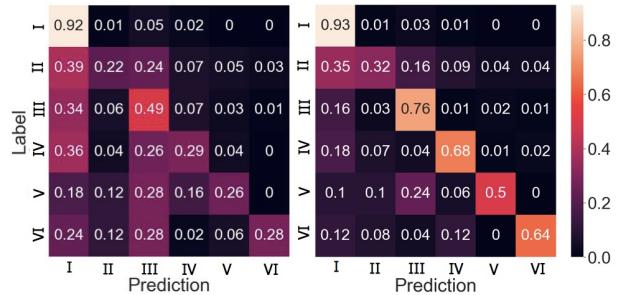
Figure 4. The statistics of the annotated portamento types (x-axis) in note degrees (y-axis) for mode 1.
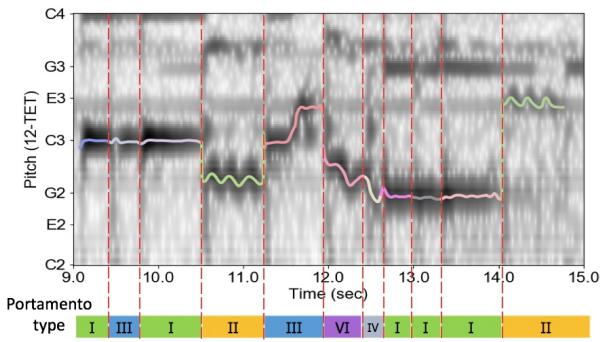
Figure 5. The attention map with feature map for 62 high- and mid- level features: music, event, and intra-event features. The red dotted lines indicate different feature types; the yellow dotted lines indicate features from 6 windows.

Figure 6. The confusion matrix of playing technique detection with CQT (left) and with CQT and high-, mid-level features (right).

Figure 7. The example of CQT (the background figure) and pitch contour segments (color curves) for different portamento types (color blocks at the bottom).

CQT and mid-, high-level features combination. As can be observed in the graph, the music-level features receive the most attention in the classification model (mean value = 0.982), compared to the event-level features (mean value = 0.500) and intra-event-level features (mean value = 0.293). Moreover, in intra-event features, an obvious peak occurs in each window, which corresponds to a specific feature: the distance between the maximum and minimum value within the window. Above observations suggest that high-level features (music and event features) are more effective to facilitate technique classification. And in mid-level features (intra-event features), the distance between the maximum and minimum value is the most adequate element to represent the pitch contour constitution.

To analyze individual portamento types in depth, it is noted in the confusion matrix (Figure 6) that types II to IV (especially type II, vibrato) are easily confused with type 1 (no variation) portamento. It is an unexpected result contradicting the studies on other instruments [10, 29], where vibrato can be easily identified. The inter-type confusion is mainly caused by the data imbalance in guqin performance, in which type I occurs much more frequently than other types (see Figure 4). In addition, we further examine the spectrogram and pitch contours for different portamento types (Figure 7), and notice that the pitch contours of type 1 portamento are not straight lines as expected, but exhibit unstable and irregular pitch drift, which can be easily confused with weak type 2 portamento, partic-

ularly when only the spectral features are analyzed. Furthermore, the two type 3 portamento segments in this example display diverse manners. The divergent behaviour within one single portamento type can rise the difficulty for accurate classification. The insufficient quality of historical recordings may also blur the distinction between portamento types and add additional challenges for the task.

6. CONCLUDING REMARKS

In this paper, we design and compile a new dataset consulting the theoretical basis of guqin performance. Mode detection is performed on the collected dataset, and playing technique classification are conducted using neural network. The results indicate that the hierarchical construction is crucial for mode detection, and the high-level and mid-level features contribute to improving the playing technique classification task.

This work verifies the conventional theory by empirical observations, in which statistical analysis confirmed the solid connection between the mode structure and the arrangement of playing technique. This study highlights the joint-relationship between the global tonal structure and the local distribution of playing technique, as well as bridges the knowledge gap between the music theory and performance practice. The findings in this study contribute insights for constructing an auto-evaluation system for music performance, or an educational system for musicians and music listeners.

7. REFERENCES

[1] T. Wang, *Qin Zhi (The Purpose of Qin* 琴旨*, 1744/1746). Si Ku Quan Shu (Wenyuange Edition, 1784) Vol. 220.* Taipei: The Commercial Press, 1986.

[2] L. W. Kong and T. Lee, "Automatic key partition based on tonal organization information of classical music," in *Proceedings of the the 15th International Society for Music Information Retrieval Conference*, 2014.

[3] F. Korzeniowski and G. Widmer, "Genre-agnostic key classification with convolutional neural networks," in *Proceedings of the the 19th International Society for Music Information Retrieval Conference*, 2018.

[4] C. Weiß and M. Schaab, "On the impact of key detection performance for identifying classical music styles," in *Proceedings of the the 16th International Society for Music Information Retrieval Conference*, 2015.

[5] G. K. Koduri, S. Gulati, P. Rao, and X. Serra, "Raga recognition based on pitch distribution methods," *Journal of New Music Research*, vol. 41, no. 4, pp. 337–350, 2012.

[6] K. K. Ganguli, S. Gulati, X. Serra, and P. Rao, "Data-driven exploration of melodic structures in hindustani music," in *Proceedings of the the 17th International Society for Music Information Retrieval Conference*, 2016.

[7] T. Nuttall, M. García-Casado, V. Núñez-Tarifa, R. C. Repetto, and X. Serra, "Contributing to new musicological theories with computational methods: the case of centonization in arab-andalusian music," in *Proceedings of the the 20th International Society for Music Information Retrieval Conference*, 2019.

[8] N. Pretto, B. Bozkurt, R. C. Repetto, and X. Serra, "Nawba recognition for arab-andalusian music using templates from music scores," in *Proceedings of the the 15th International Sound and Music Computing Conference*, 2018.

[9] F. Scherbaum, M. Muller, and S. Rosenzweig, "Analysis of the tbilisi state conservatory recordings of artem erkomaishvili in 1966," in *Proceedings of the the 7th International Workshop on Folk Music Analysis*, 2017.

[10] Y.-P. Chen, L. Su, and Y.-H. Yang, "Electric guitar playing technique detection in real-world recordings based on f0 sequence pattern recognition," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, 2015.

[11] L. Su, L.-F. Yu, and Y.-H. Yang, "Sparse cepstral and phase codes for guitar playing technique classification," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, 2014.

[12] J. Abeber, H. Lukashevich, and G. Schuller, "Feature-based extraction of plucking and expression styles of the electric bass guitar," in *Proceedings of IEEE Acoustics, Speech and Signal Processing (ICASSP)*, 2010.

[13] P.-C. Li, L. Su, Y.-H. Yang, and A. W. Y. Su, "Analysis of expressive musical terms in violin using score-informed and expression-based audio features," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, 2015.

[14] C.-C. Shih, P.-C. Li, Y.-J. Lin, Y.-L. Wang, A. W. Y. Su, L. Su, and Y.-H. Yang, "Analysis and synthesis of the violin playing style of heifetz and oistrakh," in *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17)*, 2017.

[15] J.-I. Liang, *The Event of 'Cadenza' in Liu Shui of Pan-Chuan Qin School and Ping Sha of Mei An Qin School: A Thesis on the Great Changes of Qin Music at the Transition from the Old to the New.* Taipei: Master's thesis, Taipei National University of the Arts, 2012.

[16] ——, *The Characteristics of Modal Structure in Qin Compositions of the Florid-Inflections Style Since the Late Ming Dynasty: Analysis and Application of the 'Ti-Yong' Theory in Wang Tan's Qin Zhi.* Taipei: PhD dissertation, Taipei National University of the Arts, 2018.

[17] S.-Y. Xu, *Da Huan Ge Qin Pu (Da Huan Ge Pavilion Anthology of Qin Music* 大還閣琴譜*, 1673). Reprinted in Qin Gu Ji Cheng (Anthology of Qin Music* 琴曲集成*) Vol. 10.* Bejing: Chung Hwa Book Company, 2010.

[18] S.-J. Chen, *Qin Xue Chu Jin (Introduction to the Scholarship of Qin* 琴學初津*, 1902). Reprinted in Qin Gu Ji Cheng (Anthology of Qin Music* 琴曲集成*) Vol. 28.* Bejing: Chung Hwa Book Company, 2010.

[19] M.-G. Gu, *Qin Xue Bei Yao (Required Essentials of Qin Scholarship* 琴學備要*).* Shanghai: Shanghai Music Publishing House, 2009.

[20] W.-C. Chou, "Single tones as musical entities: An approach to structured deviations in tonal characteristics," *American Society of University Composers*, vol. 3, pp. 86–97, 1970.

[21] C. H. Lee, *Asian music.* Yang-Chih Book, 2015.

[22] W. G. Wu, "The modal system of qin and its verification 1," *Musicology in China*, vol. 1, pp. 5–30, 1997.

[23] ——, "The modal system of qin and its verification 2," *Musicology in China*, vol. 2, pp. 88–109, 1997.

[24] P. Guo, *Inimitable Sound and Treasures: The Collection of Historical Audio and Video Tracks from Guqin Legends by GuoPeng (*絕響: 國鵬輯近世琴人音像遺珍*).*

[25] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[26] J. W. Kim, J. Salamon, P. Li, and P. Bello, "Crepe: A convolutional representation for pitch estimation," in *Proceedings of IEEE Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[28] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[29] C. Wang, E. Benetos, V. Lostanlen, and E. Chew, "Adaptive time-frequency scattering for periodic modulation recognition in music signals," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019.

SEMI-SUPERVISED LEARNING USING TEACHER-STUDENT MODELS FOR VOCAL MELODY EXTRACTION

Sangeun Kum[1]   **Jing-Hua Lin**[2]   **Li Su**[2]   **Juhan Nam**[1]

[1] Graduate School of Culture Technology, KAIST, South Korea
[2] Institute of Information Science, Academia Sinica, Taiwan

keums@kaist.ac.kr, jhlin@iis.sinica.edu.tw, lisu@iis.sinica.edu.tw, juhan.nam@kaist.ac.kr

ABSTRACT

The lack of labeled data is a major obstacle in many music information retrieval tasks such as melody extraction, where labeling is extremely laborious or costly. Semi-supervised learning (SSL) provides a solution to alleviate the issue by leveraging a large amount of unlabeled data. In this paper, we propose an SSL method using teacher-student models for vocal melody extraction. The teacher model is pre-trained with labeled data and guides the student model to make identical predictions given unlabeled input in a self-training setting. We examine three setups of teacher-student models with different data augmentation schemes and loss functions. Also, considering the scarcity of labeled data in the test phase, we artificially generate large-scale testing data with pitch labels from unlabeled data using an analysis-synthesis method. The results show that the SSL method significantly increases the performance against supervised learning only and the improvement depends on the teacher-student models, the size of unlabeled data, the number of self-training iterations, and other training details. We also find that it is essential to ensure that the unlabeled audio has vocal parts. Finally, we show that the proposed SSL method enables a baseline convolutional recurrent neural network model to achieve performance comparable to state-of-the-arts.

1. INTRODUCTION

One of the key elements in the success of deep learning is a large amount of labeled data. However, when the labeled data is scarce in a given task, it can be a bottleneck in leveraging the power of deep neural networks. The issue has been found in many music information retrieval (MIR) tasks as well. Among others, melody extraction research has suffered from it as pitch labeling requires experienced annotators to handle the annotation tool and the process is extremely labor-intensive [1].

The lack of labeled data in melody extraction research has been tackled in several different ways. A popular

method to alleviate the issue is data augmentation which increases labeled data by transforming the input audio, for example, using pitch-shifting [2–4]. Data augmentation, however, has the limitation in covering the diversity in the input space. Another approach is using multi-track audio data [5–7]. This allows to use monophonic pitch tracking algorithms for the melodic source and therefore it expedites laborious the pitch labeling. However, multi-track recording datasets often maintain individual tracks as stem files where multiple similar sound sources can be mixed (e.g., main vocal and backing vocal). Therefore, obtaining clean pitch labels from multi-track audio can be not straightforward [8, 9]. Recently, melody MIDI files, which are more easily accessible, have been utilized to guide melody extraction from audio with transfer learning techniques from the symbolic to audio domain [3, 10]. MIDI data exhibit greater flexibility than audio on data augmentation, but still face limitations on representing natural pitch contours of singing voice, which usually contain subtle variations such as vibrato and portamento.

Semi-supervised learning (SSL) is another but more general strategy to address the lack of labeled data. SSL uses a large amount of unlabeled data, which is usually easy to collect, jointly with labeled data. A popular class of SSL methods is based on self-training in the teacher-student framework. Recent works have combined random data augmentation with the SSL methods to encourage the model to produce robust output even when input is perturbed. This approach has achieved state-of-the-art performance on image classification [11–13], speech recognition [14], and audio classification [15]. There are a few MIR researches that used the teacher-student framework to address the lack of labeled data, for example, in automatic drum transcription [16] and singing voice detection [17, 18]. However, to the best of our knowledge, recent advances in SSL methods that leverage the power of deep neural networks and random data augmentation in the teacher-student framework have been not studied yet in the music domain.

In this paper, we apply the SSL methods to vocal melody extraction with the following contributions. First, we present the SSL methods for vocal melody extraction leveraging large-scale unlabeled music datasets. This prevents the model from overfitting to small labeled data and improve the performance. Second, we compare three setups of teacher-student models along with various audio

data augmentation techniques. We show the model with the consistency regularization is most effective. Third, we investigate effective SSL strategies by exploring joint training, the size of unlabeled data, and the number of self-training iterations. Fourth, we show that the proposed teacher-student training method enables a baseline convolutional recurrent neural network model to achieve performance comparable to state-of-the-arts. Finally, apart from the SSL method, we propose large-scale testing data artificially generated from unlabeled data using an analysis-synthesis framework, considering the lack of labeled data even at the testing stage. Evaluation on the diverse and sizable test set will reinforce the effectiveness of the proposed method. For reproducibility, the source code and pre-trained model used in this paper are available online [1].

2. RELATED WORK

The teacher-student framework has been previously studied in several MIR tasks to address the lack of labeled data. Wu and Lerch applied the approach to automatic drum transcription [16]. They used multiple teacher models based on non-negative matrix factorization (NMF) trained with different datasets and a student model based on deep neural network trained with labels from the teachers. They showed that the student model outperforms the teacher models. However, it was not a self-training setting where the teacher model is repeatedly replaced with an improved student model. Schlüter explored the self-training for singing voice detection [17]. They first trained a convolutional neural network (CNN) on the original labels with low-granularity, then a second network on pseudo-labels with high-granularity from the first network, and a third network on the summarized saliency maps from the second network. They showed this self-improvement worked up to the third network. However, they conducted the self-training on weakly-labeled data in the context of multiple-instance learning and did not used any unlabeled data. Recently, Meseguer-Brocal et al. used the teacher-student paradigm for singing voice detection to create a large-scale time-aligned vocal melody and lyrics dataset [18]. They consistently improved the teacher model by increasing the correlation between the prediction of the model and the time-aligned lyrics annotation.

3. METHODS

3.1 Model Architecture

Recent melody extraction algorithms have used CNN [9, 19, 20] and its variants [4, 21, 22] as a standard architecture. Since we focus on the effectiveness of SSL in this paper, we employ a previously proposed convolutional recurrent neural network (CRNN) which was a baseline architecture in [4]. The CRNN architecture consists of 4 ResNet blocks and a bi-directional long short-term memory layer. We first merge the audio waveforms into a mono channel and downsample them to 8 kHz. We then calculate the logarithmic-magnitude spectrogram using short-time

[1] https://github.com/keums/melodyExtraction_SSL

Algorithm 1: Train SSL Models

Train a teacher network T_1 on labeled data
  $\mathcal{D} = \{(x_d, y_d) : d \in (1, ..., N)\}$;
Generate augmented data
  $\tilde{\mathcal{U}} = \{\tilde{x}_u = RAA(x_u) : u \in (1, ..., M)\}$ from
  unlabeled data $\mathcal{U} = \{x_u : u \in (1, ..., M)\}$;
for $i = 1$ to k **do**
  | Use T_i to generate pseudo labels for $\mathcal{U}$ (or $\tilde{\mathcal{U}}$);
  | Train student network S_i using both $\mathcal{D}$ and $\mathcal{U}$
  | (or $\tilde{\mathcal{U}}$) as training data;
  | $T_{i+1} = S_i$;
end

Fourier transform with a 1024-point Hann window and an 80-point hop size. The CRNN architecture takes 31 consecutive frames of the spectrogram as input and predicts a pitch label quantized with a resolution of 1/8 semitone and ranged from E2 (82.4 Hz) to B6 (1975.7 Hz). The size of the output layer is 442, including a non-vocal label.

3.2 SSL in the Teacher-Student Framework

Our SSL method is based on self-training in the teacher-student framework where the teacher model is first trained with labeled data and then the student model is trained with artificial labels generated from the teacher model using unlabeled data. The artificial labels can be the prediction distribution vector [11, 12] or one-hot vector determined by the class with a highest confidence [13, 23]. We formally describe the overall procedure in Algorithm 1. We first train the initial teacher model T_1 using only labeled data $\mathcal{D}$ where x_d are labeled examples and y_d are one-hot reference labels. For unlabeled data $\mathcal{U}$ where x_u are unlabeled examples, we use random data augmentation to generate noisy input data $\tilde{\mathcal{U}}$ where $\tilde{x}_u$ are noisy unlabeled examples. RandAudioAugment (RAA) is an audio version of random data augmentation method which is described in Section 3.4. While it is more effective to use random data augmentation on the student model only in image classification [12], we also try applying it for both teacher and student models for ablation study. Once we train the student model jointly with the labeled data and unlabeled data (with pseudo labels), we replace the teacher model with the student model. We repeat the same pseudo labeling and the training with a new student model.
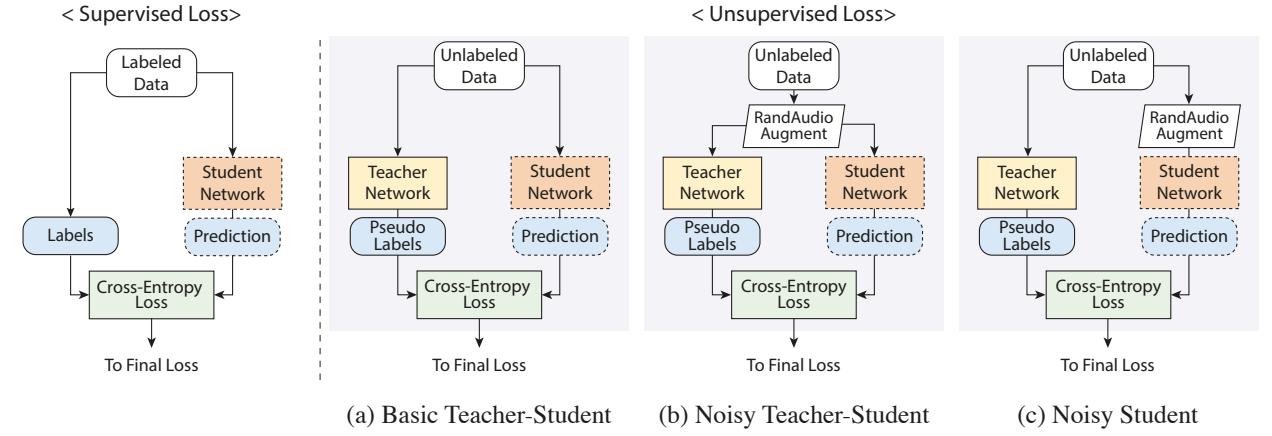
3.3 Proposed Teacher-Student Models

Our proposed Teacher-Student models are illustrated in Figure 1. The supervised loss $\mathcal{L}_D$ is computed with labeled data and defined as:

$$\mathcal{L}_D = \frac{1}{N} \sum_{d=1}^{N} H(y_d, p(y|x_d; \theta_s)) \qquad (1)$$

where $H(\cdot)$ denotes the cross-entropy between the pitch label y_d and pitch prediction $p(y|x)$, and θ_s denotes a set of parameters of the student model. The supervised loss is a common loss term of the three investigated

(a) Basic Teacher-Student   (b) Noisy Teacher-Student   (c) Noisy Student

Figure 1. Diagram of the three Teacher-Student models.

teacher-student models. Each of them are explained below.

Basic Teacher-Student is a fundamental teacher-student framework that uses the unlabeled data $\mathcal{U}$ but trains the student network with the pseudo labels generated from the teacher network. The final loss of Basic Teacher-Student $\mathcal{L}_B$ is defined as

$$\mathcal{L}_B = \mathcal{L}_D + \frac{1}{M} \sum_{u=1}^{M} H(y_u, p(y|x_u; \theta_s)) \qquad (2)$$

where y_u is the pseudo labels on $\mathcal{U}$ generated by the teacher network, i.e. $y_u = p(y|x_u; \theta_t)$ where θ_t to denote the parameters of teacher network. The basic teacher-student model is illustrated in Figure 1(a).

Noisy Teacher-Student takes noisy unlabeled data $\tilde{\mathcal{U}}$ for both of the teacher and student networks using RAA and the rest is the same as the basic teacher-student model. The final loss of Noisy Teacher-Student $\mathcal{L}_N$ is defined as

$$\mathcal{L}_N = \mathcal{L}_D + \frac{1}{M} \sum_{u=1}^{M} H(\tilde{y}_u, p(y|\tilde{x}_u; \theta_s)) \qquad (3)$$

where $\tilde{y}_u$ is a prediction on $\tilde{\mathcal{U}}$ generated by the teacher network, i.e. $\tilde{y}_u = p(y|\tilde{x}_u; \theta_t)$. The noisy teacher-student model is illustrated in Figure 1(b).

Noisy Student takes noisy unlabeled data $\tilde{\mathcal{U}}$ only for the student network while the teacher network takes unnoised input $\mathcal{U}$ to generate the pseudo labels. The idea is that the student should produce consistent outputs that minimize the difference from the teacher even though the input is perturbed [12]. This notion is also similar to consistency regularization [24, 25]. The final loss of Noisy Student $\mathcal{L}_C$ is defined as

$$\mathcal{L}_C = \mathcal{L}_D + \frac{1}{M} \sum_{u=1}^{M} H(y_u, p(y|\tilde{x}_u; \theta_s)) \qquad (4)$$

The noisy student model is illustrated in Figure 1(c).

3.4 Data Augmentation

We conducted pitch-shift by $\pm$ 1,2 semitone on the labeled data $\mathcal{D}$ (audio and corresponding labels). In the melody extraction task, it has shown that pitch-shifting can improve the generality and performance of the model by increasing the amount of audio and label pairs for different f_0 [2, 26]. For data augmentation of unlabeled data $\mathcal{U}$, we propose RandAudioAugment (RAA) inspired by RandAugment [27], which is a method of randomly applying different kinds of transformations to increase image data. RAA converts audio by randomly selecting multiple audio effects as follows: audio equalizer (low-shelf, high-shelf), filters (low-pass, high-pass), overdrive, phaser, and reverb. Here, we use *pysndfx* that is a Python library designed for applying effects to audio files [2]. We sampled a random magnitude of each transformation from a predefined range. The implementation details for RAA are also described in the source code.

3.5 Data Selection

The SSL algorithm using large-scale unlabeled data may suffer from labeling noise. Unlabeled data are highly likely to have audio without vocals. Filtering only high-confidence examples or the top-K examples in image classification has demonstrated to be an effective method to handle the labeling noise [12, 28]. Likewise, we performed data selection so that only the tracks with vocal ratios exceeding a threshold were used for training. To estimate the ratio of vocals included in the track, we used our singing voice detector [3] based on CNN based on [29]. Considering the distribution of vocal ratio in the FMA, we set the threshold to 0.3.

4. DATASETS

Table 1 shows the simple statistics of the labeled and unlabeled training datasets and test datasets.

---

[2] https://github.com/carlthome/python-audio-effects
[3] https://github.com/keums/SingingVoiceDetection

	Dataset	Number of Tracks	Total Length
Training (Labeled)	RWC	100	6h 47m
	MedleyDB	61	2h 39m
	iKala	262	2h 6m
Training (Unlabeled)	In-house	535	6h 21m
	FMA_small	3,521 / 8,000	25h / 60h
	FMA_medium	10,639 / 25,000	89h / 208h
	FMA_large	40,505 / 106,574	337h / 888h
Test	ADC04	12	4m
	MIREX05	9	4m
	MedleyDB	12	43m
	AST218	218	14h 53m

Table 1. Description of datasets. In FMA, The two numbers indicate tracks with vocal (the vocal ratio above 0.3) and all tracks respectively.

4.1 Labeled Data

We used the three labeled datasets (RWC [30], MedleyDB [6], and iKala [7]) and split them into a train and validation set following [9]. We augmented the training data by pitch-shifting with $\pm$ 1,2 semitone. The total length of the labeled training data amounts to about 55 hours after the data augmentation.

4.2 Unlabeled Data

As to unlabeled data, we used an in-house dataset crawled from YouTube and the Free Music Archive (FMA) [31]. The in-house dataset is pop songs with vocals recorded in a variety of environments. It includes both public-released and user-uploaded tracks. FMA is a large-scale open dataset containing up to 106,574 tracks and covers 161 genres of music. We used FMA for performance comparison on data scalability. The FMA has three different subsets depending on the number of the track and genre included: FMA_small (FMA_S), FMA_medium (FMA_M), and FMA_large (FMA_L). We selected vocal tracks from them as described in Section 3.5 and denote the selected versions as FMA_{Sv}, FMA_{Mv}, and FMA_{Lv}, respectively. We augmented the unlabeled datasets via RAA during training as described in Section 3.4.

4.3 Test Data

4.3.1 Public Test Sets

We used three public test sets (ADC04 [4], MIREX05 [4], and MedleyDB) to evaluate the performance of vocal melody extraction. In this study, we excluded non-vocal tracks from ADC04 and MIREX05, and used songs not included in training data for MedleyDB. To obtain the ground truth for singing voice in MedleyDB, we adopted its 'MELODY2' annotations. These three datasets have been commonly used to compare the performance of melody extraction. However, the number of tracks and the total length are very limited as shown in Table 1.

4.3.2 Proposed Large-Scale Test Set

To make up the scarcity of testing data for evaluating singing voice extraction algorithms, we propose a new test set composed of DSD100 [32] and MusDB18 [33]. The two multitrack datasets were originally designed for source separation. Each track has four isolated stems: vocals, drums, bass, and others. Following the analysis/synthesis framework [8], the singing melodies for 218 selected tracks [5] were synthesized with automatically generated f_0 contours. In detail, for each song, we extracted the melody of the vocals with five different pitch trackers, and each f_0 information along with the vocal audio was fed into the WORLD [34] (D4C edition [35]) vocoder to reproduce five monophonic variations of the vocal stem. The original vocal audio was parameterized into harmonic and aperiodic spectral envelopes, and then resynthesized with provided pitch contours. Then a mask was applied to filter intervals without f_0 information. For remixing, the amplitude of the synthesized vocal was weighted to that of the original vocal stem, and the rest stems were directly summed up as accompaniments, then mixed with the weighted synthesized vocal that perfectly matched the f_0 annotation. These 1,090 polyphonic mixtures with accurate and automatic annotations constitute the proposed analysis/synthesis test set, AST218 [6].
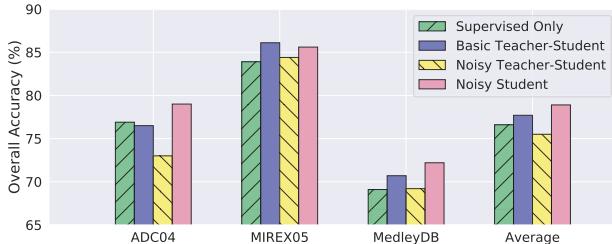
Each track in AST218 has five variations whose vocal melody was annotated separately with five different pitch estimators: CREPE [36] (with confidence threshold of 0.5 and 0.7), pYIN [37], and Lu&Su [3] (with time step of 10 and 20ms), as they have different merits. Since there is no exact way to pinpoint a common optimal confidence threshold across the entire dataset, we chose two different threshold values for CREPE: one is 0.5, suffering from high false positive (FP) but preserving details; the other threshold is 0.7, acceptable FP though sacrificing some recall. pYIN was chosen for it has even lower FP while producing stable and continuous melodic lines when the vocal stem is monophonic. However, it is not stable in the pholyphonic scenario, which is universal in DSD100 and MusDB18. In need of other polyphonic-based melody estimators to balance the f_0 quality, we chose two time step setups of the Lu&Su model: 20ms, at which this model is optimized; and 10ms, which provides more continuous predictions and offers alternative pitch contours when encountering multiple melodic vocal lines.

The analysis/synthesis framework has been practiced successfully in evaluating monotonic pitch trackers [36]. As a sanity check, we evaluated several patchCNN [19] setups on the original and resynthesized ADC04, MIREX05, and MedleyDB. The differences of OA are within $\pm$ 2–5%, which is acceptable, meaning this framework is also applicable for polyphonic test set generation.

When evaluating vocal extraction algorithms on AST218, we averaged the scores from the five variations. Our pilot study shows that these five pitch contours reach

---

[4] http://labrosa.ee.columbia.edu/projects/melody/

[5] Songs that appear in MedleyDB were excluded for they were part of the training data, but songs in MusDB18 having counterparts in DSD100 were not removed for they are not exactly identical. Additionally, 12 songs that do not have discernible vocal melodies were also excluded.

[6] https://sites.google.com/view/mctl/resource

Figure 2. Comparison with supervised-learning model and three student models on three test sets.

Figure 3. Comparison with pre-training, fine-tuning, and joint training methods on three test sets.

consensus over a majority of frames, while the estimations differ for tricky frames. Rather than manually check on the estimated f_0, we used AST218 in an ensemble manner, fully leveraging the spirit of automatic pitch annotation.

5. EXPERIMENTS

5.1 Experimental Setup

5.1.1 Training Details

We used the CRNN architecture with residual connections and bi-directional long short-term memory in all experiments. The implementation of the model was consistent with that of the main network of [4]. We trained our models using Adam optimizer for 70 epochs on 2 GPUs. The initial learning rate was set to 0.003 in all the experiments. We used a learning rate schedule that reduces the learning rate by 0.7 times if validation accuracy did not increase within three epochs. The model and the training procedures were implemented using Keras [7] [38].
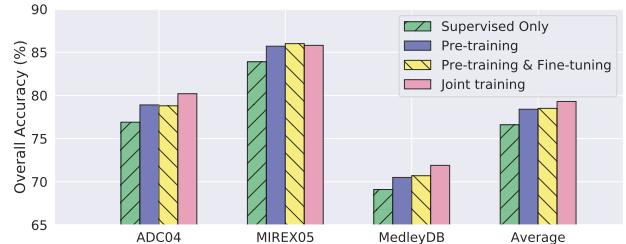
5.1.2 Evaluation

To evaluate the performance of melody extraction, we mainly used overall accuracy (OA) which combines the accuracy of pitch estimation with voice detection. We also used three metrics raw pitch accuracy (RPA) for pitch estimation, and voicing recall (VR) and voicing false alarm (VFA) for voice detection [39]. These metric are computed by *mir_eval* [40] library designed.

5.2 Experiment 1: Teacher-Student Models

Our first experiment is to demonstrate the efficacy of the proposed Teacher-Student models for SSL. In this experiment, we trained three Teacher-Student models described in Section 3.3 using an in-house dataset as unlabeled data. We evaluated the performance of each model on ADC04, MIERX05, and MedleyDB, which have been used as standard test sets for evaluation. As shown in Figure 2, the basic teacher-student model can achieve 1.1% higher average OA than the supervised-only model which has 77.7% average OA. This confirms the possibility of using unlabeled data to improve the performance of melody extraction. Our experiment also shows that the noisy student model outperforms all the others, having 78.9% average OA.

The noisy student model increases OA by 3.1% with respect to the supervised-only model in MedleyDB, which is

especially a challenging dataset because it contains tracks that are difficult to distinguish between vocals and background music, or tracks with excessive audio effects. The results indicate that the student network can be trained reliably using the noisy student model, even if the initial teacher network is not robust to diverse noise. Meanwhile, the performance of the noisy teacher-student has deteriorated, being worse than the supervised-only model. This degradation is probably because the noised teacher model is not generating reliable pseudo labels.
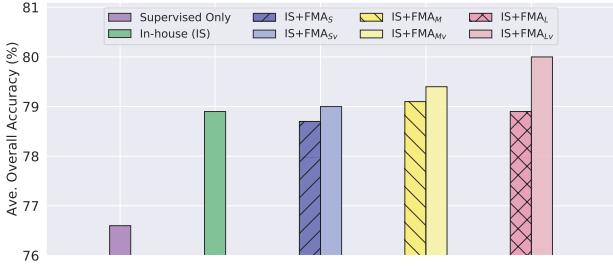
5.3 Experiment 2: Joint Training vs. Fine-Tuning

The training methods of the teacher-student framework can be divided into three approaches depending on how $\mathcal{D}$ and $\mathcal{U}$ are used for training: pre-training on only $\mathcal{U}$ and then fine-tuning on $\mathcal{D}$; joint-training on both $\mathcal{U}$ and $\mathcal{D}$ simultaneously. Figure 3 compares the results among pre-training, fine-tuning, and joint training for the noisy student model. The jointly trained model achieves 0.8% higher average OA than the fine-tuned model, with the highest results on MedleyDB. This indicates that joint training on unlabeled data and labeled data would help the networks produce a decision boundary that better reflects real music [41]. Interestingly, the average OA of the pre-trained model only on unlabeled data is higher than that of the supervised learning model. This suggests that the distribution of unlabeled data is similar to that of labeled data. Considering that the in-house dataset consists of pop songs with vocals, the in-house dataset can be seen as having a similar tendency to the labeled data. It provides insight into the data selection in the next experiment.
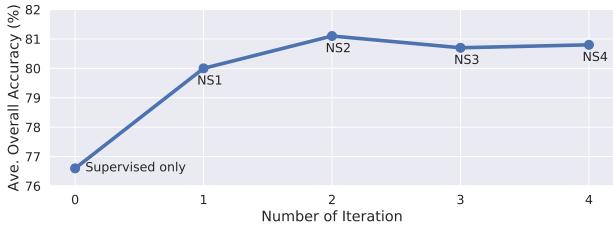
5.4 Experiment 3: Size of Training Data

We investigated the importance of the size and validity of unlabeled data. To explore the effect of the size of unlabeled data, we started with the in-house dataset as training data for the noisy student model and progressively included larger subsets of FMA. The results can be seen in Figure 4. Although the FMA data set contains more numerous tracks than the in-house dataset, the average OA of FMA_S and FMA_L is lower than that of the model trained only with the in-house dataset. Note that the proposed model focuses only on vocal melodies. As a result, teacher models may suffer from labeling noise generated by numerous instrument tracks included in the FMA. In addition, all labels on the instrumental track are classified as non-vocal pitch, resulting in data imbalance.

---

[7] We used Keras 2.3.0, Accessed: 15 May 2020

Figure 4. Comparison with Noisy Students on varied sizes of unlabeled datasets. The subscript 'v' denotes a selected subset of FMA whose vocal ratio exceeds a threshold. We use the average of OA for three public test sets.

Figure 5. Effect of iteration training for Noisy Students.

To confirm the validity of the dataset, we performed data selection for each FMA subset as mentioned in Section 3.5 and used them to train each student model. Interestingly, as the size of the $\mathcal{U}$ increases, the performance of each model tends to be significantly improved. For example, FMA_{Lv} achieves an average OA of 80.2%, which is 3.6% higher than the supervised-only model. This indicates that effective SSL requires a large amount of $\mathcal{U}$ with a similar distribution for $\mathcal{D}$.

5.5 Experiment 4: Iterative Training

We iterated the self-training 4 times for the noisy student model using the in-house dataset and FMA_{Lv}. The results are illustrated in Figure 5. We observe that the performance continuously increases up to 2 iterations achieving the highest average OA of 81.1%. Generally, self-training tends to amplify the error caused by labelling noise during training. However, the noisy student model trained on large-scale unlabeled data can help overcome this difficulty. Nevertheless, increasing the number of training iterations three or more times does not improve performance, and rather slightly lower the accuracy.

5.6 Comparison with State-of-the-Arts

We compared the supervised-only model (as a baseline) and proposed the noisy student model (NS) with four recent melody extraction algorithms based on deep neural networks: the patch-based CNN (patchCNN) [19], the deep salience map (DSM) [9], the streamlined encoder/decoder network (segNet) [21], and the joint detection and classification model (JDC) [4], which have open-sourced codes with vocal mode. Each method was run with its default parameters, and then evaluated on the three conventional test sets and the newly introduced AST218.

Methods	ADC04	MIREX05	MedleyDB	AST218
PatchCNN [19]	76.9 / 72.9	69.7 / 73.8	44.0 / 59.3	42.3 / 59.7
DSM [9]	89.2 / 72.2	87.7 / 80.1	80.6 / 75.4	38.9 / 68.3
SegNet [21]	88.7 / 83.3	82.6 / 80.0	70.6 / 75.5	41.5 / 68.1
JDC [4]	**90.6 / 83.5**	**91.4 / 87.4**	72.7 / 78.1	55.8 / **75.4**
Baseline	78.7 / 76.8	79.9 / 81.5	57.2 / 70.7	**56.3** / 69.7
Proposed (NS)	90.4 / 82.2	90.4 / 85.9	**76.3 / 79.2**	54.2 / 74.2

Table 2. Vocal melody extraction results in terms of (RPA / OA) of the proposed and other methods on various test sets. The proposed model is iterated the self-training two times using the in-house dataset and FMA_{Lv}.

Methods	ADC04	MIREX05	MedleyDB	AST218
PatchCNN	91.8 / 46.1	80.3 / **11.6**	60.1 / 22.4	61.6 / 26.0
DSM	95.7 / 61.1	93.9 / 29.4	**85.4** / 26.6	44.6 / 7.7
SegNet	95.2 / 38.5	92.2 / 24.0	78.8 / 21.7	51.7 / 10.0
JDC	96.7 / 40.2	**97.5** / 18.5	80.5 / 18.3	64.7 / **8.6**
Baseline	92.6 / **33.8**	89.1 / 15.2	71.0 / **16.7**	**72.0** / 19.2
Proposed (NS)	**97.4** / 42.1	97.3 / 20.4	83.3 / 19.1	61.6 / 9.4

Table 3. Voicing detection results in terms of (VR / VFA) of the proposed and other methods on various test sets.

Besides, we report the frame-level scores instead of song-level ones to settle uneven song lengths.

Table 2 and Table 3 list the results of each method on the four test sets. In general, performances of the proposed NS model are comparable to other supervised-learning-based methods and even outperforms others in MedleyDB, and it effectively improves the OA of the baseline by 4.5–8.5%. The overall rankings of VR and VFA vary across the test sets, but the behavior converges in terms of OA. One can also observe that the AST218 is the most challenging in the majority of cases. In such a dataset, the performance of the NS model shows that the proposed method is robust to large-scale evaluation. However, the NS model improves the baseline except for VR and RPA in AST218. This result might be because the simple rule-based remixing of vocal and accompaniment tracks in AST218 is different from the artistic practice of mixing engineers, which can affect voicing detection and, in turn, RPA.

6. CONCLUSION

This study provides a framework of semi-supervised learning using the teacher-student model for vocal melody extraction. We compared three setups of teacher-student models and revealed that the NS model is the most effective and robust to real-world music where various noises can be present. We showed that large-scale unlabeled data is effective when they are properly selected. We found that iterative training for the teacher-student model helps improve performance. We also confirmed the effectiveness of the proposed method by evaluating it on artificial large-scale test data generated from automatically annotated multitrack data. Although these findings are based only on vocal melody extraction, we believe our method can be extended to other MIR tasks that suffer from the lack of labeled data such as automatic music transcription and chord recognition.

7. ACKNOWLEDGEMENT

8. REFERENCES

[1] J. Salamon, "What's broken in music informatics research? three uncomfortable statements," in *Proc. of the 36th International Conference on Machine Learning, PMLR 97*, 2019.

[2] S. Kum, C. Oh, and J. Nam, "Melody extraction on vocal segments using multi-column deep neural networks," in *Proc. ISMIR*, 2016, pp. 819–825.

[3] W.-T. Lu and L. Su, "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning," in *Proc. ISMIR*, 2018, pp. 521–528.

[4] S. Kum and J. Nam, "Joint detection and classification of singing voice melody using convolutional recurrent neural networks," *Applied Sciences*, vol. 9, no. 7, p. 1324, 2019.

[5] C.-L. Hsu and J.-S. R. Jang, "On the improvement of singing voice separation for monaural recordings using the mir-1k dataset," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 2, pp. 310–319, 2009.

[6] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research." in *Proc. ISMIR*, 2014, pp. 155–160.

[7] T.-S. Chan, T.-C. Yeh, Z.-C. Fan, H.-W. Chen, L. Su, Y.-H. Yang, and R. Jang, "Vocal activity informed singing voice separation with the ikala dataset," in *Proc. ICASSP*, 2015, pp. 718–722.

[8] J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gómez, and J. P. Bello, "An analysis/synthesis framework for automatic f0 annotation of multitrack datasets." in *Proc. ISMIR*, 2017, pp. 71–78.

[9] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for f0 estimation in polyphonic music," in *Proc. ISMIR*, 2017.

[10] Y. Gao, B. Zhu, W. Li, K. Li, Y. Wu, and F. Huang, "Vocal melody extraction via dnn-based pitch estimation and salience-based pitch refinement," in *Proc. ICASSP*, 2019, pp. 1000–1004.

[11] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 5050–5060.

[12] Q. Xie, E. Hovy, M.-T. Luong, and Q. V. Le, "Self-training with noisy student improves imagenet classification," *arXiv preprint arXiv:1911.04252*, 2019.

[13] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *arXiv preprint arXiv:2001.07685*, 2020.

[14] L. Mošner, M. Wu, A. Raju, S. H. K. Parthasarathi, K. Kumatani, S. Sundaram, R. Maas, and B. Hoffmeister, "Improving noise robustness of automatic speech recognition via parallel data and teacher-student learning," in *Proc. ICASSP*. IEEE, 2019, pp. 6475–6479.

[15] K. Lu, C.-S. Foo, K. K. Teh, H. D. Tran, and V. R. Chandrasekhar, "Semi-supervised audio classification with consistency-based regularization," *Proc. Interspeech*, pp. 3654–3658, 2019.

[16] C.-W. Wu and A. Lerch, "Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data," in *Proc. ISMIR*, 2017, pp. 613–620.

[17] J. Schlüter, "Learning to pinpoint singing voice from weakly labeled examples." in *Proc. ISMIR*, 2016, pp. 44–50.

[18] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Dali: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm," in *Proc. ISMIR*, 2018.

[19] L. Su, "Vocal melody extraction using patch-based CNN," in *Proc. ICASSP*, 2018, pp. 371–375.

[20] M.-T. Chen, B.-J. Li, and T.-S. Chi, "Cnn based two-stage multi-resolution end-to-end model for singing melody extraction," in *Proc. ICASSP*, 2019, pp. 1005–1009.

[21] T.-H. Hsieh, L. Su, and Y.-H. Yang, "A streamlined encoder/decoder architecture for melody extraction," in *Proc. ICASSP*. IEEE, 2019, pp. 156–160.

[22] H. Chou, M.-T. Chen, and T.-S. Chi, "A hybrid neural network based on the duplex model of pitch perception for singing melody extraction," in *Proc. ICASSP*, 2018, pp. 381–385.

[23] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, 2013, p. 2.

[24] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *Advances in neural information processing systems*, 2016, pp. 1163–1171.

[25] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," *arXiv preprint arXiv:1904.12848*, 2019.

[26] R. M. Bittner, B. McFee, and J. P. Bello, "Multitask learning for fundamental frequency estimation in music," *arXiv preprint arXiv:1809.00381*, 2018.

[27] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical data augmentation with no separate search," *arXiv preprint arXiv:1909.13719*, 2019.

[28] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, "Billion-scale semi-supervised learning for image classification," *arXiv preprint arXiv:1905.00546*, 2019.

[29] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks." in *Proc. ISMIR*, 2015, pp. 121–126.

[30] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: popular, classical and jazz music databases." in *Proc. ISMIR*, 2002, pp. 287–288.

[31] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," in *Proc. ISMIR*, 2017.

[32] A. Liutkus, F.-R. Stöter, Z. Rafii, D. Kitamura, B. Rivet, N. Ito, N. Ono, and J. Fontecave, "The 2016 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation - 12th International Conference*. Springer, 2017, pp. 323–332.

[33] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[34] M. Morise, F. Yokomori, and K. Ozawa, "World: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[35] M. Morise, "D4c, a band-aperiodicity estimator for high-quality speech synthesis," *Speech Communication*, vol. 84, pp. 57–65, 2016.

[36] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *Proc. ICASSP*. IEEE, 2018, pp. 161–165.

[37] M. Mauch and S. Dixon, "pyin: A fundamental frequency estimator using probabilistic threshold distributions," in *Proc. ICASSP*. IEEE, 2014, pp. 659–663.

[38] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[39] J. Salamon, E. Gómez, D. P. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

[40] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "mir_eval: A transparent implementation of common mir metrics," in *Proc. ISMIR*, 2014.

[41] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," in *Advances in Neural Information Processing Systems*, 2018, pp. 3235–3246.

MUSPY: A TOOLKIT FOR SYMBOLIC MUSIC GENERATION

Hao-Wen Dong   Ke Chen   Julian McAuley   Taylor Berg-Kirkpatrick

University of California San Diego

`{hwdong, knutchen, jmcauley, tberg}@ucsd.edu`
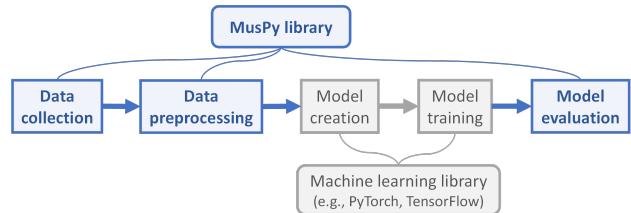
ABSTRACT

In this paper, we present MusPy, an open source Python library for symbolic music generation. MusPy provides easy-to-use tools for essential components in a music generation system, including dataset management, data I/O, data preprocessing and model evaluation. In order to showcase its potential, we present statistical analysis of the eleven datasets currently supported by MusPy. Moreover, we conduct a cross-dataset generalizability experiment by training an autoregressive model on each dataset and measuring held-out likelihood on the others—a process which is made easier by MusPy's dataset management system. The results provide a map of domain overlap between various commonly used datasets and show that some datasets contain more representative cross-genre samples than others. Along with the dataset analysis, these results might serve as a guide for choosing datasets in future research. Source code and documentation are available at `https://github.com/salu133445/muspy`.

1. INTRODUCTION

Recent years have seen progress on music generation, thanks largely to advances in machine learning [1]. A music generation pipeline usually consists of several steps—data collection, data preprocessing, model creation, model training and model evaluation, as illustrated in Figure 1. While some components need to be customized for each model, others can be shared across systems. For symbolic music generation in particular, a number of datasets, representations and metrics have been proposed in the literature [1]. As a result, an easy-to-use toolkit that implements standard versions of such routines could save a great deal of time and effort and might lead to increased reproducibility. However, such tools are challenging to develop for a variety of reasons.

First, though there are a number of publicly-available symbolic music datasets, the diverse organization of these collections and the various formats used to store them presents a challenge. These formats are usually designed for different purposes. Some focus on playback capability

Figure 1. An example of a learning-based music generation system. MusPy provides basic routines specific to music as well as interfaces to machine learning frameworks.

(e.g., MIDI), some are developed for music notation softwares (e.g., MusicXML [2] and LilyPond [3]), some are designed for organizing musical documents (e.g., Music Encoding Initiative (MEI) [4]), and others are research-oriented formats that aim for simplicity and readability (e.g., MuseData [5] and Humdrum [6]. Oftentimes researchers have to implement their own preprocessing code for each different format. Moreover, while researchers can implement their own procedures to access and process the data, issues of reproducibility due to the inconsistency of source data have been raised in [7] for audio datasets.
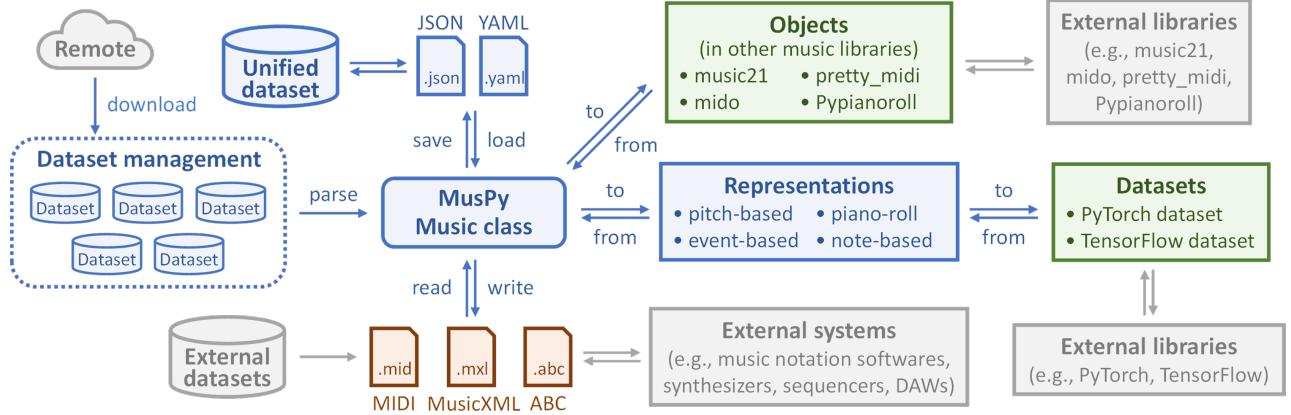
Second, music has hierarchy and structure, and thus different levels of abstraction can lead to different representations [8]. Moreover, a number of music representations designed specially for generative modeling of music have also been proposed in prior art, for example, as a sequence of pitches [9–12], events [13–16], notes [17] or a time-pitch matrix (i.e., a piano roll) [18, 19].

Finally, efforts have been made toward more robust objective evaluation metrics for music generation systems [20] as these metrics provide not only an objective way for comparing different models but also indicators for monitoring training progress in machine learning-based systems. Given the success of mir_eval [21] in evaluating common MIR tasks, a library providing implementations of commonly used evaluation metrics for music generation systems could help improve reproducibility.

To manage the above challenges, we find a toolkit dedicated for music generation a timely contribution to the MIR community. Hence, we present in this paper a new Python library, MusPy, for symbolic music generation. It provides essential tools for developing a music generation system, including dataset management, data I/O, data preprocessing and model evaluation.

With MusPy, we provide a statistical analysis on the eleven datasets currently supported by MusPy, with an eye

Figure 2. System diagram of MusPy. The MusPy Music object at the center is the core element of MusPy.
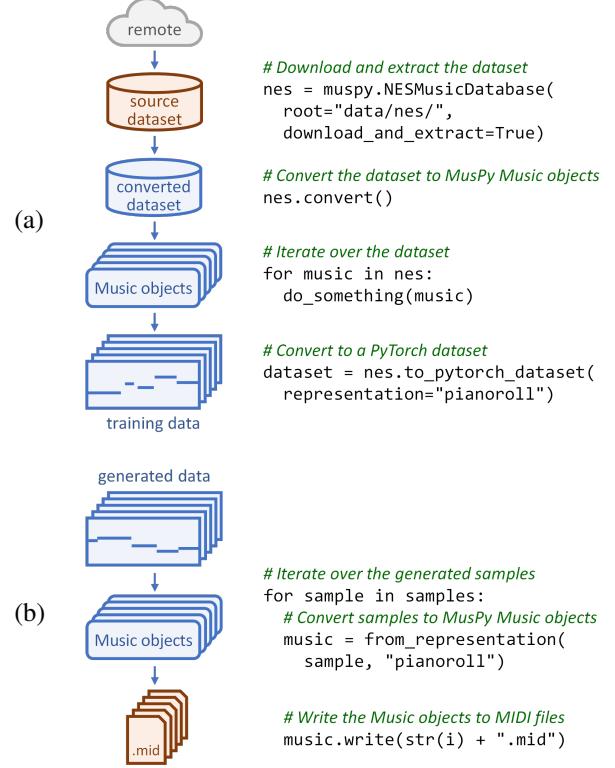
to unveiling statistical differences between them. Moreover, we conduct three experiments to analyze their relative diversities and cross-dataset domain compatibility of the various datasets. These results, along with the statistical analysis, together provide a guide for choosing proper datasets for future research. Finally, we also show that combining multiple heterogeneous datasets could help improve generalizability of a music generation system.

2. RELATED WORK

Few attempts, to the best of our knowledge, have been made to develop a dedicated library for music generation. The Magenta project [22] represents the most notable example. While MusPy aims to provide fundamental routines in data collection, preprocessing and analysis, Magenta comes with a number of model instances, but is tightly bound with TensorFlow [23]. In MusPy, we leave the model creation and training to dedicated machine learning libraries, and design MusPy to be flexible in working with different machine learning frameworks.

There are several libraries for working with symbolic music. music21 [24] is one of the most representative toolkits and targets studies in computational musicology. While music21 comes with its own corpus, MusPy does not host any dataset. Instead, MusPy provides functions to download datasets from the web, along with tools for managing different collections, which makes it easy to extend support for new datasets in the future. jSymbolic [25] focuses on extracting statistical information from symbolic music data. While jSymbolic can serve as a powerful feature extractor for training supervised classification models, MusPy focuses on generative modeling of music and supports different commonly used representations in music generation. In addition, MusPy provides several objective metrics for evaluating music generation systems.

Related cross-dataset generalizability experiments [15] show that pretraining on a cross-domain data can improve music generation results both qualitatively and quantitatively. MusPy's dataset management system makes it easier for us to thoroughly verify this hypothesis by examining pairwise generalizabilities between various datasets.

Figure 3. Examples of (a) training data preparation and (b) result writing pipelines using MusPy.

3. MUSPY

MusPy is an open source Python library dedicated for symbolic music generation. Figure 2 presents the system diagram of MusPy. It provides a core class, MusPy Music class, as a universal container for symbolic music. Dataset management system, I/O interfaces and model evaluation tools are then built upon this core container. We provide in Figure 3 examples of data preparation and result writing pipelines using MusPy.

3.1 MusPy Music class and I/O interfaces

We aim at finding a middle ground among existing formats for symbolic music and design a unified format dedicated

Dataset	Format	Hours	Songs	Genre	Melody	Chords	Multitrack
Lakh MIDI Dataset (LMD) [26]	MIDI	>9000	174,533	misc	△	△	△
MAESTRO Dataset [27]	MIDI	201.21	1,282	classical			
Wikifonia Lead Sheet Dataset [28]	MusicXML	198.40	6,405	misc	✓	✓	
Essen Folk Song Database [29]	ABC	56.62	9,034	folk	✓	✓	
NES Music Database [30]	MIDI	46.11	5,278	game	✓		✓
Hymnal Tune Dataset [31]	MIDI	18.74	1,756	hymn	✓		
Hymnal Dataset [31]	MIDI	17.50	1,723	hymn			
music21 Corpus [24]	misc	16.86	613	misc	△		△
Nottingham Database (NMD) [32]	ABC	10.54	1,036	folk	✓	✓	
music21 JSBach Corpus [24]	MusicXML	3.46	410	classical			✓
JSBach Chorale Dataset [11]	MIDI	3.21	382	classical			✓
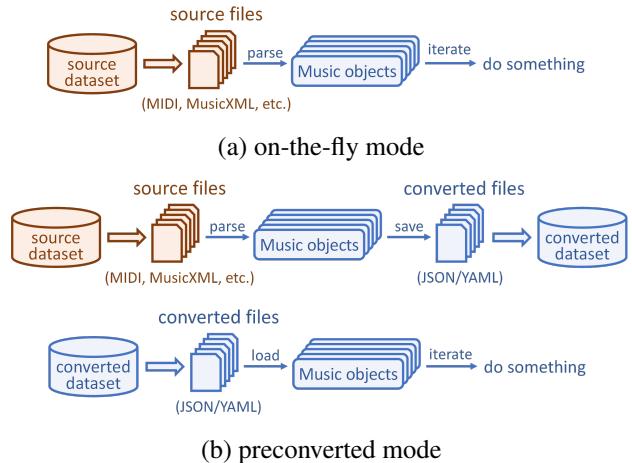
Table 1. Comparisons of datasets currently supported by MusPy. Triangle marks indicate partial support. Note that, in this version, only MusicXML and MIDI files are included for the music21 Corpus.

	MIDI	MusicXML	MusPy
Sequential timing	✓		✓
Playback velocities	✓	△	✓
Program information	✓	△	✓
Layout information		✓	
Note beams and slurs		✓	
Song/source meta data	△	✓	✓
Track/part information	△	✓	✓
Dynamic/tempo markings		✓	✓
Concept of notes		✓	✓
Measure boundaries		✓	✓
Human readability		△	✓

Table 2. Comparisons of MIDI, MusicXML and the proposed MusPy formats. Triangle marks indicate optional or limited support.

(a) on-the-fly mode

(b) preconverted mode

Figure 4. Two internal processing modes for iterating over a MusPy Dataset object.

for music generation. MIDI, as a communication protocol between musical devices, uses velocities to indicate dynamics, beats per minute (bpm) for tempo markings, and control messages for articulation, but it lacks the concepts of notes, measures and symbolic musical markings. In contrast, MusicXML, as a sheet music exchanging format, has the concepts of notes, measures and symbolic musical markings and contains visual layout information, but it falls short on playback-related data. For a music generation system, however, both symbolic and playback-specific data are important. Hence, we follow MIDI's standard for playback-related data and MusicXML's standard for symbolic musical markings.

In fact, the MusPy Music class naturally defines a universal format for symbolic music, which we will refer to as the MusPy format, and can be serialized into a human-readable JSON/YAML file. Table 2 summarizes the key differences among MIDI, MusicXML and the proposed MusPy formats. Using the proposed MusPy Music class as the internal representation for music data, we then provide I/O interfaces for common formats (e.g., MIDI, MusicXML and ABC) and interfaces to other symbolic music libraries (e.g., music21 [24], mido [33], pretty_midi [34]

and Pypianoroll [35]). Figure 3(b) provides an example of result writing pipeline using MusPy.

3.2 Dataset management

MusPy provides an easy-to-use dataset management system similar to torchvision datasets [36] and TensorFlow Dataset [37]. Table 1 presents the list of datasets currently supported by MusPy and their comparisons. Each supported dataset comes with a class inherited from the base MusPy Dataset class. The modularized and flexible design of the dataset management system makes it easy to handle local data collections or extend support for new datasets in the future. Figure 4 illustrates the two internal processing modes when iterating over a MusPy Dataset object. In addition, MusPy provides interfaces to PyTorch [38] and TensorFlow [23] for creating input pipelines for machine learning (see Figure 3(a) for an example).

3.3 Representations

Music has multiple levels of abstraction, and thus can be expressed in various representations. For music generation in particular, several representations designed for

Representation	Shape	Values	Default configurations
Pitch-based	$T \times 1$	$\{0, 1, \ldots, 129\}$	128 note-ons, 1 hold, 1 rest (*support only monophonic music*)
Event-based	$T \times 1$	$\{0, 1, \ldots, 387\}$	128 note-ons, 128 note-offs, 100 time shifts, 32 velocities
Piano-roll	$T \times 128$	$\{0, 1\}$ or $\mathbb{R}^+$	$\{0, 1\}$ for binary piano rolls; $\mathbb{R}^+$ for piano rolls with velocities
Note-based	$N \times 4$	$\mathbb{N}$ or $\mathbb{R}^+$	List of $(time, pitch, duration, velocity)$ tuples

Table 3. Comparisons of representations supported by MusPy. T and N denote the numbers of time steps and notes, respectively. Note that the configurations can be modified to meet specific requirements and use cases.

generative modeling of symbolic music have been proposed and used in the literature [1]. These representations can be broadly categorized into four types—the pitch-based [9–12], the event-based [13–16], the note-based [17] and the piano-roll [18,19] representations. Table 3 presents a comparison of them. We provide in MusPy implementations of these representations and integration to the dataset management system. Figure 3(a) provides an example of preparing training data in the piano-roll representation from the NES Music Database using MusPy.

3.4 Model evaluation tools

Model evaluation is another critical component in developing music generation systems. Hence, we also integrate into MusPy tools for audio rendering as well as score and piano-roll visualizations. These tools could also be useful for monitoring the training progress or demonstrating the final results. Moreover, MusPy provides implementations of several objective metrics proposed in the literature [17, 19, 39]. These objective metrics, as listed below, could be used to evaluate a music generation system by comparing the statistical difference between the training data and the generated samples, as discussed in [20].
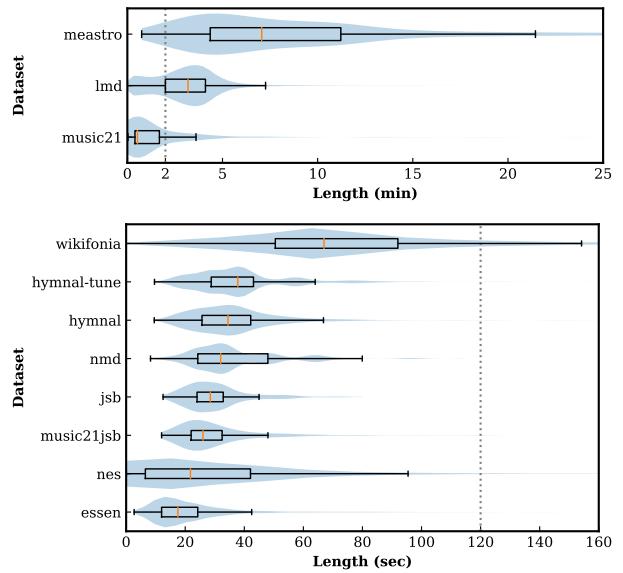
- *Pitch-related metrics*—polyphony, polyphony rate, pitch-in-scale rate, scale consistency, pitch entropy and pitch class entropy.

- *Rhythm-related metrics*—empty-beat rate, drum-in-pattern rate, drum pattern consistency and groove consistency.

3.5 Summary

To summarize, MusPy features the following:

- Dataset management system for commonly used datasets with interfaces to PyTorch and TensorFlow.

- Data I/O for common symbolic music formats (e.g., MIDI, MusicXML and ABC) and interfaces to other symbolic music libraries (e.g., music21, mido, pretty_midi and Pypianoroll).

- Implementations of common music representations for music generation, including the pitch-based, the event-based, the piano-roll and the note-based representations.

- Model evaluation tools for music generation systems, including audio rendering, score and piano-roll visualizations and objective metrics.

All source code and documentation can be found at
`https://github.com/salu133445/muspy`.
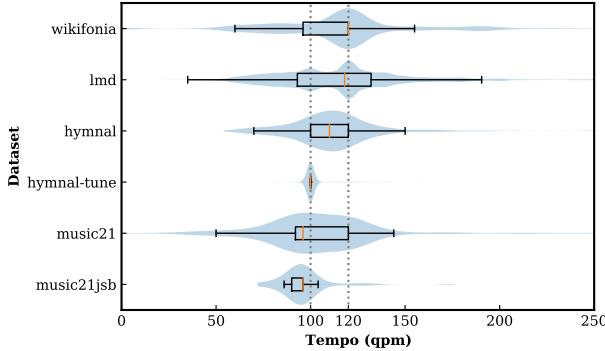
Figure 5. Length distributions for different datasets.

4. DATASET ANALYSIS
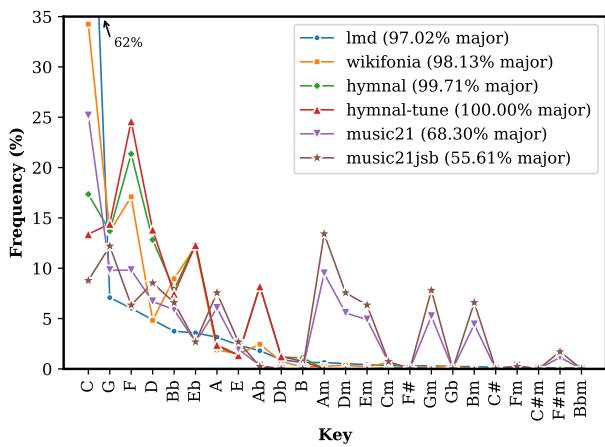
Analyzing datasets is critical in developing music generation systems. With MusPy's dataset management system, we can easily work with different music datasets. Below we compute the statistics of three key elements of a song—length, tempo and key using MusPy, with an eye to unveiling statistical differences among these datasets. First, Figure 5 shows the distributions of song lengths for different datasets. We can see that they differ greatly in their ranges, medians and variances.

Second, we present in Figure 6 the distributions of initial tempo for datasets that come with tempo information. We can see that all of them are generally bell-shaped but with different ranges and variances. We also note that there are two peaks, 100 and 120 quarter notes per minute (qpm), in Lakh MIDI Dataset (LMD), which is possibly because these two values are often set as the default tempo values in music notation programs and MIDI editors/sequencers. Moreover, in Hymnal Tune Dataset, only around ten percent of songs have an initial tempo other than 100 qpm.

Finally, Figure 7 shows the histograms of keys for different datasets. We can see that the key distributions are rather imbalanced. Moreover, only less than 3% of songs are in minor keys for most datasets except the music21 Corpus. In particular, LMD has the most imbalanced key distributions, which might be due to the fact that C major is

Figure 6. Initial-tempo distributions for different datasets (those without tempo information are not presented).

Figure 7. Key distributions for different datasets. The keys are sorted w.r.t. their frequencies in Lakh MIDI Dataset.

often set as the default key in music notation programs and MIDI editors/sequencers. [1] These statistics could provide a guide for choosing proper datasets in future research.
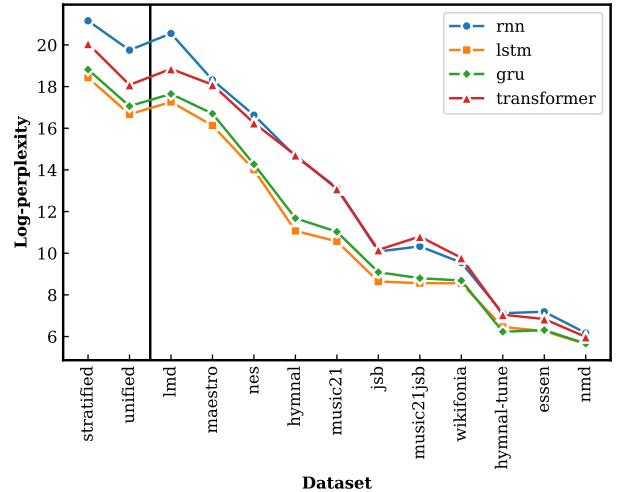
5. EXPERIMENTS AND RESULTS

In this section, we conduct three experiments to analyze the relative complexities and the cross-dataset generalizabilities of the eleven datasets currently supported by MusPy (see Table 1). We implement four autoregressive models—a recurrent neural network (RNN), a long short-term memory (LSTM) network [40], a gated recurrent unit (GRU) network [41] and a Transformer network [42].

5.1 Experiment settings

For the data, we use the event representation as specified in Table 3 and discard velocity events as some datasets have no velocity information (e.g., datasets using ABC format). Moreover, we also include an end-of-sequence event, leading to in total 357 possible events. For simplicity, we downsample each song into four time steps per quarter note and fix the sequence length to 64, which is equivalent to

___

[1] Note that key information is considered as a meta message in a MIDI file. It does not affect the playback and thus can be unreliable sometimes.

Figure 8. Log-perplexities for different models on different datasets, sorted by the values for the LSTM model.

four measures in 4/4 time. In addition, we discard repeat information in MusicXML data and use only melodies in Wikifonia dataset. We split each dataset into train–test–validation sets with a ratio of 8 : 1 : 1. For the training, the models are trained to predict the next event given the previous events. We use the cross entropy loss and the Adam optimizer [43]. For evaluation, we randomly sample 1000 sequences of length 64 from the test split, and compute the perplexity of these sequences. We implement the models in Python using PyTorch. For reproducibility, source code and hyperparmeters are available at `https://github.com/salu133445/muspy-exp`.
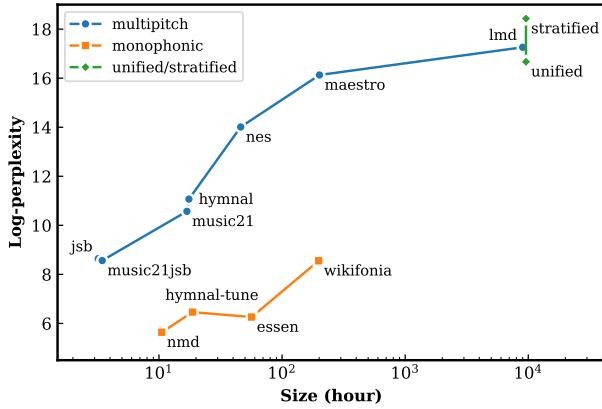
5.2 Autoregressive models on different datasets

In this experiment, we train the model on some dataset $\mathcal{D}$ and test it on the same dataset $\mathcal{D}$. We present in Figure 8 the perplexities for different models on different datasets. We can see that all models have similar tendencies. In general, they achieve smaller perplexities for smaller, homogeneous datasets, but result in larger perplexities for larger, more diverse datasets. That is, the test perplexity could serve as an indicator for the diversity of a dataset. Moreover, Figure 9 shows perplexities versus dataset sizes (in hours). By categorizing datasets into multi-pitch (i.e., accepting any number of concurrent notes) and monophonic datasets, we can see that the perplexity is positively correlated to the dataset size within each group.

5.3 Cross-dataset generalizability

In this experiment, we train a model on some dataset $\mathcal{D}$, while in addition to testing it on the same dataset $\mathcal{D}$, we also test it on each other dataset $\mathcal{D}'$. We present in Figure 10 the perplexities for each train–test dataset pair. Here are some observations:

- Cross dataset generalizability is not symmetric in general. For example, a model trained on LMD generalizes well to all other datasets, while not all models trained on

Figure 9. Log-perplexities for the LSTM model versus dataset size in hours. Each point corresponds to a dataset.

other datasets generalize to LMD, which is possibly due to the fact that LMD is a large, cross-genre dataset.
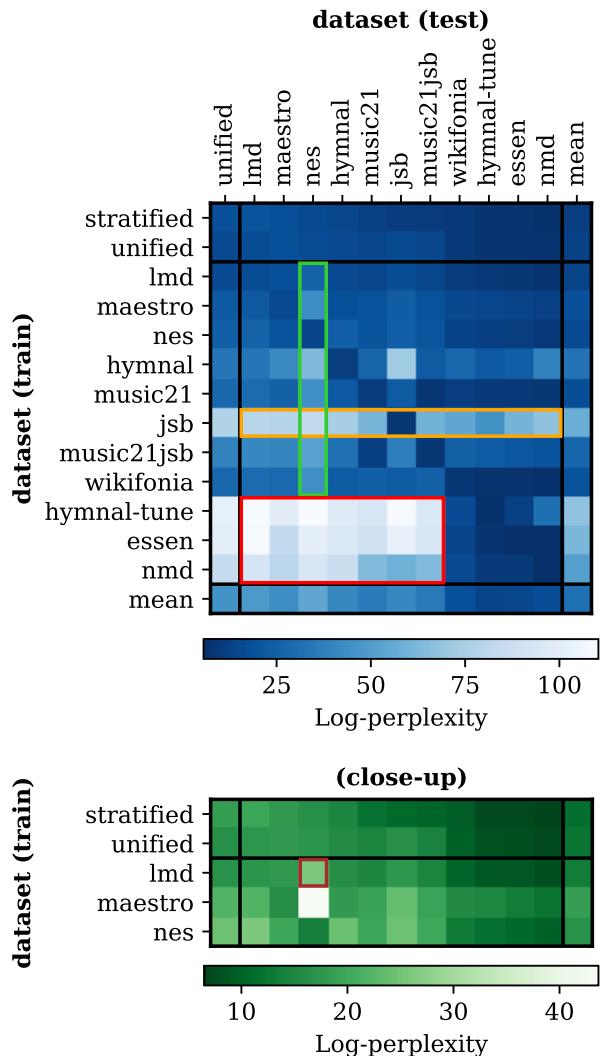
- Models trained on multi-pitch datasets generalize well to monophonic datasets, while models trained on monophonic datasets do not generalize to multi-pitch datasets (see the red block in Figure 10).

- The model trained on JSBach Chorale Dataset does not generalize to any of the other datasets (see the orange block in Figure 10). This is possibly because its samples are downsampled to a resolution of quarter note, which leads to a distinct note duration distribution.

- Most datasets generalize worse to NES Music Database compared to other datasets (see the green block in Figure 10). This is possibly due to the fact that NES Music Database contains only game soundtracks.

5.4 Effects of combining heterogeneous datasets

From Figure 10 we can see that LMD has the best generalizability, possibly because it is large, diverse and cross-genre. However, a model trained on LMD does not generalize well to NES Music Database (see the brown block in the close-up of Figure 10). We are thus interested in whether combing multiple heterogeneous datasets could help improve generalizability.

We combine all eleven datasets listed in Table 1 into one large *unified* dataset. Since these datasets differ greatly in their sizes, simply concatenating the datasets might lead to severe imbalance problem and bias toward the largest dataset. Hence, we also consider a version that adopts stratified sampling during training. Specifically, to acquire a data sample in the *stratified* dataset, we uniformly choose one dataset out of the eleven datasets, and then randomly pick one sample from that dataset. Note that stratified sampling is disabled at test time.

We also include in Figures 8, 9 and 10 the results for these two datasets. We can see from Figure 10 that combining datasets from different sources improves the generalizability of the model. This is consistent with the finding in [15] that models trained on certain cross-domain

Figure 10. Cross-dataset generalizability results. The values and colors represent the log-perplexities of a LSTM model trained on a specific dataset (row) and tested on another dataset (column). The datasets are sorted by the diagonal values, i.e., trained and tested on the same dataset.

datasets generalize better to other unseen datasets. Moreover, stratified sampling alleviates the source imbalance problem by reducing perplexities in most datasets with a sacrifice of an increased perplexity on LMD.

6. CONCLUSION

We have presented MusPy, a new toolkit that provides essential tools for developing music generation systems. We discussed the designs and features of the library, along with data pipeline examples. With MusPy's dataset management system, we conducted a statistical analysis and experiments on the eleven currently supported datasets to analyze their relative diversities and cross-dataset generalizabilities. These results could help researchers choose appropriate datasets in future research. Finally, we showed that combining heterogeneous datasets could help improve generalizability of a machine learning model.

7. REFERENCES

[1] J.-P. Briot, G. Hadjeres, and F. Pachet, "Deep learning techniques for music generation: A survey," *arXiv preprint arXiv:1709.01620*, 2017.

[2] M. Good, "Musicxml for notation and analysis," in *The Virtual Score: Representation, Retrieval, Restoration*, W. B. Hewlett and E. Selfridge-Field, Eds. Cambridge, Massachusetts: MIT Press, 2001, ch. 8, pp. 113–124.

[3] "Lilypond," https://lilypond.org/.

[4] A. Hankinson, P. Roland, and I. Fujinaga, "The music encoding initiative as a document-encoding framework," in *Proc. of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[5] W. B. Hewlett, "MuseData: Multipurpose representation," in *Beyond MIDI: The Handbook of Musical Codes*, E. Selfridge-Field, Ed. Cambridge, Massachusetts: MIT Press, 1997, ch. 27, pp. 402–447.

[6] D. Huron, "Humdrum and Kern: Selective feature encoding," in *Beyond MIDI: The Handbook of Musical Codes*, E. Selfridge-Field, Ed. Cambridge, Massachusetts: MIT Press, 1997, ch. 27, pp. 375–401.

[7] R. M. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi, and T. Kell, "mirdata: Software for reproducible usage of datasets," in *Proc. of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[8] R. B. Dannenberg, "A brief survey of music representation issues, techniques, and systems," *Computer Music Journal*, vol. 17, no. 3, pp. 20–30, 1993.

[9] M. Mozer, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing," *Connection Science*, vol. 6, pp. 247–280, 1994.

[10] D. Eck and J. Schmidhuber, "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks," in *Proc. of the IEEE Workshop on Neural Networks for Signal Processing*, 2002, pp. 747–756.

[11] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proc. of the 29th International Conference on Machine Learning (ICML)*, 2012.

[12] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. of the 35th International Conference on Machine Learning (ICML)*, 2018.

[13] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, vol. 32, 2018.

[14] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *Proc. of the 7th International Conference for Learning Representations (ICLR)*, 2019.

[15] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," in *Proc. of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[16] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Generating music with rhythm and harmony," *arXiv preprint arXiv:2002.00212*, 2020.

[17] O. Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," in *NeuIPS Workshop on Constructive Machine Learning*, 2016.

[18] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[19] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proc. of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[20] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, pp. 4773–4784, 2018.

[21] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mir_eval: A transparent implementation of common MIR metrics," in *Proc. of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[22] "Magenta," https://magenta.tensorflow.org/.

[23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *Proc. of the 12th USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, 2016.

[24] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," in *Proc. of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010.

[25] C. Mckay and I. Fujinaga, "JSymbolic: A feature extractor for MIDI files," in *Proc. of the 2006 International Computer Music Conference (ICMC)*, 2006.

[26] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching," Ph.D. dissertation, Columbia University, 2016.

[27] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proc. of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[28] "Wikifonia," http://www.wikifonia.org/.

[29] "Essen folk song database," https://ifdo.ca/~seymour/runabc/esac/esacdatabase.html.

[30] C. Donahue, H. H. Mao, and J. McAuley, "The NES music database: A multi-instrumental dataset with expressive performance attributes," in *Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[31] "Hymnal," https://www.hymnal.net/.

[32] "Nottingham database," https://ifdo.ca/~seymour/nottingham/nottingham.html.

[33] "Mido: Midi objects for python," https://github.com/mido/mido.

[34] C. Raffel and D. P. W. Ellis, "Intuitive analysis, creation and manipulation of MIDI data with pretty_midi," in *Late-Breaking Demos of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[35] H.-W. Dong, W.-Y. Hsiao, and Y.-H. Yang, "Pypianoroll: Open source Python package for handling multitrack pianorolls," in *Late-Breaking Demos of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[36] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proc. of the 18th ACM International Conference on Multimedia*, 2010.

[37] "Tensorflow datasets," https://www.tensorflow.org/datasets.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019, pp. 8024–8035.

[39] S.-L. Wu and Y.-H. Yang, "The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures," in *Proc. of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[41] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of the 3rd International Conference for Learning Representations (ICLR)*, 2014.

MUSIC FADERNETS: CONTROLLABLE MUSIC GENERATION BASED ON HIGH-LEVEL FEATURES VIA LOW-LEVEL FEATURE MODELLING

Hao Hao Tan[1]   **Dorien Herremans**[1]

[1] Singapore University of Technology and Design

{haohao_tan, dorien_herremans}@sutd.edu.sg

ABSTRACT

High-level musical qualities (such as emotion) are often abstract, subjective, and hard to quantify. Given these difficulties, it is not easy to learn good feature representations with supervised learning techniques, either because of the insufficiency of labels, or the subjectiveness (and hence large variance) in human-annotated labels. In this paper, we present a framework that can learn high-level feature representations with a limited amount of data, by first modelling their corresponding quantifiable *low-level* attributes. We refer to our proposed framework as Music FaderNets, which is inspired by the fact that low-level attributes can be continuously manipulated by separate "sliding faders" through feature disentanglement and latent regularization techniques. High-level features are then inferred from the low-level representations through semi-supervised clustering using Gaussian Mixture Variational Autoencoders (GM-VAEs). Using arousal as an example of a high-level feature, we show that the "faders" of our model are disentangled and change linearly w.r.t. the modelled low-level attributes of the generated output music. Furthermore, we demonstrate that the model successfully learns the intrinsic relationship between arousal and its corresponding low-level attributes (rhythm and note density), with only 1% of the training set being labelled. Finally, using the learnt high-level feature representations, we explore the application of our framework in style transfer tasks across different arousal states. The effectiveness of this approach is verified through a subjective listening test.

1. INTRODUCTION

We consider *low-level* musical attributes as attributes that are relatively straightforward to quantify, extract and calculate from music, such as rhythm, pitch, harmony, etc. On the other hand, *high-level* musical attributes refer to semantic descriptors or qualities of music that are relatively abstract, such as emotion, style, genre, etc. Due to the nature of abstractness and subjectivity in these high-level musical qualities, obtaining labels for these qualities typically requires human annotation. However, training conditional models on top of these human-annotated labels using supervised learning might result in sub-par performance because firstly, obtaining such labels can be costly, hence the amount of labels collected might be insufficient to train a model that can generalize well [1]; Secondly, the annotated labels could have high variance among raters due to the subjectivity of these musical qualities [2, 3].

Instead of inferring high-level features directly from the music sample, we propose to use low-level features as a "bridge" between the music and the high level features. This is because the relationship between the sample and its low-level features can be learnt relatively easier, as the labels are easier to obtain. In addition, we learn the relationship between the low-level features and the high-level features in a data-driven manner. In this paper, we show that the latter works well even with a limited amount of labelled data. Our work relies heavily on the concept that each high-level feature is intrinsically related to a set of low-level attributes. By tweaking the levels of each low-level attribute in a constrained manner, we can achieve a desired change on the high-level feature. This idea is heavily exploited in rule-based systems [4–6], however rule-based systems are often not robust enough as their capabilities are constrained by the fixed set of predefined rules handcrafted by the authors. Hence, we propose an alternative path which is to *learn* these implicit relationships with semi-supervised learning techniques.

To achieve the goals stated above, we intend to build a framework which can fulfill these two objectives:

- Firstly, the model should be able to control multiple low-level attributes of the music sample in a continuous manner, as if it is controlled by sliding knobs on a console (or also known as *faders*). Each knob should be independent from the others, and only controls one single feature that it is assigned to.

- Secondly, the model should be able to learn the relationship between the levels of the sliding knobs controlling the low-level features, and the selected high-level feature. This is analogous to learning a *preset* of the sliding knobs on a console.

We named our model "Music FaderNets", with reference to musical "faders" and "presets" as described above. Achieving the first objective requires representation learning and feature disentanglement techniques. This motivates us to use *latent variable models* [7] as we can learn

separate latent spaces for each low-level feature to obtain disentangled controllability. Achieving the second objective requires the latent space to have a hierarchical structure, such that high-level information can be inferred from low-level representations. This is achieved by incorporating Gaussian Mixture VAEs [8] in our model.

2. RELATED WORK

2.1 Controllable Music Generation

The application of deep learning techniques for music generation has been rapidly advancing [9–13], however, embedding *control* and *interactivity* in these systems still remains a critical challenge [10]. Variants of conditional generative models (such as CGAN [14] and CVAE [15]) are used to allow control during generation, which have attained much success mainly in the image domain. Fader Networks [16] is one of the main inspirations of this work (hence the name Music FaderNets), in which users can modify different visual features of an image using "sliding faders". However, their approach is built upon a CVAE with an additional adversarial component, which is very different from our approach. Recently, controllable music generation has gained much research interest, both on modelling low-level [17–20] and high-level features [21, 22]. Specifically, [18] and [19] each proposed a novel latent regularization method to encode attributes along specific latent dimensions, which inspired the "sliding knob" application in this work.

2.2 Disentangled Representation Learning for Music

Disentangled representation learning has been widely used across both the visual [23–26] and speech domain [1, 27, 28] to learn disjoint subsets of attributes. Such techniques have also been applied to music in several recent works, both in the audio [29–31] and symbolic domain [32–34]. The discriminator component in our model draws inspiration from both the explicit conditioning component in the EC^2-VAE model [33], and the *extraction* component in the Ext-Res model [34]. We find that most of the work on disentanglement in symbolic music focuses on low-level features, and is done on monophonic music.

This research distinguishes itself from other related work through the following novel contributions:

- We combine latent regularization techniques with disentangled representation learning to build a framework that can control various continuous low-level musical attribute values using "faders", and apply the framework on *polyphonic* music modelling.

- We show that it is possible to infer high-level features from low-level latent feature representations, even under weakly supervised scenario. This opens up possibilities to learn good representations for abstract, high-level musical qualities even under data scarcity conditions. We further demonstrate that the learnt representations can be used for controllable generation based on high-level features.

3. PROPOSED FRAMEWORK

3.1 Gaussian Mixture Variational Autoencoders

VAEs [35] combine the power of both latent variable models and deep generative models, hence they provide both representation learning and generation capabilities. Given observations $\mathbf{X}$ and latent variables $\mathbf{z}$, the VAE learns a graphical model $\mathbf{z} \to \mathbf{X}$ by maximizing the evidence lower bound (ELBO) of the marginal likelihood $p(\mathbf{X})$ as below:

$$\mathcal{L}(p, q; \mathbf{X}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{X})}[\log p(\mathbf{X}|\mathbf{z})] - \mathcal{D}_{KL}(q(\mathbf{z}|\mathbf{X})||p(\mathbf{z}))$$

where $q(\mathbf{z}|\mathbf{X})$ and $p(\mathbf{z})$ represent the learnt posterior and prior distribution respectively. In vanilla VAEs, $p(\mathbf{z})$ is an isotropic, unimodal Gaussian. Gaussian Mixture VAEs (GM-VAE) [8] extend the prior to a mixture of K Gaussian components, which corresponds to learning a graphical model with an extra hierarchy of dependency $c \to \mathbf{z} \to \mathbf{X}$. The newly introduced categorical variable $c \in \mathcal{C}$, whereby $|\mathcal{C}| = K$, is a discrete representation of the observations. Hence, a new distribution $q(c|\mathbf{X})$ is introduced to infer the class of each observation, which enables semi-supervised and unsupervised clustering applications.

Following [8], the ELBO of a GM-VAE is derived as:

$$\mathcal{L}(p, q; \mathbf{X}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{X})}[\log p(\mathbf{X}|\mathbf{z})]$$
$$- \sum_{k=1}^{K} q(c_k|\mathbf{X})\mathcal{D}_{KL}(q(\mathbf{z}|\mathbf{X})||p(\mathbf{z}|c_k))$$
$$- \mathcal{D}_{KL}(q(c|\mathbf{X})||p(c))$$

The original KL loss term from the vanilla VAE is modified into two new terms: (i) the KL divergence between the approximate posterior $q(\mathbf{z}|\mathbf{X})$ and the conditional prior $p(\mathbf{z}|c_k)$, marginalized over all Gaussian components; (ii) the KL divergence between the cluster inferring distribution $q(c|\mathbf{X})$, and the categorical prior $p(c)$.
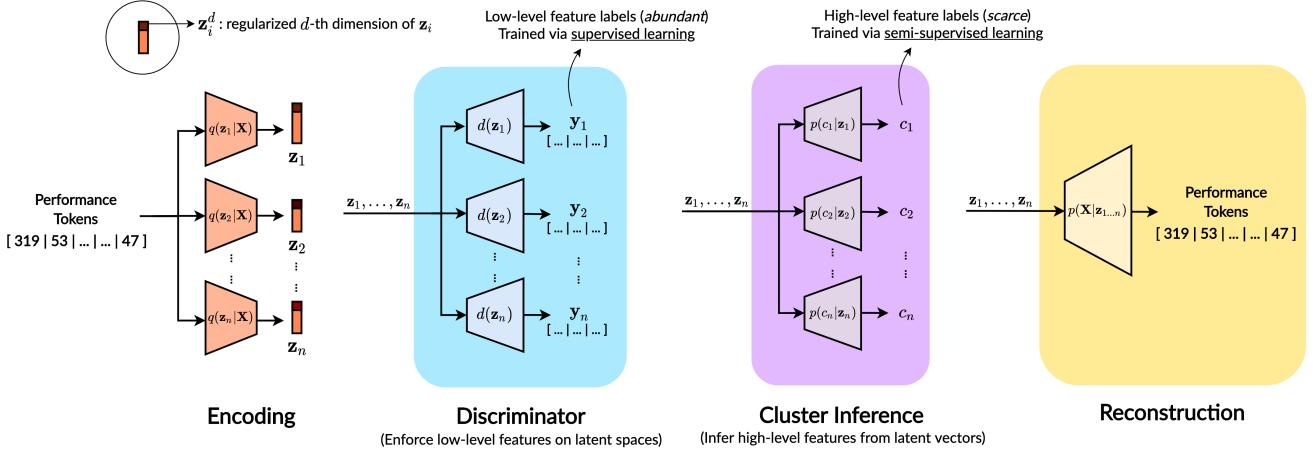
3.2 Model Formulation

Figure 1 shows the model formulation of our proposed Music FaderNets. Input $\mathbf{X}$ is a sequence of performance tokens converted from MIDI following [12, 13]. Assume that we want to model a high-level feature with K discrete states, which is related to a set of N low-level features. We denote the latent variables learnt for each low-level feature as $\mathbf{z}_{1...N}$; the labels for each low-level feature as $\mathbf{y}_{1...N}$; and the class inferred from each latent variable as $c_{1...N}$.

The joint probability of $\mathbf{X}, \mathbf{z}_{1...N}, c_{1...N}$ is written as:

$$p(\mathbf{X}, \mathbf{z}_{1...N}, c_{1...N}) = p(\mathbf{X}|\mathbf{z}_{1...N}) \prod_{i=1}^{N} p(\mathbf{z}_i|c_i) \prod_{i=1}^{N} p(c_i)$$

We assume that each categorical prior $p(c_i)$, $i \in [1, N]$ is uniformly distributed, and the conditional distributions $p(\mathbf{z}_i|c_i) = \mathcal{N}(\mu_{c_i}, \text{diag}(\sigma_{c_i}))$ are diagonal-covariance Gaussians with learnable means and constant variances. For each low-level attribute, we learn an approximate posterior $q(\mathbf{z}_i|\mathbf{X})$, parameterized by an encoder neural network, that samples latent code $\mathbf{z}_i$ which represents the i-th low-level feature.

Figure 1. Music FaderNets model architecture.

The latent codes $\mathbf{z}_{1...N}$ are then passed through the remaining three components: (1) **Discriminator**: To ensure that $\mathbf{z}_i$ incorporates information of the assigned low-level feature, it is passed through a discriminator represented by a function $d(\mathbf{z}_i)$ to reconstruct the low-level feature label $\mathbf{y}_i$; (2) **Reconstruction**: All latent codes are fed into a global decoder network which parameterizes the conditional probability $p(\mathbf{X}|\mathbf{z}_{1...n})$ to reconstruct the input $\mathbf{X}$; (3) **Cluster Inference**: This component parameterizes the cluster inference probability $q(c|\mathbf{X})$, with c representing the selected high-level feature. It can be approximated by $q(c|\mathbf{X}) \approx \mathbb{E}_{q(\mathbf{z}|\mathbf{X})}p(c|\mathbf{z})$ [36], where the cluster state is predicted from each latent code $\mathbf{z}_i$ instead of $\mathbf{X}$.

To incorporate the "sliding knob" concept, we need to map the change of value of an arbitrary dimension on $\mathbf{z}_i$ (denoted as $\mathbf{z}_i^d$, shown on Figure 1 as the darkened dimension) linearly to the change of value of the low-level feature label $\mathbf{y}_i$. After comparing across previous methods on conditioning and regularization [15, 16, 18, 19], we choose to adopt [19] which applies a latent regularization loss term written as $\mathcal{L}_{\text{reg}}(\mathbf{z}_i^d, \mathbf{y}_i) = \text{MSE}(\tanh(\mathcal{D}_{\mathbf{z}_i^d}), \text{sign}(\mathcal{D}_{\mathbf{y}_i}))$, where $\mathcal{D}_{\mathbf{z}_i^d}$ and $\mathcal{D}_{\mathbf{y}_i}$ denotes the *distance matrix* of values $\mathbf{z}_i^d$ and $\mathbf{y}_i$ within a training batch respectively. We provide a detailed comparison study across each proposed method in Section 4.2. Hence, if we define:

$$\mathcal{L}_\phi^i(p, q; \mathbf{X}) = \begin{cases} \sum_{k=1}^{K} q(c_{i,k}|\mathbf{X})\mathcal{D}_{KL}(q(\mathbf{z}_i|\mathbf{X})||p(\mathbf{z}_i|c_{i,k})) \\ \quad + \mathcal{D}_{KL}(q(c_i|\mathbf{X})||p(c_i)), \text{if unsupervised} \\ \\ \mathcal{D}_{KL}(q(\mathbf{z}_i|\mathbf{X})||p(\mathbf{z}_i|c_i)), \text{if supervised} \end{cases}$$
(1)

then the entire training objective can be derived as:

$$\mathcal{L}(p, q; \mathbf{X}) = \mathbb{E}_{q(\mathbf{z}_1|\mathbf{X})...q(\mathbf{z}_N|\mathbf{X})}[\log p(\mathbf{X}|\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N)]$$
$$- \beta \cdot \sum_{i=1}^{N} \mathcal{L}_\phi^i(p, q; \mathbf{X}) + \sum_{i=1}^{N} \mathcal{L}_{\text{reg}}(\mathbf{z}_i^d, \mathbf{y}_i)$$
$$+ \mathbb{E}_{q(\mathbf{z}_1|\mathbf{X})...q(\mathbf{z}_n|\mathbf{X})}[\log p(\mathbf{y}_1|\mathbf{z}_1)...p(\mathbf{y}_N|\mathbf{z}_N)]$$
(2)

where β is the KL weight hyperparameter [24]. The first term in Eq. 2 represents the reconstruction loss. The second KL loss term (derived from the ELBO function of GM-VAE) correspond to the cluster inference component, which allows both *supervised* and *unsupervised* training setting, depending on the availability of label c. If we omit the cluster inference component, it could conform to a vanilla VAE by replacing this term with the KL loss term of VAE. The third term is the latent regularization loss applied during the encoding process. The last term is the reconstruction loss of the low-level feature labels, which corresponds to the discriminator component. All encoders and decoders are implemented with gated recurrent units (GRUs), and teacher-forcing is used to train all decoders.

4. EXPERIMENTAL SETUP

In this work, we chose *arousal* (which refers to the energy level conveyed by the song [37]) as the high-level feature to be modelled. In order to select relevant low-level features, we refer to musicology papers such as [6, 38, 39], which suggest that arousal is related to features including rhythm density, note density, key, dynamic, tempo, etc. Among these low-level features, we focus on modelling the score-level features in this work (i.e. rhythm, note and key).

4.1 Data Representation and Hyperparameters

We use two polyphonic piano music datasets for training: the **Yamaha Piano-e-Competition dataset** [12], and the **VGMIDI dataset** [3], which contains piano arrangements of 95 video game soundtracks in MIDI, annotated with valence and arousal values in the range of -1 to 1. The arousal labels are used to guide the cluster inference component in our GM-VAE model using semi-supervised learning. We extract every 4-beat segment from each music sample, with a beat resolution of 4 (quarter-note granularity). Each segment is encoded into event-based tokens following [12] with a maximum sequence length of 100. This results in a total of 103,934 and 1,013 sequences from the Piano e-Competition and VGMIDI dataset respectively, which are split into train/validation/test sets with a ratio of 80/10/10.

Inspired by [33], we represent each rhythm label, $\mathbf{y}_{\text{rhythm}}$, as a sequence of 16 one-hot vectors with 3 dimensions, denoting an onset for any pitch, a holding state, or a rest. The *rhythm density* value is calculated as the number of onsets in the sequence divided by the total sequence length. Each note label, $\mathbf{y}_{\text{note}}$, is represented by a sequence of 16 one-hot vectors with 16 dimensions, each dimension denoting the number of notes being played or held at that time step (we assume a minimum polyphony of 0 and a maximum of 15). The *note density* value is the average number of notes being played or held for per time step. For key, we use the key analysis tool from music21 [40] to extract the estimated global key of each 4-beat segment. The key is represented using a 24-dimension one-hot vector, accounting for major and minor modes. In this work, we directly concatenate the key vector as a conditioning signal with $\mathbf{z}_{\text{rhythm}}$ and $\mathbf{z}_{\text{note}}$ as an input to the global decoder for reconstruction. For representing arousal, we split the arousal ratings into two clusters ($K = 2$): *high* arousal cluster for positive labels, and *low* arousal cluster for negative labels. We remove labels annotated within the range [-0.1, 0.1] so as to reduce ambiguity in the annotations.
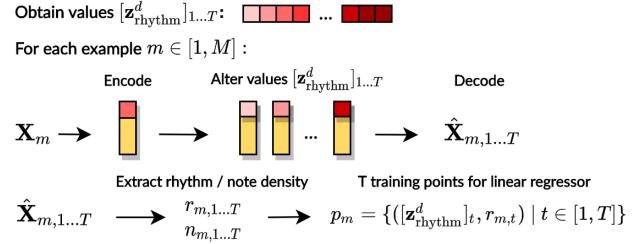
The hyperparameters are tuned according to the results on the validation set using grid search. The mean vectors of $p(c|\mathbf{z})$ are all randomly initialized with Xavier initialization, whereas the variance vectors are kept fixed with value e^{-2}. We observe that the following annealing strategy for β leads to the best balance between reconstruction and controllability: β is set to 0 in the first 1,000 training steps, and is slowly annealed up to 0.2 in the next 10,000 training steps. We set the batch size to 128, all hidden sizes to 512, and the encoded $\mathbf{z}$ dimensions to 128. The Adam optimizer is used with a learning rate of 10^{-3}.

4.2 Measuring the Controllability of Latent Features

The proposed Music FaderNets model should meet two requirements: (i) Each "fader" independently controls one low-level musical feature without affecting other features (disentanglement), and (ii) the "faders" should change linearly with the controlled attribute of the generated output (linearity). For disentanglement, we follow the definition proposed in [41] which decomposes the concept of disentanglement into *generator consistency* and *generator restrictiveness*. Using rhythm density as an example:

- *Consistency* on rhythm density means that for the same value of $\mathbf{z}_{\text{rhythm}}^{d}$, the value of the output's rhythm density should be consistent.

- *Restrictiveness* on rhythm density means that changing the value of $\mathbf{z}_{\text{rhythm}}^{d}$ does not affect the attributes other than rhythm density (in our case, note density).

- *Linearity* on rhythm density means that the value of rhythm density is directly proportional to the value of $\mathbf{z}_{\text{rhythm}}^{d}$, which is analogous to a sliding fader.

We will be evaluating all three of these points in our experiment. For evaluating linearity, [19] proposed a slightly modified version of the interpretability metric by [42],

Figure 2. Workflow of obtaining evaluation metrics for "faders" controlling rhythm density.

which includes the following steps: (1) encode each sample in the test set, obtain the rhythm latent code and the dimension $\mathbf{z}^{d}$ which has the maximum mutual information with regards to the attribute; (2) learn a linear regressor to predict the input attribute values based on $\mathbf{z}^{d}$. The linearity score is hence the coefficient of determination (R^2) score of the linear regressor. However, this method evaluates only the encoder and not the decoder. As we want the sliding knobs to directly impact the output, we argue that the relationship between $\mathbf{z}^{d}$ and the output attributes should be more important. Hence, we propose to "slide" the values of the regularized dimension $\mathbf{z}^{d}$ within a given range and decode them into reconstructed outputs. Then, instead of predicting the *input* attributes given the encoded $\mathbf{z}^{d}$, the linear regressor learns to predict the corresponding *output* attributes given the "slid" values of $\mathbf{z}^{d}$.

We demonstrate a single workflow to calculate the consistency, restrictiveness and linearity scores of a given model based on the low-level features (we use rhythm density as an example low-level feature for the discussion below), as depicted in Figure 2. After obtaining the rhythm density latent code for all samples in the training set and finding the minimum and maximum value of $\mathbf{z}_{\text{rhythm}}^{d}$, we "slide" for $T = 8$ steps by calculating $\min(\mathbf{z}_{\text{rhythm}}^{d}) + \frac{t}{T}(\max(\mathbf{z}_{\text{rhythm}}^{d}) - \min(\mathbf{z}_{\text{rhythm}}^{d}))$, with $t \in [1, T]$. This results in a list of values denoted as $[\mathbf{z}_{\text{rhythm}}^{d}]_{1...T}$. Then, we conduct the following steps:

1. Randomly select $M = 100$ samples from the test set, and encode each sample into $\mathbf{z}_{\text{rhythm}}$ and $\mathbf{z}_{\text{note}}$;
2. Alter the d-th element in $\mathbf{z}_{\text{rhythm}}$ using the values in the range $[\mathbf{z}_{\text{rhythm}}^{d}]_{1...T}$, to obtain $[\hat{\mathbf{z}}_{\text{rhythm}}]_{m,1...T}$ for each sample m;
3. Decode each new rhythm density latent code together with the unchanged note density latent code $\mathbf{z}_{\text{note}}$ to get $\hat{\mathbf{X}}_{m,1...T}$;
4. Calculate rhythm density $r_{m,1...T}$ and note density $n_{m,1...T}$ for each reconstructed output;
5. Pair up the new rhythm density latent code with the resulting rhythm density of the output as T training data points $p_m = \{([\mathbf{z}_{\text{rhythm}}^{d}]_t, r_{m,t}) \mid t \in [1, T]\}$ for a linear regressor.

The final evaluation scores are then calculated as follows:

$$\text{Consistency score} = 1 - \frac{1}{T}\sum_{t=1}^{T}\sigma_t(r_{1...M,t}) \qquad (3)$$

	Consistency		Restrictiveness		Linearity	
	Rhythm Density	Note Density	Rhythm Density	Note Density	Rhythm Density	Note Density
Proposed (Vanilla VAE)	0.4367 ± 0.0258	0.3490 ± 0.0360	0.6645 ± 0.0169	0.6481 ± 0.0154	$\mathbf{0.7805 \pm 0.0142}$	$\mathbf{0.8255 \pm 0.0107}$
Proposed (GM-VAE)	$\mathbf{0.5096 \pm 0.0248}$	0.4207 ± 0.0309	0.6603 ± 0.0164	0.6457 ± 0.0132	0.7580 ± 0.0124	0.7792 ± 0.0177
Pati et al. [19]	0.4625 ± 0.0264	$\mathbf{0.5100 \pm 0.0150}$	0.6417 ± 0.0171	0.5497 ± 0.0206	0.7613 ± 0.0171	0.8220 ± 0.0143
CVAE [15]	0.2613 ± 0.0376	0.4997 ± 0.0355	$\mathbf{0.6863 \pm 0.0221}$	0.7140 ± 0.0130	0.4969 ± 0.0166	0.3997 ± 0.0411
Fader Networks [16]	0.2730 ± 0.0366	0.4983 ± 0.0425	0.6861 ± 0.0163	$\mathbf{0.7379 \pm 0.0149}$	0.5482 ± 0.0283	0.4647 ± 0.0292
GLSR [18]	0.1891 ± 0.0346	0.1969 ± 0.0831	0.6365 ± 0.0276	0.7136 ± 0.0185	0.2465 ± 0.0197	0.1799 ± 0.0209

Table 1. Experimental results (conducted on the Yamaha dataset test split) on the controllability of low-level features (rhythm density and note density) using disentangled latent variables. Bold marks the best performing model.

$$\text{Restrictiveness score} = 1 - \frac{1}{M} \sum_{m=1}^{M} \sigma_{m}(n_{m,1..T}) \quad (4)$$

$$\text{Linearity score} = R^2(\mathcal{M}(p_{1...M})) \quad (5)$$
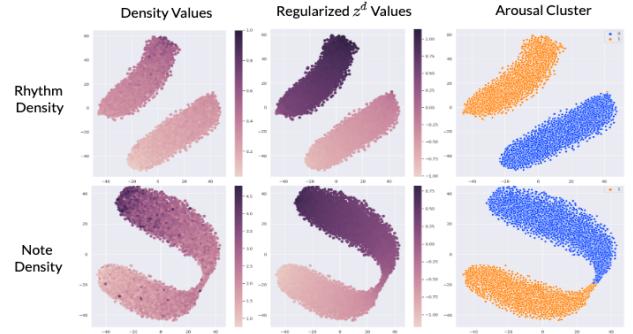
where $\sigma(\cdot)$ denotes the standard deviation, and $\mathcal{M}$ denotes the linear regressor model. In other words, consistency calculates the average standard deviation across all output rhythm density values given the same $\mathbf{z}^d_{\text{rhythm}}$, whereas restrictiveness calculates the average standard deviation across all output note density values given the changing $\mathbf{z}^d_{\text{rhythm}}$. In a perfectly disentangled and linear model, the consistency, restrictiveness and linearity scores should be equal to 1, and higher scores indicate better performance.

5. EXPERIMENTS AND RESULTS

We compare the evaluation scores of our proposed model, using both a vanilla VAE (omitting the cluster inference component) and GM-VAE, with several models proposed in related work on controllable synthesis: CVAE [15], Fader Networks [16], GLSR [18] and Pati et al. [19]. We repeat the above steps for 10 runs for each model and report the mean and standard deviation of each score. Table 1 shows the evaluation results. Overall, our proposed models achieve a good all-rounded performance on every metric as compared to other models, especially in terms of linearity, models that use [19]'s regularization method largely outperform other models. Our model shares similar results with [19], however as compared to their work, we encode a multi-dimensional, regularized latent space instead of a single dimension value for each low-level feature, thus allowing more flexibility. Our model can also be used for "generation via analogy" as mentioned in EC^2-VAE [33], by mix-matching $\mathbf{z}_{\text{rhythm}}$ from one sample with $\mathbf{z}_{\text{note}}$ from another. Moreover, the feature latent vectors can be used to infer interpretable and semantically meaningful clusters.

5.1 Inferring High-Level Features from Latent Low-Level Representations

Figure 3 visualizes the rhythm and note density latent space learnt by GM-VAE using t-SNE dimensionality reduction. We observe that both spaces successfully learn a Gaussian-mixture space with two well-separated components, which correspond to high and low arousal clusters, even though it was trained with only around 1% of labelled data. We also find that the regularized $\mathbf{z}^d$ values capture
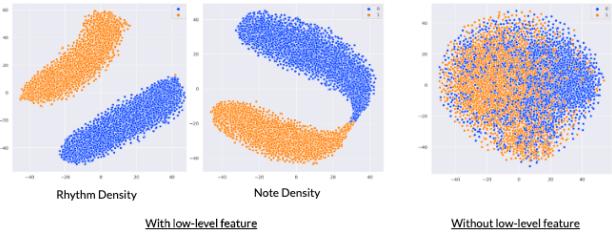
Figure 3. Visualization of rhythm (top) and note (bottom) density latent space in the GM-VAE. Each column is colored in terms of: (left) original density values, (middle) regularized $\mathbf{z}^d$ values, (right) arousal cluster labels (0 refers to low arousal and 1 refers to high arousal).
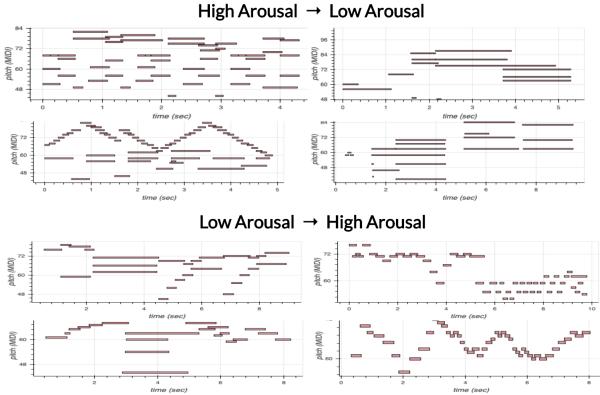
the overall trend of the actual rhythm and note density values. Interestingly, the model learns the implicit relationship between high/low arousal and the corresponding levels of rhythm/note density. From Figure 3, we observe that the high arousal cluster corresponds to higher rhythm density and lower note density, whereas the low arousal cluster corresponds to lower rhythm density and higher note density. This is reasonable as music segments with high arousal often consist of fast running notes and arpeggios, being played one note at a time, whereas music segments with low arousal often exhibit a chordal texture with more sustaining notes and relatively less melodic activity.

To further inspect the importance of using low-level features, we train a separate GM-VAE model with only one encoder (without discriminator component), which encodes only a single latent vector for each segment. The model is trained to infer the arousal label with the single latent vector similarly in a semi-supervised manner, and the hyperparameters are kept the same. From Figure 4, we can observe that the latent space learnt without using low-level features is not well-segregated into two separate components, suggesting that the right choice of low-level features helps the learning of a more discriminative and disentangled feature latent space.

The major advantage demonstrated from the results above is that by carefully choosing low-level features supported by domain knowledge, semi-supervised (or weakly supervised) training can be leveraged to learn interpretable representations that can capture implicit relationships between high-level and low-level features, overcoming the

Figure 4. Arousal cluster visualization of GM-VAE with (left), and without (right) using low-level features.
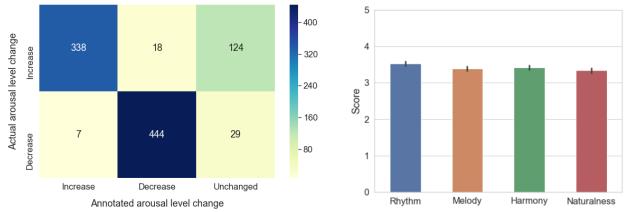
Figure 5. Examples of arousal transfer on music samples.

difficulties mentioned in the introduction section. This is an important insight for learning representations of abstract musical qualities under label scarcity conditions in future.

5.2 Style Transfer on High Level Features

Utilizing the learnt high-level feature representations enables the application of feature style transfer. Following [29], given the means of each Gaussian component, $\mu_{\text{arousal}=0}$ and $\mu_{\text{arousal}=1}$, the "shifting vector" from high arousal to low arousal is $s_{\text{low\_shift}} = \mu_{\text{arousal}=0} - \mu_{\text{arousal}=1}$, and vice versa. To shift a music segment from high to low arousal, we modify the latent codes by $\mathbf{z}'_{\text{rhythm}} = \mathbf{z}_{\text{rhythm}} + s_{\text{low\_shift}}$, $\mathbf{z}'_{\text{note}} = \mathbf{z}_{\text{note}} + s_{\text{low\_shift}}$. Both new latent codes $\mathbf{z}'_{\text{rhythm}}$ and $\mathbf{z}'_{\text{note}}$ are fed into the global decoder for reconstruction. For cases where $c_{\text{rhythm}} \neq c_{\text{note}}$, we choose to perform shifting only on the latent codes which are not lying within the target arousal cluster. Figure 5 shows several examples of arousal shift performed on given music segments. We can observe that the shift is clearly accompanied with the desired changes in rhythm density and note density, as mentioned in Section 5.1. More examples are available online. [1] We also conducted a subjective listening test to evaluate the quality of arousal shift performed by Music FaderNets. We randomly chose 20 music segments from our dataset, and performed a low-to-high arousal shift on 10 segments and a high-to-low arousal shift on the other 10. Each subject listened to the original sample and then the transformed sample, and was asked whether (1) the arousal level changes after the transformation, and; (2) how well the transformed sample sounds in terms of rhythm, melody, harmony and naturalness, on a

---

[1] https://music-fadernets.github.io/

Figure 6. Subjective listening test results. Left: Heat map of annotated arousal level change against actual arousal level change. Right: Bar plot of opinion scores for each musical quality, with 95% confidence interval.

Likert scale of 1 to 5 each.

A total of 48 subjects participated in the survey. We found that 81.45% of the responses agreed with the actual direction of level change in arousal, shifted by the model. This showed that our model is capable of shifting the arousal level of a piece to a desired state. From the heat map shown in Figure 6, we observe that shifting from high to low arousal has a higher rate of agreement (92.5%) than shifting from low to high arousal (70.41%). Meanwhile, the mean opinion score of rhythm, melody, harmony and naturalness were reported at 3.53, 3.39, 3.41 and 3.33 respectively, showing that the quality of the generated samples are generally above moderate level.

6. CONCLUSION AND FUTURE WORK

We propose a novel framework called Music FaderNets [2], which can generate new variations of music samples by controlling levels ("sliding knobs") of low-level attributes, trained with latent regularization and feature disentanglement techniques. We also show that the framework is capable of inferring high-level feature representations ("presets", e.g. arousal) on top of latent low-level representations by utilizing the GM-VAE framework. Finally, we demonstrate the application of using learnt high-level feature representations to perform arousal transfer, which was confirmed in a user experiment. The key advantage of this framework is that it can learn interpretable mixture components that reveal the intrinsic relationship between low-level and high-level features using semi-supervised learning, so that abstract musical qualities can be quantified in a more concrete manner with limited amount of labels.

While the strength of arousal transfer is gradually increased, we find that the identity of the original piece is also gradually shifted. A recent work on text generation using VAEs [43] observed this similar trait and attributed its cause to the "latent vacancy" problem by topological analysis. A possible solution is to adopt the Constrained-Posterior VAE [43], in which we aim to explore in future work. Future work will also focus on applying the framework on other sets of abstract musical qualities (such as valence [37], tension [44], etc.), and extending the framework to model multi-track music with longer duration to produce more complete music.

---

[2] Source code available at: https://github.com/gudgud96/music-fader-nets

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] R. Habib, S. Mariooryad, M. Shannon, E. Battenberg, R. Skerry-Ryan, D. Stanton, D. Kao, and T. Bagby, "Semi-supervised generative modeling for controllable speech synthesis," in *International Conference of Learning Representations*, 2020.

[2] A. Aljanaki, Y.-H. Yang, and M. Soleymani, "Emotion in music task at mediaeval 2015." in *MediaEval*, 2015.

[3] L. N. Ferreira and J. Whitehead, "Learning to generate music with sentiment," in *Proc. of the International Society for Music Information Retrieval Conference*, 2019.

[4] R. Bresin and A. Friberg, "Emotional coloring of computer-controlled music performances," *Computer Music Journal*, vol. 24, no. 4, pp. 44–63, 2000.

[5] S. R. Livingstone, R. Muhlberger, A. R. Brown, and W. F. Thompson, "Changing musical emotion: A computational rule system for modifying score and performance," *Computer Music Journal*, vol. 34, no. 1, pp. 41–64, 2010.

[6] S. K. Ehrlich, K. R. Agres, C. Guan, and G. Cheng, "A closed-loop, music-based brain-computer interface for emotion mediation," *PloS one*, vol. 14, no. 3, 2019.

[7] Y. Kim, S. Wiseman, and A. M. Rush, "A tutorial on deep latent variable models of natural language," *arXiv preprint arXiv:1812.06834*, 2018.

[8] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," *arXiv preprint arXiv:1611.05148*, 2016.

[9] D. Herremans and C.-H. Chuan, "The emergence of deep learning: new opportunities for music and audio technologies," *Neural Computing and Applications*, vol. 32], p. 913–914, 2020.

[10] J.-P. Briot, G. Hadjeres, and F. Pachet, *Deep learning techniques for music generation*. Springer, 2019, vol. 10.

[11] D. Herremans, C.-H. Chuan, and E. Chew, "A functional taxonomy of music generation systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–30, 2017.

[12] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, pp. 1–13, 2018.

[13] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long term structure," in *International Conference of Learning Representations*, 2019.

[14] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[15] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, 2015, pp. 3483–3491.

[16] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, "Fader networks: Manipulating images by sliding attributes," in *Advances in Neural Information Processing Systems*, 2017, pp. 5967–5976.

[17] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International Conference of Machine Learning*, 2018.

[18] G. Hadjeres, F. Nielsen, and F. Pachet, "Glsr-vae: Geodesic latent space regularization for variational autoencoder architectures," in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017, pp. 1–7.

[19] A. Pati and A. Lerch, "Latent space regularization for explicit control of musical attributes," in *ICML Machine Learning for Music Discovery Workshop (ML4MD), Extended Abstract, Long Beach, CA, USA*, 2019.

[20] J. Engel, M. Hoffman, and A. Roberts, "Latent constraints: Learning to generate conditionally from unconditional generative models," in *International Conference of Learning Representations*, 2017.

[21] S. Dai, Z. Zhang, and G. G. Xia, "Music style transfer: A position paper," in *Proc. of International Workshop on Musical Metacreation*, 2018.

[22] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with transformer autoencoders," in *International Conference of Machine Learning*, 2020.

[23] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2016, pp. 2172–2180.

[24] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework." in *International Conference of Learning Representations*, 2017.

[25] H. Kim and A. Mnih, "Disentangling by factorising," in *International Conference of Machine Learning*, 2018.

[26] L. Yingzhen and S. Mandt, "Disentangled sequential autoencoder," in *International Conference on Machine Learning*, 2018, pp. 5670–5679.

[27] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," in *Advances in neural information processing systems*, 2017, pp. 1878–1889.

[28] Y. Wang, D. Stanton, Y. Zhang, R. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, F. Ren, Y. Jia, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *International Conference of Machine Learning*, 2018.

[29] Y.-J. Luo, K. Agres, and D. Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders," in *Proc. of the International Society for Music Information Retrieval Conference*, 2019.

[30] Y.-N. Hung, Y.-A. Chen, and Y.-H. Yang, "Learning disentangled representations for timber and pitch in music audio," *arXiv preprint arXiv:1811.03271*, 2018.

[31] Y.-N. Hung, I. Chiang, Y.-A. Chen, Y.-H. Yang *et al.*, "Musical composition style transfer via disentangled timbre representations," in *International Joint Conference of Artificial Intelligence*, 2019.

[32] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer," in *Proc. of the International Society for Music Information Retrieval Conference*, 2018.

[33] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proc. of the International Society for Music Information Retrieval Conference*, 2019.

[34] T. Akama, "Controlling symbolic music generation based on concept learning from domain knowledge," in *Proc. of the International Society for Music Information Retrieval Conference*, 2019.

[35] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference of Learning Representations, ICLR*, 2014.

[36] W.-N. Hsu, Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Y. Wang, Y. Cao, Y. Jia, Z. Chen, J. Shen *et al.*, "Hierarchical generative modeling for controllable speech synthesis," in *International Conference of Learning Representations*, 2019.

[37] J. A. Russell, "A circumplex model of affect." *Journal of personality and social psychology*, vol. 39, no. 6, p. 1161, 1980.

[38] A. Gabrielsson and E. Lindström, "The influence of musical structure on emotional expression." 2001.

[39] P. Gomez and B. Danuser, "Relationships between musical structure and psychophysiological measures of emotion." *Emotion*, vol. 7, no. 2, p. 377, 2007.

[40] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," in *Proc. of the International Society for Music Information Retrieval Conference*, 2010.

[41] R. Shu, Y. Chen, A. Kumar, S. Ermon, and B. Poole, "Weakly supervised disentanglement with guarantees," in *International Conference of Learning Representations*, 2020.

[42] T. Adel, Z. Ghahramani, and A. Weller, "Discovering interpretable representations for both deep generative and discriminative models," in *International Conference on Machine Learning*, 2018, pp. 50–59.

[43] P. Xu, J. C. K. Cheung, and Y. Cao, "On variational learning of controllable representations for text without supervision," in *International Conference on Machine Learning*, 2020.

[44] D. Herremans and E. Chew, "Morpheus: generating structured music with constrained patterns and tension," *IEEE Transactions on Affective Computing*, 2017.

FEW-SHOT DRUM TRANSCRIPTION IN POLYPHONIC MUSIC

Yu Wang[1]   **Justin Salamon**[2]   **Mark Cartwright**[1]   **Nicholas J. Bryan**[2]   **Juan Pablo Bello**[1]

[1] Music and Audio Research Lab, New York University, USA

[2] Adobe Research, San Francisco, USA

{wangyu, mark.cartwright, jpbello}@nyu.edu, {salamon, nibryan}@adobe.com
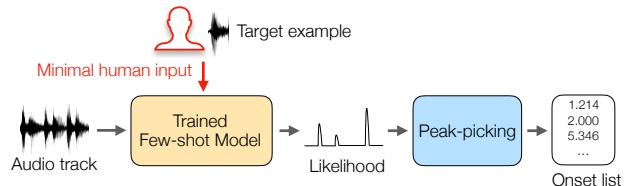
ABSTRACT

Data-driven approaches to automatic drum transcription (ADT) are often limited to a predefined, small vocabulary of percussion instrument classes. Such models cannot recognize out-of-vocabulary classes nor are they able to adapt to finer-grained vocabularies. In this work, we address open vocabulary ADT by introducing few-shot learning to the task. We train a Prototypical Network on a synthetic dataset and evaluate the model on multiple real-world ADT datasets with polyphonic accompaniment. We show that, given just a handful of selected examples at inference time, we can match and in some cases outperform a state-of-the-art supervised ADT approach under a fixed vocabulary setting. At the same time, we show that our model can successfully generalize to finer-grained or extended vocabularies unseen during training, a scenario where supervised approaches cannot operate at all. We provide a detailed analysis of our experimental results, including a breakdown of performance by sound class and by polyphony.

1. INTRODUCTION

Automatic Drum Transcription (ADT) aims at deriving a symbolic annotation of percussion instrument events from a music audio recording. It is a subtask of Automatic Music Transcription, where the aim is to transcribe all events within a musical piece. An accurate ADT system enables diverse applications in music education, music production, music search and recommendation, and computational musicology.

Early studies on ADT often combined multiple signal processing, information retrieval, and machine learning techniques such as support vector machines (SVM) and hidden Markov models (HMM) [1–3]. While these methods work well when applied to solo drum recordings, they often generalize poorly when applied to polyphonic music [4]. Recent approaches utilizing non-negative matrix factorization (NMF) [5, 6] and deep neural networks [7, 8] have shown promising performance in the presence of polyphonic music. However, such systems are often

Figure 1. Few-shot drum transcription. We input a music recording and one or more target example sounds into our trained model, output the likelihood of the selected event over time, and then post-process to generate onset times per percussion instrument.

limited to transcribing a very small subset of percussive sound classes, such as the bass drum (BD), snare drum (SD), and hi-hat (HH). For deep learning methods, in particular, this is mainly due to the limited number and size of ADT datasets, and the small class vocabulary size of the annotations in these datasets [9–12]. Recently, studies have utilized synthetic data and deep learning to expand ADT systems to support transcribing larger vocabularies of 10 or more instruments [11, 13]. However, 10–20 classes are still far from the wide gamut of percussive instruments used in recorded music. For example, rare or non-western percussion sounds are usually considered out-of-vocabulary. Moreover, when transcribing different datasets, we often need to manually map the percussion instruments in a dataset to the limited output vocabulary of an existing ADT system with reduced granularity. It can also be challenging for ADT systems that utilize fully-supervised learning to generalize to different musical genres or diverse drum sounds [11].

Recently, few-shot learning has been proposed for recognizing and detecting novel sound events [14–17], which is of great relevance to ADT. Under this paradigm, a model is trained to learn to recognize novel classes, unseen during training, given only very few examples from each class at inference time. Expanding on this, a few-shot sound event detection system for open vocabularies was recently proposed [14], yielding a search-by-example paradigm where a human first selects a handful of target example sounds that are passed to a trained model that automatically locates similar sounding events within the same audio track. This work, however, was developed for speech data, while other studies have focused on environmental sound. The main challenges in applying few-shot learning to music audio are the limited size of available datasets and the polyphonic

nature of music audio. While few-shot learning methods are designed to work with few labeled examples at inference time, they still require large amounts of labeled data at train time. Standard few-shot models are designed to solve multi-class problems where only one class is active at a time, while polyphonic music is inherently multi-label since multiple instruments can be active at once.

In this work, we propose a new paradigm for drum transcription in polyphonic music by introducing few-shot learning. Instead of trying to train a standard supervised model with more data for better generalizability, we propose to incorporate minimal human input with a few-shot model. Our proposed few-shot drum transcription system is shown in Figure 1, which can easily adapt to detecting a novel percussion instrument given a handful (e.g., five) of labeled target examples. By doing so, we can support open-vocabulary drum transcription, while minimizing the human labeling effort in order to make the transcription process as close to automatic as possible. To address the aforementioned challenges to applying few-shot learning to polyphonic music audio, we propose (1) utilizing a large synthetic dataset for training and (2) transcribing one percussion instrument at a time as a binary classification problem during inference. We evaluate our proposed model on multiple real music ADT datasets, compare it to a state-of-the-art supervised learning benchmark, and provide a detailed analysis of our model's performance including breakdowns by instrument class and polyphony. We show that our approach not only matches or outperforms past methods, but enables open vocabulary drum transcription, which is highly-advantageous for real-world applications.

2. METHODS

2.1 Prototypical Networks with Episodic Training

In our work, we focus on metric learning-based few-shot methods and, in particular, prototypical networks [18–21]. Prototypical networks have been found to perform well on several audio-related tasks [14, 15, 17, 22], rely on a simple training framework, and support efficient feed-forward inference [20], all of which are advantageous for our problem domain. They are designed to project an input audio example into a discriminative embedding space such that similar sounding events are clustered around a single prototype (average class embedding) via a neural network. Classification is then performed for an embedded query point by simply finding the nearest class prototype via the squared Euclidean distance.

Few-shot learning models, and prototypical networks specifically, are typically trained to solve a *C-way K-shot* multi-class classification task. In this setup, the method is tasked with labeling a query recording with one of C novel class labels, given K labeled examples per class at inference time, where K is typically a small number in the range of one to five. The availability of only very few examples of the new classes limits our ability to fine-tune a pre-trained model. To address this, *episodic training* has been proposed to train a prototypical network, which mim-

ics the few-shot inference problem during training, improving model generalizability [19]. In each training iteration, a training *episode* is formed by randomly selecting C classes from the training set. For each selected class, K samples are first selected to build a support set $\mathcal{S}$ of size of $C \times K$, while a disjoint set of q samples are selected to form a query set $\mathcal{Q}$ of size $C \times q$. Prototypes $\mathcal{M} = \{\mu_1, ..., \mu_C\}$ are the mean vectors of the embedded support samples belonging to each class:

$$\mu_c = \frac{1}{K} \sum_{(x,y) \in \mathcal{S}_c} f_\theta(x), \qquad (1)$$

where $\mathcal{S}_c$ denotes a set of examples labeled with class c and f_θ is parametrized by a neural network. Given a sample $\mathbf{x}_q$ in $\mathcal{Q}$, we take a softmax over distances to the prototypes in the embedding space to obtain per-class likelihoods:

$$p_\theta(y = c \mid \mathbf{x}_q) = \frac{\exp(-d(f_\theta(\mathbf{x}_q), \mu_c))}{\sum_{c'} \exp(-d(f_\theta(\mathbf{x}_q), \mu_{c'}))}, \quad (2)$$

where d is the squared Euclidean distance function. The training objective is to minimize the negative log-likelihood of the true class c:

$$\mathcal{L}(\theta) = -\log p_\theta(y = c \mid \mathbf{x}_q). \qquad (3)$$

Therefore, in each training episode, the model is learning to solve a *C-way K-shot* classification task. By training with a large collection of episodes, each consisting of a different set of C classes, the model learns *how to learn* from limited labeled data and obtains a class-agnostic discriminative ability. In this work, we train a prototypical network on a *10-way 5-shot* classification task [14] as the few-shot model in our proposed system.

2.2 Few-shot Drum Transcription

While the training task is a specific *C-way K-shot* classification, the trained few-shot model provides an embedding function that projects the input data into a discriminative space in which sound events are classified by finding the nearest class prototype, where each prototype is derived from a few examples. We propose to use this embedding space for percussion sound event detection by providing a support set containing examples for both the positive (target) and negative (non-target) classes, and classify a given query by measuring its distance to the positive and negative class prototypes. Here, the trained few-shot model is essentially performing a binary, 2-way, classification at inference time for each target class, ultimately resulting in a multi-label prediction.

Given a target instrument and an audio track, we first slice the track into a series of query frames. To construct a support set of labeled examples for the few-shot model, we randomly sample target examples from the track as positive examples, simulating the human input in Figure 1, and take all frames within the track as negative examples to model the non-target class. Note that, while the full track will also contain the target class, previous work has shown that since the target class is relatively sparse compared to

the full track, this strategy works well [14]. Given the support set, the trained few-shot model outputs the likelihood of each query frame containing the target class. Finally, we perform peak-picking on these probabilities to get a list of onset locations as is commonly done for ADT [11, 13].

3. EXPERIMENTAL DESIGN

To evaluate the proposed few-shot drum transcription paradigm, we first train a prototypical network as our few-shot model on a large synthetic dataset. Then, we apply the trained model to three real-music ADT datasets to get transcription performance. We focus on one target instrument at a time and use randomly selected target examples to simulate human input at inference time.

3.1 Dataset for Episodic Training: Slakh2100

We use the Slakh2100 dataset to train our few-shot model [23]. Slakh2100 is synthesized from the Lakh MIDI Dataset [24] using professional-grade sample-based virtual instruments. It contains 2100 automatically mixed tracks and accompanying MIDI files, totaling 145 hours of mixtures. Slakh2100 is synthesized using eight different drum patches, where each patch can be viewed as a unique drummer playing a unique drum kit. In each patch, there are around 25 to 45 different percussion classes, each consisting of a combination of a percussion instrument with a playing technique. For episodic training, we alternatively define a class as a specific percussion class (e.g. snare drum side stick) played by a specific patch, resulting in a total of 282 classes. Each drum patch has its own MIDI note-instrument mapping, which does not follow the general MIDI convention. We manually check the mapping to group duplicates and remove empty ones. Each patch is used to synthesize approximately 250 songs. We partition the dataset into patch-conditional train, validation and test splits using 5, 1, 2 patches per split, respectively.
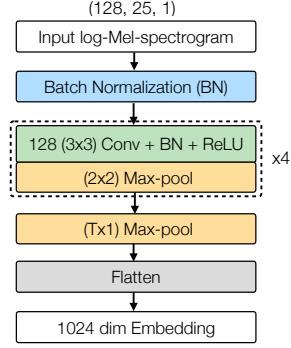
3.2 Evaluation Datasets

3.2.1 ENST-Drums

ENST-Drums is a dataset of recordings from three drummers each playing a different drum kit [9]. It contains drum onset annotations for 20 classes of percussion sounds. While the dataset also contains many solo drum recordings, we only use the subset of 64 recordings with accompaniment for evaluation. The accompaniments are mixed with corresponding drum tracks using a scaling factor of 1/3 and 2/3 to get natural-sounding mixtures and to be consistent with prior studies [6, 11, 25].

3.2.2 MDB-Drums

MDB-Drums is a set of 23 fully-produced music tracks from the MedleyDB dataset [10, 26]. It contains two levels of drum onset annotations — we use the finer level which divides the classes by instrument and playing technique, resulting in 21 classes.

Figure 2. Our backbone prototypical network embedding model architecture.

3.2.3 RBMA13

RBMA13 consists of 30 fully-produced music tracks in the genres of electronic dance music, singer-songwriter, and fusion-jazz [11]. The drum sounds of this set are more diverse compared to the previous sets, and it is considered a particularly difficult dataset [11]. It contains annotations for 23 percussive classes.

3.3 Training

For each percussion instrument onset in Slakh2100, we center a 250 ms context window around the onset as the input to the model. We compute a log-scaled Mel-spectrogram from the context window with `librosa` [27] using a window size of 46 ms (2048 samples for a sample rate of 44.1 kHz) and a hop size of 10 ms. In preliminary experiments, we studied a range of short (160 ms) to long (500 ms) context windows and found that a 250 ms window yields consistent, well-performing results across different datasets. We conjecture that a 250 ms window is wide enough to capture most percussive onsets, while also capturing some context around the onset.

To construct a *10-way 5-shot* training episode, we randomly sample a drum patch from the training set, sample 10 percussion instrument classes from the drum patch, and sample 5 instances per class as the support set. Note that, while each instance is guaranteed to contain the target class, it may also contain other sound classes if they overlap in time with the target class. The query set is comprised of 16 separate instances per each of the 10 classes [28].

We use a backbone convolutional neural network (CNN) to embed the input as shown in Figure 2. It consists of four convolution blocks, each of which has a convolutional layer with a 3×3 kernel, a batch normalization layer, a ReLU activation layer, and a 2×2 max-pooling layer. To allow our model to handle varying-duration input, we apply max-pooling along the time dimension to the output of the convolution blocks (rather than a fully connected layer). Finally, we flatten the feature map to get an embedding with a dimensionality of 1024. We train our model using the `Adam` optimizer [29] in `PyTorch` [30] with a learning rate of 0.001 for 100,000 episodes with early stopping. We choose the best model based on the

Vocab size	Model	Target examples	ENST-Drums			MDB-Drums			RBMA13		
			micro	macro	macro*	micro	macro	macro*	micro	macro	macro*
18	CRNN [11]	-	**0.67**	-	0.74	**0.60**	-	0.78	0.47	-	0.64
	Proto. Net	Include	0.55	**0.54**	**0.80**	0.58	**0.60**	**0.87**	**0.56**	**0.55**	**0.81**
		Exclude	0.49	0.45	0.76	0.51	0.49	0.83	0.54	0.50	0.79

Table 1. F-measure evaluated on real polyphonic music datasets under a fixed 18-class vocabulary [11].

Vocab size	Model	Target examples	ENST-Drums			MDB-Drums			RBMA13		
			micro	macro	macro*	micro	macro	macro*	micro	macro	macro*
All	Proto. Net	Include	0.55	0.54	0.89	0.60	0.61	0.90	0.54	0.53	0.83
		Exclude	0.49	0.45	0.87	0.53	0.48	0.88	0.52	0.48	0.81

Table 2. F-measure evaluated on real polyphonic music datasets under all classes that exist in each dataset.

few-shot classification loss on the validation set.

3.4 Evaluation

To evaluate our proposed few-shot drum transcription paradigm, we apply the prototypical network trained on Slakh2100 to perform drum transcription on three real music datasets. For each dataset, we first evaluate transcription under the fixed vocabulary scenario by mapping percussion instruments to a predefined 18-class vocabulary used in a state-of-the-art ADT system [11] for comparison. Then, we transcribe all classes that exist in the test set, including those classes that do not exist in our training data, mimicking the open vocabulary scenario.

Given a target class and an audio track, we first pre-process the track into a series of overlapping query frames with a 250 ms window size, matching the context window used during training, and 10 ms hop size. To simulate human selections at inference time, we randomly sample 5 target examples from the track as positive examples. Then, we take all frames within the track as negative examples and predict each query frame as described in Section 2.2. We run 10 iterations of this prediction process to account for randomness and concatenate all predictions to compute performance metrics. We estimate target onset locations from the model output using the peak picking method described in [31]. A frame n is selected as an onset if the corresponding output probability $p(n)$ meets the following criteria:

1. $p(n) = max(p(n - w_1) : p(n + w_2))$,

2. $p(n) \geq mean(p(n - w_3) : p(n + w_4)) + \delta$,

3. $n - n_{last\ onset} > w_5$,

where δ is a threshold parameter, w_1 to w_4 are sample offset values defining the windows for the max and $mean$ functions, and w_5 is the minimum allowed number of samples between onsets.

We divide each dataset used for evaluation into three splits for 3-fold cross-validation. For each fold, we tune the peak-picking parameters on the validation split using a randomized search with 1000 iterations, and perform drum

transcription on the test split. Finally, we report the model performance averaged over the three test splits. For ENST-Drums, each split contains a different drummer. For MDB-Drums and RBMA13, we use the same splits as [11].

3.5 Metrics

We compute performance metrics by first using the `onset_evaluation` function in madmom [32] to find matching onset locations with a 20 ms tolerance window. We then compute F-measure as the primary performance metric, using both *micro* and *macro* aggregation. For *micro* F-measure, we aggregate all true positives (TP), false positives (FP), and false negatives (FN) over all classes and tracks in the entire dataset. For *macro* F-measure, we first compute a track-level F-measure for each track by averaging all class-level F-measures in the track, and we then average over all track-level F-measures in the dataset to compute the final metric.

When computing *macro* F-measure, if an instrument does not exist in a track and the ADT model under evaluation does not predict any corresponding positive labels, previous work defined its class-level F-measure to be 1 [11]. This convention is informative for a standard supervised approach since the model may produce false positives for non-existing classes. However, a few-shot model would never predict non-existing classes since there are no positive examples to begin with, which is one of the advantages of the few-shot drum transcription paradigm. Therefore, when evaluating our few-shot model, it makes more sense to exclude non-existing classes in a track from the evaluation to avoid artificially inflating the *macro* F-measure. For completeness and comparison to previous work, however, we report both variants, either excluding non-existing classes in a track (*macro* F-measure) or following the convention of setting the F-measure for such classes to 1 (*macro** F-measure).

Given that our model requires five user-labeled examples from the test data in each iteration of prediction, we can compute the aforementioned performance metrics either including or excluding the user-labeled examples. The former represents the joint human-computer performance of the proposed paradigm that a user would experience for

a track, while the latter represents our model's performance on strictly unlabeled data (i.e., the rest of the track). We report both variants for each metric described in the previous paragraph (labeled as Include/Exclude in the Tables).

4. RESULTS
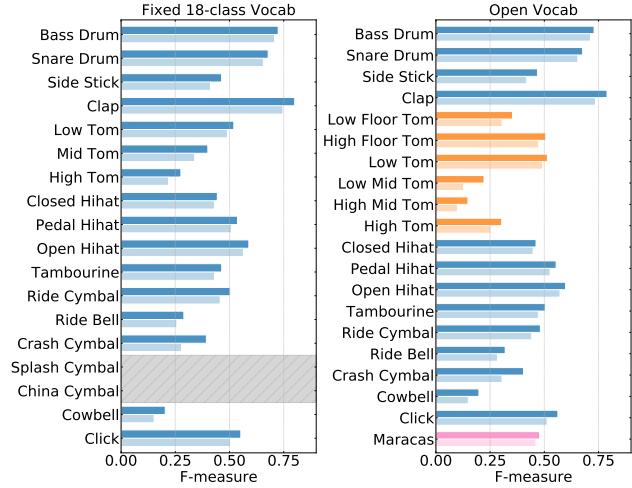
4.1 Fixed Vocabulary

We first compare our few-shot drum transcription approach to a state-of-the-art CRNN model, which was trained on synthetic data and fine-tuned on real music data, under a fixed 18-class vocabulary [11]. Note that while NMF-based methods can be considered more closely related to our approach, they require iterative optimization at test time and the determination of the non-negative rank for the decomposition process can be difficult. Most previous NMF-based ADT systems were evaluated on solo drum tracks and a small subset of percussive sound classes [5, 12, 33]. Therefore, in this work, we choose the CRNN model as the baseline system and plan to compare our approach to NMF-based methods as part of future work.

In Table 1 we present model performance on three real music datasets. From these results, we find three distinct insights applicable to the fixed vocabulary ADT tasks. First, the results show that our approach, a prototypical network trained on synthetic data with only five examples provided at inference time, gives comparable and in some cases better performance compared to previous, fully supervised state-of-the-art results. Second, we see that our model performance is relatively stable across different datasets. Third, our proposed approach outperforms the supervised model on RBMA13 by a large margin, which is considered a difficult dataset with diverse drum sounds [11]. For instance, snare drum sounds on different tracks in RBMA13 are very diverse and can sound very different. Standard supervised approaches typically struggle to generalize well for classes with high intra-class variation. However, while a percussive sound class may exhibit large intra-class variation across different tracks (e.g. different tracks may have very different snare drum sounds), it's often the case that such sound classes display far less intra-class variation within the same track (e.g., the same snare drum is used throughout a track). Since our few-shot model detects a sound class based on target examples from the same track, it is considerably more robust when it comes to intra-class variation, as evidenced by the quantitative results. This highlights the strength of the few-shot drum transcription paradigm which instead of aiming at generalization, aims for quick adaptation with minimal human input.

4.2 Open Vocabulary

Next, we evaluate our few-shot model under an open-vocabulary scenario. Here, we evaluate the model against all the classes in each test set, including classes that were never seen by the model during training. Specifically, we evaluate on 20 classes for ENST-Drums, 19 classes for MDB-Drums, and 20 classes for RBMA13. Classes that do

Figure 3. F-measure for each percussion instrument in the RBMA13 dataset. (Left) Under 18-class vocabulary. (Right) Under all 20 classes. For each instrument, we show the metrics computed with target examples included (dark bar) and excluded (light bar).

not appear more than five times in any track in the dataset are filtered out. Note that 6, 4, and 1 classes within the full vocabulary of ENST, MDB, and RBMA respectively do not exist in our training data. The results are presented in Table 2. Note that we do not compare our model to the fully supervised model in this scenario since, as noted earlier, such a model would fail (by design) to recognize classes that are outside of the training vocabulary. When we compare these results to those in Table 1 (fixed vocabulary), we note that there is no drop in performance when moving from a fixed known vocabulary to an open vocabulary with previously unseen sound classes. This is a direct result of adapting few-shot learning to ADT and highlights the benefit of our proposed approach.

Next, we break down the performance of the few-shot model by instrument class under both the fixed and open vocabulary scenarios on RBMA13, presented in Figure 3. Here, two out of 18 predefined classes, splash cymbal and china cymbal, do not exist in RBMA13 annotations and thus the results for these classes are absent from the figure. We see that in the open vocabulary scenario, we are able to transcribe fine-grained classes such as six different tom drums (orange bars in Figure 3) with comparable performance to predicting a coarser, fixed vocabulary. We can also transcribe Maracas (pink bar in Figure 3) which our model has not seen at training.

4.3 Transcribing Novel Classes

In the previous section, we saw that the model can detect a class that is out of the training vocabulary. To evaluate this more quantitatively, we re-train our few-shot model on the Slakh2100 dataset while completely excluding three classes from the training data: bass drum, tambourine, and clap, representing both common and rare classes. We then evaluate the model on predicting these three classes in the Slakh2100 test set and compare the results to those we ob-

Training data	Target examples	$\mathcal{H}$ in Slakh2100 test set		
		micro	macro	macro*
Slakh2100	Include	0.66	0.66	0.83
	Exclude	0.64	0.64	0.83
Slakh2100 - $\mathcal{H}$	Include	0.62	0.62	0.81
	Exclude	0.61	0.60	0.80

Table 3. F-measure evaluated on three classes: $\mathcal{H} = \{$bass drum, tambourine, clap$\}$ in the Slakh2100 test set when training with the entire Slakh2100 training set or with three classes held out from the training data.
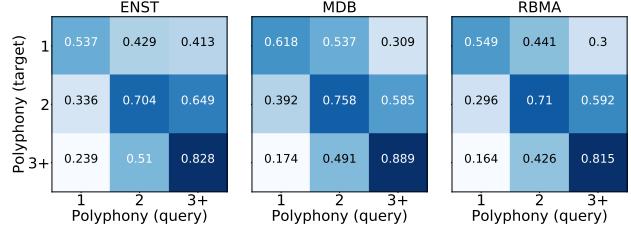
tain when the three classes are included in the training data.

We present the results in Table 3. We see that the performance of the model is very stable, with only a minor decrease in F-measure when predicting classes that are completely excluded from the training set. This confirms that with the few-shot training paradigm, our model can detect entirely unseen classes given just a few examples at inference time.

4.4 Performance Breakdown by Polyphony

Next, we focus on the target example selection process at inference time. Due to the polyphonic nature of music audio, when a target example is selected, it can include non-target instrument sounds, played at the same time. We want to investigate how the polyphony of these selected examples affects transcription performance. To do so, we repeat our evaluation process three more times, each time varying the support set such that all positive target examples have the same degree of polyphony: 1, 2, and 3 or more (3+). To define the polyphony of each example, we look at a 20 ms window around its onset and count the number of percussion instruments that co-occur within the window. To assess how the performance is affected by the polyphony of the *query*, we break down the performance by the polyphony of the query frames.

We present the results in Figure 4, which show similar trends across the three evaluation datasets. First, we see high performance on the diagonal, where the polyphony between target and query examples match. Along the diagonal, performance also increases with increasing polyphony, for which a possible explanation is that the chance of having exactly matched instrument sources between target and query examples increases at high polyphony. That is, the number of different percussion instrument combinations decreases with increasing polyphony, due to the underlying pattern of drum playing. Another insight is that when there is a mismatch between target and query polyphony, having lower polyphony in target examples than in query examples gives better performance than the other way around (comparing the upper right triangle to the lower-left triangle in each figure). This matches the intuition that when the target examples have high polyphony, it is difficult for the few-shot model to latch onto the correct target instrument, result-

Figure 4. Break down F-measure by polyphony in both target and query examples.

ing in poor performance even on query examples with low polyphony. On the other hand, it is easier for the model to find the target class in common within examples with lower polyphony.

Overall, the results show that the performance of our few-shot drum transcription approach can significantly depend on the target examples selected for the support set. In future work, we plan to build on top of these results to investigate the best strategy of composing a support set, and how we can inform user to make effective selections.

5. LIMITATION AND FUTURE WORK

The main limitation of our proposed approach is that it requires the user to provide a few examples at inference time. If the vocabulary is known in advance and there is sufficient training data for each sound class, a fully supervised approach has the advantage of not requiring any user intervention. However, when the vocabulary is not known in advance, or when there isn't enough training data for every class, the few-shot paradigm is clearly advantageous. For future work, we plan to investigate the example selection process at inference time and the corresponding human element. This includes studying how we can compose sets of support examples to maximize performance, and how we can guide the user to those selections.

6. CONCLUSION

In this work, we address open vocabulary ADT by proposing a few-shot drum transcription paradigm, a combination of a few-shot model with minimal human input. We train a prototypical network on the Slakh2100 dataset as the few-shot model, and evaluate the proposed few-shot drum transcription system on multiple real-world ADT datasets with polyphonic accompaniment. We show that, given just a handful of target examples, we can match and, in some cases outperform, a state-of-the-art supervised ADT approach under a fixed vocabulary setting. At the same time, we show that our model can successfully generalize to finer-grained class labeling and extended vocabularies unseen during training. Lastly, we investigate the dependence of few-shot model performance on the polyphony of target examples. We show that matching polyphony in target and query examples gives better performance and when there is a mismatch, having lower polyphony in target examples than in query examples gives better results.

7. ACKNOWLEDGMENT

8. REFERENCES

[1] O. Gillet and G. Richard, "Automatic transcription of drum loops," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004.

[2] G. Tzanetakis, A. Kapur, and R. I. McWalter, "Subband-based drum transcription for audio signals," in *IEEE 7th Workshop on Multimedia Signal Processing*, 2005, pp. 1–4.

[3] D. Fitzgerald, E. Coyle, and B. Lawlor, "Sub-band independent subspace analysis for drum transcription," in *Proc. of the Digital Audio Effects Conference (DAFX)*, 2002.

[4] O. Gillet and G. Richard, "Drum track transcription of polyphonic music using noise subspace projection." in *Proc. of the 6th International Society for Music Information Retrieval Conference (ISMIR)*, 2005, pp. 92–99.

[5] C. Wu and A. Lerch, "Drum transcription using partially fixed non-negative matrix factorization," *23rd European Signal Processing Conference (EUSIPCO)*, pp. 1281–1285, 2015.

[6] ——, "Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data," in *Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[7] R. Vogl, M. Dorfer, and P. Knees, "Drum transcription from polyphonic music with recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 201–205.

[8] C. Southall, R. Stables, and J. Hockman, "Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks," in *Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[9] O. Gillet and G. Richard, "ENST-Drums: an extensive audio-visual database for drum signals processing." in *Proc. of the 7th International Society for Music Information Retrieval Conference (ISMIR)*, 2006, pp. 156–159.

[10] C. Southall, C. Wu, A. Lerch, and J. A. Hockman, "MDB Drums — An Annotated Subset of MedleyDB for Automatic Drum Transcription," in *Late Breaking Demo (Extended Abstract), Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[11] R. Vogl, G. Widmer, and P. Knees, "Towards multi-instrument drum transcription," in *Proc. of the 21st International Conference on Digital Audio Effects (DAFx)*, 2018, pp. 57–64.

[12] C. Dittmar and D. Gärtner, "Real-time transcription and separation of drum recordings based on NMF decomposition," in *Proc. of the International Conference on Digital Audio Effects (DAFx)*, 2014, pp. 187–194.

[13] M. Cartwright and J. P. Bello, "Increasing drum transcription vocabulary using data synthesis," in *Proc. of the 21st International Conference on Digital Audio Effects (DAFx)*, 2018, pp. 57–64.

[14] Y. Wang, J. Salamon, N. J. Bryan, and J. P. Bello, "Few-shot sound event detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 81–85.

[15] S. Chou, K. Cheng, J. R. Jang, and Y. Yang, "Learning to match transient sound events using attentional similarity for few-shot sound recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 26–30.

[16] K. Cheng, S. Chou, and Y. Yang, "Multi-label few-shot learning for sound event recognition," in *IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, 2019, pp. 1–5.

[17] B. Shi, M. Sun, K. C. Puvvada, C. Kao, S. Matsoukas, and C. Wang, "Few-shot acoustic event detection via meta learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 76–80.

[18] G. R. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Workshop*, 2015.

[19] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 3630–3638.

[20] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 4077–4087.

[21] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 1199–1208.

[22] J. Pons, J. Serrà, and X. Serra, "Training Neural Audio Classifiers with Few Data," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 16–20.

[23] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019, pp. 45–49.

[24] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, 2016.

[25] O. Gillet and G. Richard, "Extraction and remixing of drum tracks from polyphonic music signals," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2005, pp. 315–318.

[26] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, "MedleyDB: A multitrack dataset for annotation-intensive MIR research," in *Proc. of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[27] B. McFee, V. Lostanlen, M. McVicar, A. Metsai, S. Balke, C. Thomé, C. Raffel *et al.*, "librosa/librosa: 0.7.0," 2019.

[28] W. Chen, Y. Liu, Z. Kira, Y. F. Wang, and J. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2019.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2014.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[31] S. Böck, F. Krebs, and M. Schedl, "Evaluating the online capabilities of onset detection methods." in *Proc. of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 49–54.

[32] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "Madmom: A New Python Audio and Music Signal Processing Library," in *Proc. of the 24th ACM International Conference on Multimedia*, 2016, p. 1174–1178.

[33] H. Lindsay-Smith, S. McDonald, and M. Sandler, "Drumkit transcription via convolutive nmf," in *Proc. of the 15th International Conference on Digital Audio Effects (DAFx)*, 2012.

dMELODIES: A MUSIC DATASET FOR DISENTANGLEMENT LEARNING

Ashis Pati   **Siddharth Gururani**   **Alexander Lerch**

Center for Music Technology, Georgia Institute of Technology, USA

`ashis.pati@gatech.edu, siddgururani@gatech.edu, alexander.lerch@gatech.edu`
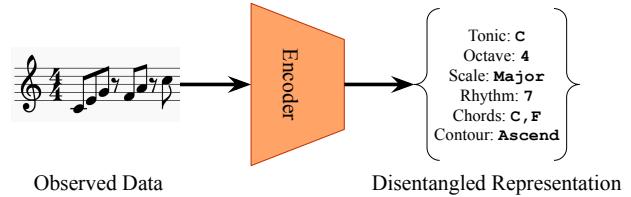
ABSTRACT

Representation learning focused on disentangling the underlying factors of variation in given data has become an important area of research in machine learning. However, most of the studies in this area have relied on datasets from the computer vision domain and thus, have not been readily extended to music. In this paper, we present a new symbolic music dataset that will help researchers working on disentanglement problems demonstrate the efficacy of their algorithms on diverse domains. This will also provide a means for evaluating algorithms specifically designed for music. To this end, we create a dataset comprising of 2-bar monophonic melodies where each melody is the result of a unique combination of nine latent factors that span ordinal, categorical, and binary types. The dataset is large enough (≈ 1.3 million data points) to train and test deep networks for disentanglement learning. In addition, we present benchmarking experiments using popular unsupervised disentanglement algorithms on this dataset and compare the results with those obtained on an image-based dataset.

1. INTRODUCTION

Representation learning deals with extracting the underlying factors of variation in a given observation [1]. Learning compact and *disentangled* representations (see Figure 1 for an illustration) from given data, where important factors of variation are clearly separated, is considered useful for generative modeling and for improving performance on downstream tasks (such as speech recognition, speech synthesis, vision and language generation [2–4]). Disentangled representations allow a greater degree of interpretability and controllability, especially for content generation, be it language, speech, or music. In the context of Music Information Retrieval (MIR) and generative music models, learning some form of disentangled representation has been the central idea for a wide variety of tasks such as genre transfer [5], rhythm transfer [6, 7], timbre synthesis [8], instrument rearrangement [9], manipulating musical attributes [10, 11], and learning music similarity [12].

Consequently, there exists a large body of research in

Figure 1: Disentanglement example where a high dimensional observed data is disentangled into a low dimensional representation comprising of semantically meaningful factors of variation.

the machine learning community focused on developing algorithms for learning disentangled representations. These span unsupervised [13–16], semi-supervised [17–19] and supervised [10, 20–22] methods. However, a vast majority of these algorithms are designed, developed, tested, and evaluated using data from the image or computer vision domain. The availability of standard image-based datasets such as dSprites [23], 3D-Shapes [24], and 3D-Chairs [25] among others has fostered disentanglement studies in vision. Additionally, having well-defined factors of variation (for instance, size and orientation in dSprites [23], pitch and elevation in Cars3D [26]) has allowed systematic studies and easy comparison of different algorithms. However, this restricted focus on a single domain raises concerns about the generalization of these methods [27] and prevents easy adoption into other domains such as music.

Research on disentanglement learning in music has often been application-oriented with researchers using their own problem-specific datasets. The factors of variation have also been chosen accordingly. To the best of our knowledge, there is no standard dataset for disentanglement learning in music. This has prevented systematic research on understanding disentanglement in the context of music.

In this paper, we introduce *dMelodies*, a new dataset of monophonic melodies, specifically intended for disentanglement studies. The dataset is created algorithmically and is based on a simple and yet diverse set of independent latent factors spanning ordinal, categorical and binary attributes. The full dataset contains ≈ 1.3 million data points which matches the scale of image datasets and should be sufficient to train deep networks. We consider this dataset as the primary contribution of this paper. In addition, we also conduct benchmarking experiments using three popular unsupervised methods for disentanglement learning and present a comparison of the results with the dSprites dataset [23]. Our experiments show that disentanglement

learning methods do not directly translate between the image and music domains and having a music-focused dataset will be extremely useful to ascertain the generalizability of such methods. The dataset is available online [1] along with the code to reproduce our benchmarking experiments. [2]

2. MOTIVATION

In representation learning, given an observation $\mathbf{x}$, the task is to learn a representation $r(\mathbf{x})$ which "makes it easier to extract useful information when building classifiers or other predictors" [1]. The fundamental assumption is that any high-dimensional observation $\mathbf{x} \in \mathcal{X}$ (where $\mathcal{X}$ is the dataspace) can be decomposed into a semantically meaningful low dimensional latent variable $\mathbf{z} \in \mathcal{Z}$ (where $\mathcal{Z}$ is referred to as the latent space). Given a large number of observations in $\mathcal{X}$, the task of disentanglement learning is to estimate this low dimensional latent space $\mathcal{Z}$ by separating out the distinct factors of variation [1]. An ideal disentanglement method ensures that changes to a single underlying factor of variation in the data changes only a single factor in its representation [27]. From a generative modeling perspective, it is also important to learn the mapping from $\mathcal{Z}$ to $\mathcal{X}$ to enable better control over the generative process.

2.1 Lack of diversity in disentanglement learning

Most state-of-the-art methods for unsupervised disentanglement learning are based on the Variational Auto-Encoder (VAE) [28] framework. The key idea behind these methods is that factorizing the latent representation to have an aggregated posterior should lead to better disentanglement [27]. This is achieved using different means, e.g., imposing constraints on the information capacity of the latent space [13, 29, 30], maximizing the mutual information between a subset of the latent code and the observations [31], and maximizing the independence between the latent variables [14, 15]. However, unsupervised methods for disentanglement learning are sensitive to inductive biases (such network architectures, hyperparameters, and random seeds) and consequently there is a need to properly evaluate such methods by using datasets from diverse domains [27].

Apart from unsupervised methods for disentanglement learning, there has also been some research on semi-supervised [18, 19] and supervised [20, 21, 32, 33] learning techniques to manipulate specific attributes in the context of generative models. In these paradigms, a labeled loss is used in addition to the unsupervised loss. Available labels can be utilized in various ways. They can help with disentangling known factors (e.g., digit class in MNIST) from latent factors (e.g., handwriting style) [34], or supervising specific latent dimensions to map to specific attributes [10]. However, most of these approaches are evaluated using image domain datasets.

Tremendous interest from the machine learning community has led to the creation of benchmarking datasets (albeit image-based) specifically targeted towards disentanglement learning such as dSprites [23], 3D-Shapes [24], 3D-chairs [25], MPI3D [35], most of which are artificially generated and have simple factors of variation. While one can argue that artificial datasets do not reflect real-world scenarios, the relative simplicity of these datasets is often desirable since they enable rapid prototyping.

2.2 Lack of consistency in music-based studies

Representation learning has also been explored in the field of MIR. Much like images, learning better representations has been shown to work well for MIR tasks such as composer classification [36, 37], music tagging [38], and audio-to-score alignment [39]. The idea of disentanglement has been particularly gaining traction in the context of interactive music generation models [5, 6, 11, 33]. Disentangling semantically meaningful factors can significantly improve the usefulness of music generation tools. Many researchers have independently tried to tackle the problem of disentanglement in the context of symbolic music by using different musically meaningful attributes such as genre [5], note density [10], rhythm [6], and timbre [8]. However, these methods and techniques have all been evaluated using different datasets which makes a direct comparison impossible. Part of the reason behind this lack of consistency is the difference in the problems that these methods were looking to address. However, the availability of a common dataset allowing researchers to easily compare algorithms and test their hypotheses will surely aid systematic research.

3. dMELODIES DATASET

The primary objective of this work is to create a simple dataset for music disentanglement that can alleviate some of the shortcomings mentioned in Section 2: first, researchers interested in disentanglement will have access to more diverse data to evaluate their methods, and second, research on music disentanglement will have the means for conducting systematic, comparable evaluation. This section describes the design choices and the methodology used for creating the proposed *dMelodies* dataset.

While core MIR tasks such as music transcription, or tagging focus more on analysis of audio signals, research on generative models for music has focused more on the symbolic domain. Considering most of the interest in disentanglement learning stems from research on generative models, we decided to create this dataset using symbolic music representations.

3.1 Design Principles

To enable objective evaluation of disentanglement algorithms, one needs to either know the ground-truth values of the underlying factors of variation for each data point, or be able to synthesize the data points based on the attribute values. The dSprites dataset [23], for instance, consists of single images of different 2-dimensional shapes with simple attributes specifying the position, scale and orientation of these shapes against a black background. The design of our

---

dataset is loosely based on the dSprites dataset. The following principles were used to finalize other design choices:

(a) The dataset should have a simple construction with homogenous data points and intuitive factors of variation. It should allow for easy differentiation between data points and have clearly distinguishable latent factors.

(b) The factors of variation should be independent, i.e., changing any one factor should not cause changes to other factors. While this is not always true for real-world data, it enables consistent objective evaluation.

(c) There should be a clear one-to-one mapping between the latent factors and the individual data points. In other words, each unique combination of the factors should result in a unique data point.

(d) The factors of variation should be diverse. In addition, it would be ideal to have the factors span different types such as discrete, ordinal, categorical and binary.

(e) Finally, the different combinations of factors should result in a dataset large enough to train deep neural networks. Based on size of the different image-based datasets [23,40], we would require a dataset of the order of at least a few hundred thousand data points.

3.2 Dataset Construction

Considering the design principles outlined above, we decided to focus on monophonic pitch sequences. While there are other options such as polyphonic or multi-instrumental music, the choice of monophonic melodies was to ensure simplicity. Monophonic melodies are a simple form of music uniquely defined by the pitch and duration of their note sequences. The pitches are typically based on the key or scale in which the melody is being played and the rhythm is defined by the onset positions of the notes.

Since the set of all possible monophonic melodies is very large and heterogeneous, the following additional constraints were imposed on the melody in order to enforce homogeneity and satisfy the other design principles:

(a) Each melody is based on a scale selected from a finite set of allowed scales. This choice of scale also serves as one of the factors of variation. The melody will also be uniquely defined by the pitch class of the tonic (root pitch) and the octave number.

(b) In order to constrain the space of all possible pitch patterns within a scale, we restrict each melody to be an arpeggio over the standard I-IV-V-I cadence chord pattern. Consequently, each melody consists of 12 notes (3 notes for each of the 4 chords).

(c) In order to vary the pitch patterns, the direction of arpeggiation of each chord, i.e. up or down, is used as a latent factor. This choice adds a few binary factors of variation to the dataset.

(d) The melodies are fixed to 2-bar sequences with 8th note as the minimum note duration. This makes the dataset uniform in terms of sequence lengths of the data points and also helps reduce the complexity of the sequences. 2-bar sequences have been used in other music generation studies as well [10, 41]. We use a tokenized data representation such that each melody is

Factor	# Options	Notes
Tonic	12	C, C#, D, through B
Octave	3	Octave 4, 5 and 6
Scale	3	major, harmonic minor, and blues
Rhythm Bar 1	28	$\binom{8}{6}$, based on onset locations of 6 notes
Rhythm Bar 2	28	$\binom{8}{6}$, based on onset locations of 6 notes
Arp Chord 1	2	up/down, for Chord 1
Arp Chord 2	2	up/down, for Chord 2
Arp Chord 3	2	up/down, for Chord 3
Arp Chord 4	2	up/down, for Chord 4

Table 1: Table showing the different factors of variation for the dMelodies dataset. Since all factors of variation are independent, the total dataset contains 1,354,752 unique melodies.
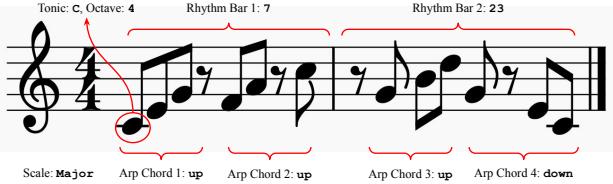
a sequence of length 16.

(e) If we consider the space of all possible unique rhythms, the number of options will explode to $\binom{16}{12}$ which will be significantly larger than other factors of variation. Hence, we choose to break the latent factor for rhythm into 2 independent factors: rhythm for bar 1 and bar 2.

(f) The rhythm of a melody is based on the metrical onset position of the notes [42]. Consequently, rhythm is dependent on the number of notes. In order to keep rhythm independent from other factors, we constrain each bar to have 6 notes (play 2 chords) thereby obtaining $\binom{8}{6}$ options for each bar.

Based on the above design choices, the dMelodies dataset consists of 2-bar monophonic melodies with 9 factors of variations listed in Table 1. The factors of variation were chosen to satisfy the design principles listed in Section 3.1. For instance, while melodic transformations such as repetition, inversion, retrograde would have made more musical sense, they did not allow creation of a large-enough dataset with independent factors of variation. The resulting dataset thus contains simple melodies which do not adequately reflect real-world musical data. A side-effect of this choice of factors is that some of them (such as arpeggiation direction and rhythm) affect only a specific part of the data. Since each unique combination of these factors results in a unique data point we get 1,354,752 unique melodies. Figure 2 shows one such melody from the dataset and its corresponding latent factors. The dataset is generated using the *music21* [43] python package.

4. BENCHMARKING EXPERIMENTS

In this section, we present benchmarking experiments to demonstrate the performance of some of the existing unsupervised disentanglement algorithms on the proposed dMelodies dataset and contrast the results with those obtained on the image-based dSprites dataset.

Figure 2: Example of a sample melody from the dMelodies dataset. Also shown are the values of the different latent factors. For rhythm latent factors, the shown value corresponds to the index from the rhythm dictionary.

4.1 Experimental Setup

We consider 3 different disentanglement learning methods: β-VAE [13], Annealed-VAE [29], and FactorVAE [15]. All these methods are based on different regularization terms applied to the VAE loss function.

4.1.1 Data Representation

We use a tokenized data representation [44] with the 8th-note as the smallest note duration. Each 8th note position is encoded with a token corresponding to the note name which starts on that position. A special continuation symbol ('__') is used which denotes that the previous note is held. A special token is used for rest.

4.1.2 Model Architectures

Two different VAE architectures are chosen to conduct these experiments. The first architecture (dMelodies-CNN) is based on Convolutional Neural Networks (CNNs) and is similar to those used for several image-based VAEs, except that we use 1-D convolutions. The second architecture (dMelodies-RNN) is based on a hierarchical recurrent model [41, 45]. Details of the model architectures are provided in the supplementary material.

4.1.3 Hyperparameters

Each learning method has its own regularizing hyperparameter. For β-VAE, we use three different values of $\beta \in \{0.2, 1.0, 4.0\}$. This choice is loosely based on the notion of normalized-β [13]. In addition, we force the KL-regularization only when the KL-divergence exceeds a fixed threshold $\tau = 50$ [41, 46]. For Annealed-VAE, we fix $\gamma = 1.0$ and use three different values of capacity, $C \in \{25.0, 50.0, 75.0\}$. For FactorVAE, we use the Annealed-VAE loss function with a fixed capacity ($C = 50$), and choose three different values for $\gamma \in \{1, 10, 50\}$.

4.1.4 Training Specifications

For each of the above methods, model, and hyperparameter combination, we train 3 models with different random seeds. To ensure consistency across training, all models are trained with a batch-size of 512 for 100 epochs. The ADAM optimizer [47] is used with a fixed learning rate of $1e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$. For β-VAE and Annealed-VAE, we use 10 warm-up epochs where $\beta = 0.0$. After warm-up, the regularization hyperparameter (β for

β-VAE and C for Annealed-VAE) is annealed exponentially from 0.0 to their target values over 100000 iterations. For FactorVAE, we stick to the original implementation and do not anneal any of the parameters in the loss function. The VAE optimizer is the same as mentioned earlier. The Factor-VAE discriminator is optimized using ADAM with a fixed learning rate of $1e-4$, $\beta_1 = 0.8$, $\beta_2 = 0.9$, and $\epsilon = 1e-8$. We found that utilizing the original hyperparameters [15] for this optimizer led to unstable training on dMelodies.

For comparison with dSprites, we present the results for all the three methods using a CNN-based VAE architecture. The set of hyperparameters and other training configurations were kept the same for the dSprites dataset, except for the FactorVAE where we use the originally proposed loss function and discriminator optimizer hyperparameters, as the model does not converge otherwise.

4.1.5 Disentanglement Metrics

The following objective metrics for measuring disentanglement are used: (a) *Mutual Information Gap (MIG)* [14], which measures the difference of mutual information between a given latent factor and the top two dimensions of the latent space which share maximum mutual information with the factor, (b) *Modularity* [48], which measures if each dimension of the latent space depends on only one latent factor, and (c) *Separated Attribute Predictability (SAP)* [16], which measures the difference in the prediction error of the two most predictive dimensions of the latent space for a given factor. For each metric, the mean across all latent factors is used for aggregation. For consistency, standard implementations of the different metrics are used [27].
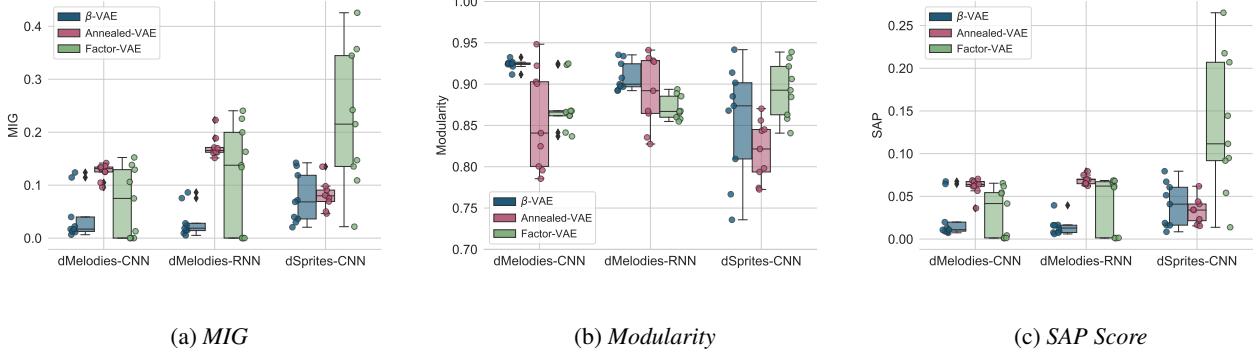
4.2 Experimental Results

4.2.1 Disentanglement
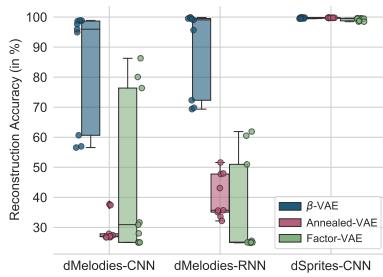
In this experiment, we present the comparative disentanglement performance of the different methods on dMelodies. The result for each method is aggregated across the different hyperparameters and random seeds. Figure 3 shows the results for all three disentanglement metrics. We group the trained models based on the architecture. The results for the dSprites dataset are also shown for comparison.

First, we compare the performance of different methods on dMelodies. Annealed-VAE shows better performance for MIG and SAP. These metrics indicate the ability of a method to ensure that each factor of variation is mapped to a single latent dimension. The performance in terms of Modularity is similar across the different methods. High Modularity indicates that each dimension of the latent space maps to only a single factor of variation. For dSprites, FactorVAE seems to be best method overall across metrics. However, the high variance in the results shows that choice of random seeds and hyperparameters is probably more important than the disentanglement method itself. This is in line with observations in previous studies [27].

Second, we observe no significant impact of model architecture on the disentanglement performance. For both the CNN and the hierarchical RNN-based VAE, the performance of all the different methods on dMelodies is

(a) *MIG*   (b) *Modularity*   (c) *SAP Score*

Figure 3: Overall disentanglement performance (higher is better) of different methods on the dMelodies and dSprites datasets. Individual points denote results for different hyperparameter and random seed combinations. Please refer to supplementary material Sec.2.1 for the best hyperparameter settings.

Figure 4: Overall reconstruction accuracies (higher is better) of the different methods on the dMelodies and dSprites datasets. Individual points denote results for different hyperparameter and random seed combinations.

comparable. This might be due to the relatively short sequence lengths used in dMelodies which do not fully utilize the capabilities of the hierarchical-RNN architecture (which has been shown to work well in learning long-term dependencies [41]). On the positive side, this indicates that the dMelodies dataset might be agnostic to the VAE-architecture.

Finally, we compare differences in the performance between the two datasets. In terms of MIG and SAP, the performance for dSprites is slightly better (especially for Factor-VAE), while for Modularity, performance across both datasets is comparable. However, once again, the differences are not significant. Looking at the disentanglement metrics alone, one might be tempted to conclude that the different methods are domain invariant. However, as the next experiments will show, there are significant differences.

4.2.2 Reconstruction Fidelity

From a generative modeling standpoint, it is important that along with better disentanglement performance we also retain good reconstruction fidelity. This is measured using the reconstruction accuracy shown in Figure 4. It is clear that all three methods fail to achieve a consistently good reconstruction accuracy on dMelodies. β-VAE gets an accuracy $\geq 90\%$ for some hyperparameter values (more on this in

Section 4.2.3). However, both Annealed-VAE and Factor-VAE struggle to cross a median-accuracy of 40% (which would be unusable from a generative modeling perspective). The performance of the hierarchical RNN-based VAE is slightly better than the CNN-based architecture. In comparison, for dSprites, all three methods are able to consistently achieve better reconstruction accuracies.

4.2.3 Sensitivity to Hyperparameters
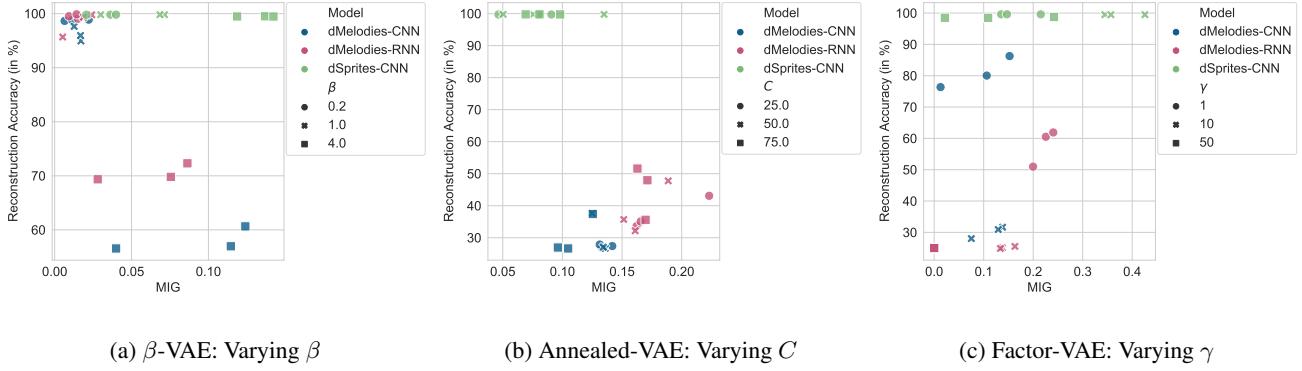
The previous experiments presented aggregated results over the different hyperparameter values for each method. Next, we take a closer look at the individual impact of those hyperparameters, i.e., the effect of changing the hyperparameters on the disentanglement performance (MIG) and the reconstruction accuracy. Figure 5 shows this in the form of scatter plots. The ideal models should lie on the top right corner of the plots (with high values of both reconstruction accuracy and MIG).

Models trained on dMelodies are very sensitive to hyperparameter adjustments. This is especially true for reconstruction accuracy. For instance, increasing β for the β-VAE model improves MIG but severely reduces reconstruction performance. For Annealed-VAE and Factor-VAE there is a wider spread in the scatter plots. For Annealed-VAE, having a high capacity C seems to marginally improve reconstruction (especially for the recurrent VAE). For FactorVAE, increasing γ leads to a drop in both disentanglement and reconstruction.
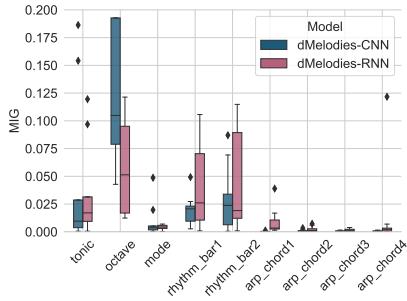
Contrast this with the scatter plots for dSprites. For all three methods, the hyperparameters seem to only significantly affect the disentanglement performance. For instance, increasing β and γ (for β-VAE and FactorVAE, respectively) result in clear improvement in MIG. More importantly, however, there is no adverse impact on the reconstruction accuracy.

4.2.4 Factor-wise Disentanglement

We also looked at how the individual factors of variation are disentangled. We consider the β-VAE model for this since it has the highest reconstruction accuracy. Figure 6

(a) β-VAE: Varying β

(b) Annealed-VAE: Varying C

(c) Factor-VAE: Varying γ

Figure 5: Effect of the hyperparameters on the different disentanglement methods. Overall, for improving disentanglement on dMelodies results in severe drop in reconstruction accuracy. The dSprites dataset does not suffer from this drawback.

Figure 6: Factor-wise MIG for the β-VAE method.

shows the factor-wise *MIG* for both the CNN and RNN-based models. Factors corresponding to octave and rhythm are disentangled better. This is consistent with some recent research on disentangling rhythm [6, 7]. In contrast, the factors corresponding to the arpeggiation direction perform the worst. This might be due to their binary type. Similar analysis for the dSprites dataset reveals better disentanglement for the scale and position based factors. Additional results are provided in the supplementary material.

5. DISCUSSION

As mentioned in Section 2, disentanglement techniques have been shown to be sensitive to the choice of hyper-parameters and random seeds [27]. The results obtained in our benchmarking experiments in the previous section using dMelodies seem to ascertain this even further. We find that methods which work well for image-based datasets do not extend directly to the music domain. When moving between domains, not only do we have to tune hyperparameters separately, but the model behavior may vary significantly when hyperparameters are changed. For instance, reconstruction fidelity is hardly effected by hyperparameter choice in the case of dSprites while for dMelodies it varies significantly. While sensitivity to hyperparameters is expected in neural networks, this is also one of the main reasons for evaluating methods on more than one dataset, preferably from multiple domains.

Some aspects of the dataset design, especially the na-

ture of the factors of variation, might have affected our experimental results. While the factors of variation in dSprites are continuous (except the shape attribute), those for dMelodies span different data-types (categorical, ordinal and binary). This might make other types of models (such as VQ-VAEs [49]) more suitable. Another consideration is that some factors of variation (such as the arpeggiation direction and rhythm) effect only a part of the data. However, the effect of this on the disentanglement performance needs further investigation since we get good performance for rhythm but poor performance for arpeggiation direction.

Unsupervised methods for disentanglement learning have their own limitations and some degree of supervision might actually be essential [27]. It is still unclear if it is possible to develop general domain-invariant disentanglement methods. Consequently, supervised and semi-supervised methods have been garnering more attention [10, 11, 19, 34]. The dMelodies dataset can also be used to explore such methods for music-based tasks. There has been some work recently in disentangling musical attributes such as rhythm and melodic contours which are considered important from an interactive music generation perspective [6,11,50]. Apart from the designed latent factors of variation, other low-level musical attributes such as rhythmic complexity and contours can also be computationally extracted using this dataset to meet task-specific requirements.

6. CONCLUSION

This paper addresses the need for more diverse modes of data for studying disentangled representation learning by introducing a new music dataset for the task. The *dMelodies* dataset comprises of more than 1 million data points of 2-bar melodies. The dataset is constructed based on fixed rules that maintain independence between different factors of variation, thus enabling researchers to use it for studying disentanglement learning. Benchmarking experiments conducted using popular disentanglement learning methods show that existing methods do not achieve performance comparable to those obtained on an analogous image-based dataset. This showcases the need for further research on domain-invariant algorithms for disentanglement learning.

7. ACKNOWLEDGMENT

8. REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, 2013.

[2] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017.

[3] W. Hsu, Y. Zhang, R. J. Weiss, Y. Chung, Y. Wang, Y. Wu, and J. R. Glass, "Disentangling correlated speaker and noise for speech synthesis via data augmentation and adversarial factorization," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019*, Brighton, United Kingdom, 2019.

[4] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum, "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding," in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, Montréal, Canada, 2018.

[5] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer," in *Proc. of 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[6] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[7] J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa, "Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 516–520.

[8] Y.-J. Luo, K. Agres, and D. Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[9] Y.-N. Hung, I.-T. Chiang, Y.-A. Chen, and Y.-H. Yang, "Musical composition style transfer via disentangled timbre representations," in *Proc. of 28th International Joint Conference on Artificial Intelligence (IJCAI)*, Macao, China, 2020.

[10] G. Hadjeres, F. Nielsen, and F. Pachet, "GLSR-VAE: Geodesic latent space regularization for variational autoencoder architectures," in *Proc. of IEEE Symposium Series on Computational Intelligence (SSCI)*, Hawaii, USA, 2017, pp. 1–7.

[11] A. Pati and A. Lerch, "Latent space regularization for explicit control of musical attributes," in *Proc. of ICML Workshop on Machine Learning for Music Discovery Workshop (ML4MD), Extended Abstract*, Long Beach, California, USA, 2019.

[12] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled multidimensional metric learning for music similarity," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 6–10.

[13] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," in *Proc. of 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[14] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud, "Isolating Sources of Disentanglement in Variational Autoencoders," in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, Montréal, Canada, 2018.

[15] H. Kim and A. Mnih, "Disentangling by Factorising," in *Proc. of 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweeden, 2018.

[16] A. Kumar, P. Sattigeri, and A. Balakrishnan, "Variational Inference of Disentangled Latent Concepts from Unlabeled Observations," in *Proc. of 5th International Conference of Learning Representations (ICLR)*, Toulon, France, 2017.

[17] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems 27 (NeurIPS)*, Montréal, Canada, 2014.

[18] N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, N. D. Goodman, P. Kohli, F. Wood, and P. H. Torr, "Learning disentangled representations with semi-supervised deep generative models," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017.

[19] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem, "Disentangling factors of variations using few labels," in *Proc. of 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

[20] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, "Fader Networks:Manipulating Images by Sliding Attributes," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017, pp. 5967–5976.

[21] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep Convolutional Inverse Graphics Network," in *Advances in Neural Information Processing Systems 28 (NeurIPS)*, Montréal, Canada, 2015, pp. 2539–2547.

[22] C. Donahue, Z. C. Lipton, A. Balsubramani, and J. McAuley, "Semantically Decomposing the Latent Spaces of Generative Adversarial Networks," in *Proc. of 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[23] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner, "dSprites: Disentanglement testing Sprites dataset," https://github.com/deepmind/dsprites-dataset, 2017, last accessed, 2nd April 2020.

[24] C. Burgess and K. Hyunjik, "3d-shapes Dataset," https://github.com/deepmind/3d-shapes, Feb. 2020, last accessed, 2nd April 2020.

[25] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic, "Seeing 3D Chairs: Exemplar Part-based 2D-3D Alignment using a Large Dataset of CAD Models," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, Ohio, USA, 2014, pp. 3762–3769.

[26] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep Visual Analogy-Making," in *Advances in Neural Information Processing Systems 28 (NeurIPS)*, Montréal, Canada, 2015, pp. 1252–1260.

[27] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations," in *Proc. of 36th International Conference on Machine Learning (ICML)*, Long Beach, California, USA, 2019.

[28] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *Proc. of 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

[29] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-VAE," in *NIPS Workshop on Learning Disentangled Representations*, Long Beach, California, USA, 2017.

[30] P. Rubenstein, B. Scholkopf, and I. Tolstikhin, "Learning Disentangled Representations with Wasserstein Auto-Encoders," in *Proc. of 6th International Conference on Learning Representations (ICLR), Workshop Track*, Vancouver, Canada, 2018.

[31] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 29 (NeurIPS)*, Barcelona, Spain, 2016, pp. 2172–2180.

[32] M. Connor and C. Rozell, "Representing closed transformation paths in encoded network latent space," in *Proc. of 34th AAAI Conference on Artificial Intelligence*, New York, USA, 2020.

[33] J. Engel, M. Hoffman, and A. Roberts, "Latent Constraints: Learning to Generate Conditionally from Unconditional Generative Models," in *Proc. of 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[34] D. Bouchacourt, R. Tomioka, and S. Nowozin, "Multi-Level Variational Autoencoder: Learning Disentangled Representations From Grouped Observations," in *Proc. of 32nd AAAI Conference on Artificial Intelligence*, New Orleans, USA, 2018.

[35] M. W. Gondal, M. Wüthrich, Đ. Miladinović, F. Locatello, M. Breidt, V. Volchkov, J. Akpo, O. Bachem, B. Schölkopf, and S. Bauer, "On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019, pp. 15 740–15 751.

[36] M. Bretan and L. Heck, "Learning semantic similarity in music via self-supervision," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[37] S. Gururani, A. Lerch, and M. Bretan, "A comparison of music input domains for self-supervised feature learning," in *Proc. of ICML Workshop on Machine Learning for Music Discovery Workshop (ML4MD), Extended Abstract*, Long Beach, California, USA, 2019.

[38] K. Choi, G. Fazekas, M. B. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proc. of 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 141–149.

[39] S. Lattner, M. Dörfler, and A. Arzt, "Learning complex basis functions for invariant representations of audio," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[40] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep Learning Face Attributes in the Wild," in *Proc. of IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 3730–3738.

[41] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music," in *Proc. of 35th*

International Conference on Machine Learning (ICML), Stockholm, Sweeden, 2018.

[42] G. Toussaint, "A Mathematical Analysis of African, Brazilian and Cuban Clave Rhythms," in *Proc. of BRIDGES: Mathematical Connections in Art, Music and Science*, 2002, pp. 157–168.

[43] M. S. Cuthbert and C. Ariza, "music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data," in *Proc. of 11th International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010.

[44] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: A steerable model for Bach chorales generation," in *Proc. of 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017, pp. 1362–1371.

[45] A. Pati, A. Lerch, and G. Hadjeres, "Learning to Traverse Latent Spaces for Musical Score Inpainting," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[46] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved Variational Inference with Inverse Autoregressive Flow," in *Advances in Neural Information Processing Systems 29 (NeurIPS)*, Barcelona, Spain, 2016, pp. 4743–4751.

[47] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. of 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

[48] K. Ridgeway and M. C. Mozer, "Learning Deep Disentangled Embeddings With the F-Statistic Loss," in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, Montréal, Canada, 2018, pp. 185–194.

[49] A. van den Oord, O. Vinyals, and k. kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017, pp. 6306–6315.

[50] T. Akama, "Controlling Symbolic Music Generation Based On Concept Learning From Domain Knowledge," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

MODELING MUSIC AND CODE KNOWLEDGE TO SUPPORT A CO-CREATIVE AI AGENT FOR EDUCATION

Jason Smith[1]   **Erin J.K. Truesdell**[2]   **Jason Freeman**[1]
Brian Magerko[2]   **Kristy Elizabeth Boyer**[3]   **Tom McKlin**[4]

[1] Center for Music Technology, Georgia Institute of Technology, Atlanta, GA, USA
[2] Expressive Machinery Lab, Georgia Institute of Technology, Atlanta, GA, USA
[3] Computer & Information Science & Engineering, University of Florida, Gainesville, FL, USA
[4] The Findings Group, Decatur, Georgia, USA

`{jsmith775, erinjktruesdell}@gatech.edu`

ABSTRACT

EarSketch is an online environment for learning introductory computing concepts through code-driven, sample-based music production. This paper details the design and implementation of a module to perform code and music analyses on projects on the EarSketch platform. This analysis module combines inputs in the form of symbolic metadata, audio feature analysis, and user code to produce comprehensive models of user projects. The module performs a detailed analysis of the abstract syntax tree of a user's code to model use of computational concepts. It uses music information retrieval (MIR) and symbolic metadata to analyze users' musical design choices. These analyses produce a model containing users' coding and musical decisions, as well as qualities of the algorithmic music created by those decisions. The models produced by this module will support future development of CAI, a Co-creative Artificial Intelligence. CAI is designed to collaborate with learners and promote increased competency and engagement with topics in the EarSketch curriculum. Our module combines code analysis and MIR to further the educational goals of CAI and EarSketch and to explore the application of multimodal analysis tools to education.

1. INTRODUCTION

Digital music creation environments often use a combination of raw audio data, symbolic information, code, and metadata to represent user-generated music. For example, a digital audio workstation might represent a song through a combination of audio files, genre and artist labels, MIDI events, and a data source indicating the placement of audio and MIDI segments on a multi-track timeline, along with device and mixer settings and automation.

In the context of music information retrieval (MIR), access to these multiple types of music representation can vastly simplify common retrieval tasks that are too complex to perform on audio alone. Examples of this include music preference modeling using a combined model of audio and metadata [1] and genre recognition using feature analysis alongside symbolic representation of the same audio through statistical descriptors of melody, harmony, and rhythm [2, 3]. A multimodal analytical approach can therefore help develop powerful MIR-driven applications within these digital music creation environments, such as creativity-support tools that generate ideas and feedback for users as they create music with the software.

Digital environments that include algorithmic composition elements offer yet another mode of analysis: the code that generates the music. A creativity support tool that incorporates code analysis in addition to these other modalities can potentially generate recommendations not only about the music users create but also about the code that they write and and the conceptual overlap thereof between the code and music.
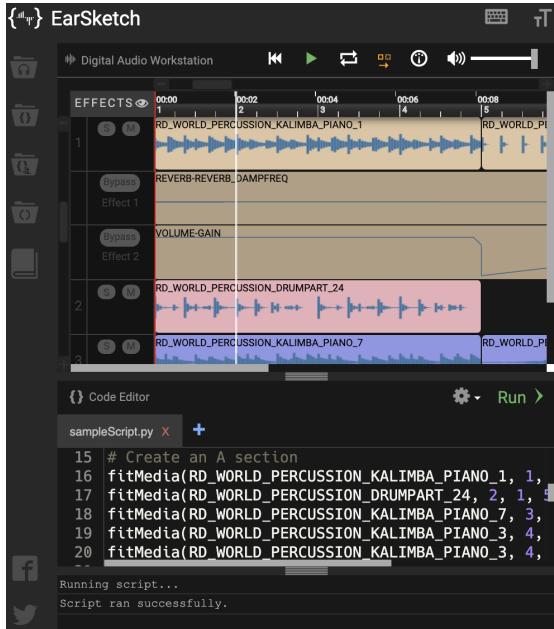
This paper describes an analysis system we have created that uses audio features, audio metadata, symbolic multi-track music data, and code from user-created projects to understand the structure of algorithmic music and to model users' knowledge of coding and musical techniques. We have created the system in the context of EarSketch [4], an expressive and collaborative learning environment for high school students that is used by roughly 120,000 students per year. In EarSketch, users write Python or JavaScript code to algorithmically create multi-track compositions remixing a library of audio loops. Figure 1 depicts the EarSketch user interface. EarSketch aims to increase student engagement in computing across diverse student populations, promoting student perceptions of authenticity through its use of a professionally-produced audio loop library and design influences from industry-standard digital audio workstations [5].

The EarSketch curriculum contains musical concepts integral to algorithmic music and relevant to coding, including repetition, form, and effect usage. The EarSketch sound library contains over 3,500 sounds from professional

Figure 1. A Screenshot of the EarSketch web-based application, containing the Digital Audio Workstation (top), and Code Editor (bottom).

artists, split into folders and labeled with artist, genre, and instrument type. Users write code with Python and JavaScript APIs to manipulate sounds and create music.

EarSketch is designed for students without formal music training. It does not focus on traditional music theory concepts such as music notation, melody, and harmony. Instead, it uses audio loops, effects, automation, and step-sequenced rhythms to facilitate music production through code without prerequisite knowledge. In addition to modeling understanding of code, our module includes music analysis to compare a user project's traits to the information conveyed by the EarSketch curriculum.

Our analysis system was designed to support a new creativity-support tool within EarSketch called CAI (Co-creative Artificial Intelligence). CAI, which is still in the early stages of design, will assist EarSketch users in learning and practicing pedagogical concepts in both computing and music. CAI will use our multimodal analysis system in combination with user interaction to suggest additions and changes to student music and code, scaffolding student learning and producing co-creative musical output.

In the following sections, we position this work in the context of recent music information retrieval research, describe each component of the analytical system in turn, explain how we coalesce this data into a user model to inform CAI, and outline future areas of work in the design and implementation of the larger CAI system that will leverage this analysis tool.

The development of this analysis module for CAI marks a unique application of MIR and code analysis to educational systems. Further development of the CAI system will continue to illuminate the role of mixed-input models for the goal of supporting learners in expressive computing environments.

2. RELATED WORK

Multimodal music information retrieval relies on representations of information to perform analysis tasks beyond what is possible using raw audio signals [6]. Digital production environments and notation software use symbolic information to represent structural and artistic elements of user-generated music. Due to its prominent use in music software, symbolic music is a significant presence in the domain of music information retrieval, with well-known tools such as Music21 [7] performing operations on symbolic data. MIDI has also been used to train style transfer models [8] and detect meter in live performance [9].

MIDI note messages are direct symbolic representations of musical notes. Other combinations of symbolic and audio analysis performed by labeling audio with statistical descriptors have been used in applications of genre recommendation [2] and pattern-based style identification [3]. Another application [10] uses symbolic music representation to vectorize multiple aspects of music for the purpose of performing song recommendations. The LFM-1b dataset [11] contains recorded listening events tagged with metadata, and has been used in conjunction with audio feature analysis to model user music preference [1]. Our system differs from these applications by using audio and symbolic data to model a user's proficiency with the techniques required to produce musical output, using EarSketch curriculum topics as evaluation criteria.

Our system extends traditional music information retrieval applications in its use of source code analysis as an input. Code analysis tools often rely on the analysis of abstract syntax trees (ASTs) generated from student code to propose edits. Contemporary AST-based systems [12, 13] rely on step calculations between a student's current code-state and a previously seen code-state or family of code-states that fulfills a set of conditions. These systems use AST analysis to simplify a wide range of projects to their most basic structure and provide suggestions in almost all situations. This application has largely focused on moving students from an "incorrect" answer to a "correct" one. In comparison, the application of AST analysis to creative problems without a defined "solution" is relatively unexplored. Our code analysis module brings the advantages of AST analysis into the expressive domain, reorienting the process towards open-ended learning and development.

Examples of software analysis applied to music information retrieval include a tool that uses Source Code Analysis and Manipulation (SCAM) to represent the structure of algorithmic music compositions [14, 15]. Music code analysis has also been performed in the context of live coding, a performance format in which programmers write algorithmic music in real time. A survey of live coding practitioners on creativity [16] discusses the link between analyzing live coding techniques and the development of a creative software agent. EarSketch has previously been examined for its ability to function as an educational live coding platform [17], and a fully developed CAI system will be able to collaborate with learners in a live coding environment.

3. CODE ANALYSIS

3.1 Code Knowledge Modeling

The code portion of the analysis module combines AST analysis and a computer science learning taxonomy to build a model of user knowledge that will be used by CAI to generate level-appropriate suggestions for EarSketch users. Previous works on computer science assessment [18–20] have adapted general learning taxonomies for computing topics; similarly, we defined a series of knowledge levels for 15 computational concepts from the EarSketch curriculum across 4 knowledge categories (see Table 1) based upon a flattened version of Bloom's Taxonomy [21]. For each concept, we define knowledge levels specific to its usage contexts. Table 2 outlines knowledge levels defined for the "String" and "User-Defined Function" concepts. Level 1 refers to usage of the concept or construct (e.g., a user includes a string in their script). Level 2 is defined as "original" usage of the concept (usage not copied directly from sample code). Our originality measures are described in greater detail in Section 3.2. Subsequent levels focus on increased complexity of use: for example, a user's script could reach level 2 of the "String" concept by merely including an original string in their script, but level 3 requires that the string to be put to use (such as using it as a function argument).

Category	Concepts
Value Types	String, Integer, Float, Boolean
Data Storage	List, Variable
Operations	String Operation, List Operation, Comparison, Boolean Logic, Mathematical Operator
Procedure	For Loop, Conditional Statement, User-defined Function, Console Input

Table 1. Concepts in the analysis module taxonomy.

Level	String	User-Defined Function
0	Does Not Use	Does Not Use
1	Uses	Uses
2	Uses Originally	Uses Originally
3	Uses Originally for Purpose	Uses and Calls Originally
4	Uses Originally and Indexes or Iterates for Purpose	Uses and Calls Originally with Return OR Arguments
5	N/A	Uses and Calls Originally with Return AND Arguments

Table 2. Knowledge levels for two concepts: "String" and "User-Defined Function."

3.2 Code Complexity Analysis

The code analysis module generates knowledge models for student scripts, producing concise information on understanding of each topic. Code knowledge models are generated by analyzing the abstract syntax tree of a user's code, which allows for fast and non-intrusive analysis. The code analysis module searches each AST node for constructs that indicate the user's knowledge level for every concept in our taxonomy.

Prior to analysis, the module performs a series of four passes over the script's AST to gather supporting data. The first pass tests student code for similarity to sample code. We use Andrei Mackenzie's Levenshtein function [1] to calculate the edit distance between each line of a user's code and all lines of EarSketch sample code; if the edit distance is below a manually-defined similarity threshold, the line is marked as "not original." Three subsequent passes gather information about user-defined functions and variable assignments and values. Once this information is collected, each individual node in the hierarchy is checked against discrete rules developed in tandem with the knowledge modeling level table.

Below is an example of a student script progressing through levels of the "user-defined function" concept. In the first code snippet, the user creates and then calls a function to make a section of music, calling `fitMedia()` to place piano and drumpad samples between measures 1 and 16. This corresponds to level 3 of the "user-defined function" item: "Uses and Calls Originally."

```
def sectionA():
  fitMedia(RD_RNB_PIANO_1,1,1,16)
  fitMedia(Y25_DRUMPAD_1,2,1,16)

sectionA()
```

In the second snippet, the function has been modified to take arguments: the function now places the samples between passed "start" and "end" measures. This corresponds to level 4 of the "user-defined function" item: "Uses and Calls Originally with Return OR Arguments."

```
def sectionA(start, end):
  fitMedia(RD_RNB_PIANO_1,1,start,end)
  fitMedia(Y25_DRUMPAD_1,2,start,end)

sectionA(1,16)
```

These node analyses populate an output object with values mapped to knowledge levels, visualized in Table 3. These analyses will be used to support future development of CAI, providing guidance to the system about appropriate code structure for programming and music suggestions when collaborating with a student.

---

[1] https://gist.github.com/andrei-m/982927

Concept	0	1	2	3	4	5
String	■	■	■	■		
Integer	■	■	■	■		
Float	■	■	■	■		
Boolean	■	■	■	■		
List	■	■	■	■	■	
Variable	■	■	■	■	■	
String Op	■					
List Op	■					
Comparison	■					
Boolean Logic	■					
Math Op	■	■	■	■		
For Loop	■	■	■			
Conditional		■				
User Function	■	■	■	■		
Console Input	■					

Table 3. Visualization of code analysis output for a sample project.

4. SYMBOLIC MUSIC ANALYSIS

4.1 EarSketch Music Representation

The music component of the analysis module uses a simplified symbolic representation generated by EarSketch to apply sounds and effects to the Digital Audio Workstation view and generate audio playback. When a learner runs a script, the parsed abstract syntax tree of the code is represented internally as a dictionary of tracks containing sound and effect usage for each measure in a piece of music.

```
#Setup
from earsketch import *
init()
setTempo(120)

# Create an A section
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_1,1,1,5) # main
fitMedia(RD_WORLD_PERCUSSION_DRUMPART_24,2,1,5) # drums
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_7,3,1,5) # bassline
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3,4,1,2) # backing
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3,4,3,4) # backing

# Create a 4 measure B section between measures 5 and 9
fitMedia(RD_WORLD_PERCUSSION_DRUMPART_3,1,5,9) # sparse drums
fitMedia(RD_WORLD_PERCUSSION_SEEDSRATTLE_1,3,5,9) # rattling
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3,4,5,6) # backing

# Then back to section A at measure 9
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_1,1,9,13) # main
fitMedia(RD_WORLD_PERCUSSION_DRUMPART_24,2,9,13) # drums
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_7,3,9,13) # bassline
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3,4,9,10)  # backing
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3,4,11,12) # backing

# Effects
setEffect(1,REVERB)
setEffect(1,VOLUME,GAIN,0,4.9,-12,5)
setEffect(1,VOLUME,GAIN,-60,5,0,9)

#Finish
finish()
```

Figure 2. Code in the Code Editor of an example project created in EarSketch.

The track listing for the example project (Figure 2) is an array of tracks, each containing a series of *clip* and *effect* objects reflecting those used in the code. Each clip object contains the name of the sound file used (*RD_WORLD_PERCUSSION_KALIMBA_PIANO_1*), as well as its start measure (*1*), and end measure (*5*). Each effect object contains each instance of a specific effect

(*VOLUME-GAIN*), its starting value and measure (*-60, 5*) and ending value/measure (*0, 9*).

The music analysis tool begins by converting this track representation to a timeline representation, in order to ascertain temporal patterns in the song. The timeline is created by converting the track dictionary to a dictionary of measures, as measure numbers are used in the EarSketch API for audio and effect sequencing. Figure 3 shows this timeline for four measures of the example, which has sounds used throughout the song such as *RD_WORLD_PERCUSSION_KALIMBA_PIANO_1* and sounds used in a single section such as *RD_WORLD_PERCUSSION_SEEDSRATTLE_1*, as well as the value of the gain adjustment at each measure.

```
0: ["RD_WORLD_PERCUSSION_KALIMBA_PIANO_1",
    "VOLUME GAIN 1",
    "REVERB REVERB_DAMPFREQ 8000",
    "RD_WORLD_PERCUSSION_DRUMPART_24",
    "RD_WORLD_PERCUSSION_KALIMBA_PIANO_7",
    "RD_WORLD_PERCUSSION_KALIMBA_PIANO_3"]

4: ["RD_WORLD_PERCUSSION_DRUMPART_3",
    "VOLUME GAIN 0",
    "REVERB REVERB_DAMPFREQ 8000",
    "RD_WORLD_PERCUSSION_SEEDSRATTLE_1",
    "RD_WORLD_PERCUSSION_KALIMBA_PIANO_3"]

6: ["RD_WORLD_PERCUSSION_DRUMPART_3",
    "VOLUME GAIN 0.4995",
    "REVERB REVERB_DAMPFREQ 8000",
    "RD_WORLD_PERCUSSION_SEEDSRATTLE_1"]

8: ["RD_WORLD_PERCUSSION_KALIMBA_PIANO_1",
    "VOLUME GAIN 0.999",
    "REVERB REVERB_DAMPFREQ 8000",
    "RD_WORLD_PERCUSSION_DRUMPART_24",
    "RD_WORLD_PERCUSSION_KALIMBA_PIANO_7",
    "RD_WORLD_PERCUSSION_KALIMBA_PIANO_3"]
```

Figure 3. Timeline representation for measures 1, 5, 7, and 9 of the sample EarSketch project (see Figure 2).

Recognition of patterns in sound and effect usage over time can be used in determining form as described in section 5. It allows CAI to propose changes to sound choices, effects, and parameters at specific points in time, or to suggest optimized code structure to realize those patterns. For example, if a user places the same sound at regular intervals and the code analysis does not observe any loops containing the variables related to that sound in their code, CAI can suggest the use of a loop.

Audio usage requires students only to select sounds from the library; conversely, effects can be manipulated through envelopes and are introduced in the EarSketch curriculum in multiple levels of complexity. The analysis module conducts a hierarchical score analysis for effect usage, while simply recording the selection of audio at each measure to inform its audio analysis tools.

4.2 Effect Usage Scores

Effect envelopes in EarSketch are applied using a start value, end value, start time, and end time. The following code example (found in Figure 2) shows a use of the volume effect on track 1 having its gain parameter changed from -60 dB to 0 dB between measures 5 and 9:

```
setEffect(1, VOLUME, GAIN, -60, 5, 0, 9)
```

The internal track representation stores a single recording of these four values for each effect parameter. Consequently, we use linear interpolation to store the value of each effect parameter at each measure when converting effect envelopes to the timeline.

This reorganization allows the analysis module to classify the level of usage the user demonstrates for each audio effect in the library, such as gain, filters, delay, and reverb. The EarSketch curriculum teaches basic effect usage, followed by use of effect parameters, and then the use of envelopes to change parameter values over time. Similarly, the levels of effect usage are *0: Does not use*, *1: Uses standard parameters*, *2: Uses non-standard parameters*, and *3: Changes parameters over time*. EarSketch's goal is to teach coding technique through music production, so students are encouraged to use increasingly complex effect parameter manipulation through increasingly complex algorithmic structure. These scores, along with the scores from the code analysis described in section 3, form a composite score to represent user knowledge.

In addition to the timeline representation and modeling of effects usage, the analysis module records the length of the piece and whether or not the user sets a tempo different than the default 120 bpm. The combination of these music analysis outputs can be used by CAI to characterize a user project and to identify which sounds or effects to include and where to include them, or areas for further improvement in a script.

5. AUDIO FEATURE ANALYSIS

In addition to storing user knowledge modeling, the analysis module is designed to analyze the sound usage, form, and genre of a user project. These are not represented in ordered levels to model conceptual understanding like effects are, but are recorded to display a detailed overview of the piece of music and to better target suggestions made by the CAI system. For example, if a user is primarily choosing sounds of a certain genre in a specific musical section, CAI might suggest other sounds in that genre for that section, while suggesting contrasting sounds for a different section.

5.1 Audio Recommendations

The analysis module includes an expansion of the existing EarSketch recommendation system [22], which uses the code of an active user project to make real-time recommendations. The recommendation system uses audio features Short-time Fourier Transform (STFT) and Mel-Frequency Cepstral Coefficients (MFCC) [23] to represent temporal and non-temporal information, respectively, of each EarSketch library sound whose name is found in the code [24]. Feature vectors for each sound are generated in offline scripts, and feature distances and relative co-usage by EarSketch users between each pair of sound vectors are uploaded to the EarSketch server. This avoids the need to calculate the features, feature distances, and co-usage data in real time. The advantage of this form of analysis in a digital production system such as EarSketch is that audio
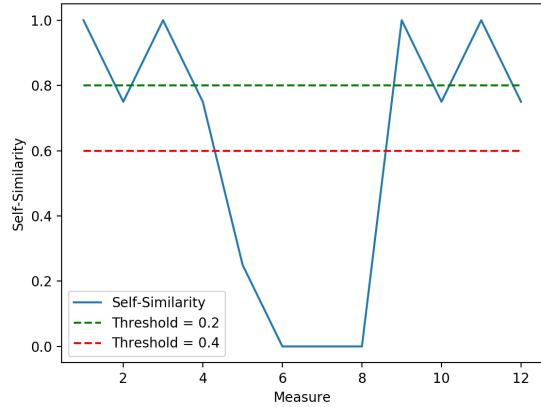
feature distances can be measured against metadata tags for artist, genre, and instrument type. The system is able to provide recommendations that either specifically conform to the user project's genre and instrumentation or allow the user to consciously explore other creative options.

The original EarSketch recommendation system [22], which used the source code of a script to generate recommendations for an entire user project. The analysis module expands this system by presenting a series of sound recommendations per measure. These recommendations can be used to increase or decrease musical contrast at specific points throughout a composition, or use stored feature distances to ascertain the best measure to include a sound.

5.2 Form and Structure Analysis

The use of symbolic music representation greatly simplifies the task of determining sections in a song through changes in instrumentation and effects. Sound and effect usage are represented for each measure, and instrument type and genre are included in metadata tags.

Self-similarity can be used in retrieving [25] and visualizing [26] musical structure. An array of self-similarity values for instrumentation at every measure in the timeline view allows the analysis module to infer musical structure from sounds used by marking the first measure in a sequence to pass above or below a threshold of similarity.

Figure 4. Self-similarity of instrumentation for the example EarSketch project (see Figure 2), indicating an A-B-A form. Dotted lines represent similarity thresholds of 0.4 and 0.2 that, when crossed, mark section beginnings.

The threshold of similarity can also be hierarchically defined to determine different granularities of sections and subsections, from a whole piece to the measure level. The example in Figure 4 shows ternary form, a musical structure represented in the EarSketch curriculum. If the similarity threshold is above 0.25, then the analysis module will cross the threshold three times and predict section beginnings at measures 1, 5, and 9. If the threshold is lower, then it will be crossed five times and predict section beginnings at measures 3 and 11 as well. When analyzing a script, the analysis module generates a list of section pre-

dictions for thresholds of every ten percent between 0.9 and 0.1. Any list of a unique length (or with unique values) is recorded. In the example of Figure 4, lists would be generated to form a section prediction of *[1, 5, 9]* with a subsection prediction of *[1, 3, 5, 9, 11]*.

5.3 Genre Analysis

Each sound in the EarSketch sound library is labeled with a tag for one of 21 genres such as Rock, Funk, Hip Hop, and EDM. These tags were manually added to the sounds by the artists and EarSketch developers who uploaded them.

Our music analyzer combines these genre tags with audio features (as described in section 5.1) to predict a genre for each measure in a user project. It applies the k-means genre clustering algorithm [27] to each measure, approximating the closest distance in audio fingerprints between the average for the measure's sounds and the average for a genre found in the EarSketch library. The genre tag for each sound used in a measure is also added to increase the confidence value for that genre. This combination affords our system the knowledge of the original genre label for each sound as presented to the user, but allows it to recognize a user who has creatively used sounds in genre applications separate from their label. This genre analysis is used to determine the likelihood of a user project belonging to a specific genre at the script, section, subsection, or measure level - depending on the granularity of the self-similarity measurement used to determine form.

This genre analysis can be used for CAI to target its audio suggestions to a song's identified genre. If a user is writing a song mainly using sounds tagged with a single genre, such as in the example code (Figure 2), CAI can suggest sounds in that genre or present options for different genres to generate contrast.

6. PRELIMINARY RESULTS, USAGE, AND INSIGHTS

Two informal testing methods have been used to aid in the iterative development of the code analysis module: a review of output correctness, and an ongoing large-scale process to identify bugs in the module.

To evaluate the ability of the module to produce a correct analysis of a script, we ran it on a series of 103 student- and researcher-generated scripts (77 Python, 26 JavaScript). Student scripts were selected to ensure the module could accurately respond to programming choices made by EarSketch users; researcher scripts were selected or written to test the module's ability to respond to complex scenarios. For each test script, the analysis module's output was judged against a researcher-generated score. Any discrepancies between the two indicated a missing component in the module. Modifications were made until the analysis module could correctly score the script. This testing identified a number of situations not originally accounted for in the module.

To efficiently locate bugs in the code analysis script, we have included a version of the Code Analysis module on EarSketch that runs each time any user runs a script. Upon the encounter of an exception while analyzing a project, the module sends a report including the exception and stack trace to an analytics engine. These reports are frequently reviewed to identify bugs and improve the ability of the code analysis module to run without error.

The music analysis tool has been integrated into the EarSketch autograder, an automatic grading tool used in evaluating the complexity of code submissions in previous course projects [4]. This autograder is a separate web page that, given any number of EarSketch script sharing IDs, generates a list of code topics, their categorized scores, the list of section markings, measure-by-measure audio recommendations, and genre predictions. This analysis can be performed on large samples of scripts, such as on multiple instances of a script to track improvement over its history, and is used in ongoing evaluation of the analysis module's ability to model form and genre. As the analysis module undergoes future iterative development, observations of emergent patterns in output will further aid the development and evaluation of the CAI recommendation system as well as its component parts.

Though the CAI system that will make use of this analysis module has yet to be completed, the development of this module has generated insights about the applications of MIR work in tandem with other disciplines, particularly in the context of education. Its combined analysis of symbolic music, audio features, and code knowledge highlights the ability of multimodal analysis to provide a comprehensive body of information to support [systems that do two things]. The use of this combination in an educational context will allow CAI to assist learners in simultaneous musical and programming development and further the goals of educational platforms that intersect domains. Additionally, the development of this module for a co-creative AI indicates potential for additional knowledge to be developed as the module is used to inform the agent's outputs.

7. FUTURE WORK

This analysis module has been implemented in the production version of EarSketch. We are still in the early stages of designing CAI, the agent that will leverage this analysis data to interact with students through dialogue and generate suggestions for their music and code. Over the past year, we have performed studies to better understand how students interact through chat with each other in student-to-student chat experiments. We have also conducted chat experiments between students and researchers posing as AI agents to simulate the intelligence that CAI will eventually provide. We are using the findings from these studies to inform the design of CAI. The initial version of CAI will exist as a chat-style interface within EarSketch, where students will choose prompts from a menu-driven system and receive natural language responses based on their interaction with the system and its analysis of their code and music as described in this paper. Through the addition of CAI to EarSketch, we hope to further increase student engagement and creativity with the platform.

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] E. Zangerle and M. Pichl, "The many faces of users: Modeling musical preference." in *Proc. of ISMIR*, 2018, pp. 709–716.

[2] T. Lidy, A. Rauber, A. Pertusa, and J. M. I. Quereda, "Improving genre classification by combination of audio and symbolic descriptors using a transcription systems." in *Proc. of ISMIR*, 2007, pp. 61–66.

[3] P. J. P. De León and J. M. Inesta, "Pattern recognition approach for music style identification using shallow statistical descriptors," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 2, pp. 248–257, 2007.

[4] B. Magerko, J. Freeman, T. Mcklin, M. Reilly, E. Livingston, S. Mccoid, and A. Crews-Brown, "Earsketch: A STEAM-based approach for underrepresented populations in high school computer science education," *ACM Transactions on Computing Education (TOCE)*, vol. 16, no. 4, pp. 1–25, 2016.

[5] McKlin, Tom, Magerko, Brian, Lee, Taneisha, Wanzer, Dana, Edwards, Doug, and Freeman, Jason, "Authenticity and personal creativity: How EarSketch affects student persistence," in *Proc. of the 49th ACM Technical Symp. on Computer Science Education*, 2018, pp. 987–992.

[6] C. C. Liem, M. Müller, D. Eck, G. Tzanetakis, and A. Hanjalic, "The need for music information retrieval with user-centered and multimodal strategies," in *Proc. of the 1st International ACM Workshop on Music information Retrieval With User-Centered and Multimodal Strategies*, 2011, pp. 1–6.

[7] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," in *Proc. of ISMIR*, 2010, pp. 637–642.

[8] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer," in *Proc. of ISMIR*, 2018, pp. 747–754.

[9] A. McLeod and M. Steedman, "Meter detection and alignment of midi performance." in *Proc. of ISMIR*, 2018, pp. 113–119.

[10] K. Watanabe and M. Goto, "Query-by-blending: a music exploration system blending latent vector representations of lyric word, song audio, and artist," in *Proc. of ISMIR*, 2019, pp. 144–151.

[11] M. Schedl, "The LFM-1b dataset for music retrieval and recommendation," in *Proc. of the 2016 ACM International Conference on Multimedia Retrieval*, 2016, pp. 103–110.

[12] T. W. Price, Y. Dong, and D. Lipovac, "iSnap: towards intelligent tutoring in novice programming environments," in *Proc. of the 2017 ACM SIGCSE Technical Symp. on Computer Science Education*, 2017, pp. 483–488.

[13] K. Rivers, E. Harpstead, and K. R. Koedinger, "Learning curve analysis for programming: Which concepts do students struggle with?" in *Proc. of the 2016 ACM Conference on International Computing Education Research*, 2016, pp. 143–151.

[14] N. Gold, "Knitting music and programming: Reflections on the frontiers of source code analysis," in *2011 IEEE 11th International Working Conference on Source Code Analysis and Manipulation*, 2011, pp. 10–14.

[15] M. Harman, M. Munro, L. Hu, and X. Zhang, "Source code analysis and manipulation," *Information and Software Technology*, vol. 44, no. 13, pp. 717–720, 2002.

[16] A. McLean and G. A. Wiggins, "Live coding towards computational creativity." in *International Conference on Computational Creativity*, 2010, pp. 175–179.

[17] J. Freeman and B. Magerko, "Iterative composition, coding and pedagogy: A case study in live coding with earsketch," *Journal of Music, Technology & Education*, vol. 9, no. 1, pp. 57–74, 2016.

[18] U. Fuller, C. G. Johnson, T. Ahoniemi, D. Cukierman, I. Hernán-Losada, J. Jackova, E. Lahtinen, T. L. Lewis, D. M. Thompson, C. Riedesel *et al.*, "Developing a computer science-specific learning taxonomy," *ACM SIGCSE Bulletin*, vol. 39, no. 4, pp. 152–170, 2007.

[19] C. W. Starr, B. Manaris, and R. H. Stalvey, "Bloom's taxonomy revisited: specifying assessable learning objectives in computer science," *ACM SIGCSE Bulletin*, vol. 40, no. 1, pp. 261–265, 2008.

[20] E. Thompson, A. Luxton-Reilly, J. L. Whalley, M. Hu, and P. Robbins, "Bloom's taxonomy for cs assessment," in *Proc. of the tenth conference on Australasian computing education-Volume 78*, 2008, pp. 155–161.

[21] D. R. Krathwohl and L. W. Anderson, *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives.* Longman, 2009.

[22] J. Smith, M. Jacob, J. Freeman, B. Magerko, and T. Mcklin, "Combining collaborative and content filtering in a recommendation system for a web-based daw," in *Proc. of the International Web Audio Conference*, 2019.

[23] A. Lerch, *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*, 1st ed. Wiley-IEEE Press, 2012.

[24] J. Smith, D. Weeks, M. Jacob, J. Freeman, and B. Magerko, "Towards a hybrid recommendation system for a sound library," in *ACM IUI Workshops*, 9.

[25] B. Martin, M. Robine, and P. Hanna, "Musical structure retrieval by aligning self-similarity matrices." in *Proc. of ISMIR*, 2009, pp. 483–488.

[26] J. Foote, "Visualizing music and audio using self-similarity," in *Proc. of the seventh ACM International Conference on Multimedia (Part 1)*, 1999, pp. 77–80.

[27] D. Turnbull and C. Elkan, "Fast recognition of musical genres using RBF networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 580–584, 2005.

THE JAZZ TRANSFORMER ON THE FRONT LINE: EXPLORING THE SHORTCOMINGS OF AI-COMPOSED MUSIC THROUGH QUANTITATIVE MEASURES

Shih-Lun Wu[1,2] **and Yi-Hsuan Yang**[2,3]

[1] National Taiwan University, [2] Taiwan AI Labs, [3] Academia Sinica, Taiwan
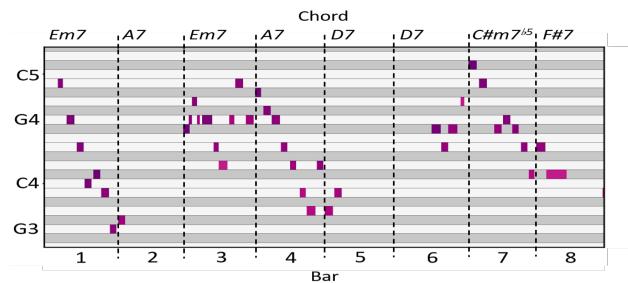
`b06902080@csie.ntu.edu.tw`, `yang@citi.sinica.edu.tw`

ABSTRACT

This paper presents the Jazz Transformer, a generative model that utilizes a neural sequence model called the Transformer-XL for modeling lead sheets of Jazz music. Moreover, the model endeavors to incorporate structural events present in the Weimar Jazz Database (WJazzD) for inducing structures in the generated music. While we are able to reduce the training loss to a low value, our listening test suggests however a clear gap between the ratings of the generated and real compositions. We therefore go one step further and conduct a series of computational analysis of the generated compositions from different perspectives. This includes analyzing the statistics of the pitch class, grooving, and chord progression, assessing the structureness of the music with the help of the fitness scape plot, and evaluating the model's understanding of Jazz music through a MIREX-like continuation prediction task. Our work presents in an analytical manner why machine-generated music to date still falls short of the artwork of humanity, and sets some goals for future work on automatic composition to further pursue.

1. INTRODUCTION

Music is a heart-touching form of art that strikes a chord with people's emotions, joyful or sorrowful; intense or relieved, through the twists and turns of notes. Despite its ubiquity in our everyday lives, the composition and arrangement of music often requires substantial human effort. This is a major reason why automatic music composition is such a fascinating field of study. Over the years, researchers have sought strenuously ways for machines to generate well-formed music; such methods include meticulously designed non deep learning-based algorithms like the Markov chains [6] and formal grammars [19]; and, a proliferation of deep learning-based solutions in the past decade [8]. In this work, we exclusively study the extension and evaluation of Transformer-based models [43] for

Figure 1. The first 8 bars of a piece (filename `sample_B01.mp3` in Google Drive) composed by the Jazz Transformer, exhibiting clear rests between phrases.

its claimed successes in natural language processing and music generation in recent years [12, 13, 20, 38].

The dataset chosen for our work is the Weimar Jazz Database (WJazzD) [2, 37]. As opposed to the commonly used piano MIDIs in recent works [20, 21], the choice of this dataset represents a fresh endeavor to train the Transformer on Jazz music, and grants us the unique opportunity to integrate structure-related events, precisely annotated in the WJazzD, to the model. However, such an attempt involves no short of technical challenges, including the quantization of the numerous short notes in Jazz improvisations; and, dealing with the complex chord representations used in the WJazzD. In Section 3, we will elaborate on how these difficulties are tackled in a detailed manner.

Furthermore, while recent works in Transformer-based music generation often praised the model's capabilities, like being able to compose "compelling" music, or generate pieces with "expressiveness, coherence, and clear structures" as claimed in [20] and [21] respectively, rarely do we admit that the machine is still far behind humans, as shown in our user study (Section 4), and take a step back to "face the music", in other words, to identify what exactly goes wrong in the model's compositions.

Therefore, the goal of the paper is two-fold. First, to deploy Transformers to a new, more complex music genre, Jazz, asking the model to compose melody lines, chord progression, and structures all at once. Second, to develop a set of objective metrics (Section 5) that evaluate the generated music's pitch usages, rhythmicity, consistency in chord progression, and structureness (see Sec. 5.4 for definition), to discover the culprits behind the model's incompetence (Section 6).

Figure 1 shows an example of a composition generated by our model, in which we may find reasonable combinations of chords and melody; and, clear rests between phrases. Audio samples can be found in a Google Drive folder, [1] which we encourage readers to listen to. We have also open-sourced our implementation of the Jazz Transformer [2] and the proposed objective metrics. [3]

2. RELATED WORK

There has been a great body of research work on computational analysis of human performance of Jazz [3, 4, 15, 18, 44]. One prominent example is the Jazzomat Project [5], which established the WJazzD [2] to study the creative processes underpinning Jazz solo improvisations [37]. Weiss *et al.* [44], for instance, used the dataset to explore the evolution of tonal complexity of Jazz improvisations in the past century. See Sec. 3.1 for more details of the dataset.

The use of Transformer-like architectures for training music composition models has drawn increasing attention recently. These works enhanced the Transformer's capability in modeling music through relative positional encoding schemes [20, 36], cross-domain pre-training [13], and event token design [13, 21]. To the best of our knowledge, this work represents the first attempt in the literature to employ Transformers to compose exclusively Jazz music.

Automatic composition of general lead sheets has been investigated lately, mostly based on recurrent neural network (RNN) models [7, 29, 30]. As for inducing structures in the generated music, several RNN-based solutions have also been proposed [24, 31, 39]. Since Transformers have been shown to outperform RNNs in various tasks [9, 20, 26], we strive to be the forerunner in bringing them to these realms of research.

Relatively little work has been done to train a model for Jazz composition. JazzGAN [42] is a model employing a generative adversarial network (GAN) architecture for chord-conditioned melody composition, using a dataset of only 44 lead sheets, approximately 1,700 bars. Another model presented in [22] explores the use of recurrent variational auto-encoders for generating both the melody and chords of a lead sheet from scratch.

A number of objective metrics have been employed for measuring the performance of deep learning for music composition [10, 14, 41, 45]. However, most of them focused on surface-level statistics only (e.g., pitch class histograms, note onset intervals, etc.). The introduction of structureness indicators and the MIREX-like metric (see Sec. 5.4–5.5) in this paper provide new insights into assessing music's quality at piece level, and evaluating the model's overall understanding of a certain music genre.

3. THE JAZZ TRANSFORMER

Transformers use self-attention modules to aggregate information from the past events when predicting the next

---

	# solos	Total duration	Total # events	Avg. # events per solo
Train	409	11h 19m	1,220 K	2,983
Val.	22	33m	56 K	2,548

Table 1. Statistics of the dataset we compile from the WJazzD [37]. See Section 3.2 for details of the "events".

events [11, 27, 43]. Accordingly, it is natural that we model music as a language, namely, to represent each composition by a sequence of event tokens. In this section, we will explain in detail how we break down the components of the WJazzD [37] to construct the vocabulary of events, and how the pieces are converted into sequences that can be fed into a Transformer-like model for training.

3.1 Dataset

The WJazzD dataset [2, 37] comprises of 456 monophonic Jazz solos. Each solo is arranged in the lead sheet style and comes with two tracks: the melody track and the beat track. The melody track contains every note's pitch, onset time and duration (in seconds), with additional information on loudness (in decibels), phrase IDs and "midlevel units" (MLUs) [17], a structure of finer granularity than a phrase to capture the distinctive short-time ideas in Jazz improvisations. The beat track contains the beat onsets (in seconds), chord progressions and form parts (or sections, e.g., A1, B1). The highlight of this dataset is that all the contents, including the notes, chords, metrical and structural markings, are human-annotated and cross-checked by the annotators [37], ensuring the data cleanliness that is often crucial for machine learning tasks. To simplify the subsequent processings, we retain only the pieces marked solely with 4/4 time signature, resulting in 431 solos. For objective analysis, we leave 5% of the solos as the held-out validation data. See Table 1 for the statistics of the data.

3.2 Data Representation

The event representation adopted here is a modified version of the "REvamped MIDI-derived event representation" recently proposed in [21], extended to integrate the chord system and structural events of WJazzD. The resulting event encodings can be broken down into the following 4 categories: **note-related**—NOTE-VELOCITY, NOTE-ON, NOTE-DURATION; **metric-related**—BAR, POSITION, TEMPO-CLASS, TEMPO; **chord-related**—CHORD-TONE, CHORD-TYPE, CHORD-SLASH; and **structure-related**—PHRASE, MLU, PART, REPETITION.

3.2.1 Note-related Events

Each note in the melody is represented by three events, i.e., NOTE-VELOCITY, NOTE-ON, and NOTE-DURATION.

The NOTE-VELOCITY event decides how hard the note should be played. We derive it according to the estimated loudness (in decibels) provided by the dataset, and quantize it into 32 bins, corresponding to MIDI velocities

---

[1] https://drive.google.com/drive/folders/
1-09SoxumYPdYetsUWHIHSugK99E2tNYD?usp=sharing
[2] https://github.com/slSeanWU/jazz_transformer
[3] https://github.com/slSeanWU/MusDr

$[3, 7, \ldots, 127]$, through $v = \lfloor (80 + 3 \cdot (dB - 65))/4 \rfloor$, where dB is the decibel value of the note, and v, clipped such that $v \in [1, 32]$, is the resulting NOTE-VELOCITY(v) event. This mapping scheme comes in handy in the process of converting the model's compositions to MIDIs.

The NOTE-ON events, ranging from 0 to 127, correspond directly to the MIDI numbers, indicating the note's pitch. The NOTE-DURATION events represent the note's length in 64th note multiples, ranging from 1 to 32, obtained by taking the ratio of the note's duration (in seconds) to the duration of the beat (also in seconds) where the note situates. The reason why we use such a fine-grained quantum, while previous work mainly consider only 16th note multiples (e.g., [20, 21]), is as follows. Most notes in WJazzD are quite short, with a significant portion being 32th and 64th notes (12.9% and 2.7% respectively). The quantum is chosen such that the coverage of the 32 NOTE-DURATION events encompasses the most notes, which is 99.6% with our choice of the 64th note. [4]

3.2.2 Metric-related Events

To model the progression of time, we use a combination of BAR and POSITION events; as demonstrated in [21], this combination leads to clearer rhythmic structure in the generated music compared to using TIME-SHIFT events introduced in [20]. In addition, the pace the music should be played at is set by TEMPO-CLASS and TEMPO events.

A BAR event is added at the beginning of each bar, and a bar is quantized into 64 subunits, each represented by a POSITION event; for example, POSITION(16) marks the start of the 2nd beat in a bar. A POSITION event occurs whenever there is a note onset, chord change, or tempo change. It is worth mentioning that to minimize the quantization error, a note's onset position is justified with the beat it is in through the formula:

$$p_n = p_b + 16 \cdot (t_n - t_b)/d_b, \qquad (1)$$

where p_b, t_b, d_b are the beat's position (note that $p_b \in \{0, 16, 32, 48\}$), onset time, and duration; and t_n is the note's onset time. The resulting p_n is then rounded to the nearest integer to determine the note's onset position.

The TEMPO-CLASS and TEMPO events always co-occur at every beat position. The 5 TEMPO-CLASS events represent the general "feeling" of speed (i.e. fast, or slow) with interval boundaries of $[50, 80, 110, 140, 180, 320]$ beats per minute (bpm), while the 12 TEMPO events assigned to each tempo class in evenly-spaced steps (within the interval, e.g., $50, 52.5, 55$ bpm...) determine the exact pace. The events can be derived simply by taking the reciprocal of a beat's duration (provided by WJazzD). The frequent appearance of these tempo events facilitates smooth local tempo changes common in Jazz performances.

3.2.3 Chord-related Events

Chord progressions serve as the harmonic foundation of Jazz improvisations [25], hence a complex chord representation system is used in the WJazzD dataset. If we were to treat each of the 418 unique chord representations present in the WJazzD as a token, the majority of chord tokens will have very few occurrences—in fact, 287 (69%) of them appear in less than 5 solos, making it hard for the model to learn the meaning of those chords well; plus, the process of translating chords to individual notes during the conversion to MIDIs would be extremely cumbersome.

Fortunately, thanks to the detailed clarification provided in [37], we are able to decompose each chord into 3 events, namely, the CHORD-TONE, CHORD-TYPE, and CHORD-SLASH events, with the help of regular expressions (regex) and some rule-based exception handling.

The 12 CHORD-TONE events, one for each note on the chromatic scale (i.e. C, C#, D, ...), determine the root note, hence the tonality, of the chord. The 47 CHORD-TYPE events affect the chord's quality and emotion by the different combination of notes played relative to the root note (or, *key template* as we call it); e.g., the key template of a Dominant 7th chord (CHORD-TYPE(7)) is $[0, 4, 7, 10]$. Finally, the 12 CHORD-SLASH events allow the freedom to alter the bass note to slightly tweak the chord's quality. If a chord contains no slash, its CHORD-SLASH event will share the same key as its CHORD-TONE. For instance, the chord C7/G, a C Dominant 7th over G, is represented by [CHORD-TONE(C), CHORD-TYPE(7), CHORD-SLASH(G)].

Note that after our decomposition, the number of unique chord-related events is greatly reduced to 71; and, the resulting set of events is still able to represent all 418 chords in WJazzD. It is easy to use the manually-constructed key template accompanying each CHORD-TYPE, together with the CHORD-TONE and CHORD-SLASH events to map each chord to notes during the conversion to MIDIs.

3.2.4 Structure-related Events

For the melodies, we prepend a PHRASE event to the notes marked as the start of a phrase. The presence of phrases may be important as it informs the model to "take a breath" between streams of notes. And, we retain several common types and subtypes of midlevel units (e.g., *line, rhythm, lick* etc.) as MLU events [17], likewise prepended to the starting note of each MLU, hoping that the model could capture the short-term note patterns described by the MLU types. PART and REPETITION events are added to each beginning and end of a form part, [5] guiding the model to generate repetitive chord progression and coherent melody lines for the parts marked with the same letter.

3.3 Model and Training Setups

Due to the large number of events per solo (check Table 1), it is hard to feed the entire pieces into a Transformer at once because of memory constraint. Therefore, we choose as the backbone sequence model the Transformer-XL [11], an improved variant of the Transformer which introduces recurrence to the architecture. It remedies the memory constraint and the resulting context fragmentation issue by caching the computation record of the last segment, and

---

[4] All notes shorter than a 64th note are discarded and those longer than a half note are clipped.

[5] For example, the entire A1 part is represented as [PART-START(A), REPETITION-START(1), ... other events ..., REPETITION-END(1), PART-END(A)].

allowing the current segment to attend to the cache in the self-attention process. This allows information to flow across the otherwise separated segments, inducing better coherence in the generated music.

To evaluate the effectiveness of adding the structure-related events (cf. Section 3.2.4), we consider the following two variants in our objective analysis:

- **Model (A)**: trained with no structure-related events.

- **Model (B)**: trained with the complete set of events.

They both consist of 12 layers, 8 attention heads and about 41 million learnable parameters. We train them on a single NVIDIA GTX 1080-Ti GPU (with 11 GB memory) with Adam optimizer, learning rate $1e-4$, batch size 8, segment length 512 and 100% teacher forcing. Besides, following the Music Transformer's data augmentation setting [20], we randomly transpose each solo in the range of -3 to $+3$ keys in every epoch. It takes roughly a full day for the negative log-likelihood losses of the models to drop to 0.25, a level at which they are able to produce music of distinctive Jazz feeling (see Section 6 for justifications).

4. SUBJECTIVE STUDY
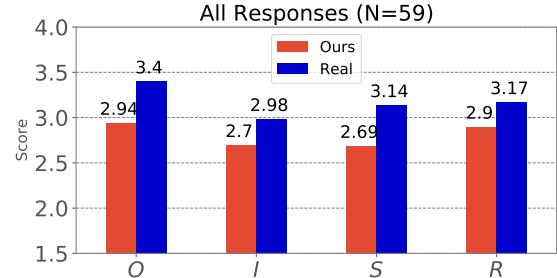
To discover how users feel about the Jazz Transformer's compositions, we set up a blind listening test in which test-takers listen to four one-minute long pieces, two from the Model (B)'s compositions (at loss level 0.25), and two from real data. We do not include Model (A) here to reduce the burden on the test-takers, assuming that Model (B) is better. We inform them that the pieces are independent of one another, and they will be asked the same set of questions after listening to each piece, namely, to rate it in a five-point Likert scale on the following aspects:

- **Overall Quality (O):** Does it sound good overall?

- **Impression (I):** Can you remember a certain part or the melody?

- **Structureness (S):** Does it involve recurring music ideas, clear phrases, and coherent sections?

- **Richness (R):** Is the music diverse and interesting?

We distribute five test suites to our social circles and collect responses from 59 anonymized subjects, of which 27 are classified as "pros" for they rate their musical background (in general, not restricted to Jazz) as $4/5$ or $5/5$ (i.e., also on a five-point scale). The result shown in Figure 2 indicates that the Jazz Transformer receives mediocre scores and falls short of humans in every aspect, especially in *overall quality* (**O**) and *structureness* (**S**). Moreover, performed *one-tailed Z-tests for the difference of means* also suggests the significance of the gaps ($p < 0.05$ for all aspects), providing concrete evidence of the model's defeat.

5. OBJECTIVE EVALUATION METRICS

The result of our subjective study poses to us an intriguing question: If the machine is still inferior to humans in creating music, then what exactly are the causes? To unravel the

Figure 2. Result of subjective study (**O:** Overall Quality, **I:** Impression, **S:** Structureness, **R:** Richness), comparing from-scratch compositions created by the proposed model with structure-related events (i.e., 'Model (B)') against the real pieces from the WJazzD. We note that the gaps in all aspects are statistically significant ($p < 0.05$).

mystery, we develop a set of objective metrics which enables us to scrutinize the Jazz Transformer's compositions from various perspectives, and make comparisons with real data. These metrics include the analyses of event distributions, namely, the pitch class histogram, the grooving pattern, and the chord progressions; assessing the structureness with the help of the fitness scape plot; and, judging the model's performance on a discriminative task through the MIREX-like continuation prediction challenge.

5.1 Pitch Class Histogram Entropy

To gain insight into the usage of different pitches, we first collect the notes appeared in a certain period (e.g., a bar) and construct the 12-dimensional pitch class histogram $\overrightarrow{\mathbf{h}}$, according to the notes' pitch classes (i.e. C, C#, ..., A#, B), normalized by the total note count in the period such that $\sum_i h_i = 1$. Then, we calculate the entropy of $\overrightarrow{\mathbf{h}}$:

$$\mathcal{H}(\overrightarrow{\mathbf{h}}) = -\sum_{i=0}^{11} h_i \log_2(h_i). \qquad (2)$$

The entropy, in information theory, is a measure of "uncertainty" of a probability distribution [40], hence we adopt it here as a metric to help assessing the music's quality in tonality. If a piece's tonality is clear, several pitch classes should dominate the pitch histogram (e.g., the tonic and the dominant), resulting in a low-entropy $\overrightarrow{\mathbf{h}}$; on the contrary, if the tonality is unstable, the usage of pitch classes is likely scattered, giving rise to an $\overrightarrow{\mathbf{h}}$ with high entropy.

5.2 Grooving Pattern Similarity

The grooving pattern represents the positions in a bar at which there is at least a note onset, denoted by $\overrightarrow{\mathbf{g}}$, a 64-dimensional binary vector in our setting.[6] We define the similarity between a pair of grooving patterns $\overrightarrow{\mathbf{g}}^a$, $\overrightarrow{\mathbf{g}}^b$ as:

$$\mathcal{GS}(\overrightarrow{\mathbf{g}}^a, \overrightarrow{\mathbf{g}}^b) = 1 - \frac{1}{Q} \sum_{i=0}^{Q-1} \text{XOR}(g_i^a, g_i^b), \qquad (3)$$

---

[6] For example, if a bar contains only two note onsets, at the beginning of the 1st beat and 2nd beat respectively, then the corresponding $\overrightarrow{\mathbf{g}}$ will have $g_0, g_{16} = 1$, and the rest dimensions 0.

where Q is the dimensionality of $\overrightarrow{\mathbf{g}}^a$, $\overrightarrow{\mathbf{g}}^b$, and $\text{XOR}(\cdot, \cdot)$ is the exclusive OR operation. Note that the value of $\mathcal{GS}(\cdot, \cdot)$ would always lie in between 0 and 1.

The grooving pattern similarity helps in measuring the music's rhythmicity. If a piece possesses a clear sense of rhythm, the grooving patterns between pairs of bars should be similar, thereby producing high $\mathcal{GS}$ scores; on the other hand, if the rhythm feels unsteady, the grooving patterns across bars should be erratic, resulting in low $\mathcal{GS}$ scores.

5.3 Chord Progression Irregularity

To measure the irregularity of a chord progression, we begin by introducing the term *chord trigram*, which is a triple composed of 3 consecutive chords in a chord progression; for example, (`Dm7`, `G7`, `CM7`). Then, *the chord progression irregularity* ($\mathcal{CPI}$) is defined as the percentage of *unique chord trigrams* in the chord progression of an entire piece. Please note that 2 chord trigrams are considered different if any of their elements does not match.

It is common for Jazz compositions to make use of 8- or 12-bar-long templates of chord progressions (known as the *8-*, or *12-bar blues*), which themselves can be broken down into similar substructures [25, 35], as the foundation of a section, and more or less "copy-paste" them to form the complete song with, say, `AABA` parts. Therefore, a well-composed Jazz piece should have a chord progression irregularity that is not too high.

5.4 Structureness Indicators

The *structureness* of music is induced by the repetitive musical content in the composition. It can involve multiple granularities, ranging from an instant musical idea to an entire section. From a psychological perspective, the appearance of repeated structures is the essence of the catchiness and the emotion-provoking nature of music [28].

The fitness scape plot algorithm [32, 33] and the associated SM Toolbox [34] offer an aesthetic way of detecting and visualizing the presence of repeating structures in music. The fitness scape plot is a matrix $S_{N \times N}$,[7] where $s_{ij} \in [0, 1]$ is the *fitness*, namely, the degree of repeat in the piece derived from the *self-similarity matrix* (SSM) [16], of the segment specified by (i, j).
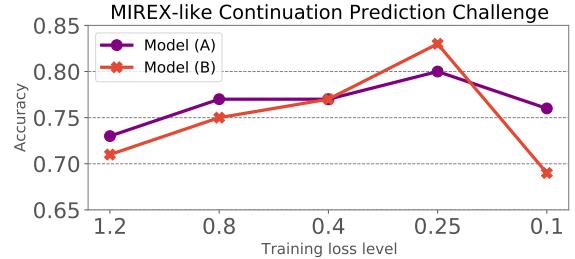
Our *structureness indicator* is based on the fitness scape plot and designed to capture the most salient repeat within a certain duration interval. For brevity of the mathematical representation, we assume the sampling frame rate of S is 1 Hz (hence N will be the piece's duration in seconds), and define the structureness indicator as follows:

$$\mathcal{SI}_l^u(S) = \max_{\substack{l \leq i \leq u \\ 1 \leq j \leq N}} S, \tag{4}$$

where l, u [8] are the lower and upper bounds of the duration interval (in seconds) one is interested in. In our experiments, we choose the structureness indicators of $\mathcal{SI}_3^8$, $\mathcal{SI}_8^{15}$, and $\mathcal{SI}_{15}$ to examine the short-, medium-, and long-term structureness respectively.

---

[7] N is the number of frames sampled from the audio of a piece, the 1st axis represents the segment duration (in frames), and the 2nd axis represents the center of segment (in frames).

[8] If present, otherwise l defaults to 1, and u defaults to N.

Figure 3. Result of the MIREX-like continuation prediction challenge, each checkpoint is asked 100 questions. Notice that the accuracy of both Model (A) and (B) peaks at the loss level of 0.25, at 80% and 83% respectively.

5.5 MIREX-like Continuation Prediction Challenge

Being inspired by the "Patterns for Prediction Challenge" held as part of the Music Information Retrieval Evaluation eXchange (MIREX) 2019 [1, 23], we developed a method to test the model's capability to predict the correct continuation given a musical prompt. The challenge is carried out as follows: First, the model is fed with the beginning 8 bars of a piece, denoted by $\overrightarrow{\mathbf{s}}$; then, it is presented with a set of four 8-bar continuations $\mathcal{X} = \{\overrightarrow{\mathbf{x}}^0, \overrightarrow{\mathbf{x}}^1, \overrightarrow{\mathbf{x}}^2, \overrightarrow{\mathbf{x}}^3\}$, in which one is the true continuation, and the rest are wrong answers randomly drawn from other pieces. The way the model attempts to answer the multiple choice question is by calculating the average probability of generating the events of each continuation:

$$\mathcal{P}(\overrightarrow{\mathbf{x}}^i) = \frac{1}{L} \sum_{j=0}^{L-1} p(x_j^i \mid \tilde{x}_{j-1}, \ldots, \tilde{x}_0; \overrightarrow{\mathbf{s}}), \ i \in \{0, 1, 2, 3\},$$
$$\tag{5}$$

where L is the length of the shortest given continuation (in # events) in $\mathcal{X}$, x_j^i is the j-th event token in $\overrightarrow{\mathbf{x}}^i$, and $\tilde{x}_{j-1}, \ldots, \tilde{x}_0$ are the events sampled from the model's output, hence the conditional probability $p(x_j^i)$ at each timestep can be obtained straightforward. Finally, the model returns $\arg\max_i \mathcal{P}(\overrightarrow{\mathbf{x}}^i)$ as its answer, of which the correctness we can check.

If the model can achieve high accuracy on this continuation prediction task, we may say it possesses a good overall understanding of Jazz music, enough for it to tell right from wrong when given multiple choices.

6. EXPERIMENT RESULTS AND DISCUSSIONS

We begin with the evaluation on the MIREX-like challenge (Section 5.5). We pick 5 checkpoints of both Model (A) and Model (B) at different training loss levels to ask each of them 100 multiple choice questions (the prompt and continuation choices of each question are randomly drawn from the held-out validation data). The result shown in Figure 3 indicates that, similarly for both models, the accuracy steadily goes up as the training loss decreases, peaks at the loss level of 0.25, and drops afterwards. This shows that the models are gradually gaining knowledge about Jazz music along the training process until a certain point, where they potentially start to overfit.
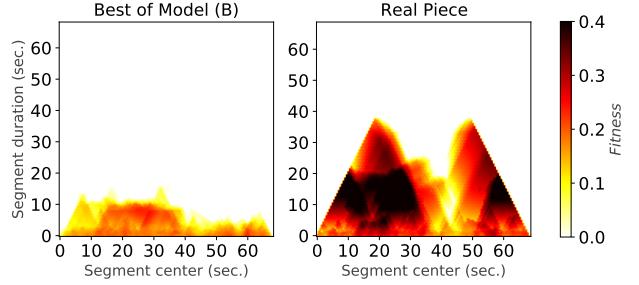
	Model (A)		Model (B)			Real
loss	*0.80*	*0.25*	*0.80*	*0.25*	*0.10*	- -
$\mathcal{H}_1$	2.29	2.45	2.26	2.20	**2.17**	1.94
$\mathcal{H}_4$	3.12	3.05	3.04	**2.91**	2.94	2.87
$\mathcal{GS}$	**0.76**	0.69	0.75	**0.76**	**0.76**	0.86
$\mathcal{CPI}$	81.2	77.6	79.2	**72.6**	75.9	40.4
$\mathcal{SI}_3^8$	0.18	0.22	0.25	**0.27**	0.26	0.36
$\mathcal{SI}_8^{15}$	0.15	0.17	**0.18**	**0.18**	0.17	0.36
$\mathcal{SI}_{15}$	0.11	**0.14**	0.10	0.12	0.11	0.35

Table 2. Results of objective evaluations. $\mathcal{H}_1$, $\mathcal{H}_4$ are the 1-, and 4-bar pitch class histogram entropy (see Sec. 5.1); $\mathcal{GS}$ is the grooving pattern similarity (Sec. 5.2) measured on all pairs of bars within a piece; $\mathcal{CPI}$ is the chord progression irregularity (in %; Sec. 5.3); finally, $\mathcal{SI}_3^8$, $\mathcal{SI}_8^{15}$, and $\mathcal{SI}_{15}$ are the short-, medium-, and long-term structureness indicators (Sec. 5.4). **Bold** texts indicate the model checkpoint performing the closest to real data, which is considered to be the best. It is observed that Model (B) (i.e., the model trained with structure-related events) with a loss of 0.25 outperforms its counterparts at other loss levels and Model (A) on most of the metrics. Moreover, consistent with the result of the MIREX-like challenge (Fig. 3), the performance of Model (B) plunges when the loss goes too low (0.1 in this case).

Following the MIREX-like challenge, we pick several checkpoints of both Models (A) and (B) for objective evaluations described in Sections 5.1–5.4. The chosen checkpoints are at loss levels 0.8, 0.25, and 0.1 (for Model (B) only, since in the MIREX-like challenge (Fig. 3), its accuracy drastically drops when the loss reduces from 0.25 to 0.1). In the experiments, 50 32-bar-long from-scratch compositions from each checkpointed model are compared against the 409 pieces in the training dataset.

From the results (Table 2), we can summarize the model's deficiencies as follows: 1) the *erraticity* of the generated musical events; and, 2) the *absence* of medium- and long-term repetitive structures. Comparing with the real data, the first argument can be justified by the higher $\mathcal{H}_1$ and $\mathcal{H}_4$, manifesting the unstable usage of pitches at local scale; and, the lower $\mathcal{GS}$ and higher $\mathcal{CPI}$ of the entire pieces, marking the lack of consistency in rhythm and harmony from a global point of view; meanwhile, the second argument can be explained directly by the significantly lower values of structureness indicators $\mathcal{SI}_8^{15}$ and $\mathcal{SI}_{15}$, suggesting that while the model might be able to repeat some short fragments of music, creating structures of a longer time span is still beyond its capability.

Much to our delight, the introduction of structure-related events seems to be functional to some extent, noticeable from the numbers that Model (B) at 0.25 loss level is for most of the time the closest competitor to humans, with a substantial lead on the metrics focusing on shorter timespans (i.e., $\mathcal{H}_1$, $\mathcal{H}_4$, and $\mathcal{SI}_3^8$) when placed in comparison with Model (A). This suggests that the use of PHRASE and MLU events provides some assistance to the model in modeling music. Furthermore, resonating with the accu-

Figure 4. The fitness scape plots of Model (B)'s best composition (according to the *structureness* (**S**) score in our subjective study, see Sec. 4) versus a human composition in the WJazzD. Note that the piece by Model (B) contains almost no signs of repetition longer than 10 seconds, while the real piece's repetitive structures extend well into the 20–30 seconds range.

racy trend in the MIREX-like challenge, the performance worsens when the loss is reduced to an overly low level.

To visualize the deficiency in structureness of the model's compositions, we choose the piece which scores the highest, 3.14, in the *structureness* (**S**) aspect in our subjective study; and, a human composition of the same duration, receiving 3.54 in the aspect **S**, for a head-to-head comparison of their fitness scape plots. The rivalry (see Figure 4) reveals the stark contrast between their fitness values across all timescales. In the model's work, all traces of repetitive structures disappear at the timescale of 10 seconds; whereas in the human composition, not only do the fitness values stay high in longer timespans, but a clear sense of section is also present, as manifested by the 2 large, dark "triangles" in its scape plot.

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented the Jazz Transformer, whose incorporation of structure-related events has been shown useful here in enhancing the quality of machine-generated music. Moreover, we have proposed a series of objective metrics that shed light on the shortcomings of machine-composed pieces, including the erratic usage of pitch classes, inconsistent grooving pattern and chord progression; and, the absence of repetitive structures. These metrics not only show that the Transformer is in fact not that good a music composer, but also serve as effective quantitative measures for future efforts in automatic music composition to assess their models' performance, which by now still relies heavily on human evaluation.

In the future, we plan to carry out larger-scale studies to explicate the correlations between those quantitative metrics and the aspects of subjective evaluation; and, to continue working on inducing structures in machine-composed music; such endeavors may not stay on revamping events that fit into Transformers as done, but involve a complete redesign of the Transformer architecture, enabling it to read the structural information directly computable from data, say, the fitness scape plot, to grasp the blueprint of a piece before composing music at finer scales.

8. ACKNOWLEDGEMENTS

9. REFERENCES

[1] The "Patterns for Prediction Challenge" of Music Information Retrieval Evaluation eXchange. [Online] https://www.music-ir.org/mirex/wiki/ 2019:Patterns_for_Prediction.

[2] The Weimar Jazz Database. [Online] https:// jazzomat.hfm-weimar.de/.

[3] Jakob Abeßer, Stefan Balke, Klaus Frieler, Martin Pfleiderer, and Meinard Müller. Deep learning for Jazz walking bass transcription. In *Proc. AES International Conference on Semantic Audio*, 2017.

[4] Jakob Abeßer, Estefanía Cano, Klaus Frieler, Martin Pfleiderer, and Wolf-Georg Zaddach. Score-informed analysis of intonation and pitch modulation in Jazz solos. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, pages 823–829, 2015.

[5] Jakob Abeßer, Klaus Frieler, Martin Pfleiderer, and Wolf-Georg Zaddach. Introducing the Jazzomat project – Jazz solo analysis using music information retrieval methods. In *Proc. International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2013.

[6] Christopher Anderson, Arne Eigenfeldt, and Philippe Pasquier. The generative electronic dance music algorithmic system (GEDMAS). In *Proc. Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.

[7] Cedric De Boom, Stephanie Van Laere, Tim Verbelen, and Bart Dhoedt. Rhythm, chord and melody generation for lead sheets using recurrent neural networks. *arXiv preprint arXiv:2002.10266*, 2020.

[8] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. *Deep Learning Techniques for Music Generation, Computational Synthesis and Creative Systems*. Springer, 2019.

[9] Tsung-Ping Chen and Li Su. Harmony Transformer: Incorporating chord segmentation into harmony recognition. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, pages 259–267, 2019.

[10] Ching-Hua Chuan and Dorien Herremans. Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. In *Proc. AAAI Conference on Artificial Intelligence*, 2018.

[11] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2978–2988, 2019.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] Chris Donahue, Huanru Henry Mao, Yiting Ethan Li, Garrison W. Cottrell, and Julian McAuley. LakhNES: Improving multi-instrumental music generation with cross-domain pre-training. In *Proc. International Society for Music Information Retrieval (ISMIR)*, pages 685–692, 2019.

[14] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proc. AAAI Conference on Artificial Intelligence*, pages 34–41, 2018.

[15] Vsevolod Eremenko, Emir Demirel, Baris Bozkurt, and Xavier Serra. Audio-aligned Jazz harmony dataset for automatic chord transcription and corpus-based research. In *Proc. International Conference on Music Information Retrieval (ISMIR)*, pages 483–490, 2018.

[16] Jonathan Foote. Visualizing music and audio using self-similarity. In *Proc. ACM International Conference on Multimedia*, pages 77–80, 1999.

[17] Klaus Frieler, Martin Pfleiderer, Wolf-Georg Zaddach, and Jakob Abeßer. Midlevel analysis of monophonic Jazz solos: A new approach to the study of improvisation. *Musicae Scientiae*, 20:143–162, 2016.

[18] Jeff Gregorio and Youngmoo Kim. Phrase-level audio segmentation of jazz improvisations informed by symbolic data. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[19] Ryan Groves. Automatic melodic reduction using a supervised probabilistic context-free grammar. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[20] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music Transformer: Generating music with long-term structure. In *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[21] Yu-Siang Huang and Yi-Hsuan Yang. Pop Music Transformer: Beat-based modeling and generation of expressive Pop piano compositions. In *Proc. ACM International Conference on Multimedia*, 2020.

[22] Hsiao-Tzu Hung, Chung-Yang Wang, Yi-Hsuan Yang, and Hsin-Min Wang. Improving automatic jazz melody generation by transfer learning techniques. In *Proc. Asia Pacific Signal and Information Processing Association Annual Summit and Conf. (APSIPA ASC)*, 2019.

[23] Berit Janssen, Tom Collins, and Iris Ren. Algorithmic ability to predict the musical future: Datasets and evaluation. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, pages 208–215, 2019.

[24] Harsh Jhamtani and Taylor Berg-Kirkpatrick. Modeling self-repetition in music generation using structured adversaries. In *Proc. Machine Learning for Media Discovery Workshop, extended abstract*, 2019.

[25] Philip Johnson-Laird. How Jazz musicians improvise. *Music Perception — MUSIC PERCEPT*, 19:415–442, 2002.

[26] Shigeki Karita et al. A comparative study on Transformer vs RNN in speech applications. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pages 449–456, 2019.

[27] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proc. International Conference on Machine Learning*, 2020.

[28] Daniel J. Levitin. *This is Your Brain on Music: The Science of a Human Obsession*. Dutton, 2006.

[29] Hyungui Lim, Seungyeon Rhyu, and Kyogu Lee. Chord generation from symbolic melody using BLSTM networks. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, pages 621–627, 2017.

[30] Hao-Min Liu, Meng-Hsuan Wu, and Yi-Hsuan Yang. Lead sheet generation and arrangement via a hybrid generative model. In *Proc. International Society for Music Information Retrieval Conference (ISMIR), late-breaking demo*, 2018.

[31] Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, Samer Abdallah, Marco Selvi, Ed Newton-Rex, and Kevin Webster. StructureNet: Inducing structure in generated melodies. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, pages 725–731, 2018.

[32] Meinard Müller, Peter Grosche, and Nanzhu Jiang. A segment-based fitness measure for capturing repetitive structures of music recordings. In *Proc. International Conference on Music Information Retrieval (ISMIR)*, pages 615–620, 2011.

[33] Meinard Müller and Nanzhu Jiang. A scape plot representation for visualizing repetitive structures of music recordings. In *Proc. International Conference on Music Information Retrieval (ISMIR)*, pages 97–102, Porto, Portugal, 2012.

[34] Meinard Müller, Nanzhu Jiang, and Harald G. Grohganz. SM Toolbox: MATLAB implementations for computing and enhancing similarity matrices. In *Proc. Audio Engineering Society (AES)*, 2014.

[35] Simon John Nelson. *Melodic improvisation on a twelve bar blues model: an investigation of physical and historical aspects and their contribution to performance*. PhD thesis, City University London, 2001.

[36] Christine McLeavy Payne. MuseNet. *OpenAI Blog*, 2019.

[37] Martin Pfleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart, editors. *Inside the Jazzomat — New Perspectives for Jazz Research*. Schott Campus, 2017.

[38] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *Open AI Blog*, 1(8), 2019.

[39] Shakeel Raja. Music generation with temporal structure augmentation. *arXiv preprint arXiv:2004.10246*, 2020.

[40] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[41] Bob L. Sturm and Oded Ben-Tal. Taking the models back to music practice: Evaluating generative transcription models built using deep learning. *Journal of Creative Music Systems*, 2(1), 2017.

[42] Nicholas Trieu and Robert M. Keller. JazzGAN: Improvising with generative adversarial networks. In *Proc. International Workshop on Musical Metacreation*, 2018.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[44] Christof Weiss, Stefan Balke, Jakob Abeßer, and Meinard Müller. Computational corpus analysis: A case study on Jazz solos. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, pages 416–423, 2018.

[45] Li-Chia Yang and Alexander Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 2018.

EXPLAINING PERCEIVED EMOTION PREDICTIONS IN MUSIC: AN ATTENTIVE APPROACH

Sanga Chaki[1]   **Pranjal Doshi**[2]   **Sourangshu Bhattacharya**[2]
Priyadarshi Patnaik[3]

[1] Advanced Technology Development Centre, IIT Kharagpur, India
[2] Department of Computer Science and Engineering, IIT Kharagpur, India
[3] Department of Humanities and Social Sciences, IIT Kharagpur, India

`s.chaki27@gmail.com, sourangshu@gmail.com`

ABSTRACT

Dynamic prediction of perceived emotions of music is a challenging problem with interesting applications. Utilization of relevant context in audio sequence is essential for effective prediction. Existing methods have used LSTMs with modest success. In this work we describe three attentive LSTM based approaches for dynamic emotion prediction from music clips. We validate our models through extensive experimentation on standard dataset annotated with arousal-valence values in continuous time, and choose the best performer. We find that the LSTM based attention models perform better than the state of the art transformers for the dynamic emotion prediction task, both in terms of R^2 and Kendall-τ metrics. We explore individual smaller feature sets in search of a more effective one and to understand how different features contribute to perceived emotion. The spectral features are found to perform at par with the generic ComPare feature set [1]. Through attention map analysis we visualize how attention is distributed over music clips' frames for emotion prediction. It is observed that the models attend to frames which contribute to changes in reported arousal-valence values and chroma to produce better emotion predictions, effectively capturing long-term dependencies.

1. INTRODUCTION

Automatic determination of perceived emotion in music is an active and major area of focus for the music information retrieval (MIR) community. The aim of dynamic perceived emotion prediction task is to output a sequence of time-synchronized arousal-valence labels when a music clip is given as input. It finds varied applications in the domains of personalized and/or generalized music recommendations, organizing music databases, automatic music creation, mood based music search etc. This task is challenging because: 1) perceived emotion might depend on

the inherent relationship between different frames of music, distributed over time, and 2) emotion perception is inherently subjective in nature, highly contextual and personal. Thus, it is understandable that the emotions related to music are a time-continuous process, where the context of the sequential music frames play an immense role on the associated emotion. Relating this to the machine learning perspective, one can discern the need of context sensitive models like recurrent neural networks (RNNs) for the task at hand. In this study, we use attention mechanism with a deep RNN-LSTMs (Long Short Term Memory) and the Transformer [2], to predict the perceived emotion in each defined time frame of music continuously. We compare our approach with recent works [3] using only LSTM. We also attempt to understand the importance of types of features contributing to dynamic perceived emotion. Lastly, attention is visualized with the help of attention map analysis. The following are the major contributions of this work: 1) The LSTM based attention models are found to perform better than the state of the art Transformers for the dynamic emotion prediction task. 2) Spectral features are found to perform at par with the generic ComPare feature set [1]. 3) Attention maps are interpreted to observe that the attention models are able to focus on relevant music frames for dynamic emotion prediction task.

This paper is organized as follows. In section 2, relevant literature regarding music emotion recognition and attention is reviewed. Section 3 provides details of the attention based models and Transformer used in this work. All the experiments carried out and the observations are reported in section 4. Finally, the conclusions drawn from the present study are detailed in section 5.

2. RELATED WORK

2.1 Music Emotion Recognition

In the past, most music emotion prediction systems used features of timbre, pitch, MFCCs and/or lyrics and applied to classifiers like SVMs [4]. Current state-of-the-art methods for music emotion prediction are mostly based on deep neural networks like RNN-LSTMs. Coutinho et al. [5] proposed the use of this model for this task. Weninger et al. [3, 6, 7] used RNN-LSTM networks successfully to perform continuous time music emotion regression, using

a modified cost function, on the *1000 Songs for Emotional Analysis of Music* dataset [8]. Giamusso et al. [9] used neural networks to predict playlist emotions based on lyrics. Fan et al. [10] performed ranking based emotion recognition from experimental music. Delbouys et al. [11] used LSTM and ConvNet models on the Million Song Dataset [12] for audio and lyrics based bimodal music emotion detection.

2.2 Emotion Representation

Over the years, Discrete and Dimensional models of emotion representation have been used in MIR. Studies using discrete model either tag their musical data with single [13] or cluster [14] of simple tags. In dimensional models like Russel's Circumplex model [15], emotion is mapped into a 2-D plane, spanned by two axes denoting *arousal* and *valence*. Using this well known and satisfyingly exhaustive emotion representation, the problem of emotion recognition/prediction is turned into a two dimensional regression problem [16].

2.3 Attention in MIR tasks

Recently, attention mechanism and Transformer models have found application in a wide range of MIR tasks, with success. Balke et al. [17] used a soft-attention mechanism on input of synthesized piano data for audio sheet music retrieval. Their results indicate that attention increases the robustness of the retrieval system by focusing on different parts of the input representation based on the tempo of the audio. The improved results led them to argue for the potential of attention models as a very general tool for many MIR tasks. Gururani et al. [18] explored an attention mechanism for handling weakly labeled data for multi-label instrument recognition. Their results show that incorporating attention leads to overall improvement in classification accuracy metrics and enables models to *attend to* specific time segments in the audio relevant to each instrument label leading to interpretable results. Donahue et al. [19] used the Transformer architecture to improve performance for the task of generating multi-instrumental music scores. Chen et al. [20] proposed the Harmony Transformer, a multi-task music harmony analysis model aiming to improve chord recognition. Park et al. [21] utilized a bi-directional Transformer for chord recognition (BTC) which showed competitive performance. Through attention map, they visualized how attention was performed, and it was observed that the model was able to divide segments of chords by utilizing adaptive receptive field of the attention mechanism and capture long-term dependencies. These and other works have explored various feature sets like CQT (in [21]), Chroma (in [20]), along with other standard feature sets [1] (in [3]). These recent successes in varied MIR tasks in terms of model accuracy and interpretability, motivated us to apply the same in the music emotion regression task. To the best of our knowledge, neither attention models nor Transformers have been applied before to the task under examination.

3. ATTENTION BASED MODELS FOR EMOTION PREDICTION IN MUSIC

3.1 Attention Model (AT)

In the past, traditional LSTM-RNN approach has provided good results in music emotion regression [3]. In this work we propose the use of attention mechanism for dynamic emotion prediction in music. According to the *attention* model [22], to compute each output of a encoder-decoder architecture, a distinct *context vector* is used, which is a function of all the hidden states at the encoder side and not just the last one. The encoder encodes the input into a set of hidden states and attention is applied on them to produce target arousal and valence values over fixed length segments or time frames of the music audio signal. The encoder reads the input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, which is a sequence of vectors, and produces the hidden states $(h_1, h_2, \ldots, h_T)$, using some RNN approach. In this work LSTM is used. In traditional attention mechanism [22], the whole set of hidden states $(h_1, h_2, \ldots, h_T)$ are available to compute the context vectors. Each time, the context vector c_i is calculated as a weighted sum of all the hidden states. Let the output be $\mathbf{y} = (y_1, y_2, \ldots, y_T)$. For the current problem, $\mathbf{y}$ can be defined as set of arousal or valence values associated with each music time frame. The tth output, y_t, will be a function $\mathbf{g}()$ of a) the present hidden state h_t, b) the previous output y_{t-1}, c) the unique context vector c_t, as given by equation 1.

$$p(y_t|y_1, y_2, \ldots, y_{t-1}, \mathbf{x}) = g(h_t, y_{t-1}, c_t) \qquad (1)$$

The unique context vector c_t depends on the sequence of annotations $(h_1, h_2, \ldots, h_T)$, and is computed as a weighted sum of these annotations h_j, as given in equation 2.

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_j \qquad (2)$$

So, the model at time t, *attends* to each h_j corresponding to each of the inputs, with a weight of α_{tj}. To obtain each weight α_{tj} for each output y_t, the alignment between the corresponding h_t and each of h_j need to be calculated, where $1 \leq j \leq T$. So, the alignment model, when attending to h_j, is given by equation 3.

$$e_{tj} = a(h_{t-1}, h_j), 1 \leq j \leq (t-1) \qquad (3)$$

This alignment is the measure of how well the inputs around position j and the output at position t match. Then, each of these scores e_{tj} are used to calculate the attention weights for each h_j as given in equation 4.

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T} exp(e_{tk})} \qquad (4)$$

So, for each output, the context vector will *attend* or focus on those parts of the entire input sequence, which are more relevant for that particular output, by assigning higher weights to the associated encoder-side hidden states, using an *alignment* model. These models are referred to as the *AT* models from hereon. The naming convention of the models is the acronym *AT* for attention, followed by the hidden layer dimensions.

Table 1: Model Selection for Dynamic Arousal Prediction

Model	Parameter Search (Layer Size)	Best Model	R_A^2	$\overline{\tau}_A$	MAE_A
Baseline [3]	400	-	0.60	0.14	0.11
LSTM (Single Layer)	128, 300, 400, 512, 700, 1024, 2048	1024	0.73	0.12	0.12
LSTM (Multi Layer)	(700_128), (700_400), (2048_1024), (2048_1024_700)	(700_128)	0.69	0.20	0.12
AT (Single Layer)	32, 64, 128, 300, 400, 512, 700, 1024, 2048	300	0.75	0.15	0.13
AT (Multi Layer)	(300_128), (400_128), (1024_400), (2048_1024), (2048_1024_512)	(2048_1024)	**0.78**	0.24	0.11
BAT (Single Layer)	400, 1024, 2048	1024	0.55	0.04	0.12
BAT (Multi Layer)	(300_128), (400_128), (1024_400), (1024_512), (2048_1024), (2048_1024_512)	(2048_1024)	0.58	0.06	0.12
Transformer	1-Layer, 2-Layer, 4-Layer	2-Layer	0.64	0.61	0.27

Table 2: Model Selection for Dynamic Valence Prediction

Model	Parameter Search (Layer Size)	Best Model	R_V^2	$\overline{\tau}_V$	MAE_V
Baseline [3]	400	-	0.29	0.08	0.16
LSTM (Single Layer)	128, 300, 400, 512, 700, 1024, 2048	700	0.39	0.10	0.15
LSTM (Multi Layer)	(700_128), (700_400), (2048_1024), (2048_1024_700)	(2048_1024)	0.29	0.17	0.15
AT (Single Layer)	32, 64, 128, 300, 400, 2048, 512, 700, 1024, 2048	400	**0.53**	0.08	0.16
AT (Multi Layer)	(300_128), (400_128), (1024_400), (2048_1024), (2048_1024_512)	(300_128)	0.51	0.04	0.16
BAT (Single Layer)	400, 1024, 2048	2048	0.16	0.13	0.15
BAT (Multi Layer)	(300_128), (400_128), (1024_400), (1024_512), (2048_1024), (2048_1024_512)	(400_128)	0.21	0.16	0.14
Transformer	1-Layer, 2-Layer, 4-Layer	1-Layer	0.12	0.11	0.10

3.2 Backward Attention Model (BAT)

A modified form of the traditional attention mechanism [22] is also used in the current work, called Backward Attention (BAT) models. In these models, for emotion prediction at each tth time frame, attention is distributed only among h_k hidden states, where, $1 \le k \le (t-1)$.

3.3 Transformers

The transformer architecture as proposed in Vaswani et. al. [2] is used in this work, with changes in the number of encoder side layers, as appropriate for the experiments. Attention is calculated as in equation 5.
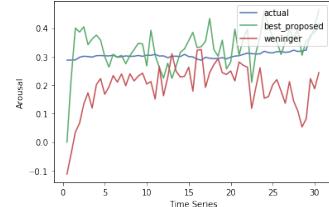
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (5)$$

where, Q, K and V are matrices representing the set of queries, keys and values respectively and d_k is the key dimension.
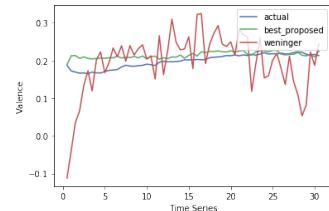
4. EXPERIMENTS

4.1 Data Description and Experimental Setup

We use the *1000 Songs for Emotional Analysis of Music* dataset [8] for all experiments. Of the thousand clips, the dataset provides arousal and valence annotations for only 744 clips, which are used as *ground truth* values. According to the dataset manual [8], arousal-valence continuous annotations for each song (second 15-45), with 2Hz sampling frequency are available in the dataset. We define each non-overlapping 500ms of the clips as *one music frame*. Thus, the last 30s or the last 61 frames of each clip are used for this work, since only those 61 emotion (arousal-valence) tags are available. 10-fold cross validation was used on the training and test sets. We used the Mean squared error (MSE) as the loss function. RMSProp, with the default learning rate of 0.001 was used for optimizing

(a) Arousal Comparison

(b) Valence Comparison

Figure 1: Dynamic Emotion Predictions for Clip 584
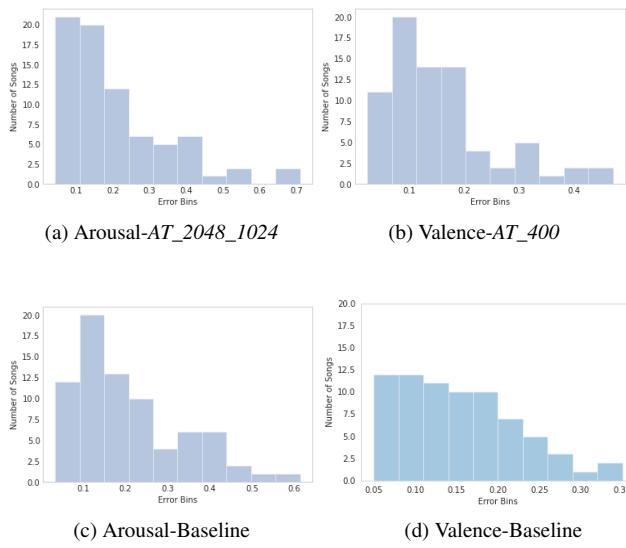
the loss with a batch size 20, and maximum 50 epochs. An early stopping strategy is also used, if validation error shows no improvement over 10^{-4} after 5 epochs, processing is stopped. Sequences are presented in random order during training. All hyper-parameters not explicitly mentioned here are left to their default values as in Tensorflow 1.14. The feature sets used for different experiments are described below.

4.1.1 ComPare Feature Set

The 2013 Computational Paralinguistics Evaluation (ComParE) tasks featureset [1], containing 6670 features is used for all experiments in sections 4.2 and 4.4. TUM's opensource *openSMILE* feature extractor [23] is used to extract the ComParE featureset for each frame of each clip. Standard normalization was performed on the extracted feature values before the experiments. So, each clip is characterised by 61 feature vectors, each of size 6670.

4.1.2 Other Feature Sets

In experiments reported in section 4.3, subsets of the Compare feature set [1] and some other features are explored. These features extracted using Librosa [24] are detailed here. The *Chroma(STFT+CQT)* features [24] consist of chroma values derived using both STFT analysis and constant-Q transform (CQT) analysis implementations. The *CQT on Audio clip* features [24] are derived from the core Spectrogram operations of Librosa [24] suitable for pitch-based signal analysis. The *Spectral Features* [24] denote the distributions of energy over a set of frequencies and are very important in many MIR analysis techniques. These consist of: Chroma(24), CENs (12) MFCC (20), RMS (1), Mel-scaled spectrogram (128), spectral centroid (1), spectral bandwidth (1), spectral contrast (7), spectral flatness (1), spectral roll-off (1), zero crossing rate (1). All clips were re-sampled to 44100 Hz before feature extraction. All features were extracted for non-overlapping frames of 500 ms each, corresponding to the available arousal-valence labels of the dataset.

(a) Arousal-*AT_2048_1024*   (b) Valence-*AT_400*

(c) Arousal-Baseline   (d) Valence-Baseline

Figure 2: Emotion Error Histograms over Validation Set

4.1.3 Metrics

The metrics used for reporting the results are Coefficient of determination (R^2), average Kendall's τ per song ($\overline{\tau}$)and mean absolute error (MAE). The determination coefficient (R^2) is a key output of regression analysis, which provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model. It can vary between 0 and 1. If a data set has n values marked ($y_1 \ldots y_n$), and each associated with a predicted value ($f_1 \ldots f_n$). So, R^2 is defined as $R^2 \equiv 1 - \frac{SS_{res}}{SS_{tot}}$ where, $SS_{res} = \sum_i (y_i - f_i)^2$ and $SS_{tot} = \sum_i (y_i - \overline{y})^2$, given $\overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$. Kendall's τ per song ($\overline{\tau}$) is a measure of how well the emotional profile of each song is captured by the regressor, as opposed to overall correlation. It measures the correspondence between two rankings. Values close to 1 indicate strong agreement, values close to -1 indicate strong disagreement. It is defined $\overline{\tau} = \frac{P-Q}{\sqrt{(P+Q+T)*(P+Q+U)}}$ where, P is the number of concordant pairs, Q the number of discordant pairs, T the number of ties only in target set ($y_1 \ldots y_n$), and U the number of ties only in predicted set ($f_1 \ldots f_n$). The mean absolute error (MAE) is given for reference. In the next section, we report the results of applying the proposed model for dynamic music emotion regression.

Baseline: It has been shown by Weninger et. al. [3, 6] that LSTMs can be used to produce good performance in emotion prediction, using the ComParE featureset. We try to reproduce their results using single layer LSTM-RNNs with hidden layer size of 400 units. These results are considered as *Baseline* in this work and are reported in the "Baseline" annotated rows of Table 1 and Table 2 for arousal and valence respectively.

4.2 Experiment 1: Model Selection

In the first set of experiments, we aim to find the best model for dynamic arousal and valence prediction, among the

Table 3: Feature Sets for Arousal Prediction

Features Used	# Features	Best Model	R^2_A	$\overline{\tau}_A$	MAE_A
Chroma(STFT+CQT)	24	AT_64	0.15	0.04	0.19
CQT on Audio clip	252	AT_64	0.45	0.06	0.17
Chroma+CQT	276	AT_64	0.57	0.07	0.14
Spectral Features	197	AT_64	**0.70**	0.03	0.12

Table 4: Feature Sets for Valence Prediction

Features Used	# Features	Best Model	R^2_V	$\overline{\tau}_V$	MAE_V
Chroma(STFT+CQT)	24	AT_64	0.01	0.002	0.09
CQT on Audio clip	252	AT_64	0.07	0.01	0.17
Chroma+CQT	276	AT_64	0.17	0.06	0.14
Spectral Features	197	AT_128	**0.35**	0.07	0.16
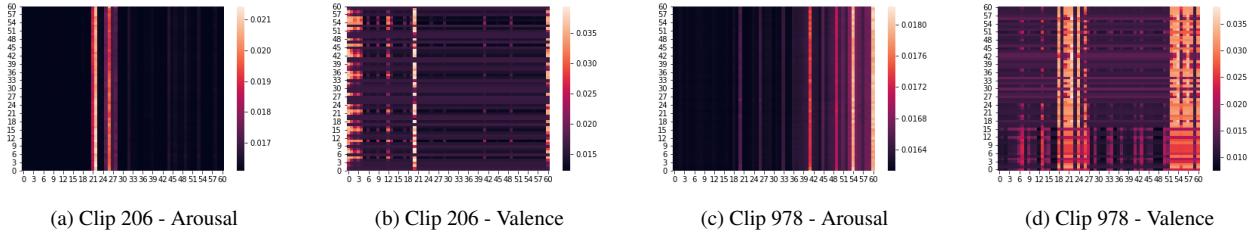
ones proposed in section 3. Accordingly, the models with attention (*AT*, *BAT*, *Transformers*) and without attention (*LSTM*) are executed with varying layer sizes and layer numbers. The findings for arousal and valence are reported in Table 1 and Table 2 respectively. For dynamic arousal prediction (Table 1) using the ComPare feature set [1] (sec 4.1.1), the best result is obtained with the multi-layer attention model *AT_2048_1024*. Comparable result is also obtained with single-layer attention model *AT_300*. The best model for dynamic valence prediction (Table 2) is found to be the single-layer attention model *AT_400*. Comparable result is also obtained with multi-layer attention model *AT_300_128*.

The following are observed from this experiment: a) The best prediction performances reported in this section are better than that reported by the baseline methods (sec 4.1.3). b) Among all the experiments conducted, *AT* models fare best in dynamic arousal-valence prediction using the full ComPare feature set [1]. c) The best single and multi layer *AT* models' performances are comparable. d) Performance for arousal prediction (R^2_A and $\overline{\tau}_A$) in general is much better than valence (R^2_V and $\overline{\tau}_V$) - across all models tested. Though performance with respect to MAE are comparable.
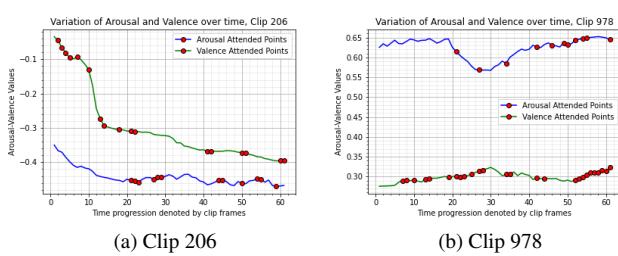
In the following subsections, we demonstrate an illustrative example of dynamic emotion prediction using a clip chosen at random, followed by an error analysis of the predictions by the best proposed models, over the validation set clips.

4.2.1 Illustrative examples

In this section, we demonstrate an illustrative example of dynamic emotion prediction pattern, with respect to ground truth (sec 4.1) and baseline (sec 4.1.3), using a clip chosen at random from the dataset [8]. The best models, *AT_2048_1024* for arousal and *AT_400* for valence, obtained in section 4.2 are used for dynamic (per 500 ms) arousal and valence prediction of music clip *584.mp3*. This is presented in Figure 1. Figure 1a and Figure 1b denote the time varying arousal and valence predictions respectively. In the figures, X-axis denote the time (in seconds), and the Y-axis denote arousal and valence values respectively. It is seen that the proposed best models follow the pattern of reported emotions more closely than baseline model.

(a) Clip 206 - Arousal   (b) Clip 206 - Valence   (c) Clip 978 - Arousal   (d) Clip 978 - Valence

Figure 3: Attention Maps using AT models. X-axis = attention points (500ms clip frames), Y-axis = prediction points (clip's progression through time)

(a) Clip 206   (b) Clip 978

Figure 4: Comparing attended frames with ground truth Emotion ratings of dataset [8]
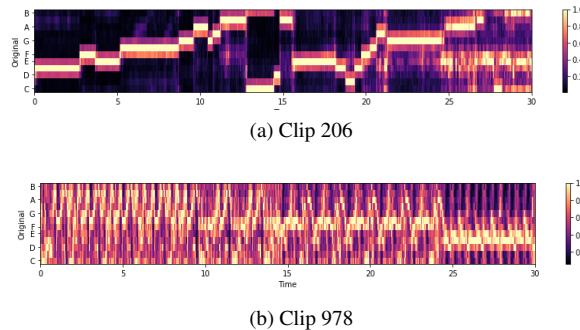
4.2.2 Errors Analysis

In this section we aim to observe patterns and biases in the best proposed models' (sec 4.1) emotion predictions, with respect to the baseline (sec 4.1.3). The respective predictions are utilized to group the validation set clips into error bins for this study. These are shown as histograms in figure 2. The X-axis denote the error bins of the models over the validation set clips. The Y-axis denote the number of clips of the validation set, which fall into each error bin. Comparing Figure 2a and Figure 2c, it can be seen that, for the proposed model, the number of clips with higher values of errors are less, in case of arousal. In case of valence, for the proposed model, almost all the clips are grouped into the error bins ≤ 0.05 (Figure 2b). Whereas for the baseline model ((Figure 2d)), a significant number of clips across bins are present.

4.3 Experiment 2: Exploring Other Feature Sets

In section 4.2, all the experiments use the full ComPare feature set [1]. Though it performs well in dynamic emotion prediction in music, it might be noted that it is generic, not music specific. It is large, which causes models to have large number of parameters. Also, there might be other relevant features, which might be used for this task, eg. Constant Q Transform features. In this section, we explore some smaller feature sets detailed in section 4.1.2, which might possibly produce similar or better results, over the same dataset [8], with the additional benefit of being smaller in size.

Single layer *AT* models were used to train on these new feature sets, since, it was observed in section 4.2 that they perform best and at par with multi layer models for emo-

(a) Clip 206

(b) Clip 978
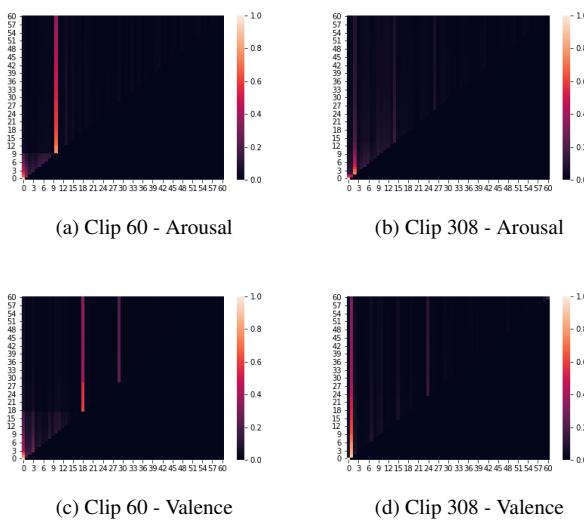
Figure 5: Chromagrams for Attention Map Analysis. X-axis = time (in seconds), Y-axis = Chroma. Vertical bars=Chroma intensities

tion prediction. The results are presented in Table 3 and Table 4 for arousal and valence respectively. For arousal (Table 3), it is observed that *AT_64* performs well, when using the *Spectral Features* set, with a R_A^2 comparable to the best model *AT_2048_1024* using full ComPare [1] feature set. It is evident that Chroma features alone have negligible contribution in arousal prediction. *CQT* set performs moderately. For valence prediction (Table 4) also, *Spectral features* set performs best among all. *CQT* set does not contribute much to valence prediction. Thus, we conclude that there might be a possibility of a smaller featureset for emotion prediction.

4.4 Attention Maps for Emotion Prediction

Attention maps demonstrate the relative importance of layer activations at different 2D spatial locations with respect to arousal and valence predictions. In this section, the best *AT* and *BAT* models are used to generate the attention maps for both arousal and valence, for some clips chosen at random from the dataset [8], presented in Figure 3 and Figure 6. These maps provide information about those frames of the clip, which are attended to during emotion prediction. This in turn can yield valuable insights into specific audio features of those frames, conducive to certain emotion perception. For all the maps, X-axis signifies the attention points, which are the 500 ms frames of the clip the model attends to. The Y-axis signifies the prediction points, the clip's progression through time. It is to be noted that these 61 frames in the maps, correspond to the

(a) Clip 60 - Arousal
(b) Clip 308 - Arousal
(c) Clip 60 - Valence
(d) Clip 308 - Valence

Figure 6: Attention Maps using BAT models. X-axis = attention points (500ms clip frames), Y-axis = prediction points (clip's progression through time)

last 30 seconds of each clip, as per the dataset [8]. So, the sth frame of a clip, is actually the $(15 + \frac{x-1}{2})^{th}$ second of the entire 45 second clip. The vertical bars on the right of each attention map give the attention weight values present in each map. The observations are discussed in the following subsections.

4.4.1 Attention Maps Using AT models

The attention maps for arousal and valence prediction, generated using *AT_2048_1024* and *AT_400*, for clips 128.mp3, 171.mp3, 206.mp3, and 978.mp3 from the dataset [8] are presented in Figure 3. Figure 3a and Figure 3b demonstrates the attention maps for arousal and valence prediction in clip 206.mp3 . As evident from the figure 3a, the model attends mostly to the clip frames 20-22, 26-28, and then again frames between 43-44, 49, 53-54 and 58 to predict arousal. From figure 3b, it is observed that the model attends to the frames 1-4, 6, 9, 12-13, 17, 20-21, 40-41, 49-50 and 59-61 to predict valence. Similar observations can be made about the other clips as well from Figure 3.

Observations: For arousal prediction, the model attends to comparatively fewer frames of the clip. These attended frames are observed to occur around 10 seconds (20 frames) after the clip has started. It can be concluded that the arousal generated in the later part of the music clip plays a significant role in determining the arousal perception of the entire clip. The attended frames have arousal ratings which are approximately average of all the arousal ratings for a particular clip. On the other hand, for valence prediction, attention is distributed across the clip, whenever there is perceptible change in valence ratings. Thus it can be concluded that reports of valence depends on momentary perception. Even small changes are registered. The attended frames have quite varied valence rating values within a particular clip.

For further investigation, we juxtapose our findings with a) The dynamic arousal and valence ratings provided by the dataset [8] - ground truth, given in Figure 4, and b) Chromagrams of the clips obtained using Librosa [24], presented in Figure 5. In each line graph of Figure 4, the X-axis denotes time frames, and Y-axis denotes the arousal and valence values.

It is to be noted here that the clips 206 and 978 are so chosen that they have significantly different arousal and valence ground truth values. In clip 206, the arousal values are lesser than the valence values. In clip 978, the reported arousal values are greater than the valence values. The blue and green lines denote arousal and valence respectively, the red dots highlight the time frames attended to by the *AT* models, as evident from Figure 3. In each subplot of Figure 5, the X-axis denotes time (in seconds), and the Y-axis denotes the Chroma. The vertical bars indicate the intensities of the Chroma. Figure 5a demonstrates the chromagram for clip 206.mp3.
Observations: For arousal prediction, the model attends on those frames with stable presence of higher notes (eg. A, B). For valence prediction, model attends all over the chroma bins, specially when there is a change in notes in the chroma sequence of the clip. Similar observations might be made from the other chromagrams as well.

4.4.2 Attention Maps using BAT models

The attention maps generated using the BAT models, *BAT_2048_1024* for arousal are presented in Figure 6. Figure 6a gives the attention map for arousal prediction in clip 60.mp3 of the dataset [8]. As evident from the figure, the attention of the model shifts continuously throughout the clip, as it progresses in time, though Segments 11-12 receive maximum attention overall. Similar trends are observed in Figure 6c as well, which represents the map for valence prediction for the same clip. Initially, the first few segments are attended to. As the clip progresses in time, the attention is shifted to later segments, with segments 18-19 and 29-30 being more prominent. As the clip progresses, the attention to initial segments reduces, rendering the lower right triangular region of the maps devoid of any attention traces.

5. CONCLUSION

We demonstrate that the state of the art models for continuous-time emotion prediction perform modestly, thus emphasizing the need for further research in this area. We have proposed an attentive LSTM based model which improves the state of the art performance significantly, on standard benchmark dataset with standard metrics. Further, we observe that a reduced, music-specific feature set achieves similar performance to the new state of the art model on arousal prediction, leading to much smaller models. Finally, we analyse attention maps for the full attention model to conclude that the model indeed attends to critical portions of the music in order to predict the dynamic emotions. We also observe that the nature of attention is different in case of arousal and valence prediction tasks.

6. REFERENCES

[1] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi *et al.*, "The interspeech 2013 computational paralinguistics challenge: social signals, conflict, emotion, autism," in *Proceedings INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France*, 2013.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[3] F. Weninger, F. Eyben, and B. Schuller, "On-line continuous-time music mood regression with deep recurrent neural networks," in *ICASSP*. IEEE, 2014, pp. 5412–5416.

[4] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, "Music emotion recognition: A state of the art review," in *ISMIR*, vol. 86, 2010, pp. 937–952.

[5] E. Coutinho, F. Weninger, B. W. Schuller, and K. R. Scherer, "The munich lstm-rnn approach to the mediaeval 2014 "emotion in music" task." in *MediaEval*, 2014.

[6] F. Weninger, F. Ringeval, E. Marchi, and B. W. Schuller, "Discriminatively trained recurrent neural networks for continuous dimensional emotion recognition from audio." in *IJCAI*, vol. 2016, 2016, pp. 2196–2202.

[7] F. Weninger, F. Eyben, and B. Schuller, "The tum approach to the mediaeval music emotion task using generic affective audio features," in *Proceedings MediaEval 2013 Workshop*, 2013.

[8] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha, and Y.-H. Yang, "1000 songs for emotional analysis of music," in *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia*. ACM, 2013, pp. 1–6.

[9] S. Giammusso, M. Guerriero, P. Lisena, E. Palumbo, and R. Troncy, "Predicting the emotion of playlists using track lyrics," *ISMIR, Late Breaking Session*, 2017.

[10] J. Fan, K. Tatar, M. Thorogood, and P. Pasquier, "Ranking-based emotion recognition for experimental music." in *ISMIR*, 2017, pp. 368–375.

[11] R. Delbouys, R. Hennequin, F. Piccoli, J. Royo-Letelier, and M. Moussallam, "Music mood detection based on audio and lyrics with deep neural net," *ISMIR*, pp. 370–375, 2018.

[12] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," *ISMIR*, 2011.

[13] Y. Song, S. Dixon, and M. T. Pearce, "Evaluation of musical features for emotion classification." in *ISMIR*. Citeseer, 2012, pp. 523–528.

[14] R. Panda, R. Malheiro, and R. P. Paiva, "Musical texture and expressivity features for music emotion recognition." in *ISMIR*, 2018, pp. 383–391.

[15] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[16] Y.-H. Yang, Y.-C. Lin, Y.-F. Su, and H. H. Chen, "A regression approach to music emotion recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 16, no. 2, pp. 448–457, 2008.

[17] S. Balke, M. Dorfer, L. Carvalho, A. Arzt, and G. Widmer, "Learning soft-attention models for tempo-invariant audio-sheet music retrieval," *ISMIR*, pp. 216–222, 2019.

[18] S. Gururani, M. Sharma, and A. Lerch, "An attention mechanism for musical instrument recognition," *ISMIR*, pp. 83–90, 2019.

[19] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," *ISMIR*, pp. 685–692, 2019.

[20] T.-P. Chen and L. Su, "Harmony transformer: Incorporating chord segmentation into harmony recognition," *ISMIR*, pp. 259–267, 2019.

[21] J. Park, K. Choi, S. Jeon, D. Kim, and J. Park, "A bi-directional transformer for musical chord recognition," *ISMIR*, pp. 620–627, 2019.

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[23] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: the munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 1459–1462.

[24] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

PANDEMICS, MUSIC, AND COLLECTIVE SENTIMENT: EVIDENCE FROM THE OUTBREAK OF COVID-19

Meijun Liu[1]   **Eva Zangerle**[2]   **Xiao Hu**[1]
Alessandro Melchiorre[3,4]   **Markus Schedl**[3,4]

[1] The University of Hong Kong, Hong Kong SAR, China
[2] University of Innsbruck, Innsbruck, Austria
[3] Johannes Kepler University Linz, Austria
[4] Linz Institute of Technology (LIT), Linz, Austria

`Correspondence:xiaoxhu@hku.hk`

ABSTRACT

The COVID-19 pandemic causes a massive global health crisis and produces substantial economic and social distress, which in turn may cause stress and anxiety among people. Real-world events play a key role in shaping collective sentiment in a society. As people listen to music daily everywhere in the world, the sentiment of music being listened to can reflect the mood of the listeners and serve as a measure of collective sentiment. However, the exact relationship between real-world events and the sentiment of music being listened to is not clear. Driven by this research gap, we use the unexpected outbreak of COVID-19 as a natural experiment to explore how users' sentiment of music being listened to evolves before and during the outbreak of the pandemic. We employ causal inference approaches on an extended version of the LFM-1b dataset of listening events shared on Last.fm, to examine the impact of the pandemic on the sentiment of music listened to by users in different countries. We find that, after the first COVID-19 case in a country was confirmed, the sentiment of artists users listened to becomes more negative. This negative effect is pronounced for males while females' music emotion is less influenced by the outbreak of the COVID-19 pandemic. We further find a negative association between the number of new weekly COVID-19 cases and users' music sentiment. Our results provide empirical evidence that public sentiment can be monitored based on collective music listening behaviors, which can contribute to research in related disciplines.

1. INTRODUCTION

Music listening has various functions in people's daily lives, especially in terms of psychological aspects. Mood management and regulation are two major psychological

uses of music listening [31, 41–43] and have been widely investigated in previous studies [18,21,40,44]. In the literature, there are mainly two kinds of music mood being addressed. One is "expressed" mood, the other is "induced" mood. The former refers to the mood that is intended to be expressed by a piece of music whereas the latter refers to listeners' emotional state or feeling induced by listening to a music piece or situations with which people associate a song [5, 58]. Both kinds of music mood have been extensively studies in the field of Music Information Retrieval (MIR) in recent decades [59]. It is well known that people often search and select music based on the mood of music that fits their current (emotional) needs [12] and it has been found that people's mood is responsive to events in daily life [10, 25]. However, the influence of real-world events, such as human or natural disasters, on users' mood-based music selection remains less explored, especially in the scale of collective choices of users in a society.

As of May 4, 2020, with 3.57 million cases and over 250 thousand deaths reported, COVID-19 has posed a severe threat to public health. To contain the virus, multiple measures have been taken, such as city lock-downs, social distancing, travel restrictions, and university closures, disrupting people's daily life and routine. High unemployment and the economic damages caused by COVID-19 make people suffer from increasing economic stress [17, 24]. Causing economic and social pressure, the pandemic is putting enormous stress on all of us and might trigger feelings of distress and anxiety [8, 56]. Large-scale disasters are often accompanied by increases in depression, a broad range of other psychological stress and behavioral disorders [16, 20, 34]. Investigating whether and the extent to which collective sentiment inferred from music listening behavior is influenced by the pandemic can not only deepen our understanding of the relationship between real-world events and users' sentiment reflected by the music they listen to, but also contribute to mitigating negative mental health impacts caused by the pandemic, and help people adapt, and be resilient during distress times.

Inspired by the research gap and driven by the emerging pandemic, this study aims to explore the association between real-world events and the sentiment of music being listened to based on users' music listening history at

a large scale. In this study, we use the term *users' music sentiment* to refer to the aggregated collective sentiment of music listened to by a population of users. We use the difference in differences (hereafter DD) method [2], a widely used causal inference approach, and base our analysis on an extension of the LFM-1b dataset [45]. This dataset provides us with a substantial set of listening histories of users from around the world as well as user-generated tags that we derive sentiment information from. We use the unanticipated outbreak of COVID-19 as a natural experiment to investigate the causality between pandemics and collective sentiment reflected by music people choose to listen to.

Specifically, we aim to answer the following research questions:

RQ1: How did the first COVID-19 case in a country affect the music sentiment of users in that country?

RQ2: How did the number of new COVID-19 cases and hence, the spread of the disease in a country affect the music sentiment of users in that country?

Using the DD model, our analyses can reveal causal relationship between real-world events, a pandemic in our case, and users' music sentiment. Results of this study could shed light on designs of music recommenders to be more sensitive to real-life events. More importantly, our results provide convincing evidence that collective sentiment is hampered by the pandemic. Beyond the field of MIR, findings of this study could provide empirical evidence on the extent to which public sentiment could be monitored by music listening behaviors of a concerned population. This could contribute to science in related disciplines such as social psychology, sociology, and journalism.

2. RELATED WORK

Two streams of research show a possible relationship between real-world events and users' sentiment of music being listened to, while the causal pathway remains unexplored. Links between real-worlds events and people's mood have been widely documented in previous research [50, 51]. However, to our best knowledge, there has been no study on the association between real-world events and users' collective sentiment as reflected by music they listen to.

2.1 Real-world events linked to mood

Mood indicates a set of transient, fluctuating affective states in terms of individuals' feelings [4, 26]. A variety of factors are related to individuals' mood state, including personality [14], ongoing events, experiences, or the environmental milieu [57]. Real-world events or the daily experiences, conceptualized as situational or contextual factors, have long been recognized as important determinants of daily mood in abundant research [50, 51].

Pioneering studies on the association between daily events and mood are often based on self-reported data on the same day. For example, Lewinsohn and colleagues [29] found a negative correlation between the number of pleasant events and depression by exploiting subjects' self-reported data on daily experiences and self-ratings of mood

states. Similar results have been found by Rehm [36] and Stone [49, 51]. In particular, Stone found a same-day association between major daily events and mood, based on self-reported events and moods of 50 men [51]. However, studies based on self-reported data in the same-day context are not sufficient to demonstrate a causal relationship because mood could impact the reliability of event reporting [9]. More convincing evidence of the causal relationship between real-world events and mood could be provided based on longitudinally self-reported data on events and moods over a period of time.

The occurrences of events have an impact on changes in positive and negative mood, respectively. Pleasant events are normally related to positive mood, such as exercise [13, 28], family, friends, and leisure time [50] and other pleasant daily events [27]. There is also evidence of the links between various negative daily events and negative mood, such as interpersonal conflict [6], negative interpersonal interactions [38], stressful work-related events [38], daily hassle [11], undesirable daily events [1], and other daily stressors [15, 52]. For instance, using the data extracted from diaries, Zuckerman demonstrated that young adults reported significantly higher levels of depression when they reported interpersonal conflicts [7].

Current research on the effect of major real-world events on mood is limited [47]. The majority of previous work is built on the self-reported data of small samples. On one hand, it is unclear whether small samples were sufficiently representative; on the other hand, the self-rating of mood state can be subjective.

2.2 Real-world events and music listening

Schedl et al. [46] took an initial step to explore the correlation between real-world events and music consumption behavior by leveraging listening events from Last.fm and world-wide events from Google Trends. Performing an intervention time series analysis, the authors found that changes in listening behavior might be correlated to real-world events. However, only two variables were taken into account: the number of events identified by Google Trends and the absolute number of listening events. Whether or not real-world events are linked to users' music sentiment remains uncovered.

Inspired by this research gap , this study aims to examine the association between real-world events (the COVID-19 pandemic) and the collective sentiment reflected in people's music listening behaviors. Based on (an extension of) the LFM-1b dataset, one of the largest datasets of music listening histories available to date, this study examines the relationship between the outbreak of COVID-19 and the collective music sentiment.

3. DATA AND METHODOLOGY

In the following, we present the data utilized and subsequently, we describe the methods underlying our analyses.

3.1 Data

We built upon the LFM-1b dataset [45] to explore the relationship between the outbreak of the COVID-19 pandemic

and users' music sentiment. More precisely, we gathered the listening records of users in the LFM-1b dataset between November 1, 2019, one month before the first COVID-19 case in the world was confirmed in China [1], and March 27, 2020. Even though the outbreaks are still ongoing in many places as of the writing of this paper, this time frame captures the times of first confirmed COVID-19 cases in all countries involved in this study. We obtained more than 28 million listening events shared during this period of time, 21 weeks in total, generated by 12,278 Last.fm users from 40 countries from the LFM-1b dataset. From the Our World in Data website, [2] we collected data on confirmed COVID-19 new cases by date for these countries. In particular, we identified the date of the first confirmed COVID-19 case in each country and define it as the date when the COVID-19 outbreak started in this country.

3.2 Measuring music sentiment

In Last.fm, tracks and artists are associated with tags created by users. To capture the sentiment of music listened to by our users, we collect the artists whose music pieces are included in our dataset, that is, have been listened to by included users during the concerned period of time. Although tracks also have tags, track-level tags indicating sentiment might be too sparse as there are a much larger number of tracks than artists. Therefore, using artist sentiment (as opposed to track sentiment) provides us with a substantially larger set of tags with sentiment. The sentiment of an artist can then be calculated by methods described in the next paragraphs.

We capture the sentiment values assigned to all artists by crawling the user-created tags assigned to those artists from Last.fm. Subsequently, we follow Zangerle et al.'s [60] approach for the computation of sentiment values based on the tags. Specifically, we utilize sentiment lexica, a widely used unsupervised sentiment detection method for the extraction of sentiment information from the tags. A sentiment lexicon is a list of words, where each word is assigned a sentiment value (e.g., on a scale from 0 to 1, describing a range of sentiments from negative to positive) [54]. We rely on a set of widely used dictionaries that have been shown to provide the best coverage and accuracy according to Ribeiro's benchmark of sentiment dictionaries [39]: AFINN [32], Opinion Lexicon [22], SentiStrength [55], and Vader [23]. The different lexica provide different notions of polarity and strength of polarity. Hence, we normalize the set of sentiments to a range between 0 and 1 by using linear min-max feature scaling.

The following steps are taken to perform the sentiment computation based on these dictionaries. First, we employ whitespace as delimiters to tokenize the tags as tags may consist of several words such as in "nothing left to fear". We then use the tokens as the input to the matching step between tokens and dictionaries. To do so, we apply lemmatization to the tokens contained in the sentiment dictionaries (utilizing the lemmatization method of

the Python's NLTK package). Next, we match each tag token to the set of dictionaries. For each matched token, we extract the assigned sentiment value. In the case that a token matches multiple entries of the dictionaries (i.e., if it is contained in multiple dictionaries), we utilize the arithmetic mean of those values as the sentiment value of the token. After having computed the sentiment value for each token, we assign the tag the mean of the sentiment values of all tokens contained in the original tag.

The sentiment of an artist is then defined as the weighted average sentiment values of the tags assigned to the artist, as shown in Equation 1:

$$AS_j = \sum_{i=1}^{n} v_i \cdot \frac{f_i}{F} \qquad (1)$$

where (AS_j) stands for the sentiment value of the jth artist; v_i denotes the sentiment value of the ith tag for artist j; f_i refers to the frequency of tag i for this artist, and F denotes the total frequency of tags for the artist.

We consider this a reasonable estimate as the tags applied to each artist reflect the collective sentiment a large number of listeners had towards the artist over a long period of time. It can thus be assumed that tags are not biased by certain (subgroups of) users or any short term events.

3.3 Difference in differences approach

To capture the relationship between the outbreak of the COVID-19 pandemic and the sentiment of music listened to by users, the DD model is used [2]. This method measures the differential effect of certain changes on the dependent variable of treatment groups versus that of control groups [19]. One distinct advantage of the DD model is that it can disclose causality. In this study, the concerned change is the outbreak of COVID-19. The dependent variable is the users' music sentiment extracted from their listening behaviors (cf. Section 3.2). In this study, for a given period of observation (i.e., a specific week), users in countries with confirmed COVID-19 cases are in the treatment group, whereas users in countries without confirmed cases form the control group.

Besides, to control for users' other characteristics that may affect their sentiment of music, we introduce fixed effects regression, the major method for regression analysis of panel data. It is an extension of multiple regression that exploits panel data to control for variables that differ across entities but are constant over time [48]. This allows controlling unobserved changes during our observation period. We incorporate user gender, age, and country as control variables.

The DD model employed for the COVID-19 pandemic is shown in Equation 2.

$$Y_{it} = \alpha + \beta_1 (I_{it}) + \beta_2 Gender_i + \\ \beta_3 Age_i + C_i + D_t + \epsilon_{it} \qquad (2)$$

where Y is a dependent variable, I are the independent variables, it indicates observations for user i in week t and ϵ_{it} indicates the error term in the equation. We detail the utilized dependent and independent variables in

---

[2] https://ourworldindata.org/covid-cases

Sections 3.4 and 3.5, respectively. Furthermore, the equation also contains controls and fixed-effect variables as described above. Specifically, fixed-country effects (C_i in the equation) allow controlling the time-invariant, country-specific characteristics of users as e.g., socio-cultural background may be related to users' artist listening behavior. The observable changes of global events that might influence users' listening behavior worldwide can be controlled by adding a week fixed effect (D_t). Controls of gender and age of users are also included in Equation 2.

The coefficient for the dependent variable is an effective difference-in-differences estimate of the average impact of the COVID-19 outbreak on the sentiment of music listened to by users. The effect of the outbreak of COVID-19 on user's music sentiment in a week is estimated using the Ordinary Least Squares method, which is appropriate for the continuous dependent variable.

3.4 Dependent variables

We use two dependent variables to build two models, by which we can cross check robustness of the results. Both are on user's music sentiment calculated based on the measure of music sentiment described in Section 3.2. The first dependent variable is the average sentiment of artists a user listened to (hereafter referred to as *USE*) in a given week. The values of USE range from 0 to 1. The larger the USE value is, the more positive the user's music sentiment. USE captures the average sentiment of artists the user listened to in each day, weighted by the relative consumption frequency of the artists, as shown by Equation 3:

$$USE = \sum_{i=1}^{n} AS_i \cdot \frac{c_i}{C} \qquad (3)$$

where AS_i denotes the sentiment value of the ith artist the user listened to on the given day; c_i indicates the playcount (number of times the user listened to artist i); and C denotes the total playcount of the user on that day. Based on daily USE values, we then aggregate this user's daily USE to the weekly level through averaging. The weekly average USE of a user is the first dependent variable.

The second dependent variable we propose to use is whether the music sentiment of a user in a given week is extremely positive (i.e., whether or not the weekly USE value of a user is in the 90th percentile of USE values of all users in our dataset). We refer to this variable as *POS*.

3.5 Independent variables

For our DD analyses, we propose to use the outbreak of COVID-19, denoted as *COVID-19* hereafter, as the independent variable. Specifically, it is a binary variable indicating whether or not the first COVID-19 case has been confirmed in a given country in a given week. The variable is 0 for observations (a.k.a samples) before the outbreak in a country and 1 for observations/samples after the outbreak. In addition, to explore the association between the number of new COVID-19 cases and user's music sentiment, we also include the weekly number of new COVID-19 cases in a country as the second independent variable, denoted as *COVID19 cases*.

Variable	Mean	Std. Dev.	Min	Max
USE	0.3332	0.0782	0	0.9813
POS	0.0976	0.2967	0	1
COVID-19	0.2038	0.4028	0	1
COVID-19 cases	0.8344	1.9249	0	10.0542
Age	32.3674	13.1637	6	126
Gender	0.1731	0.3783	0	1

Table 1. Descriptive statistics of variables modeled in this study; N =184,277 (weekly observations of 12,278 users for 21 weeks); gender indicates male (0) or female (1).
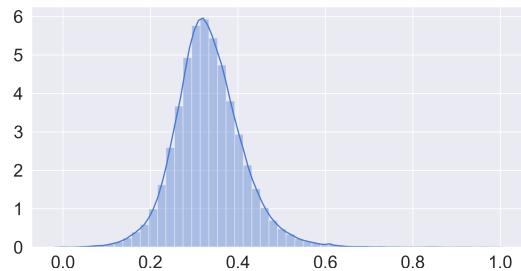
Table 1 summarizes the descriptive statistics of dependent and independent variables in this study. The distribution of USE for the samples is presented in Figure 1.

3.6 Interaction effect between the COVID-19 and gender

To explore whether there is a gender difference regarding the effect of the COVID-19 pandemic on user' music sentiment, we generate two interaction terms between gender and the independent variables respectively, *COVID#Gender* and *Case#Gender*. Specifically, we multiply COVID-19 and COVID-19 cases with gender respectively and add each of the interaction terms to Equation 2 for separate modeling.

Besides, after running regression models on all sampled users, we conduct DD analysis for females and males separately, to further examine possible gender differences.

In sum, for each dependent variable (USE, POS), we run four regression models on each independent variable (COVID-19 and COVID19 cases): the original one indicated by Equation 2, the one with added interaction term, and two on female and male users only.

Figure 1. Distribution of USE (average sentiment of artists the user listened to), with USE on x-axis and probability density on y-axis.

4. RESULTS

4.1 COVID-19 outbreak and users' music sentiment

The effects of the independent variable, COVID-19, on dependent variable, weekly USE (user's music sentiment) are shown in columns 1 to 4 in Table 2 (Panel 1). We observe a significant and negative association between the outbreak of COVID-19 in a country and users' music sentiment (USE): The coefficient on COVID-19 for USE is -0.002 ($p < 0.01$, t-test) in column 1 of Table 2 (Panel 1),

suggesting that the sentiment values of the users in countries with COVID-19 outbreak is 0.002 lower than that of the users in countries where COVID-19 has not appeared yet. In other words, compared to sentiment of artists listened to by users in countries without COVID-19, that of users exposed to this pandemic is more negative.

In the USE model, the coefficient for Gender in column 1 (Panel 1) is 0.014 at the significance level of 0.01, indicating that the sentiment of artists listened to by females is more positive than that of males. The coefficient for age is significantly negative in column 1 (Panel 1), suggesting that younger users' average weekly music sentiment is more positive than older users, though the difference is small and nearly close to zero.
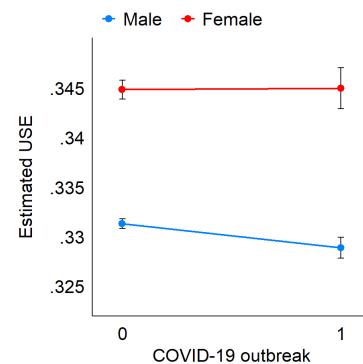
Gender differences are also found in the model of USE with the intersection term (column 2, Panel 1). With a value of 0.0025 ($p < 0.05$), the coefficient on the interaction term between COVID-19 outbreak and gender, COVID#Gender, is significantly positive, indicating that females are less influenced by the pandemic, as compared to their male peers. In column 2 of Table 2 (Panel 1). The estimated USE for females and males before and after the outbreak of COVID-19 is shown in Figure 2. The figure suggests a very slight decrease in USE after the outbreak of COVID-19 for females, as compared to a clear decline in users' USE for males. The regression results are consistent based on the models that only include either female (column 3, Panel 1) or male users (column 4, Panel 1): there is no significant impact of COVID-19 outbreak on USE in the model of females (column 3), while there is a significantly negative effect for males as shown in column 4 of Table 2 (Panel 1).

The results of the analyses of the POS variable (i.e., whether music sentiment of users is extremely positive) provide consistent findings that user's music sentiment turns more negative after the outbreak of COVID-19. Column 5 of Table 2 (Panel 1) reveals that there is a significantly negative relationship between the outbreak of COVID-19 and the positiveness of users' music sentiment (POS). After the outbreak of COVID-19, the probability that users' weekly averaged music sentiment reaches extremely positive values decreased by nearly 2.8% ($p < 0.01$). The coefficient on the interaction term between COVID-19 and gender is not significant (column 6, Panel 1). Columns 7 and 8 in Panel 1 both show that the likelihood that the weekly USE for females and males get extremely positive declines by 1.6% ($p < 0.05$) and 3.2% ($p < 0.01$), respectively.

4.2 COVID-19 cases and users' music sentiment

Users' sentiment of music is negatively associated with the number of COVID-19 cases in a country. In other words, in a country where a larger number of people are infected with COVID-19 in a given week, the weekly average sentiment of artists the user listened to becomes more negative. From column 1 in Table 2 (Panel 2), one unit growth in COVID-19 new cases in the week leads to a decrease of 0.05 in weekly USE of users , i.e., 64.10% (i.e., 0.05 divided by standard deviation of USE) standard deviation of

users' USE. Again, we find the negative effect of COVID-19 cases only for males, as shown in column 4 of Table 2 (Panel 2). For the dependent variable POS, results in column 5 of Table 2 (Panel 2) shows that, one unit increase in COVID-19 new cases in the week is related to a 67% lower probability that the weekly user sentiment reaches extremely positive values. There is no gender difference regarding the effect of new COVID-19 cases on users' POS as the interaction term between new COVID-19 cases and Gender is not significant (column 6, Panel 2), and the coefficients of COVID-19 cases on POS for females and males are very close, as shown in columns 7 and 8 of Table 2 (Panel 2).

Figure 2. Estimated interaction effect between COVID-19 outbreak and gender.

5. DISCUSSION

Our results show that, after the outbreak of COVID-19 in a country, the weekly sentiment of artists users listened to turns more negative. These results are robust regardless of whether the dependent variable is the weekly average sentiment value of users' artist listening (USE) or whether or not users' music sentiment is extremely positive (POS). The findings hence show significant causality between the real-world event (i.e., the COVID-19 pandemic) and users' music sentiment. We find that, after the first COVID-19 case in a country was confirmed, the weekly sentiment of artists users listened to decreased by 0.002, nearly 3% of standard deviation of users' USE (column 1 Panel 1 in Table 2. This negative effect is pronounced for males, whereas females' music emotion is less influenced by the outbreak of the COVID-19 pandemic. We further find a negative association between the number of COVID-19 new cases in the current week and users' music sentiment: one unit growth in COVID-19 new cases in a given week brings a decrease of 0.05, i.e., 64.10% of standard deviation of user's weekly music sentiment (column 1, Panel 2 in Table 2. One unit increase in COVID-19 new cases in a week is related to a 67% lower probability that user's weekly music sentiment reaches extremely positive (column 5, Panel 2 in Table 2).

We provide convincing evidence to show a negative impact of public health crisis on collective sentiment using the unexpected outbreak of COVID-19 pandemic as

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Panel 1								
Model	**USE**				**POS**			
	All	All	Female	Male	All	All	Female	Male
COVID19	-0.0020***	-0.0024***	0.0008	-0.0028***	-0.0284***	-0.0290***	-0.0161***	-0.0316***
	(0.0006)	(0.0006)	(0.0015)	(0.0007)	(0.0023)	(0.0024)	(0.0062)	(0.0025)
Gender	0.0141***	0.0136***			0.0335***	0.0329***		
	(0.0005)	(0.0005)			(0.0018)	(0.0020)		
COVID#Gender		0.0025**				0.0036		
		(0.0012)				(0.0047)		
Age	-0.0000**	-0.0000**	0.0001**	-0.0000***	-0.0001	-0.0001	0.0003*	-0.0001**
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0001)	(0.0001)	(0.0002)	(0.0001)
Constant	0.3321***	0.3322***	0.3428***	0.3328***	0.0999***	0.1000***	0.1199***	0.1025***
	(0.0005)	(0.0005)	(0.0013)	(0.0005)	(0.0020)	(0.0020)	(0.0053)	(0.0020)
Observations	184,277	184,277	31,894	152,383	184,277	184,277	31,894	"152,383"
R-squared	0.030	0.030	0.036	0.025	0.014	0.014	0.020	0.011
Panel 2								
Model	**USE**				**POS**			
	All	All	Female	Male	All	All	Female	Male
COVID19 cases	-0.0005**	-0.0005**	-0.0003	-0.0006**	-0.0067***	-0.0065***	-0.0069***	-0.0068***
	(0.0002)	(0.0002)	(0.0006)	(0.0002)	(0.0008)	(0.0009)	(0.0023)	(0.0009)
Gender	0.0141***	0.0141***			0.0340***	0.0351***		
	(0.0005)	(0.0005)			(0.0018)	(0.0020)		
Case#Gender		0.0001				-0.0014		
		(0.0003)				(0.0010)		
Age	-0.0000**	-0.0000**	0.0001**	-0.0000***	-0.0001	-0.0001	0.0003*	-0.0001**
	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0001)	(0.0001)	(0.0002)	(0.0001)
Constant	0.3321***	0.3322***	0.3432***	0.3327***	0.0996***	0.0995***	0.1224***	0.1017***
	(0.0005)	(0.0005)	(0.0013)	(0.0006)	(0.0020)	(0.0020)	(0.0055)	(0.0021)
Observations	184,277	184,277	31,894	152,383	184,277	184,277	31,894	152,383
R-squared	0.030	0.030	0.036	0.025	0.013	0.013	0.020	0.011

Table 2. Estimated effect of the outbreak of COVID-19 (Panel 1) and number of new COVID-19 cases (Panel 2) on users' music sentiment. Robust standard errors are in parentheses; gender indicates female (1) or male (0). All models employ country and week-fixed effects (*** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$). For columns (1) to (4), the dependent variable is USE. From columns (5) to (8), the dependent variable is POS. Columns (1) and (5) report the regression results based on the models that include control variables, i.e., gender and age for all sampled users; columns (2) and (6) indicates the regression results based on the models with control variables and an interaction term between COVID-19/Case and gender; columns (3), (4) and columns (7), (8) show the results based on the models with either female or male users.

an example that ensures a causal relationship through the different of differences method. Wide spread outbreaks of infectious diseases often cause psychological distress and symptoms of mental illness [3, 35]. Our results suggest that the average sentiment of artists users listened to becomes more negative after the first COVID-19 case was confirmed in a country, which is consistent with reported detrimental effect of pandemics on public mental health [30, 33, 37]. We further find that in countries where more new COVID-19 are confirmed, users' music sentiment turns more negative. During this acute public health crisis, people might experience fears of infection and worry about the pandemic's consequences. The disruption of daily routines and the social isolation imposed by the "stay at home orders" adopted in many countries also cause compounding personal stress and anxiety. The unemployment and financial losses caused by physical isolation may also strengthen feelings of distress. These negative emotions, as shown by our results, have been reflected by the sentiment of artists users listened to during the pandemic.We find that females' music sentiment is less influenced by the outbreak of COVID-19, which is probably due to the gender differences in socio-economic roles that

females might be less financially stressed than males [53].

6. CONCLUSION

This study applies the difference of differences method to find that, after the first COVID-19 case in a country was confirmed, the weekly sentiment of artists users listened to became more negative. This negative effect is pronounced for males. We further find a negative association between the number of new COVID-19 cases in a week and users' music sentiment. The results could provide useful implications for the design of music recommendation systems during public health crises or other disasters, which could help manage users' mood, enhance mental health, and mitigate negative psychological impacts of pandemics. The findings also provide empirical evidence that large-scale aggregation of music listening data can help monitor collective sentiment of listener populations.

7. ACKNOWLEDGEMENT

8. REFERENCES

[1] Glenn Affleck, Howard Tennen, Susan Urrows, and Pamela Higgins. Person and contextual features of daily stress reactivity: individual differences in relations of undesirable daily events with mood disturbance and chronic pain intensity. *Journal of Personality and Social Psychology*, 66(2):329, 1994.

[2] Joshua D Angrist and Jörn-Steffen Pischke. *Mostly harmless econometrics: An empiricist's companion.* Princeton university press, 2008.

[3] Yanping Bao, Yankun Sun, Shiqiu Meng, Jie Shi, and Lin Lu. 2019-ncov epidemic: address mental health care to empower society. *The Lancet*, 395(10224):e37–e38, 2020.

[4] Bonnie G Berger and Robert W Motl. Exercise and mood: A selective review and synthesis of research employing the profile of mood states. *Journal of applied sport psychology*, 12(1):69–92, 2000.

[5] Kerstin Bischoff, Claudiu S Firan, Raluca Paiu, Wolfgang Nejdl, Cyril Laurier, and Mohamed Sordo. Music mood and theme classification-a hybrid approach. In *ISMIR*, pages 657–662, 2009.

[6] Niall Bolger, Anita DeLongis, Ronald C Kessler, and Elizabeth A Schilling. Effects of daily stress on negative mood. *Journal of personality and social psychology*, 57(5):808, 1989.

[7] Niall Bolger and Adam Zuckerman. A framework for studying personality in the stress process. *Journal of personality and social psychology*, 69(5):890, 1995.

[8] Wenjun Cao, Ziwei Fang, Guoqiang Hou, Mei Han, Xinrong Xu, Jiaxin Dong, and Jianzhong Zheng. The psychological impact of the covid-19 epidemic on college students in china. *Psychiatry research*, page 112934, 2020.

[9] David M Clark and John D Teasdale. Diurnal variation in clinical depression and accessibility of memories of positive and negative experiences. *Journal of abnormal psychology*, 91(2):87, 1982.

[10] Lee Anna Clark and David Watson. Mood and the mundane: Relations between daily life events and self-reported mood. *Journal of personality and social psychology*, 54(2):296, 1988.

[11] Anita DeLongis, Susan Folkman, and Richard S Lazarus. The impact of daily stress on health and mood: psychological and social resources as mediators. *Journal of personality and social psychology*, 54(3):486, 1988.

[12] J Stephen Downie and Sally Jo Cunningham. Toward a theory of music information retrieval queries: System design implications. 2002.

[13] Kathryn M Fritz and Patrick J O'Connor. Acute exercise improves mood and motivation in young men with adhd symptoms. *Medicine and science in sports and exercise*, 48(6):1153–1160, 2016.

[14] Scott K Fry and Bernd G Heubeck. The effects of personality and situational variables on mood states during outward bound wilderness courses: An exploration. *Personality and individual differences*, 24(5):649–659, 1998.

[15] Shelly L Gable, Harry T Reis, and Andrew J Elliot. Behavioral activation and inhibition in everyday life. *Journal of personality and social psychology*, 78(6):1135, 2000.

[16] Sandro Galea, Arijit Nandi, and David Vlahov. The epidemiology of post-traumatic stress disorder after disasters. *Epidemiologic reviews*, 27(1):78–91, 2005.

[17] Anuj Gangopadhyaya and A Bowen Garrett. Unemployment, health insurance, and the covid-19 recession. *Health Insurance, and the COVID-19 Recession (April 1, 2020)*, 2020.

[18] Walter Gantz, Howard M Gartenberg, Martin L Pearson, and Seth O Schiller. Gratifications and expectations associated with pop music among adolescents. *Popular Music & Society*, 6(1):81–89, 1978.

[19] Paul J Gertler, Sebastian Martinez, Patrick Premand, Laura B Rawlings, and Christel MJ Vermeersch. *Impact evaluation in practice.* The World Bank, 2016.

[20] Emily Goldmann and Sandro Galea. Mental health consequences of disasters. *Annual review of public health*, 35:169–183, 2014.

[21] David J Hargreaves and Adrian C North. The functions of music in everyday life: Redefining the social in music psychology. *Psychology of music*, 27(1):71–83, 1999.

[22] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[23] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[24] Wolfram Kawohl and Carlos Nordt. Covid-19, unemployment, and suicide. *The Lancet Psychiatry*, 7(5):389–390, 2020.

[25] Peter Kuppens, Zita Oravecz, and Francis Tuerlinckx. Feelings change: Accounting for individual differences in the temporal dynamics of affect. *Journal of personality and social psychology*, 99(6):1042, 2010.

[26] Andrew M Lane and Peter C Terry. The nature of mood: Development of a conceptual model with a focus on depression. *Journal of applied sport psychology*, 12(1):16–33, 2000.

[27] Christopher A Langston. Capitalizing on and coping with daily-life events: Expressive responses to positive events. *Journal of Personality and Social Psychology*, 67(6):1112, 1994.

[28] Arthur R LaPerriere, Michael H Antoni, Neil Schneiderman, Gail Ironson, Nancy Klimas, Panagiota Caralis, and Mary Ann Fletcher. Exercise intervention attenuates emotional distress and natural killer cell decrements following notification of positive serologic status for hiv-1. *Biofeedback and Self-regulation*, 15(3):229–242, 1990.

[29] Peter M Lewinsohn and Julian Libet. Pleasant events, activity schedules, and depressions. *Journal of abnormal psychology*, 79(3):291, 1972.

[30] Nianqi Liu, Fan Zhang, Cun Wei, Yanpu Jia, Zhilei Shang, Luna Sun, Lili Wu, Zhuoer Sun, Yaoguang Zhou, Yan Wang, et al. Prevalence and predictors of ptss during covid-19 outbreak in china hardest-hit areas: Gender differences matter. *Psychiatry research*, page 112921, 2020.

[31] Adam J Lonsdale and Adrian C North. Why do we listen to music? a uses and gratifications analysis. *British Journal of Psychology*, 102(1):108–134, 2011.

[32] Finn Årup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*, 2011.

[33] Paul C Perrin, O Lee McCabe, George S Everly, and Jonathan M Links. Preparing for an influenza pandemic: mental health considerations. *Prehospital and disaster medicine*, 24(3):223–230, 2009.

[34] Robert H Pietrzak, Melissa Tracy, Sandro Galea, Dean G Kilpatrick, Kenneth J Ruggiero, Jessica L Hamblen, Steven M Southwick, and Fran H Norris. Resilience in the face of disaster: prevalence and longitudinal course of mental disorders following hurricane ike. *PLoS One*, 7(6), 2012.

[35] Jianyin Qiu, Bin Shen, Min Zhao, Zhen Wang, Bin Xie, and Yifeng Xu. A nationwide survey of psychological distress among chinese people in the covid-19 epidemic: implications and policy recommendations. *General psychiatry*, 33(2), 2020.

[36] Lynn P Rehm. Mood, pleasant events, and unpleasant events: Two pilot studies. *Journal of Consulting and Clinical Psychology*, 46(5):854, 1978.

[37] Dori B Reissman, Patricia J Watson, Richard W Klomp, Terri L Tanielian, and Stephen D Prior. Pandemic influenza preparedness: adaptive responses to an evolving challenge. *Journal of Homeland Security and Emergency Management*, 3(2), 2006.

[38] Rena L Repetti. Short-term effects of occupational stressors on daily mood and health complaints. *Health Psychology*, 12(2):125, 1993.

[39] Filipe N Ribeiro, Matheus Araújo, Pollyanna Gonçalves, Marcos André Gonçalves, and Fabrício Benevenuto. Sentibench-a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ Data Science*, 5(1):1–29, 2016.

[40] Keith Roe. Swedish youth and music: Listening patterns and motivations. *Communication Research*, 12(3):353–362, 1985.

[41] Suvi Saarikallio and Jaakko Erkkilä. The role of music in adolescents' mood regulation. *Psychology of music*, 35(1):88–109, 2007.

[42] Thomas Schäfer and Peter Sedlmeier. From the functions of music to music preference. *Psychology of Music*, 37(3):279–300, 2009.

[43] Thomas Schäfer and Peter Sedlmeier. What makes us like music? determinants of music preference. *Psychology of Aesthetics, Creativity, and the Arts*, 4(4):223, 2010.

[44] Thomas Schäfer, Peter Sedlmeier, Christine Städtler, and David Huron. The psychological functions of music listening. *Frontiers in psychology*, 4:511, 2013.

[45] Markus Schedl. The LFM-1B Dataset for Music Retrieval and Recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ICMR '16, pages 103–110, New York, NY, USA, 2016. ACM.

[46] Markus Schedl, Eelco Wiechert, and Christine Bauer. The effects of real-world events on music listening behavior: An intervention time series analysis. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 75–76, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[47] Alexander J Shackman, Jennifer S Weinstein, Stanton N Hudja, Conor D Bloomer, Matthew G Barstead, Andrew S Fox, and Edward P Lemay Jr. Dispositional negativity in the wild: Social environment governs momentary emotional experience. 2017.

[48] James H Stock and Mark W Watson. *Introduction to econometrics*. Prentice Hall, 3rd edition, 2015.

[49] Arthur A Stone. The association between perceptions of daily experiences and self-and spouse-rated mood. *Journal of Research in Personality*, 15(4):510–522, 1981.

[50] Arthur A Stone. Event content in a daily survey is differentially associated with concurrent mood. *Journal of Personality and Social Psychology*, 52(1):56, 1987.

[51] Arthur A Stone and John M Neale. Effects of severe daily events on mood. *Journal of personality and social psychology*, 46(1):137, 1984.

[52] Arthur A Stone, John M Neale, and Saul Shiftman. Daily assessments of stress and coping and their association with mood. *Annals of Behavioral Medicine*, 15(1):8–16, 1993.

[53] Joanna Syrda. Spousal relative income and male psychological distress. *Personality and Social Psychology Bulletin*, page 0146167219883611, 2019.

[54] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

[55] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558, 2010.

[56] Cuiyan Wang, Riyu Pan, Xiaoyang Wan, Yilin Tan, Linkang Xu, Cyrus S Ho, and Roger C Ho. Immediate psychological responses and associated factors during the initial stage of the 2019 coronavirus disease (covid-19) epidemic among the general population in china. *International journal of environmental research and public health*, 17(5):1729, 2020.

[57] David Watson. *Mood and temperament*. Guilford Press, 2000.

[58] Zhongzhe Xiao, Emmanuel Dellandrea, Weibei Dou, and Liming Chen. What is the best segment duration for music mood analysis? In *2008 International Workshop on Content-Based Multimedia Indexing*, pages 17–24. IEEE, 2008.

[59] Yi-Hsuan Yang and Homer H Chen. Machine recognition of music emotion: A review. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):40, 2012.

[60] Eva Zangerle, Chih-Ming Chen, Ming-Feng Tsai, and Yi-Hsuan Yang. Leveraging affective hashtags for ranking music recommendations. *IEEE Transactions on Affective Computing*, 2018.

Paper Session 2

IMPROVING POLYPHONIC MUSIC MODELS WITH FEATURE-RICH ENCODING

Omar Peracha

Humtap (research conducted independently)

omar.peracha@gmail.com

ABSTRACT

This paper explores sequential modelling of polyphonic music with deep neural networks. While recent breakthroughs have focussed on network architecture, we demonstrate that the representation of the sequence can make an equally significant contribution to the performance of the model as measured by validation set loss. By extracting salient features inherent to the training dataset, the model can either be conditioned on these features or trained to predict said features as extra components of the sequences being modelled. We show that training a neural network to predict a seemingly more complex sequence, with extra features included in the series being modelled, can improve overall model performance significantly. We first introduce TonicNet, a GRU-based model trained to initially predict the chord at a given time-step before then predicting the notes of each voice at that time-step, in contrast with the typical approach of predicting only the notes. We then evaluate TonicNet on the canonical JSB Chorales dataset and obtain state-of-the-art results.

1. INTRODUCTION

Computational modelling of polyphonic music is now a decades-old practice, with documented attempts dating back over sixty years [1]. Recent years have seen great progress in these models' ability to capture the semantics of a musical corpus, with much of this progress due to advances in artificial neural network algorithms and their application to the music domain.

Some of the most significant breakthroughs of late have experimented with applying newly-developed architectures to the problem of modelling symbolic music [2-3], training or pre-training on a large cross-domain corpus [2][4], or introducing Gibbs-like sampling methods to orderless models [5-6]. We instead focus on the sequence being modelled itself, and provide observations and enhancements that might improve results across a wide range of approaches.

We conduct experiments with two architectures: a multi-layer Transformer encoder [7] with input masking

and a model based on the Gated Recurrent Unit [8] to which we give the nickname TonicNet. Using the JSB chorales dataset, split into training, validation and test sets as per [9], the models are trained to predict each token of the samples one-by-one in a sequential manner.

We observe that increasing the amount of musical information these models are trained to predict tends to improve overall performance of both models as measured by validation set loss. In particular, by training to first predict the chord at a given time step before then predicting the notes of each voice at that time step, both models show improvements in validation loss, despite the modelled sequences being longer and containing a larger possible output space than would be the case if predicting only the notes. We also train TonicNet on smaller subsets of the samples by restricting the number of voices being modelled, and again observe that results improve when more musical information is contained in the sequence being predicted.

These observations allow better results to be achieved without the need to use models with an ever-larger number of parameters. Concretely, TonicNet is approximately an order of magnitude smaller than Music Transformer, yet obtains a lower validation set loss on the JSB chorales dataset than that reported by [3] as a result of being trained to predict both chords and notes. It also generates samples much faster than reported in [5] by avoiding Gibbs-like sampling, and achieves state-of-the-art validation loss without requiring pre-training on a larger cross-domain corpus, saving a significant amount of training time.

All code for this paper is made publicly available, including the ability to load and sample from the pre-trained TonicNet model. [1] Samples generated by TonicNet are also made available in both MIDI and audio form.

2. RELATED WORK

Recurrent Neural Networks have been noted for their ability to model sequential data, with the LSTM [10] in particular being a favoured approach for a number of applications. RNNs have been widely-used in recent attempts to model musical data, which lends itself well to a sequence-based representation. The earliest such examples used RNNs to model monophonic music [11-12].

In [9] RNNs were combined with Restricted Boltzmann Machines to model polyphonic music, while RNNs and LSTMs were combined with Deep Belief Networks for the

---

[1] https://github.com/omarperacha/TonicNet

same task by [13] and [14] respectively. BachBot [15] also uses an LSTM-based neural network, and both the architecture and hyperparameters described in their paper are influential in this work. All of these polyphonic models have been evaluated on the Bach chorales, although not all with the same dataset split. BachBot is trained to predict not only the notes, but also whether a fermata, the symbol used by Bach to mark phrase endings, coincides with each given note. This is deemed to improve sample quality, however the version of the dataset used in this work does not contain information regarding presence of fermatas.

The Transformer and its derived architectures have been popular choices for more recent polyphonic music models, and are emerging as strong alternatives to RNNs more broadly due to the self-attention mechanism demonstrating great effectiveness at capturing long-term dependencies in training data [7]. Music Transformer adds a novel relative attention mechanism to the vanilla Transformer and is evaluated on the JSB chorales dataset. LakhNES [4] uses the Transformer-XL architecture [16] and is pretrained on a large corpus of four-part music before being fine-tuned on the NES Music Database [17]. An event-based encoding is preferred which allows for more precise rhythm than the most commonly seen approach of slicing the input sample along a 16th-note (or some other suitable, dataset-dependent value) grid. MuseNet [2] is based on the GPT-2 Transformer model [18] and is trained on a massive corpus of polyphonic music. The data representation used has similar qualities to the event-based representation described by [4], and allows the trained model to sample while taking into account specific instrumentation and musical style. The encoding also includes information relating to note loudness.

Chords have been used in some previous approaches to improve the quality of monophonic music models. Chords are used as extra inputs to aid the generation of melodies by [19], who use a dual LSTM network Product of Experts system [20], and by [21], who propose training a convolutional generative adversarial network to generate melodies one bar at a time. However, neither of methods learn to predict the chords being used as input for each proceeding step, but rather provide them as fixed inputs.

HARMONET [22], comprises an ensemble of multiple neural networks trained with the ultimate goal of harmonising a given melody in the style of Bach. The first step is to derive the chords at each quarter-note step from the melody, relative to the key, including the inversion (and therefore the bass part). The inner parts are then predicted in a second step.

In the method proposed in this paper, chords are in fact included in the sequence as a de facto extra voice which must be predicted along with the other four voices, rather than being used as a secondary conditioning input. Inversions are not encoded in the chord representation, as the bass part is included later in the sequence.

DeepBach [5] is also trained on a representation including extra musical features as added voices. In this case, it is the presence of a fermata at each time step. The model is not trained to predict this, however, but fermata information is instead provided as a fixed input which helps guide the musical structure of the generated samples. Chord tokens are not used by the DeepBach model.

Both DeepBach and COCONET [6] are trained with the primary goal of completing partially-filled musical scores, for example harmonising a given melody, though both are able to generate entire four-voice samples from an empty or randomly-initialised score. They do both require the length of the sample in time-steps to be preset in order to facilitate the orderless Gibbs-like sampling methods used, and DeepBach further requires fermata information to be provided. TonicNet instead uses ancestral sampling to generate scores, and is trained to predict successive tokens in a purely autoregressive fashion, requiring absolutely no pre-set information relating to length or phrasing. This ultimately inhibits sample quality as the phrase lengths may not display the consistent symmetry over time observed in the training corpus, however the model still obtains a lower validation set loss on the JSB chorales dataset than the upper bound reported by [3] for orderless evaluation of CO-CONET (i.e. evaluating COCONET's ability to correctly fill in the missing notes in partially-completed scores), when averaging purely over the note predictions and ignoring the chords which do not originally appear in the benchmark dataset.

A powerful advantage of DeepBach and COCONET is that they lend themselves far more naturally to interactive applications. Theoretically one could fix the chords or the notes of a given voice when sampling from TonicNet by ignoring predicted output for the relevant part, instead using the token from the corresponding time-step of the fixed sequence as input to the model at the next time-step. However, this kind of forced sampling has not been tested empirically and so sample quality under these conditions cannot be attested.

3. DATASET

3.1 JSB Chorales

The JSB chorales are the most commonly-used benchmark for measuring the performance of polyphonic music models to date. They are a set of short, four-voice pieces well-noted for their stylistic homogeneity. The chorales were originally composed by Johann Sebastian Bach in the 18th century. He wrote them by first taking pre-existing melodies from contemporary Lutheran hymns and then harmonising them to create the parts for the remaining three voices. The version of the dataset used here consists of 382 such chorales, with a train/validation/test split of 229, 76 and 77 samples respectively. [2]

---

[2] The version of the dataset used can be accessed at https://github.com/czhuang/JSB-Chorales-dataset

3.2 Representation

3.2.1 Serialisation

The dataset is pre-serialised onto a 16th-note grid, which captures full resolution of the original chorales. Only the pitch information of the four voices at each time-step is encoded in the canonical dataset; other symbolic data that may appear in a musical score, such as loudness or fermata, are absent. Furthermore, information regarding note boundaries in the case of repeated pitches is not present. The consequence of this is that it is not possible to truly accurately model Bach's original rhythms, unlike other approaches, e.g. [5], where the version of the dataset used allows note boundary information to be preserved using a special 'hold' token, despite 16th-note serialisation. A partial workaround to lack of repeated note boundaries during sampling is to simply tie together consecutive occurrences of the same pitch in a voice, which we refer to as rhythmic 'smoothing', however this inevitably sacrifices some rhythmic integrity of the original chorales. While it may seem unideal to serialise music onto a fine-resolution rhythmic grid, as this has the consequence of vastly extending the length of the sequences being modelled, it in fact acts as a highly effective form of data augmentation; brief experiments which encoded the true duration values of the notes, using the Bach chorales as made available by the music21 toolkit [23], ultimately performed significantly worse than the representation presented in this work, both in terms of validation loss and sample quality.

We include chords in our encoding. We first derive the chords for each 16th-note time-step by analysing the pitches of the four voices at said time-steps, using the music21 chord module. We then create a single ordered sequence for each sample in the form C_0, S_0, B_0, A_0, T_0, C_1, S_1, B_1, A_1, T_1... C_{N-1}, S_{N-1}, B_{N-1}, A_{N-1}, T_{N-1}, <END>, where C, S, A, T and B represent the Chord, Soprano, Alto, Tenor and Bass inputs at each respective step, and N is the total number of 16th-note time-steps in the given sample. The model is fed the elements of the sequence one-by-one and tasked with predicting the next element in each case, thus it must predict the chord governing each 16th-note time-step before then predicting the actual pitch values observed in the dataset at that time-step. The effect of predicting the Bass note before the Alto and Tenor notes seems to be negligible, but has not been thoroughly tested and so may be considered arbitrary. The longest sequence observed in the dataset given the described representation, is 2,881 tokens in length, taking into account the appended <END> token, but the lengths vary quite considerably across the dataset.

3.2.2 Symbolic Encoding

Pitches and chords are all represented by distinct integer values. We restrict pitches purely to those observed in the dataset (MIDI values 36-81 inclusive assuming a value of 60 to be Middle C). The MIDI pitch value 37, despite not in fact appearing in the dataset, is included so as to better facilitate data augmentation by transposition, described in the next section. Chords are represented as belonging to one of 50 classes, comprised by 12 major chords (one chord per pitch class in the western chromatic scale), 12 minor chords, 12 diminished chords, 12 augmented chords and a special <OTHER> token which accounts for any chord which is not interpreted as fitting into the previous 48 classes. A <CHORD REST> token completes the set used to represent chord classes, and denotes instances when all four voices have rests at that time-step. Any voice-wise occurrence of a rest in the dataset is itself represented by a distinct <REST> token, and finally the <END> token completes the set of possible model input/output classes, taking the total to 98.

3.3 Conditioning

In addition to chord/pitch inputs, which we refer to as X-input, we condition the model a second input that relays information about note repetition. Concretely, alongside each X-input value $X_{n,i}$, where i is used to index the chords and voices, we also input an integer, $Z_{n,i}$, corresponding to the number of consecutive times the value represented by $X_{n,i}$ has so far appeared in voice i, resetting to 0 each time a new value is observed in that voice or if $Z_{n,i}$ exceeds 79 (equating to 5 bars in 4/4 timing). This value was chosen because only one sample features Z-input values greater than 79, with this sample's maximum Z-input value of 143 being a clear outlier.

The motivation for this is that we might more explicitly capture some of the inter-voice rhythmic relationships that exist in the music, and indeed we observe that it improves model performance (Table 1). We also experimented with instrument labelling, as in [3], and found that while it somewhat improved model performance, repetition encoding had a more significant impact and combining both repetition encoding and instrument labels did not perform better than repetition encoding alone. From this we can infer that repetition encoding, as well as helping the model to better learn timing information relating to note and chord changes, also fulfils a role similar to instrument labelling. We refer to repetition encoding inputs as Z-inputs.

3.4 Augmentation

We perform two kinds of dataset augmentation on the training set alone, leaving the validation and test sets unchanged. Firstly, we transpose all pieces as many times as possible so that each piece only contains pitches that are within the set of pitches observed in the original dataset, and so that there are no instances of a pitch exceeding the natural range of the voice-type in which it appears. This takes the total number of training examples up to 1,968. We found that transposition makes a significant impact on the model's ability to generalise (Table 1).

We also crudely convert all major pieces to minor, and vice versa, by raising all occurrences of the minor 3rd, 6th and 7th in minor pieces and flattening occurrences of the major 3rd and 6th in major pieces, leaving the 7th raised. This ultimately had negligible impact on model performance and harmed sample quality, while significantly in-

creasing the training time due to effectively doubling the number of samples. We therefore do not consider this an effective technique for dataset augmentation in the context of the JSB chorales. We hypothesise that this weakness may be due to the use of chromaticism and presence of key modulations within samples in the dataset, which may not maintain their stylistic integrity when the overall mode of the piece is converted in the naive manner described.

4. MODELS

4.1 TonicNet

TonicNet takes two inputs: the integer corresponding to the previous time-step's class label (X-input) and the integer corresponding to its repetition count (Z-input). These inputs are each converted to one-hot vector representations and by a 256-dimension and 32-dimension embedding respectively. The embedding outputs are then endconcatenated. Both the X embedding and the Z embedding are learned during training.

The concatenated embedding outputs have Variational Dropout [24] applied with a rate of 0.1, before then being input to a three-layer, 256-unit GRU, with dropout applied after each of the first two GRU layers, using a rate of 0.3.
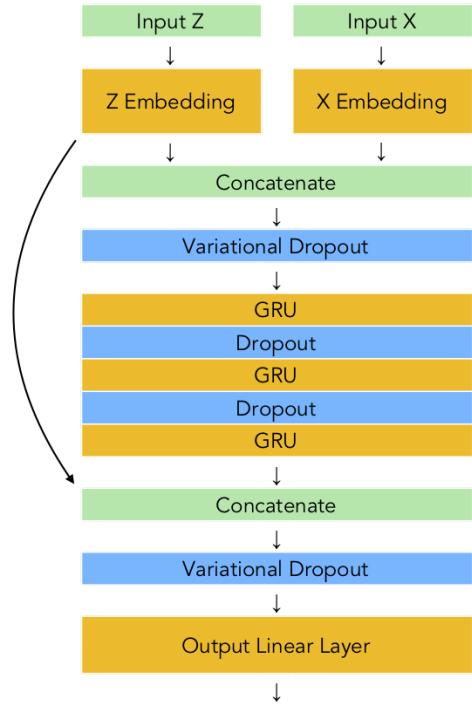
The Z embedding output is reintroduced by endconcatenating with the GRU output, before applying Variational Dropout with a probability of 0.3. The resulting tensor is then fed to a final 98-unit affine layer.

We train TonicNet using a batch size of one for 60 epochs, employing the 1cycle policy [25] with Stochastic Gradient Descent as the optimiser. We begin training with an initial learning rate of 0.008, which is increased to 0.2 over the first 18 epochs. The learning rate is then decreased to 0.0002 over the remaining epochs via cosine annealing. We also cycle momentum inversely to the learning rate between values of 0.8 and 0.95. During training we clip the norm of the gradients to 5, which prevents gradient explosion. Training on the transposed dataset took roughly 3.25hrs on a T4 Tensor Core GPU.

Model hyperparameters, including number of recurrent layers, hidden units and dropout rate, were inspired by [15], and corroborated by initial experimentation. The sizes of both the Z and X embeddings were chosen naively and the effect of varying these has not been determined through experimentation, so there may be room to further tune the parameters of this model and improve results, which we leave to future work.

4.2 Transformer

We use a 5-layer Transformer encoder model. We encode absolute position in the sequence using a fixed sinusoidal position embedding as described in [7], with 256 dimensions. The model also learns a 256-dimension input embedding. The outputs of the two embeddings are endconcatenated. We use 8 attention heads and set the model dimension, D, to 512. The feedforward layer within the encoder module has 1024 units, and dropout is set to 0.1. The hyperparameters of this model are largely derived from the

Figure 1. Diagram of the TonicNet model architecture..

Vanilla Transformer decoder model used as a baseline in [3]. Input is masked to ensure the model can only attend to previous time-steps when making a prediction. Neither instrument labelling or repetition encoding are used when training this model.

We again utilise a batch size of one and employ the 1cycle policy with SGD. In this case we train for 30 epochs, increasing the learning rate from 0.0006 to 0.06 over the first 9 epochs before decreasing to 0.00006 with cosine annealing over the remaining epochs. Momentum is cycled inversely to learning rate between values of 0.8 and 0.95, and the gradient norm is clipped to 5. This model trains faster than TonicNet when measuring time taken per batch, but direct comparison is not possible as it was trained on CPU in our experiments.

4.3 Common Implementation Details

In both cases, we derive the maximum learning rate used during training by first performing an LR Range Test [26]. The models are trained using cross-entropy as the objective function, and always see the ground truth values for the previous steps of the sequence when predicting the current step.

While the benchmark dataset contains only notes, and therefore we are arguably more interested in minimising the loss when predicting the pitches of the voices than when predicting chords, we train the model to minimise the average loss across the entire sequence with no bias shown to steps including note predictions. Both models are implemented and trained using the PyTorch library [27].

5. EVALUATION

5.1 Quantitative Evaluation

The Transformer model is evaluated only on the un-augmented dataset, due to resource constraints. We compare performance when training on just notes of the four voices (SATB), and when training on sequences also including chords (CSATB). TonicNet is tested on a wider range of related tasks including modelling a variety of part combinations, from one part only up to the maximum five parts. We also show the effect of repetition encoding and training set augmentation. Table 1 shows the results of these experiments as measured by validation set loss, and compares performance against two high-performing baselines on the dataset, namely COCONET and Music Transformer.

In Table 1, each model variant shows the parts trained and evaluated on in parentheses, where NLL = Negative Log Likelihood, C = Chords, S = Soprano, A = Alto, T = Tenor and B = Bass. NCL stands for No Chord Loss, indicating that the model was only evaluated on the note predictions, ignoring the loss at time-steps corresponding to chord predictions when averaging NLL. The use of transposition to augment the dataset is denoted by Tr, and MM signifies training set augmentation via major-to-minor key conversion (and vice versa). TonicNet_Z here indicates the inclusion of repetition encoding Z-inputs when training the model.

The results show a trend whereby training the network to predict more musical information improves overall performance. Both models perform better when trained to predict chords before predicting notes at each 16th-note time-step, as compared with training to predict only the notes. Even when evaluating on the entire CSATB sequence, including the chords in the reported loss, we see that TonicNet_Z with transposition outperforms Music Transformer, the previous highest-performing ordered model as evaluated on the pure SATB dataset, despite being an order of magnitude smaller. When we ignore loss on chord time-steps, the average NLL is significantly lower, performing better than the upper bound for unordered evaluation of COCONET. The improvement when comparing performance on pitch-wise loss alone is echoed by the Transformer (CSATB) model. From this we can derive that predicting the chords has the impact of significantly improving model confidence when predicting pitches, which is perhaps intuitive.

5.2 Qualitative Evaluation

We evaluate the quality of samples from TonicNet via human domain expert analysis. We define a domain expert as someone holding a post-graduate degree in a subject directly related to Western Classical Music, and who has formally studied Bach's chorales as part requirement for obtaining an academic qualification. This is favoured as it allows for a more objective, thorough and strict criticism than a layperson-targeted listening test. We use random sampling to generate chorales from TonicNet, after first

Model & Dataset Variation	Validation NLL
Transformer (SATB)	0.544
Transformer (CSATB)	0.503
Transformer (CSATB, NCL)	0.394
Music Transformer* (SATB)	0.335
COCONET* (SATB, chronological)	0.436
COCONET* (SATB, orderless)	≤ 0.238
TonicNet (C)	0.936
TonicNet (B)	0.716
TonicNet (S)	0.521
TonicNet (CS)	0.588
TonicNet (SB)	0.555
TonicNet (CSB)	0.516
TonicNet (SATB)	0.523
TonicNet_Z (SATB)	0.497
TonicNet_Z (CSATB)	0.422
TonicNet_Z (CSATB, Tr)	0.321
TonicNet_Z (CSATB, Tr+MM)	0.317
TonicNet_Z (CSATB, Tr, NCL)	**0.224**
TonicNet_Z (CSATB, Tr+MM, NCL)	**0.220**

Table 1. Validation loss on JSB chorales at 16th-note time-steps.

selecting a starting minor or major chord at random. Experiments with beam search decoding tended to produce overly-short samples, even when normalising sample probability for length, therefore random sampling according to the output probability distribution is preferred. Stochasticity during beam search has not been subject to experimentation.

We find that TonicNet_Z (CSATB, Tr) produces the best samples. Voice leading is typically stylistic, especially on a local scale, as is the generated melodic contour, although there is a tendency to diverge from what is clearly the intended phrase within a part for a single 16th-note, usually by a single scale degree or semitone, before then returning, causing an undesirable ornamentation effect.

Harmonisation and harmonic trajectory is also consistently plausible, however there are occasional instances of a phrase which clearly starts in a major key suddenly modulating to a minor key, or vice versa, in a manner that is uncharacteristic to the corpus. The worst generated samples may in fact display poor, overly-chromatic harmonisation and lack stylistic harmonic direction. Some instances of sample weakness may be artefacts of the exposure bias introduced by using teacher forcing when training TonicNet.

The most significant issue detected in samples ultimately relates to phrasing; Bach's chorales feature symmetric phrases, typically two, four or eight bars long, ending in a cadence. Generated pieces have a tendency to feature asymmetry between consecutive phrases, which is not stylistic. Including fermata or other phrase-based information in the modelled sequence could help mitigate this issue, as demonstrated in [5] and [15]. Samples do consistently end on a perfect cadence as expected, and voices never misalign due to a misordering in the generated se-

---

*Figures reproduced directly from [3]

	Val NLL	Val Acc.	Test NLL	Test Acc.
Full	0.317	90.928	0.311	90.787
NCL	0.220	93.468	0.214	93.419

Table 2. Model loss and accuracy when evaluating Tonic-Net_Z (CSATB, Tr+MM) on validation and test sets, both when including and ignoring chord predictions (Full versus NCL).

quence; rather, each voice clearly completes its phrase, and the duration of each voice's phrase coincides exactly with the others. Samples displaying a range of quality are included in the code repository for fair analysis.

6. CONCLUSIONS

We first extracted salient features from the existing dataset, in the form of chords and intra-voice token repetition, and then included these extra features among the training inputs. The fact that exposing the model to more features should improve results is unsurprising; more unexpected is that including new features as extra elements within the series being modelled should dramatically enhance performance. Furthermore, it was noted that confidence when predicting pitches is much higher if the model is first tasked with predicting chords. This suggests a worthwhile area for further research is to improve confidence when predicting the chords, and we conjecture that a method to achieve this may be to include yet more related features in the sequences themselves, such as fermata information or a representation of floating tonality, given our findings.

State-of-the-art validation loss on the JSB chorales dataset was achieved with a variation of TonicNet and effective dataset augmentation, and we demonstrated that despite this there are still some specific weaknesses in sample quality which other approaches have mitigated. We also noted the superior human interactability of COCONET and DeepBach, which we believe gives those proposals a greater potential for real-world application. However, the findings presented in this paper could be applicable to a wide range of approaches to statistical modelling of polyphonic music, and their merit was demonstrated on two such approaches.

We also surveyed the effects of serialising music by splitting notes across a fine-resolution temporal grid. While the benefit of this in terms of data augmentation was noted, we also presented weaknesses relating to true rhythmic integrity in the case of repeated note boundaries, and vast extension of sequence length which has the effect of increasing both training and sampling time. Ultimately we would like to move towards utilising encodings including true rhythmic duration, with the aim of being able to train more general polyphonic music models that are not confined to a temporal grid, and therefore to styles of music whose rhythmic resolution can be encompassed by this grid. It is not viable to simply serialise music into increasingly finer-resolution rhythmic units in order to accommodate datasets which include some occurrences of tuplet du-

rations, for example, as this continues to extend the overall sequence length. Future work will explore solutions to this drawback.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] L. Hiller and L. Isaacson, *Experimental Music*. New York: McGraw-Hill, 1959.

[2] C. Payne, "MuseNet", *OpenAI*, 2019. [Online]. Available: https://openai.com/blog/musenet/. [Accessed: 5-May- 2019].

[3] C. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. Dai, M. Hoffman, M. Dinculescu and D. Eck, "Music Transformer", ArXiv:1809.04281 [cs], 2018

[4] C. Donahue, H. Mao, Y. Li, G. Cottrell and J. McAuley, "LakhNES: Improving Multi-Instrumental Music Generation with Cross-Domain Pre-Training", ArXiv:1907.04868 [cs], 2019.

[5] G. Hadjeres, F. Pachet and F. Nielsen, "DeepBach: a Steerable Model for Bach Chorales Generation", in *International Conference on Machine Learning*, 2017, pp. 1362-1371.

[6] C. Huang, T. Coojimans, A. Roberts, A. Courville and D. Eck, "Counterpoint by Convolution", in *Proc. of the International Conference on Music Information Retrieval*, 2017, pp. 211–218.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser and I. Polosukhin, "Attention is All You Need", in *Advances in Neural Information Processing Systems*, 2017, pp. 5998-6008.

[8] K. Cho, B. van Merrienboer, D. Bahdanau and Y. Bengio, "On The Properties of Neural Machine Translation: Encoder-Decoder Approaches", ArXiv:1409.1259 [cs], 2014.

[9] N. Boulanger-Lewandowski, Y. Bengio and P. Vincent, "Modelling Temporal Dependencies in High- Dimensional Sequences: Application to Polyphonic Music Generation and Transcription", in *International Conference on Machine Learning*, 2012.

[10] S. Hochreiter and J. Schmidhuber, "Long Short- Term Memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] P. Todd, "A Connectionist Approach to Algorithmic Composition", *Computer Music Journal"*, vol. 13, no. 4, pp. 27–43, 1989.

[12] D. Eck and J. Schmidhuber. "Finding Temporal Structure in Music: Blues Improvisation with LSTM Recurrent Networks", in *Proc. of the 12th IEEE Workshop on Neural Networks for Signal Processing*, 2002.

[13] K. Goel, R. Vohra and J. Sahoo, "Polyphonic Music Generation by Modelling Temporal Dependencies Using a RNN-DBN", in *Proc. of the International Conference on Artificial Neural Networks*, 2014, pp. 217-224.

[14] R. Vohra, K. Goel and J. Sahoo, "Modelling Temporal Dependencies in Data Using a DBN-LSTM", in *Proc. of the IEEE International Conference on Data Science and Advanced Analytics*, 2015.

[15] F. Liang, "Bachbot: Automatic Composition in the s Style of Bach Chorales," M.Phil thesis. University of Cambridge, 2016.

[16] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le and R. Salakhutdinov, "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context", ArXiv:1901.02860 [cs], 2018.

[17] C. Donahue, H. Mao and J. McAuley, "The NES Music Database: A Multi-Instrumental Dataset with Expressive Performance Attributes", in *Proc. of the International Conference on Music Information Retrieval*, pp. 475-482, 2018.

[18] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, *Language Models are Unsupervised Multitask Learners*. 2019.

[19] D. Johnson, R. Keller and N. Weintraut, "Learning to Create Jazz Melodies Using a Product of Experts", in *Proc. of the International Conference on Computational Creativity*, 2017.

[20] G. Hinton, "Products of Experts", in *Proc. of the International Conference on Artificial Neural Networks*, pp. 1-6, 2002.

[21] L. Yang, S. Chou and Y. Yang, "MidiNet: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation", in *Proc. of the International Conference on Music Information Retrieval*, pp. 324-331, 2017.

[22] H. Hild, J. Feulner and W. Menzel, "HARMONET: A Neural Net for Harmonizing Chorales in the Style of J. S. Bach", in *Advances in Neural Information Processing Systems*, pp. 267-274, 1991.

[23] M. Cuthbert and C. Ariza, "music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data," in *Proc. of the International Conference on Music Information Retrieval*, pp. 637–642, 2010.

[24] D. Kingma, T. Saliman and M. Welling, "Variational Dropout and the Local Reparameterization Trick", in *Advances in Neural Information Processing Systems*, pp. 2575-2583, 2015.

[25] L. Smith, "A Disciplined Approach to Neural Network Hyper-Parameters: Part 1 – Learning Rate, Batch Size, Momentum, and Weight Decay", ArXiv:1803.09820 [cs], 2018.

[26] L. Smith, "Cyclical Learning Rates for Training Neural Networks", ArXiv:1506.01186 [cs], 2015.

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic Differentiation in PyTorch", *Neural Information Processing Systems Autodiff Workshop*, 2017.

COMPOSER STYLE CLASSIFICATION OF PIANO SHEET MUSIC IMAGES USING LANGUAGE MODEL PRETRAINING

TJ Tsai
Harvey Mudd College
ttsai@g.hmc.edu

Kevin Ji
Harvey Mudd College
kji@g.hmc.edu

ABSTRACT

This paper studies composer style classification of piano sheet music images. Previous approaches to the composer classification task have been limited by a scarcity of data. We address this issue in two ways: (1) we recast the problem to be based on raw sheet music images rather than a symbolic music format, and (2) we propose an approach that can be trained on unlabeled data. Our approach first converts the sheet music image into a sequence of musical "words" based on the bootleg feature representation, and then feeds the sequence into a text classifier. We show that it is possible to significantly improve classifier performance by first training a language model on a set of unlabeled data, initializing the classifier with the pretrained language model weights, and then finetuning the classifier on a small amount of labeled data. We train AWD-LSTM, GPT-2, and RoBERTa language models on all piano sheet music images in IMSLP. We find that transformer-based architectures outperform CNN and LSTM models, and pretraining boosts classification accuracy for the GPT-2 model from 46% to 70% on a 9-way classification task. The trained model can also be used as a feature extractor that projects piano sheet music into a feature space that characterizes compositional style.

1. INTRODUCTION

We've all had the experience of hearing a piece of music that we've never heard before, but immediately recognizing the composer based on the piece's style. This paper explores this phenomenon in the context of sheet music. The question that we want to answer is: "Can we predict the composer of a previously unseen page of piano sheet music based on its compositional style?"

Many previous works have studied the composer classification problem. These works generally fall into one of two categories. The first category of approach is to construct a set of features from the music, and then feed the features into a classifier. Many works use manually designed features that capture musically meaningful information (e.g. [1] [2] [3] [4]). Other works feed minimally
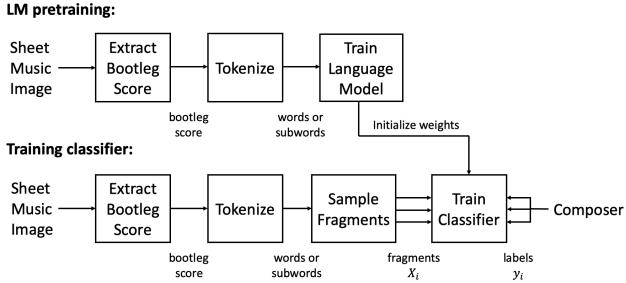
preprocessed representations of the data (e.g. 2-D piano rolls [5] [6] or tensors encoding note pitch & duration information [7] [8]) into a convolutional model, and allow the model to learn a useful feature representation. The second category of approach is to train one model for each composer, and then select the model that has the highest likelihood of generating a given sequence of music. Common approaches in this category include N-gram language models [9] [10] [11] and Markov models [12] [13].

Our approach to the composer classification task addresses what we perceive to be the biggest common obstacle to the above approaches: lack of data. All of the above approaches assume that the input is in the form of a symbolic music file (e.g. MIDI or **kern). Because symbolic music formats are much less widely used than audio, video, and image formats, the amount of training data that is available is quite limited. We address this issue of data scarcity in two ways: (1) we re-define the composer classification task to be based on sheet music images, for which there is a lot of data available online, and (2) we propose an approach that can be trained on unlabeled data.

Our work takes advantage of recent developments in transfer learning in the natural language processing (NLP) community. Prior to 2017, transfer learning in NLP was done in a limited way. Typically, one would use pretrained word embeddings such as word2vec [14] [15] or GloVe [16] vectors as the first layer in a model. The problem with this paradigm of transfer learning is that the entire model except the first layer needs to be trained from scratch, which requires a large amount of labeled data. This is in contrast to the paradigm of transfer learning in computer vision, where a model is trained on the ImageNet classification task [17], the final layer is replaced with a different linear classifier, and the model is finetuned for a different task. The benefit of this latter paradigm of transfer learning is that the entire model except the last layer is pretrained, so it can be finetuned with only a small amount of labeled data. This paradigm of transfer learning has been widely used in computer vision in the last decade [18] using pretrained models like VGG [19], ResNet [20], Densenet [21], etc. The switch to ImageNet-style transfer learning in the NLP community occurred in 2017, when Howard et al. [22] proposed a way to pretrain an LSTM-based language model on a large set of unlabeled data, add a classification head on top of the language model, and then finetune the classifier on a new task with a small amount of labeled data. This was quickly followed by sev-

Figure 1. Overview of proxy classifier training. A language model is first trained on a set of unlabeled data, the classifier is initialized with the pretrained language model weights, and then the classifier is finetuned on a small set of labeled data.

Figure 2. A short section of sheet music and its corresponding bootleg score. Staff lines in the bootleg score are shown for reference, but are not present in the actual feature representation.

eral other similar language model pretraining approaches that replaced the LSTM with transformer-based architectures (e.g. GPT [23], GPT-2 [24], BERT [25]). These pretrained language models have provided the basis for achieving state-of-the-art results on a variety of NLP tasks, and have been extended in various ways (e.g. Transformer-XL [26], XLNet [27]).

Our approach is similarly based on language model pretraining. We first convert each sheet music image into a sequence of words based on the bootleg score feature representation [28]. We then feed this sequence of words into a text classifier. We show that it is possible to significantly improve the performance of the classifier by training a language model on a large set of unlabeled data, initialize the classifier with the pretrained language model weights, and finetune the classifier on a small amount of labeled data. In our experiments, we train language models on all piano sheet music images in the International Music Score Library Project (IMSLP) [1] using the AWD-LSTM [29], GPT-2 [24], and RoBERTa [30] language model architectures. By using pretraining, we are able to improve the accuracy of our GPT-2 model from 46% to 70% on a 9-way classification task. [2]

2. SYSTEM DESCRIPTION

We will describe our system in the next four subsections. In the first subsection, we give a high-level overview and rationale behind our approach. In the following three subsections, we describe the three main stages of system development: language model pretraining, classifier finetuning, and inference.

2.1 Overview

Figure 1 summarizes our training approach. In the first stage, we convert each sheet music image into a sequence of words based on the bootleg score representation [28], and then train a language model on these words. Since this task does not require labels, we can train our language
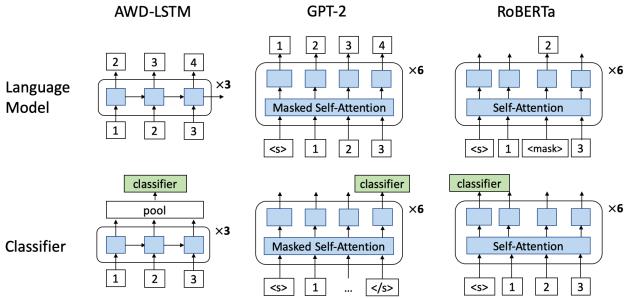
model on a large set of unlabeled data. In this work, we train our language model on all piano sheet music images in the IMSLP dataset. In the second stage, we train a classifier that predicts the composer of a short fragment of music, where the fragment is a fixed-length sequence of symbolic words. We do this by adding one or more dense layers on top of the language model, initializing the weights of the classifier with the language model weights, and then finetuning the model on a set of labeled data. In the third stage, we use the classifier to predict the composer of an unseen scanned page of piano sheet music. We do this by converting the sheet music image to a sequence of symbolic words, and then either (a) applying the classifier to a single variable length input sequence, or (b) averaging the predictions of fixed-length crops sampled from the input sequence. We will describe each of these three stages in more detail in the following three subsections.

The guiding principle behind our approach is to maximize the amount of data. This impacts our approach in three significant ways. First, it informs our choice of data *format*. Rather than using symbolic scores (as in previous approaches), we instead choose to use raw sheet music images. While this arguably makes the task much more challenging, it has the benefit of having much more data available online. Second, we choose an approach that can utilize unlabeled data. Whereas labeled data is usually expensive to annotate and limited in quantity, unlabeled data is often extremely cheap and available in abundance. By adopting an approach that can use unlabeled data, we can drastically increase the amount of data available to train our models. Third, we use data augmentation to make the most of the limited quantity of labeled data that we do have. Rather than fixating on the page classification task, we instead define a proxy task where the goal is to predict the composer given a fixed-length sequence of symbolic words. By defining the proxy task in this way, we can aggressively subsample fragments from the labeled data, resulting in a much larger number of unique training data points than there are actual pages of sheet music. Once the proxy task classifier has been trained, we can apply it to the full page classification task in a straightforward manner.

---

[1] http://imslp.org/
[2] Code can be found at https://github.com/tjtsai/PianoStyleEmbedding.

Figure 3. Overview of AWD-LSTM, GPT-2, and RoBERTa language models (top) and classifiers (bottom). Boxes in blue are trained during the language modeling phase and used to initialize the classifier.

2.2 Language Model Pretraining

The language model pretraining consists of three steps, as shown in the upper half of Figure 1. These three steps will be described in the next three paragraphs.

The first step is to convert the sheet music image into a bootleg score. The bootleg score is a low-dimensional feature representation of piano sheet music that encodes the position of filled noteheads relative to the staff lines [28]. Figure 2 shows an example of a section of sheet music and its corresponding bootleg score representation. The bootleg score itself is a $62 \times N$ binary matrix, where 62 indicates the total number of possible staff line positions in both the left and right hands, and where N indicates the total number of estimated simultaneous note onset events. Note that the representation discards a significant amount of information: it does not encode note duration, key signature, time signature, measure boundaries, accidentals, clef changes, or octave markings, and it simply ignores non-filled noteheads (e.g. half or whole notes). Nonetheless, it has been shown to be effective in aligning sheet music and MIDI [28], and we hypothesize that it may also be useful in characterizing piano style. The main benefit of using the bootleg score representation over a full optical music recognition (OMR) pipeline is processing time: computing a bootleg score only takes about 1 second per page using a CPU, which makes it suitable for computing features on the entire IMSLP dataset. [3] We use the code from [28] as a fixed feature extractor to compute the bootleg scores.

The second step is to tokenize the bootleg score into a sequence of word or subword units. We do this differently for different language models. For word-based language models (e.g. AWD-LSTM [29]), we consider each bootleg score column as a single word consisting of a 62-character string of 0s and 1s. We limit the vocabulary to the $30,000$ most frequent words, and map infrequent words to a special unknown word token <unk>. For subword-based language models (e.g. GPT-2 [24], RoBERTa [30]), we use a byte pair encoding (BPE) algorithm [32] to learn a vocabulary of subword units in an unsupervised manner. The BPE algorithm starts with an initial set of subword units

(e.g. the set of unique characters [33] or the $2^8 = 256$ unique byte values that comprise unicode characters [34]), and it iteratively merges the most frequently occurring pair of adjacent subword units until a desired vocabulary size has been reached. We experimented with both character-level and byte-level encoding schemes (i.e. representing each word as a string of 62 characters vs. a sequence of 8 bytes), and we found that the byte-level encoding scheme performs much better. We only report results with the byte-level BPE tokenizer. For both subword-based language models explored in this work, we use the same shared BPE tokenizer with a vocabulary size of $30,000$ (which is the vocabularly size used in the RoBERTa model). At the end of the second step, we have represented the sheet music image as a sequence of words or subword units.

The third step is to train a language model on a set of unlabeled data. In this work, we explore three different language models, which are representative of state-of-the-art models in the last 3-4 years. The top half of Figure 3 shows a high-level overview of these three language models. The first model is AWD-LSTM [29]. This is a 3-layer LSTM architecture that makes heavy use of regularization techniques throughout the model, including four different types of dropout. The output of the final LSTM layer is fed to a linear decoder whose weights are tied to the input embedding matrix. This produces an output distribution across the tokens in the vocabulary. The model is then trained to predict the next token at each time step. We use the fastai implementation of the AWD-LSTM model with default parameters. The second model is openAI's GPT-2 [24]. This architecture consists of multiple transformer decoder layers [35]. Each transformer decoder layer consists of a masked self-attention, along with feed-forwards layers, layer normalizations, and residual connections. While transformer encoder layers allow each token to attend to all other tokens in the input, the transformer decoder layers only allow a token to attend to previous tokens. [4] Similar to the AWD-LSTM model, the outputs of the last transformer layer are fed to a linear decoder whose weights are tied to the input embeddings, and the model is trained to predict the next token at each time step. We use the huggingface implementation of the GPT-2 model with default parameters, except that we reduce the vocabulary size from $50,000$ to $30,000$ (to use the same tokenizer as the RoBERTa model), the amount of context from 1024 to 512, and the number of layers from 12 to 6. The third model is RoBERTa [30], which is based on Google's BERT language model [25]. This architecture consists of multiple transformer encoder layers. Unlike GPT-2, each token can attend to all other tokens in the input and the goal is not to predict the next token. Instead, a certain fraction of the input tokens are randomly converted to a special <mask> token, and the model is trained to predict the masked tokens. We use the huggingface implementation of RoBERTa with default parameter settings, except that we reduce the number of layers from 12 to 6.

---

[3] In contrast, the best performing music object detectors take 40-80 seconds to process each page at inference time using a GPU [31].

[4] This is because, in the original machine translation task [35], the decoder generates the output sentence autoregressively.

2.3 Classifier Finetuning
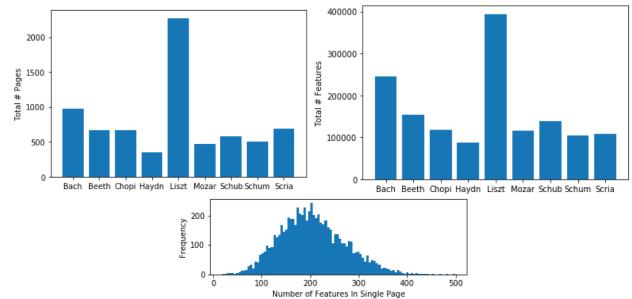
In the second main stage, we finetune a classifier based on a set of labeled data. The labeled data consists of a set of sheet music images along with their corresponding composer labels. The process of training the classifier is comprised of four steps (lower half of Figure 1).

The first two steps are to compute and tokenize a bootleg score into a sequence of symbolic words. We use the same fixed feature extractor and the same tokenizer that were used in the language model pretraining stage.

The third step is to sample short, fixed-length fragments of words from the labeled data. As mentioned in Section 2.1, we define a proxy task where the goal is to predict the composer given a short, fixed-length fragment of words. Defining the proxy task in this way has three significant benefits: (1) we can use sampling to generate many more unique training data points than there are actual pages of sheet music in our dataset, (2) we can sample the data in such a way that the classes are balanced, which avoids problems during training, and (3) using fixed-length inputs allows us to train more efficiently in batches. Our approach follows the general recommendations of a recent study on best practices for training a classifier with imbalanced data [36]. Each sampled fragment and its corresponding composer label constitute a single (X_i, y_i) training pair for the proxy task.

The fourth step is to train the classifier model. The bottom half of Figure 3 shows how this is done with our three models. Our general approach is to add a classifier head on top of the language model, initialize the weights of the classifier with the pretrained language model weights, and then finetune the classifier on the proxy task data. For the AWD-LSTM, we take the outputs from the last LSTM layer and construct a fixed-size representation by concatenating three things: (a) the output at the last time step, (b) the result of max pooling the outputs across the sequence dimension, and (c) the result of average pooling the outputs across the sequence dimension. This fixed-size representation (which is three times the hidden dimension size) is then fed into the classifier head, which consists of two dense layers with batch normalization and dropout. For the GPT-2 model, we take the output from the last transformer layer at the last time step, and then feed it into a single dense (classification) layer. Because the GPT-2 and RoBERTa models require special tokens during training, we insert special symbols <s> and </s> at the beginning and end of every training input, respectively. Because of the masked self-attention, we must use the output of the last token in order to access all of the information in the input sequence. For the RoBERTa model, we take the output from the last transformer layer corresponding to the <s> token, and feed it into a single dense (classification) layer. The <s> takes the place of the special [CLS] token described in the original paper.

We integrated all models into the fastai framework and finetuned the classifier in the following manner. We first select an appropriate learning rate using a range test, in which we sweep the learning rate across a wide range of

Figure 4. Statistics on the target dataset. The top two histograms show the distribution of the number of pages (top left) and number of bootleg score features (top right) per composer. The bottom figure shows the distribution of the number of bootleg score features per page.

values and observe the impact on training loss. We initially freeze all parameters in the model except for the untrained classification head, and we gradually unfreeze more and more layers in the model as the training converges. To avoid overly aggressive changes to the pretrained language model weights, we use discriminative finetuning, in which earlier layers of the model use exponentially smaller learning rates compared to later layers in the model. All training is done with (multiple cycles of) the one cycle training policy [37], in which learning rate and momentum are varied cyclically over each cycle. The above practices were proposed in [22] and found to be effective in finetuning language models for text classification.

2.4 Inference

The third main stage is to apply the proxy classifier to the original full page classification task. We explore two different ways to do this. The first method is to convert the sheet music image into a bootleg score, tokenize the bootleg score into a sequence of word or subword units, and then apply the proxy classifier to a single variable-length input. Note that all of the models can handle variable-length inputs up to a maximum context length. The second method is identical to the first, except that it averages the predictions from multiple fixed-length crops taken from the input sequence. The fixed-length crops are the same size as is used during classifier training, and the crops are sampled uniformly with 50% overlap. [5]

3. EXPERIMENTAL SETUP

In this section we describe the data collection process and the metrics used to evaluate our approach.

The data comes from IMSLP. We first scraped the website and downloaded all PDF scores and accompanying metadata. [6] We filtered the data based on its instrumentation in order to identify a list of solo piano scores. We

---

[5] We also experimented with applying a Bayesian prior to the classifier softmax outputs, as recommended in [36], but found that the results were not consistently better.

[6] We downloaded the data over a span of several weeks in May of 2018.
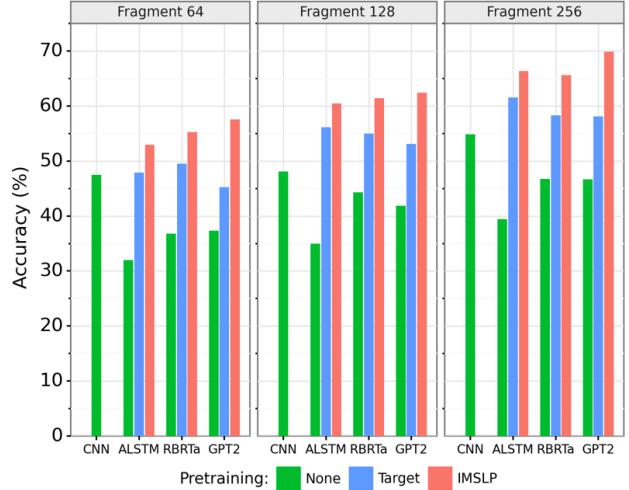
then computed bootleg score features for all of the piano sheet music images using the XSEDE supercomputing infrastructure [38], and discarded any pages that had less than a minimum threshold of features. This latter step is designed to remove non-music pages such as the title page, foreword, or table of contents. The resulting set of data contained $29,310$ PDFs, [7] $255,539$ pages and a total of 48.5 million bootleg score features. This set of data is what we refer to as the IMSLP dataset in this work (e.g. the IMSLP pretrained language model). For language model training, we split the IMSLP data by piece, using 90% for training and 10% for validation.

The classification task uses a subset of the IMSLP data. We first identified a list of composers with a significant amount of data (composers shown in Figure 4). We limited the list to nine composers in order to avoid extreme class imbalance. Because popular pieces tend to have many sheet music versions in the dataset, we select one version per piece in order to avoid over-representation of a small subset of pieces. Next, we manually labeled and discarded all filler pages, and then computed bootleg score features on the remaining sheet music images. This cleaned dataset is what we refer to as the target data in this work (e.g. the target pretrained language model). Figure 4 shows the total number of pages and bootleg score features per composer for the target dataset, along with the distribution of the number of bootleg score features per page. For training and testing, we split the data by piece, using 60% of the pieces for training (4347 pages), 20% for validation (1500 pages), and 20% for testing (1304 pages). To generate data for the proxy task, we randomly sampled fixed-length fragments from the target data. We sample the same number of fragments for each composer to ensure class balance. We experimented with fragment sizes of 64/128/256 and sampled 32400/16200/8100 fragments for training and 10800/5400/2700 fragments for validation/test, respectively. This sampling scheme ensures the same data coverage regardless of fragment length. Note that the classification data is carefully curated, while the IMSLP data requires minimal processing.

We use two different metrics to evaluate our systems. For the proxy task, accuracy is an appropriate metric since the data is balanced. For the full page classification task – which has imbalanced data – we report results in macro F1 score. Macro F1 is a generalization of F1 score to a multi-class setting, in which each class is treated as a one-versus-all binary classification task and the F1 scores from all classes are averaged.

4. RESULTS & ANALYSIS

In this section we present our experimental results and conduct various analyses to answer key questions of interest. While the proxy task is an artificially created task, it provides a more reliable indicator of classifier performance than the full page classification. This is because the test set of the full page classification task is both imbalanced and

Figure 5. Model performance on the proxy classification task. This comparison shows the effect of different pretraining conditions and fragment sizes.

very small (1304 data points). Accordingly, we will report results on both the proxy task and full page classification task.

4.1 Proxy Task
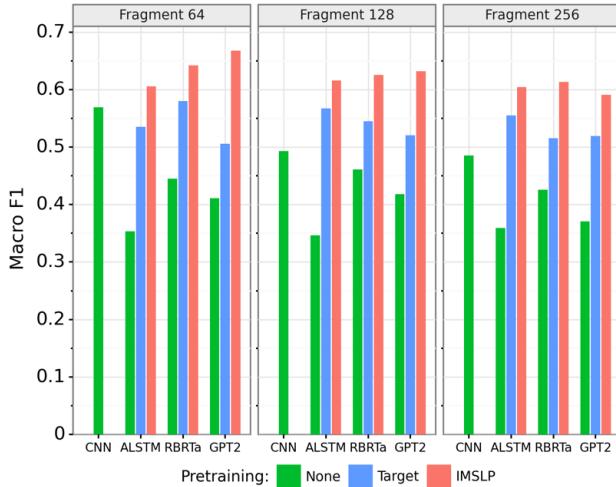
We first consider the performance of our models on the proxy classification task. We would like to understand the effect of (a) model architecture, (b) pretraining condition, and (c) fragment size.

We evaluate four different model architectures. In addition to the AWD-LSTM, GPT-2, and RoBERTa models previously described, we also measure the performance of a CNN-based approach recently proposed in [7]. Note that we cannot use the exact same model in [7] since we do not have symbolic score information. Nonetheless, we can use the same general approach of computing local features, aggregating feature statistics across time, and applying a linear classifier. The design of our 2-layer CNN model roughly matches the architecture proposed in [7].

We consider three different language model pretraining conditions. The first condition is with no pretraining, where we train the classifier from scratch only on the proxy task. The second condition is with target language model pretraining, where we first train a language model on the target data, and then finetune the classifier on the proxy task. The third condition is with IMSLP language model pretraining. Here, we train a language model on the full IMSLP dataset, finetune the language model on the target data, and then finetune the classifier on the proxy task.

Figure 5 shows the performance of all models on the proxy task. There are three things to notice. First, regarding (a), the transformer-based models generally outperform the LSTM and CNN models. Second, regarding (b), language model pretraining improves performance significantly across the board. Regardless of architecture, we see a large improvement going from no pretraining (condition 1) to target pretraining (condition 2), and another
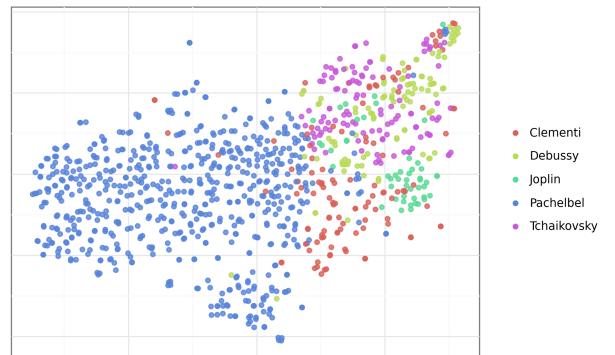
Figure 6. Results on the full page classification task.

Figure 7. t-SNE plot of the RoBERTa model activations for five novel composers. Each data point corresponds to a single page of sheet music for a composer that was *not* considered in the classification task.

large improvement going from target pretraining (condition 2) to IMSLP pretraining (condition 3). For example, the performance of the GPT-2 model increases from 37.3% to 45.2% to 57.5% across the three pretraining conditions. Because the data in conditions 1 & 2 is exactly the same, the improvement in performance must be coming from more effective use of the data. We can interpret this from an information theory perspective by noting that the classification task provides the model $log_2 9 = 3.17$ bits of information per fragment, whereas the language modeling task provides $log_2 V$ bits of information *per bootleg score feature* where V is the vocabulary size. The performance gap between condition 2 and condition 3 can also be interpreted as the result of providing more information to the model, but here the information is coming from having additional data. Third, regarding (c), larger fragments result in better performance, as we might expect.

4.2 Full Page Classification

Next, we consider performance of our models on the full page classification task. We would like to understand the effect of (a) model architecture, (b) pretraining condition, (c) fragment size, and (d) inference type (single vs. multicrop). Regarding (d), we found that taking multiple crops improved results with all models except the CNN. This suggests that this type of test time augmentation does not benefit approaches that simply average feature statistics over time. In the results presented below, we only show the optimal inference type for each model architecture (i.e. CNN with single crop, all others with multi-crop).

Figure 6 shows model performance on the full page classification task. There are two things to notice. First, we see the same general trends as in Figure 5 for model architecture and pretraining condition: the transformer-based models generally outperform the CNN and LSTM models, and pretraining helps substantially in every case. The macro F1 score of our best model (GPT-2 with fragment size 64) increases from 0.41 to 0.51 to 0.67 across the three pretraining conditions. Second, we see the *opposite* trend

as the proxy task for fragment size: smaller fragments have *better* page classification performance. This strongly indicates a data distribution mismatch. Indeed, when we look at the distribution of the number of bootleg score features in a single page (Figure 4), we see that a significant fraction of pages have less than 256 features. Because we only sample fragments that contain a complete set of 256 words, our proxy task data is biased towards longer inputs. This leads to poor performance when the classifier is faced with short inputs, which are never seen in training. Using a fragment size of 64 minimizes this bias.

4.3 t-SNE Plots

Another key question of interest is, "Can we use our model to characterize the style of any page of piano sheet music?" The classification task forces the model to project the sheet music into a feature space where the compositional style of the nine composers can be differentiated. We hypothesize that this feature space might be useful in characterizing the style of *any* page of piano sheet music, even from composers not in the classification task.

To test this hypothesis, we fed data from 5 novel composers into our models and constructed t-SNE plots [39] of the activations at the second-to-last layer. Figure 7 shows such a plot for the RoBERTa model. Each data point corresponds to a single page of sheet music from a novel composer. Even though we have not trained the classifier on these five composers, we can see that the data points are still clustered, suggesting that the feature space can describe the style of new composers in a useful manner.

5. CONCLUSION

We propose a method for predicting the composer of a single page of piano sheet music. Our method first converts the raw sheet music image into a sequence of musical words based on the bootleg score feature representation, and then feeds the sequence into a text classifier. We show that by pretraining a language model on a large set of unlabeled data, it is possible to significantly improve the performance of the classifier.

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] K. C. Kempfert and S. W. Wong, "Where does haydn end and mozart begin? composer classification of string quartets," *arXiv preprint arXiv:1809.05075*, 2018.

[2] P. Sadeghian, C. Wilson, S. Goeddel, and A. Olmsted, "Classification of music by composer using fuzzy min-max neural networks," in *Proc. of the 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2017, pp. 189–192.

[3] A. Brinkman, D. Shanahan, and C. Sapp, "Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of josquin," in *Proc. of the Biennial Int. Conf. on Music Perception and Cognition*, 2016, pp. 91–97.

[4] D. Herremans, D. Martens, and K. Sörensen, "Composer classification models for music-theory building," in *Computational Music Analysis*, 2016, pp. 369–392.

[5] G. Velarde, C. C. Chacón, D. Meredith, T. Weyde, and M. Grachten, "Convolution-based classification of audio and symbolic representations of music," *Journal of New Music Research*, vol. 47, no. 3, pp. 191–205, 2018.

[6] G. Velarde, T. Weyde, C. E. C. Chacón, D. Meredith, and M. Grachten, "Composer recognition based on 2d-filtered piano-rolls," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 115–121.

[7] H. Verma and J. Thickstun, "Convolutional composer classification," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 549–556.

[8] G. Buzzanca, "A supervised learning approach to musical style recognition," in *Proc. of the international conference on music and artificial intelligence (ICMAI)*, vol. 2002, 2002, p. 167.

[9] J. Wołkowicz and V. Kešelj, "Evaluation of N-gram-based classification approaches on classical music corpora," in *International Conference on Mathematics and Computation in Music*, 2013, pp. 213–225.

[10] M. Hontanilla, C. Pérez-Sancho, and J. M. Inesta, "Modeling musical style with language models for composer recognition," in *Iberian Conference on Pattern Recognition and Image Analysis*, 2013, pp. 740–748.

[11] R. Hillewaere, B. Manderick, and D. Conklin, "String quartet classification with monophonic models." in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 537–542.

[12] M. A. Kaliakatsos-Papakostas, M. G. Epitropakis, and M. N. Vrahatis, "Weighted markov chain model for musical composer identification," in *European Conference on the Applications of Evolutionary Computation*, 2011, pp. 334–343.

[13] E. Pollastri and G. Simoncelli, "Classification of melodies by composer with hidden markov models," in *Proc. of the First International Conference on WEB Delivering of Music*, 2001, pp. 88–95.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[16] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. of the conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F.-F. Li, "ImageNet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[18] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[22] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 328–339.

[23] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *OpenAI Blog*, 2018.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[26] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2978–2988.

[27] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5754–5764.

[28] D. Yang, T. Tanprasert, T. Jenrungrot, M. Shan, and T. Tsai, "MIDI passage retrieval using cell phone pictures of sheet music," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 916–923.

[29] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," *arXiv preprint arXiv:1708.02182*, 2017.

[30] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[31] A. Pacha, J. Hajič, and J. Calvo-Zaragoza, "A baseline for general music object detection with deep learning," *Applied Sciences*, vol. 8, no. 9, p. 1488, 2018.

[32] P. Gage, "A new algorithm for data compression," *C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[33] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 1715–1725.

[34] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya, "Multilingual language processing from bytes," in *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1296–1306.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[36] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[37] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[38] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr, "XSEDE: Accelerating scientific discovery," *Computing in Science & Engineering*, vol. 16, no. 5, pp. 62–74, Sept.-Oct. 2014. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MCSE.2014.80

[39] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

USING WEAKLY ALIGNED SCORE–AUDIO PAIRS TO TRAIN DEEP CHROMA MODELS FOR CROSS-MODAL MUSIC RETRIEVAL

Frank Zalkow, Meinard Müller

International Audio Laboratories Erlangen, Germany

{frank.zalkow,meinard.mueller}@audiolabs-erlangen.de

ABSTRACT

Many music information retrieval tasks involve the comparison of a symbolic score representation with an audio recording. A typical strategy is to compare score–audio pairs based on a common mid-level representation, such as chroma features. Several recent studies demonstrated the effectiveness of deep learning models that learn task-specific mid-level representations from temporally aligned training pairs. However, in practice, there is often a lack of strongly aligned training data, in particular for real-world scenarios. In our study, we use weakly aligned score–audio pairs for training, where only the beginning and end of a score excerpt is annotated in an audio recording, without aligned correspondences in between. To exploit such weakly aligned data, we employ the Connectionist Temporal Classification (CTC) loss to train a deep learning model for computing an enhanced chroma representation. We then apply this model to a cross-modal retrieval task, where we aim at finding relevant audio recordings of Western classical music, given a short monophonic musical theme in symbolic notation as a query. We present systematic experiments that show the effectiveness of the CTC-based model for this theme-based retrieval task.

1. INTRODUCTION

Music appears in many different modalities, for example, as audio or video recordings, in the form of symbolic representations, or as graphical sheet music [1]. In particular, audio recordings and symbolic representations are of great importance in many music information retrieval (MIR) tasks. An example is cross-modal retrieval, where a symbolic score is given as a query, and the task is to identify relevant audio recordings [2–4]. A general strategy for matching such different modalities is to use a common mid-level representation. In music processing, chroma features are widely used as mid-level [1, 5, 6]. These features, which capture the energy in the twelve chromatic pitch class bands, are robust against changes in octave, instrumentation, and timbre.
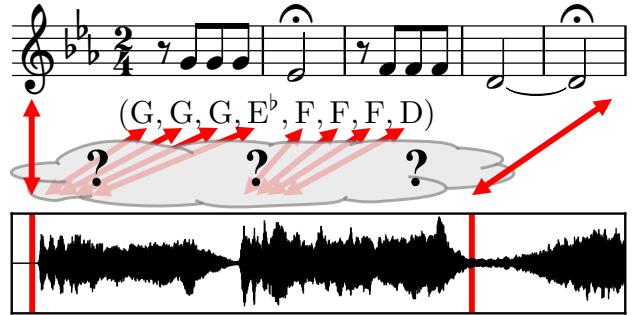
Figure 1. Illustration of a weakly aligned score–audio pair.

In recent years, many studies have shown the benefits of deep learning models to compute task-specific mid-level representations [7–10]. These learned features have proven their effectiveness in many scenarios, for example, audio–audio retrieval [11–13], chord recognition [9, 10, 14], or pitch tracking [7, 15, 16]. Training deep neural networks (DNNs) usually requires aligned training pairs, i.e., in MIR, music recordings with temporally aligned annotations. For example, the training pairs for a deep salience model by Bittner et al. [7] consist of time–frequency representations (more details in Section 2.2) with fundamental frequency annotations, where inputs and annotations correspond to each other for all time frames. For popular music, annotated data sets [17] have led to significant advances in research on pitch salience representations. However, creating such strongly aligned training pairs is labor-intensive, and, for many music scenarios, such data is hardly available. In contrast to the difficulty in annotating local alignments, it may be much easier to annotate global correspondences. In this paper, we use training pairs, where only global correspondences have been annotated. We denote these pairs as weakly aligned.

In our contribution, we use a deep learning model to compute enhanced chroma features, which we then use as a mid-level representation for a cross-modal retrieval task. Given a symbolic representation of a monophonic musical theme as a query and an audio database of Western classical music, the task is to find all audio recordings in which the theme is played [18, 19]. To obtain a task-specific chroma variant, we train a deep learning model with weakly aligned score–audio pairs, where only the beginning and end of a musical theme is annotated in an audio recording. Figure 1 illustrates such a pair for the famous first theme of Beethoven's Symphony No. 5. As our

Themes			Audio Recordings		
#	Mean Dur.	Total Dur.	#	Mean Dur.	Total Dur.
2048	00:00:09	04:54:58	1114	00:06:26	119:28:27

Table 1. Dat set overview. Duration format: hh:mm:ss.

main contribution, we combine a deep salience model [7] with a training procedure for weakly aligned data. [1] This procedure, called *Connectionist Temporal Classification* (CTC) [20], allows us to use training pairs of audio excerpts (in the form of spectral features) as input and musical themes (as sequences of chroma labels) as output. Using this CTC-based strategy, we train a model to compute enhanced chroma features for musical themes. We evaluate these features using more than 2000 themes and 1000 audio recordings and show that they improve the state of the art for our cross-modal retrieval scenario.

In Section 2, we review several prerequisites, such as cross-modal retrieval (Section 2.1), deep salience and deep chroma models (Section 2.2), and the CTC loss (Section 2.3). Then, in Section 3, we describe our adaption of the deep salience model, which computes chroma features and can be trained with the CTC loss. We present our experiments in Section 4 and conclude with Section 5.

2. PRIOR WORK AND PREREQUISITES

2.1 Cross-Modal Retrieval

For our retrieval scenario, we use a data set based on "A Dictionary of Musical Themes" by Barlow and Morgenstern (BM) [21], which contains roughly 10000 musical themes of instrumental Western classical music. Most of these themes have also been available as symbolic versions (MIDI) on the internet. [2] For a subset of the themes, we annotated their occurrences in audio recordings. In these annotations, a theme corresponds to exactly one recording, which, in turn, can correspond to several themes. The annotations comprise global correspondences, i.e., the beginning and end of the occurrences, as well as transpositions. Table 1 shows some statistics for our data set, which consists of 2048 themes from the BM book and 1114 corresponding recordings. The BM book already inspired several MIR studies [22, 23]. Some of them [18, 19] used the same subset for retrieval. We slightly corrected some annotations for this paper. A previous study [18] pointed out the challenges of the task, which are due to the differences in modality (symbolic vs. audio), tuning, transposition, tempo, and polyphony between the query and the recordings. The last point means that the themes are monophonic, but they usually appear in polyphonic context in the recordings (further discussion in Section 5). Previous work [19] has shown that pitch salience representations are capable of overcoming the differences in polyphony. In

this paper, building upon these findings, we introduce an approach for learning a task-specific salience representation.

2.2 Deep Salience and Deep Chroma Models
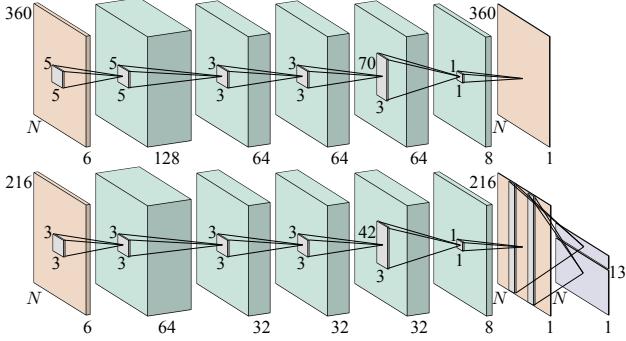
Many studies have demonstrated the effectiveness of using deep learning models to compute task-specific feature representations. One example is the use of deep salience models to compute enhanced time–frequency representations (measuring the saliency of frequencies over time) for tasks such as melody or multi-pitch tracking [7, 15, 16]. Another example is the use of deep chroma models for computing enhanced chroma features (encoding the energy in the twelve chromatic pitch class bands) for chord recognition [9, 10, 14].

This paper is inspired by the deep salience approach by Bittner et al. [7]. They introduced a feature representation named harmonic CQT (HCQT) as input for a convolutional DNN. The HCQT is a three-dimensional tensor, where the three dimensions are time, frequency (logarithmic scaling), and harmonics. The third dimension ensures that harmonically related frequency bins are neighbors across the depth of the tensor. This way, the convolutional kernels of the network can easily exploit harmonic relationships. Many studies use this deep salience representation as a baseline [16, 24] or build upon this model for diverse tasks such as dominant melody estimation [8], instrument recognition [25], tempo estimation [26], or chord recognition [27]. In Section 3, we describe how we adapt the deep salience model for computing enhanced chroma features.

The study of Wu et al. [27] is related to ours in two respects. First, they also use the HCQT representation, and, second, they use weakly aligned training data. However, they aim for chord recognition instead of learning a mid-level representation for cross-modal retrieval. Unlike us, they take a three-step approach: First, they use a pre-trained deep chroma extractor to compute features. Second, they strongly align their annotations to the chroma features using a hidden Markov model. Third, they use a frame-wise DNN classifier for chord recognition. In our paper, we present a single-step approach to realize the alignment within the DNN training procedure.

2.3 CTC Loss

Graves et al. [20] originally introduced *Connectionist Temporal Classification* (CTC) as the task of labeling unsegmented feature sequences with recurrent DNNs in the context of speech recognition. However, their training technique can be used with any DNN architecture. Furthermore, the task can be generalized to any scenario, where the aim is to map feature sequences to sequences of symbols. If the training data consisted of strongly aligned pairs of feature and symbol sequences (i.e., each vector of the feature sequences is labeled with a symbol), then a standard classification approach could be taken. The key aspect of CTC is that there is no need for strongly aligned training data, i.e., the feature and symbol sequences may be of

---

[1] Pre-trained models and code to apply them are available at `https://www.audiolabs-erlangen.de/resources/MIR/2020-ISMIR-ctc-chroma`.

[2] Unfortunately, the page is now offline. It is still reachable with the Wayback Machine without access to the MIDI files: `https://web.archive.org/web/20160209045946/http://www.multimedialibrary.com/barlow/index.asp`

Layer	Output Shape	Activation	Parameters
Input	$(N, 216, 6)$		
Conv2D $64 \times (3, 3, 6)$	$(N, 216, 64)$	LReLU	3520
Conv2D $32 \times (3, 3, 64)$	$(N, 216, 32)$	LReLU	18464
Conv2D $32 \times (3, 3, 32)$	$(N, 216, 32)$	LReLU	9248
Conv2D $32 \times (3, 3, 32)$	$(N, 216, 32)$	LReLU	9248
Conv2D $8 \times (42, 3, 32)$	$(N, 216, 8)$	LReLU	32264
Conv2D $1 \times (1, 1, 8)$	$(N, 216, 1)$	Sigmoid	9
Pooling	$(N, 13)$	Softmax	217

Table 2. Details of the used DNN model (72970 parameters in total).

Figure 2. Network architectures. **Upper:** Original architecture proposed by Bittner et al. [7]. **Lower**: Adapted architecture used in this paper. Illustration inspired by [7].

different length, and the temporal correspondence between both sequences is unknown and may be non-linear.

Several studies from the MIR community used CTC, e.g., for optical music recognition [28], monophonic audio-to-score transcription [29], lyrics alignment [30], and audio tagging [31]. An alternative to CTC for sequence learning without aligned training data is the usage of an attention mechanism, which, e.g., was used for monophonic singing voice transcription [32].

In the following, we give the main idea of the CTC loss function introduced by Graves et al. [20] We describe the computation of the CTC loss for a single pair consisting of an audio feature sequence and a symbol sequence. Let

$$X = (x_1, x_2, \ldots, x_N) \tag{1}$$

denote the feature sequence of length $N \in \mathbb{N}$, which consists of feature vectors $x_n \in \mathbb{R}^D$ for $n \in [1 : N] :=$ $\{1, 2, \ldots, N\}$ of dimensionality $D \in \mathbb{N}$. The second sequence of the pair is a symbol sequence

$$Y = (y_1, y_2, \ldots, y_M) \tag{2}$$

of length $M \in \mathbb{N}$, which consists of elements $y_m \in \mathbb{A}$ for $m \in [1 : M]$. The alphabet $\mathbb{A}$ of size $A := |\mathbb{A}|$ is the set of symbols that can occur in the symbol sequence. Typically $M \ll N$. For example, in the case of lyrics alignment, the alphabet is the set of all considered characters [30]. In our case, it is the set of the twelve different chroma labels. The feature sequence X is transformed by a DNN f_θ with parameters θ to a sequence of probability vectors

$$f_\theta(X) = P = (p_1, p_2, \ldots, p_N) \tag{3}$$

having the same length N as the feature sequence and consisting of probability vectors $p_n \in [0, 1]^A$. We interpret the probability vector element $p_{n,a}$ for $a \in [1 : A]$ as the probability that the nth feature vector x_n corresponds to the ath symbol in $\mathbb{A}$ (assuming an order of the set).

We can now compute the probability of the symbol sequence Y given the feature sequence X. For a fixed alignment between X and Y, one multiplies all values of the probability sequence P that correspond to that alignment. Since the alignment is unknown, instead of a specific one, all possible alignments between X and Y are taken into

account. Let us denote this overall probability as $\hat{p} \in \mathbb{R}$. Graves et al. [20] described how to compute $\hat{p}$ in a differentiable and efficient way using dynamic programming similar to the forward algorithm for hidden Markov models [33]. The final CTC loss for a single training pair is

$$L_\theta(X, Y) = -\log \hat{p}. \tag{4}$$

This loss function is used in batch gradient descent to update the parameters θ by averaging the loss value over multiple training pairs in a batch. By this procedure, the parameters of the network improve to produce probability sequences that make the ground-truth symbol sequences more probable.

In our explanation, we left out a crucial detail of the procedure. For a fixed alignment between X and Y, the aligned symbol sequence can be represented by an "unfolded" sequence of length N that contains the active symbol for each time step. Let us consider the case of an unfolded sequence having multiple neighboring time steps with the same active symbol. So far, we cannot tell if this means one symbol occurrence with a long duration or multiple successive occurrences of the same symbol with shorter durations. To solve this ambiguity, an additional symbol named blank ϵ is part of the alphabet $\mathbb{A}$. This symbol serves two purposes: First, it means that no symbol is active. Second, it indicates a repeated occurrence of the same symbol if a succession of the same active symbol in the unfolded sequence is only interrupted by ϵ.

3. DEEP SALIENCE MODEL ADAPTATION

The DNN model used in this paper is inspired by the deep salience model proposed by Bittner et al. [7]. In this section, we explain our adaption of the model.

Bittner et al. [7] approached the task of melody and multi-pitch tracking, using a strongly aligned data set of 10 hours. In our case, we aim to learn an enhanced chroma representation for cross-modal retrieval, employing our weakly aligned 5-hour data set of 2048 themes. We simplified the original model in several ways to reduce the number of parameters and memory requirements. Additionally, we adapted the network so that it can be trained with the CTC loss and used as a deep chroma extractor. Figure 2 illustrates the original network architecture and our adapted version, and Table 2 gives further details for our version. Compared to the model by Bittner et al. [7],

we introduce the following modifications: First, we use a frame rate of 25 Hz instead of 86 Hz. Second, we use a frequency resolution of a third semitone instead of a fifth semitone. This resolution results in 216 instead of 360 frequency bins. Third, we reduced the number of filter kernels as well as the size of some of the filter kernels. The latter reduction accounts for the decreased frequency resolution. Forth, we use leaky ReLU activations instead of ReLU activations to avoid zero gradients [34]. Fifth, we do not use batch normalization at all, which was used at the input to each layer in the original model. Instead, we ℓ^2-normalize all columns of the input to the network for being invariant to dynamic changes. Sixth, we add a pooling layer at the end, which we explain in the next paragraph.
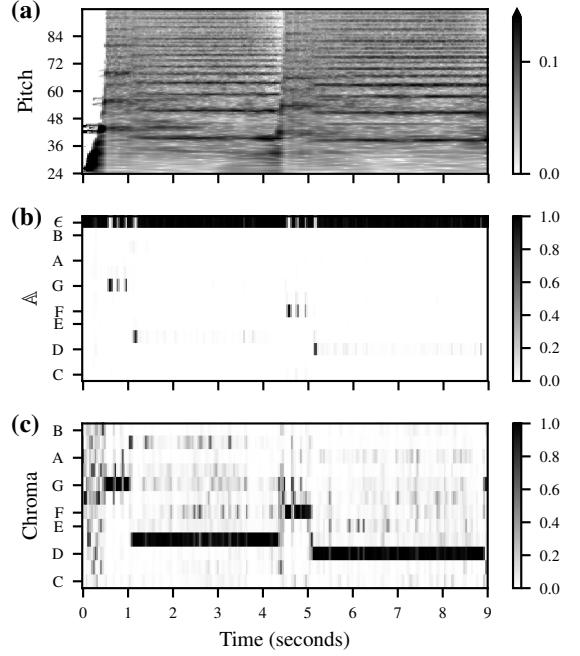
After the last convolutional layer (with sigmoid activation), we obtain a representation that we could interpret as a kind of pitch salience of size $N \times 216$. In our case, we aim for an output size of $N \times 13$, where the N columns are probability vectors over the set of the twelve chroma labels and an additional ϵ symbol:

$$\mathbb{A} := \{C, C^{\#}, D, \ldots, B\} \cup \{\epsilon\}. \tag{5}$$

Let us consider a single column of size 216 as input, which we want to transform to a probability vector of size 13. To compute the first twelve entries, we add up all pitch bins corresponding to the respective chroma bins. This fixed pooling has no learnable parameters. To compute the last entry for the ϵ symbol, we apply a standard dense layer (linear activation) to the input column. This layer has 217 learnable parameters (216 weights and a bias). Finally, we apply the softmax function to the resulting 13-dimensional vector. We repeat this process for all columns of the input.

In summary, our adapted model differs from the original model [7] in two important aspects: First, we reduced the number of parameters from 407 thousand to 73 thousand. Second, the output of the model is a probability matrix over the set $\mathbb{A}$ instead of a pitch salience representation.

We train this adapted model with the CTC loss, as described in Section 2.3. The input to the network is an HCQT tensor computed for an excerpt from an audio recording, where a musical theme is played. Figure 3a shows a slice of the HCQT features for a recording of the first theme of Beethoven's Fifth Symphony. The corresponding symbol sequence is the sequence of chroma labels of the theme with neither any rhythmic information nor any temporal alignment to the input. For our Beethoven example (see also Figure 1), this sequence is $Y = (G, G, G, E^{\flat}, F, F, F, D)$. Figure 3b visualizes the probability sequence for the Beethoven example after training. We see that the ϵ symbol has the largest probability for most of the time, and the chroma labels only have large probabilities at the beginning of the corresponding note events. To use the network output as a feature representation, we remove the row corresponding to the ϵ symbol and interpret the resulting matrix as chroma features. Finally, we ℓ^2-normalize the 12-dimensional chroma vectors to increase the energies in the time segments, where ϵ was dominating. Figure 3c shows the normalized chroma features, which correspond well with the symbol sequence.

Figure 3. Representations for the first theme of Beethoven's Fifth Symphony. **(a)** HCQT input representation X (slice corresponding to the first harmonic). **(b)** Network output P. **(c)** Features used for matching.

4. EXPERIMENTS

4.1 Training Details

We split our data set into five folds, where we use three folds for training, one for validation, and another one for testing. We ensure that all themes by a composer are part of precisely one fold. As a consequence, we do not use themes from the same composer for training and evaluation, thus avoiding a "composer overfitting." For the training folds, we perform transpositions (up to a minor third upwards and downwards) as data augmentation. We perform batch gradient descent with a batch size of eight using the Adam optimizer [35] and a learning rate annealing procedure. In the first phase of this procedure, the initial learning rate is 0.001, and we train the model until the loss for the validation fold does not improve for five epochs. In the next phase, we halve the learning rate and continue the training with the model that has the lowest validation loss among the models of all previous epochs. We repeat ten such phases. When we finished training, we use the model with the lowest validation loss as a chroma feature extractor, and evaluate its effectiveness in the retrieval scenario, using the query themes from the test fold.

4.2 Retrieval-Based Evaluation

We shortly describe our retrieval pipeline and our evaluation measures following [18, 19]. First, we have a set $\mathbb{Q}$ of symbolic (MIDI) encodings of musical themes, which serve as *queries*. Furthermore, we have a collection of audio recordings, which we denote as database *documents*. These are actual recordings, not synthesized MIDI files. For each query, there is exactly one audio document that

(a)

	Top-01	Top-05	Top-10	Top-20	Top-50	MRR
$\mathcal{C}_{\text{BG1}}$	0.754	0.835	0.861	0.885	0.913	0.792
$\mathcal{C}_{\text{Bit}}$	0.693	0.788	0.823	0.853	0.896	0.739

(b)

	Top-01	Top-05	Top-10	Top-20	Top-50	MRR
$\mathcal{C}_{\text{BG1}}$	0.820	0.892	0.910	0.925	0.952	0.854
$\mathcal{C}_{\text{Bit}}$	0.763	0.844	0.867	0.895	0.931	0.802

Table 3. Retrieval results of the baseline methods **(a)** using a feature rate of $10\,\text{Hz}$ as reported in previous work [19], **(b)** using a feature rate of $25\,\text{Hz}$.

	$|\mathbb{Q}|$	Top-01	Top-05	Top-10	Top-20	Top-50	MRR
1	559	0.891	0.946	0.961	0.971	0.977	0.918
2	373	0.823	0.887	0.917	0.938	0.954	0.855
3	372	0.839	0.911	0.933	0.944	0.954	0.872
4	372	0.903	0.949	0.952	0.962	0.976	0.922
5	372	0.855	0.919	0.935	0.949	0.976	0.885
⌀		0.865	0.925	0.941	0.955	0.968	0.893

Table 4. Retrieval results for $\mathcal{C}_{\text{CTC}}$.

contains a globally corresponding rendition of the query theme (i.e., matching duration and transposition). For a fixed symbolic query, the aim is to retrieve the corresponding audio document. To compare the query with a document, we convert both into chroma sequences. For the symbolic query, we simply compute a binary chroma representation. For converting the audio recording, we employ a salience representation (from our CTC or a baseline approach). Then, we use Subsequence Dynamic Time Warping (SDTW) to compare the query with subsequences of the document [1]. In particular, we use the cosine distance, the step size condition $\Sigma := \{(2,1),(1,2),(1,1)\}$, as well as the weights $w_{\text{vertical}} = 2$ and $w_{\text{horizontal}} = w_{\text{diagonal}} = 1$. As a result of SDTW, one obtains a matching function, where local minima point to locations with a good match between the query and a document subsequence. We consider the minimal value of the matching function as the distance between query and document.

To solve the retrieval task, we compute distances between all documents and the query. We then order the documents according to ascending distance values. The document's position in this ordered list is called the *rank* $r \in \mathbb{N}$ of the document. The top-K evaluation metric yields a value of one if the relevant document is among the top K matches, i.e., $r \leq K$. We then average this metric across all queries. Furthermore, we report the *mean reciprocal rank* (MRR), which is the average of $1/r$ across all queries.

In the cross-validation iterations of our evaluation, we only use the query themes from the respective test fold to search within the 1114 documents of our database. The reported average evaluation measures (⌀) are weighted with the number of queries from the respective test fold.

4.3 Baseline

As for our baselines, we consider the best-performing representations from a previous study [19], namely $\mathcal{C}_{\text{Bit}}$, using the original deep salience model for melody estimation by Bittner et al. [7][3], and $\mathcal{C}_{\text{BG1}}$, using a model-based salience representation by Bosch and Gómez [36]. The latter one is a combination of a source-filter model with harmonic summation, using threshold parameters (named "BG1") that are particularly suited for orchestral music [37].

Table 3a cites the results from the previous study [19], where a $10\,\text{Hz}$ feature rate was used. Since we use an in-

creased feature rate of $25\,\text{Hz}$ in this paper, we reproduced the experiments with this rate. The results are shown in Table 3b. Just by changing the feature rate, we see a substantial improvement of the results. For example, for $\mathcal{C}_{\text{Bit}}$, the top-1 rate increases from 0.693 to 0.763, which means that 7 % more themes achieved a rank of 1. Since we corrected some errors in the data set, an improvement of up to 2 % may be due to the revision, but the main improvements are due to the increased time resolution. The reason for this may be the following: A fast tempo of *Presto* corresponds to up to $200\,\text{BPM}$. Having a quarter-note beat, in such a tempo, a sixteenth note has a duration of $75\,\text{ms}$, which is shorter than the length of a frame given the feature rate of $10\,\text{Hz}$. In such cases, the increased feature rate is necessary to represent the musical content in a more meaningful way.

For both feature rates, the representation $\mathcal{C}_{\text{BG1}}$ performs better than $\mathcal{C}_{\text{Bit}}$. For example, the respective top-1 rates are 0.820 and 0.763 for the $25\,\text{Hz}$ rate. The results for $\mathcal{C}_{\text{Bit}}$ may be lower because the training data of the underlying DNN consisted mainly of popular music (for overall 240 training tracks, only 22 are tagged as "classical" in version 1 of MedleyDB [17]). Another possible reason is that the saliency characteristics in the training data (coming from the "Melody 2" definition of MedleyDB) are different from the characteristics of musical themes.

4.4 CTC-Based Results

We now discuss the results we achieved with our CTC-based approach $\mathcal{C}_{\text{CTC}}$. Table 4 shows the evaluation results for the five cross-validation iterations. The second column ($|\mathbb{Q}|$) gives the number of query themes in the respective test fold. This number is larger in the first fold (559) because this fold contains all BM themes by Ludwig van Beethoven, which is the most prominent composer of our data set. The other folds have fewer queries (372 or 373) and are more diverse in terms of composers, having 12 or 13 different composers each. The retrieval results have some diversity, ranging from a top-1 rate of 0.823 for test fold 2 up to 0.903 for test fold 4. The last row of the table shows an average of the results, weighted by the number of queries used. Overall, we see a substantial improvement compared to the baseline approaches (Table 3b). For example, the average top-1 rate is 0.865 for $\mathcal{C}_{\text{CTC}}$, compared to 0.763 for $\mathcal{C}_{\text{Bit}}$ and 0.820 for $\mathcal{C}_{\text{BG1}}$. Improvements for larger ranks can also be seen, such as in the top-50 rate (0.968 compared to 0.931 and 0.952, respectively). The results show that our approach is able to outperform the baselines, which have been the state of the art for the task [19].

---

[3] Original weights ("Melody 2"). $\mathcal{C}_{\text{Bit}}$ was denoted by $\mathcal{C}_{\text{CNN}}$ in [19].

	Top-01	Top-05	Top-10	Top-20	Top-50	MRR
$\mathcal{C}_{\text{CTC}}$	0.865	0.925	0.941	0.955	0.968	0.893
$\mathcal{C}_{\text{CCE}}$	0.814	0.890	0.907	0.929	0.951	0.849

Table 5. Retrieval results ($\varnothing$) using cross-entropy.

	Top-01	Top-05	Top-10	Top-20	Top-50	MRR
$\mathcal{C}_{\text{BG1}}$	0.820	0.892	0.910	0.925	0.952	0.854
$\mathcal{C}_{\text{CTC}}$	0.865	0.925	0.941	0.955	0.968	0.893
Oracle	0.904	0.947	0.958	0.967	0.983	0.924

Table 6. Retrieval results ($\varnothing$) for an oracle of the baseline by Bosch and Gómez [36] and our CTC approach.

4.5 Importance of CTC-Alignment

To verify the need for the CTC procedure in our scenario, we performed an additional experiment, where we assumed a linear temporal alignment between the symbolic themes and the corresponding excerpts in the audio recordings. Here, we changed the training procedure from our CTC strategy to a standard classification approach, using categorical cross-entropy (CCE). As output labels, we used binary chroma representations that we obtained by linearly scaling the symbolic themes to the same length as the corresponding audio excerpts. The ϵ symbol here only indicates rests in a theme. Note that, in this experiment, we used the rhythm information and note durations from the MIDI files, which we did not use in the CTC approach.
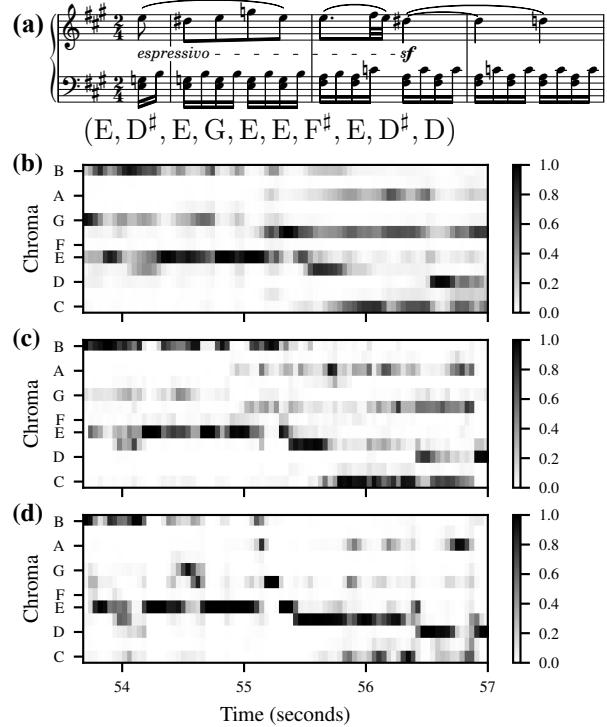
The trained model was used as chroma extractor and then evaluated in the theme retrieval context. The first row of Table 5 repeats the average evaluation measures from Table 4 for convenience and the second row presents the average results for the CCE approach. The evaluation measures are lower compared to the CTC-based results, e.g., having a top-1 rate of 0.814 compared to 0.865. This difference is due to the non-linear temporal correspondence between audio recordings and the symbolic themes.

4.6 Oracle Experiment

The model-based approach $\mathcal{C}_{\text{BG1}}$ also shows excellent performance for this task. To investigate the relationship between $\mathcal{C}_{\text{BG1}}$ and the $\mathcal{C}_{\text{CTC}}$, we evaluated both strategies with an oracle procedure. For each query, we took the better rank: either achieved with $\mathcal{C}_{\text{BG1}}$ or $\mathcal{C}_{\text{CTC}}$. Table 6 repeats the results for the baseline and CTC approaches for convenience and shows the oracle results in the third row. The oracle further improves the results for $\mathcal{C}_{\text{CTC}}$. For example, the top-1 rate is 4 % larger (0.904 instead of 0.865). For top-K rates with larger K, there are still some small improvements. The oracle indicates that for some queries, $\mathcal{C}_{\text{BG1}}$ is a slightly better feature representation than $\mathcal{C}_{\text{CTC}}$.

5. CONCLUSION

In this paper, we showed the potential of CTC [20] for training a deep salience model with weakly aligned data. Adapting a model by Bittner et al. [7] to compute a task-specific mid-level representation, we improved state-of-the-art results for a cross-modal retrieval task for musical

Figure 4. Second theme of Beethoven's Piano Sonata Op. 2, No. 2, first movement. **(a)** Full score with the chroma sequence of the theme, **(b)** standard chroma features using the full spectral content, **(c)** $\mathcal{C}_{\text{BG1}}$, **(d)** $\mathcal{C}_{\text{CTC}}$.

themes. To achieve these improvements, the feature computation procedure has to reduce the potential polyphony of the audio recording, which is a major challenge. We close our paper with a qualitative example to show the feature's properties for a representative polyphonic example.

Figure 4a shows the full score and the chroma sequence for the second theme in the first movement of Beethoven's Piano Sonata Op. 2, No. 2. In this case, the theme is played by the right hand (upper staff), and the left hand (lower staff) plays an accompaniment. The sixteenth notes of the accompaniment present a minor triad (E, G, B) in the first half and a diminished triad $(F^\sharp, A, C)$ in the second half. Ideally, for our retrieval scenario, we aim for a chroma representation that only captures energy from the theme and not from the accompaniment. Figures 4b, 4c, and 4d show chroma features for the full spectral content, the baseline salience approach $\mathcal{C}_{\text{BG1}}$, and our CTC strategy $\mathcal{C}_{\text{CTC}}$, respectively. In all representations, the main notes of the theme are well represented. However, some shorter notes of the theme (e.g., fourth note G or seventh note $F^\sharp$) are most evident in $\mathcal{C}_{\text{CTC}}$. In general, $\mathcal{C}_{\text{CTC}}$ attenuates the energy in the chroma bands corresponding to the accompaniment. The ability to represent the chroma energy of a musical theme is the main reason why our CTC-based features are a powerful tool for cross-modal retrieval.

In this study, we excluded the challenges due to differences in transposition. This could be taken into account by circularly shifting the chroma features [18], or by incorporating it into the learning procedure [38]. Furthermore, our oracle experiment suggests a possible next step of combining our strategy with traditional salience approaches [36].

6. REFERENCES

[1] M. Müller, *Fundamentals of Music Processing*. Springer Verlag, 2015.

[2] J. Pickens, J. P. Bello, G. Monti, T. Crawford, M. Dovey, M. B. Sandler, and D. Byrd, "Polyphonic score retrieval using polyphonic audio," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2002.

[3] I. S. Suyoto, A. L. Uitdenbogerd, and F. Scholer, "Searching musical audio using symbolic queries," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 372–381, 2008.

[4] N. Hu, R. B. Dannenberg, and G. Tzanetakis, "Polyphonic audio matching and alignment for music retrieval," in *Proceedings of the Workshop on Applications of Signal Processing (WASPAA)*, New Paltz, New York, USA, 2003, pp. 185–188.

[5] E. Gómez, "Tonal description of music audio signals," PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2006.

[6] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 96–104, 2005.

[7] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for F0 tracking in polyphonic music," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 63–70.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI*, Munich, Germany, 2015, pp. 234–241.

[9] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016, pp. 37–43.

[10] ——, "A fully convolutional deep auditory model for musical chord recognition," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Salerno, Italy, 2016.

[11] F. Zalkow and M. Müller, "Learning low-dimensional embeddings of audio shingles for cross-version retrieval of classical music," *Applied Sciences*, vol. 10, no. 1, 2020.

[12] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 107–114.

[13] F. Yesiler, J. Serrà, and E. Gómez, "Accurate and scalable version identification using musically-motivated embeddings," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 21–25.

[14] Y. Wu and W. Li, "Automatic audio chord recognition with midi-trained deep feature and BLSTM-CRF sequence decoding model," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 2, pp. 355–366, 2019.

[15] S. Balke, C. Dittmar, J. Abeßer, and M. Müller, "Data-driven solo voice enhancement for jazz music retrieval," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, New Orleans, Louisiana, USA, 2017, pp. 196–200.

[16] D. Basaran, S. Essid, and G. Peeters, "Main melody estimation with source-filter NMF and CRNN," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 82–89.

[17] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "MedleyDB: A multitrack dataset for annotation-intensive MIR research," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014, pp. 155–160.

[18] S. Balke, V. Arifi-Müller, L. Lamprecht, and M. Müller, "Retrieving audio recordings using musical themes," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Shanghai, China, 2016, pp. 281–285.

[19] F. Zalkow, S. Balke, and M. Müller, "Evaluating salience representations for cross-modal retrieval of western classical music recordings," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, United Kingdom, 2019, pp. 311–335.

[20] A. Graves, S. Fernández, F. J. Gomez, and J. Schmid-huber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, Pittsburgh, Pennsylvania, USA, 2006, pp. 369–376.

[21] H. Barlow and S. Morgenstern, *A Dictionary of Musical Themes*, revised edition third printing ed. Crown Publishers, Inc., 1975.

[22] L. Prechelt and R. Typke, "An interface for melody input," *ACM Transactions on Computer-Human Interaction*, vol. 8, no. 2, pp. 133–149, 2001.

[23] S. Balke, S. P. Achankunju, and M. Müller, "Matching musical themes based on noisy OCR and OMR input," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brisbane, Australia, 2015, pp. 703–707.

[24] L. Su, "Vocal melody extraction using patch-based cnn," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 371–375.

[25] Y.-N. Hung and Y.-H. Yang, "Frame-level instrument recognition by timbre and pitch," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 135–142.

[26] H. F. Aarabi and G. Peeters, "Deep-rhythm for global tempo estimation in music," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 636–643.

[27] Y. Wu, T. Carsault, and K. Yoshii, "Automatic chord estimation based on a frame-wise convolutional recurrent neural network with non-aligned annotations," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, A Coruña, Spain, 2019, pp. 1–5.

[28] J. Calvo-Zaragoza, J. J. Valero-Mas, and A. Pertusa, "End-to-end optical music recognition using neural networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 472–477.

[29] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza, "An end-to-end framework for audio-to-score music transcription on monophonic excerpts," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 34–41.

[30] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, United Kingdom, 2019, pp. 181–185.

[31] Y. Hou, Q. Kong, and S. Li, "Audio tagging with connectionist temporal classification model using sequentially labelled data," in *Proceedings of the International Conference in Communications, Signal Processing, and Systems (CSPS)*, Dalian, China, 2019, pp. 955–964.

[32] R. Nishikimi, E. Nakamura, S. Fukayama, M. Goto, and K. Yoshii, "Automatic singing transcription based on encoder-decoder recurrent neural networks with a weakly-supervised attention mechanism," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, United Kingdom, 2019, pp. 161–165.

[33] A. Hannun, "Transcribing real-valued sequences with deep neural network," Ph.D. dissertation, Stanford University, 2018.

[34] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, Georgia, USA, 2013.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference for Learning Representations (ICLR)*, San Diego, California, USA, 2015.

[36] J. J. Bosch and E. Gómez, "Melody extraction based on a source-filter model using pitch contour selection," in *Proceedings of the 13th Sound and Music Computing Conference (SMC)*, Hamburg, Germany, 2016, pp. 67–74.

[37] ——, "Melody extraction for MIREX 2016," in *Music Information Retrieval Evaluation eXchange (MIREX) System Abstracts*, 2016.

[38] A. Arzt and S. Lattner, "Audio-to-score alignment using transposition-invariant features," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 592–599.

INVESTIGATING U-NETS WITH VARIOUS INTERMEDIATE BLOCKS FOR SPECTROGRAM-BASED SINGING VOICE SEPARATION

Woosung Choi[1]   **Minseok Kim**[1]   **Jaehwa Chung**[2]
Daewon Lee[3]   **Soonyoung Jung**[1]

[1] Department of Computer Science and Engineering, Korea University, Republic of Korea
[2] Department of Computer Science, Korea National Open University, Republic of Korea
[3] Department of Computer Engineering, Seokyeong University, Republic of Korea

jsy@korea.ac.kr

ABSTRACT

Singing Voice Separation (SVS) tries to separate singing voice from a given mixed musical signal. Recently, many U-Net-based models have been proposed for the SVS task, but there were no existing works that evaluate and compare various types of intermediate blocks that can be used in the U-Net architecture. In this paper, we introduce a variety of intermediate spectrogram transformation blocks. We implement U-nets based on these blocks and train them on complex-valued spectrograms to consider both magnitude and phase. These networks are then compared on the SDR metric. When using a particular block composed of convolutional and fully-connected layers, it achieves state-of-the-art SDR on the MUSDB singing voice separation task by a large margin of 0.9 dB. Our code and models are available online. [1]

1. INTRODUCTION

Singing Voice Separation (SVS), a special case of Music Source Separation (MSS), aims at separating singing voice from a given mixed musical signal. Recently, many machine learning-based methods have been proposed for SVS and MSS tasks. They can be categorized into two groups: waveform-to-waveform models and spectrogram-based models. While the former tries to generate the vocal waveforms directly, the latter estimates spectrograms (usually magnitude) of vocal waveforms.

Typical spectrogram-based models apply Short-Time Fourier Transform (STFT) on a mixture waveform to obtain the input spectrograms. Then, they estimate the vocal spectrograms based on these inputs and finally restore the vocal waveform with inverse STFT (iSTFT). A variety of spectrogram-based models have been proposed in

the music information retrieval community and the machine learning community. For example, [1] employed the U-Net [2] architecture, an encoder-decoder structure with symmetric skip connections. These symmetric skip connections allow models to recover fine-grained details of the target object during decoding effectively. Several works [3–6] also used similar architectures.

They have revealed that U-Net-like architectures can provide promising performance for SVS and MSS. Existing works have proposed various types of neural networks for intermediate blocks. While some models [1, 3] used simple Convolutional Neural Networks (CNNs) for intermediate blocks, other advanced models tried more complex intermediate blocks. For instance, MMDenseLSTM [6] used densely connected CNNs followed by Long Short-Term Memory (LSTM) networks to efficiently model long-term structures, where LSTM is a variant of Recurrent Neural Networks (RNNs). However, a thorough search of the relevant literature indicated that there were no existing works that evaluate and directly compare these different types of blocks.

In this paper, we conduct a comparative study of U-Nets on various intermediate blocks. We designed several types of blocks based on different design strategies, which we present in section 3. For each type of block, we implemented at least one SVS model, which are all based on an identical U-Net framework for fair comparisons. In section 4, we summarize the experimental results and discuss the effect of each design choice. We validate hypotheses such as that inserting time-distributed operations (see §3.1) into intermediate blocks can significantly improve performance, which led to state-of-the-art (SOTA) performance on the MUSDB [7] SVS task.

Finally, our U-Net framework directly estimates the target complex-valued spectrogram (viewing real and imaginary as separate channels), when many existing models estimate the target magnitude without phase. In general, considering phase information improves the separation quality, as discussed in [8, 9]. Several phase-aware methods have been proposed for speech enhancement, such as phase reconstruction methods [8,9], or using raw complex-valued STFT outputs [10, 11]. In section 4, we show that the latter method is an efficient way to improve magnitude-only models, only needing a few minor adjustments.

---

[1] https://github.com/ws-choi/ISMIR2020_U_Nets_SVS
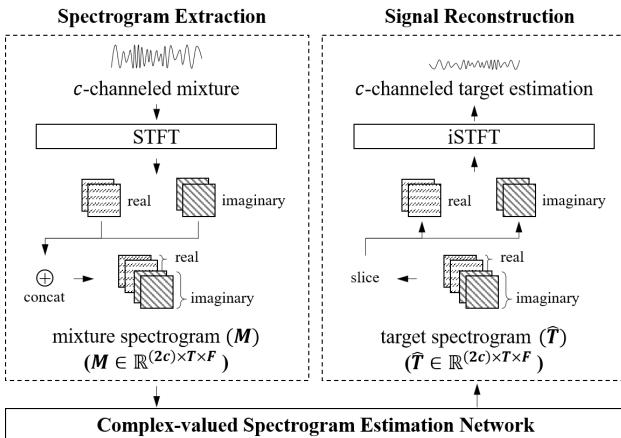
2. U-NET-BASED SVS FRAMEWORK

In this section, we describe a U-Net-based SVS framework, which is shared by several models in §4. We first introduce the 'Complex as Channel framework' (CaC), a spectrogram-based SVS framework, and then define our U-Net architecture for spectrogram estimation in CaC.

2.1 Complex as Channel Framework

CaC is a singing voice separation framework based on complex-valued spectrogram estimation. It takes a c-channeled mixture signal, and outputs c-channeled singing voice signal. As shown in Figure 1, CaC consists of three parts as follows:

1. The *spectrogram extraction layer* extracts a mixture spectrogram by applying STFT to the c-channeled input signal. The output of STFT is a complex-valued spectrogram with c-channels. Considering the imaginary and real parts as separate real-valued channels, we view the mixture spectrogram $M_{complex} \in \mathbb{C}^{c \times T \times F}$ as a $(2c)$-channeled real-valued spectrogram $M \in \mathbb{R}^{(2c) \times T \times F}$, where T denotes the number of frames and F denotes the number of the frequency bins in the spectrogram.

2. The *complex-valued spectrogram estimation network* is a neural network that takes the spectrogram $\mathbf{M}$ of a mixture signal as input and estimates the target spectrogram $\hat{T} \in \mathbb{R}^{(2c) \times T \times F}$, which is used for reconstructing the vocal signal later.

3. The *signal reconstruction layer* reshapes the estimated spectrogram $\hat{T}$ into the complex-valued spectrogram $\hat{T}_{complex} \in \mathbb{C}^{c \times T \times F}$, as shown in Figure 1. It then restores the estimated singing voice signal via inverse-STFT on $\hat{T}_{complex}$.

Figure 1. The Complex as Channel Framework
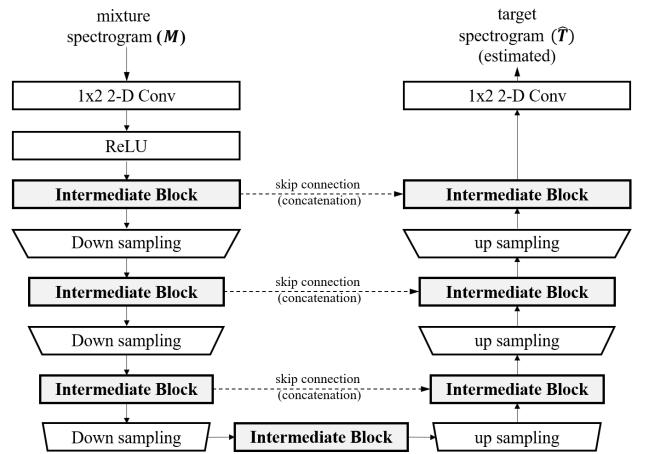
For a given mixture spectrogram $\mathbf{M}$, we train the complex-valued spectrogram estimation network in a supervised fashion to minimize the mean square error between the output $\hat{T}$ and the ground-truth spectrogram $\mathbf{T}$ of the singing voice signal.

It should be noted that the shape of $\mathbf{M}$ and $\hat{T}$ is $(2c) \times T \times F$, considering real and imaginary parts of a spectrogram as separate real-valued channels. This approach allows CaC to fully utilize the information in complex-valued spectrograms for both the input and the output. Meanwhile, current SOTA models (e.g., SA-SHN [4] and DGRU-DGConv [12]) decompose a complex-valued spectrogram into magnitude and phase, and only use the magnitude for the input of their networks. Although SA-SHN and DGRU-DGConv yielded impressive results by introducing novel attention method [4] and by adopting dilated 1-D convolutions [12] with Gated Recurrent Units (GRU) [13] respectively, they do not consider phase information. In §4.5, we compare the Source-to-Distirtion (SDR) [14] performance of models based on the CaC framework and that of models based on the Magnitude-only framework.

2.2 U-Net Architecture for Spectrogram Estimation

For spectrogram estimation in CaC, we use a U-Net-based architecture. It consists of an encoder and a decoder: the encoder transforms $\mathbf{M}$ into a downsized spectrogram-like representation, and the decoder takes it and returns the estimated target spectrogram $\hat{T}$. Before we describe them in detail, we introduce two types of main components in the architecture as follows.

- An *intermediate block* transforms an input spectrogram-like tensor into an equally-sized tensor (possibly with a different number of channels).

- A *down/up sampling layer* halves/doubles the scale of an input tensor either in the time, frequency, or Time-Frequency domain.

Figure 2. U-Net Architecture for Spectrogram Estimation

As shown in Figure 2, the number of down-sampling layers and the number of up-sampling layers are the same. Also, it uses the same number of intermediate blocks in the encoding and the decoding phase. It has an additional block in between its encoder and decoder. Thus, the total number of blocks should be an odd integer. It has skip connections that concatenate output feature maps of the same scale between the encoder and the decoder.

Besides basic components, our architecture has two additional convolution layers, as illustrated in Figure 2. We use them to increase or restore the number of channels. Before describing them, let us introduce some notations. We denote the input of the l-th intermediate block by $X^{(l-1)}$, and the output by $X^{(l)}$. The size of $X^{(l-1)}$ is denoted by $c_{in}^{(l)} \times T^{(l)} \times F^{(l)}$, where $c_{in}^{(l)}$ represents the number of channels and and $T^{(l)} \times F^{(l)}$ represents the size of the spectrogram-like tensor. Also, we denote the size of $X^{(l)}$ by $c_{out}^{(l)} \times T^{(l)} \times F^{(l)}$, where $c_{out}^{(l)}$ is the number of channels. Using these notations, we denote the input of the first block by $X^{(0)}$, and its size by $c_{in}^{(1)} \times T^{(1)} \times F^{(1)}$. To increase the number of channels, it applies a 1×2 convolution with $c_{in}^{(1)}$ output channels followed by ReLU [15] activation to the given input $\mathbf{M}$. To adjust the number of channels, it also applies a final 1×2 convolution with $(2c)$ output channels to the output of the final block. Note that the last layer is not followed by any activation function since target TF bins can be negative. We empirically set the parameter $c_{in}^{(1)}$ to be 24 in our experiments. Models with smaller $c_{in}^{(1)}$ (e.g., 12) are trained faster, but usually perform inferior than models with larger size of $c_{in}^{(1)}$.

We can implement various SVS models based on this architecture in the CaC framework because multiple options are available for intermediate blocks. In section 3, we present several neural networks which can be used as intermediate blocks in this paper.

3. INTERMEDIATE BLOCKS

We present several types of intermediate blocks based on different design strategies. We first present time-distributed blocks and then present time-frequency blocks.

3.1 Time-Distributed Blocks
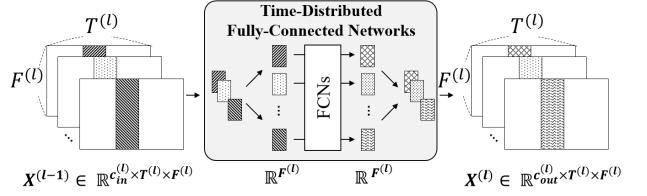
Some existing models use CNNs (e.g., [16]) for intermediate blocks to extract timbre features of the target source. However, the authors of [8] reported that conventional CNN kernels are limited for this task. They found that long-range correlations exist along the frequency axis in the spectrogram of voice signals, which Fully-connected Neural Networks (FCNs) can efficiently capture. They proposed a model named Phasen for speech enhancement, which uses the Frequency Transformation Block (FTB) that has a single-layered FCN without bias. This FCN is applied to each frame of the internal representation in a *time-distributed* manner.

Inspired by TFB, we introduce *time-distributed* blocks, which are applied to a single frame of a spectrogram-like feature map. These blocks try to extract time-independent features that help singing voice separation without using inter-frame operations. We first introduce an FCN-based block and then propose an alternative time-distributed block based on 1-D CNNs.

3.1.1 Time-Distributed Fully-connected networks

We present an FCN-based intermediate block, called Time-Distributed Fully-connected network (TDF). As illustrated

in Figure 3, a TDF block is applied to each channel of each frame separately and identically.
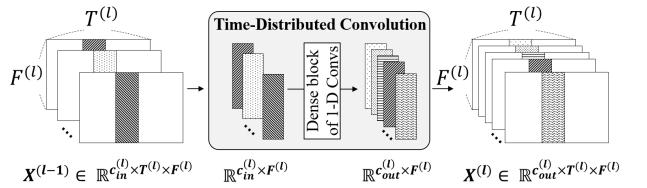
Figure 3. Time-Distributed Fully-connected networks

Suppose that the l-th intermediate block in our U-Net structure takes input $X^{(l-1)}$ into an output $X^{(l)}$. As shown in Figure 3, a fully-connected network is applied separately and identically to each frame (i.e., $X^{(l-1)}[i, j, :]$) in order to transform an input tensor in a time-distributed fashion. While an FTB of Phasen [8] is single-layered, a TDF block can be either single- or multi-layered. Each layer is defined as consecutive operations: a fully-connected layer, Batch Norm (BN) [17], and ReLU [15]. If it is multi-layered, then each internal layer maps an input to the hidden feature space, and its final layer maps the internal vector to $\mathbb{R}^{F^{(l)}}$. The number of hidden units is $\lfloor F^{(l)}/bn \rfloor$, where we denote the bottleneck factor by bf. We can reduce parameters if we use two-layered TDFs of $bf > 2$. We investigate the effect of adding additional layers in §4.2.

3.1.2 Time-Distributed Convolutions

We propose an alternative time-distributed block named Time-Distributed Convolutions (TDC), which is applied separately and identically to each multi-channeled frame. It is a series of 1-D convolution layers. Inspired by [5,6], it takes form of a *dense block* [18] structure. A dense block consists of densely connected composite layers, where each composite layer is defined as three consecutive operations: 1-D convolution, BN, and ReLU. As discussed in [5,6,18] the densely connected structure enables each layer to propagate the gradient directly to all preceding layers, making a deep CNN training more efficient.

Figure 4. Time-Distributed Convolutions
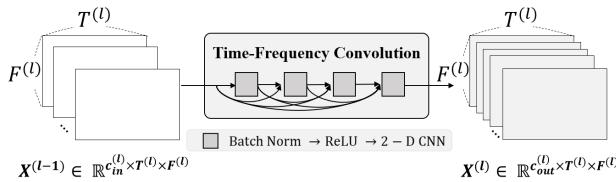
3.2 Time-Frequency Blocks

The performances of U-Nets with time-distributed blocks were above our expectation (see §4.2), but were still inferior considerably to those of current SOTA methods. The reason is that features observed in musical sources include sequential patterns (e.g., vibrato, tremolo, and crescendo) or musical patterns (e.g., rhythm, repetitive structure), which cannot be modeled by time-distributed blocks.

While time-distributed blocks cannot model the temporal context, time-frequency blocks try to extract features considering both the time and the frequency dimensions. We introduce the Time-Frequency Convolutions (TFC) block, which is used in [5]. We also propose two novel blocks that combine two different transformations.

3.2.1 Time-Frequency Convolutions
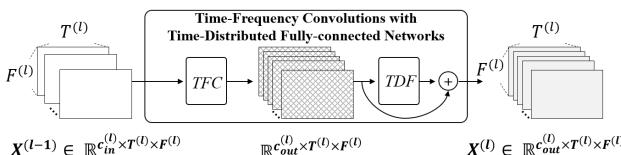
The Time-Frequency Convolutions (TFC) is a dense block of 2-D CNNs, as shown in Figure 5. The dense block consists of densely connected composite layers, where each layer is defined as three consecutive operations: 2-D convolution, BN, and ReLU. It is applied to the spectrogram-like input representation in the time-frequency domain. Every convolution layer in a dense block has kernels of size (k_F, k_T). Its 2-D filters are trained to jointly capture features along both frequency and temporal axes.

Figure 5. Time-Frequency Convolutions

3.2.2 Time-Frequency Convolutions with TDF

We propose the Time-Frequency Convolutions with Time-Distributed Fully-connected networks (TFC-TDF) block. It utilizes two different blocks inside: a TFC block and a TDF block. Figure 6 describes a TFC-TDF block. It first maps the input $X^{(l-1)}$ to a same-sized representation with $c_{out}^{(l)}$ channels by applying the TFC block. Then the TDF block is applied to the dense block output. A residual connection is also added for efficient gradient flow.

Figure 6. Time-Frequency Convolutions with TDF

Phasen [8] has shown that inserting time-distributed operations into intermediate blocks can improve speech enhancement performance. We validate whether it also works for SVS or not in §4.3.

3.2.3 Time-Distributed Convolutions with RNNs

We propose an alternative way to consider both the time and frequency dimensions. A Time-Distributed Convolutions with Recurrent Neural Networks (TDC-RNN) block uses two different blocks: a TDC block for extracting timbre features and RNNs for capturing temporal patterns. It extracts timbre features and temporal features *separately*, unlike a TFC block. We validate whether this approach can

outperform the 2-D CNN approach by comparing TDC-RNNs with TFCs in §4.3.

The structure of a TDC-RNN block is similar to that of a TFC-TDF block. It applies the TDC block to an input $X^{(l-1)}$, and obtains a same sized hidden representation with $c_{out}^{(l)}$ channels. The RNNs compute the hidden representation and output an equally sized tensor. A residual connection is added, as is a TFC-TDF block.

4. EXPERIMENT

We evaluate U-Nets with different types of blocks introduced in §3. We compare the performance of models in §4.2 and §4.3. Also, we compare our models with SOTA models in §4.4. We compare the spectrogram estimation framework in §4.5. We discuss reusable insights in §4.6.

4.1 Setup

4.1.1 Dataset

Train and test data were obtained from the MUSDB dataset [7]. The train and test sets of MUSDB have 100 and 50 musical tracks each, all stereo and sampled at 44100 Hz. Each track file consists of the mixture and its four source audios: 'vocals,' 'drums,' 'bass' and 'other.' Since we are evaluating on singing voice separation, we only use the 'vocals' source audio as the separation target for each mixture track.

4.1.2 Model Configurations

We implemented U-Nets with different blocks (§3). Each model is based on the U-Net architecture (§2.2) on the CaC framework (§2.1). We set $c_{in}^{(1)}$, the number of internal channels to be 24, as mentioned in §2.2. Each model uses a single type of block for its intermediate blocks. We usually used an FFT window size of 2048 and a hop size of 1024 for STFT. However, we used a larger window size in some models for a fair comparison with SOTA methods.

4.1.3 Training and Evaluation

Weights of each model were optimized with RMSprop [19] with learning rate $lr \in [0.0005, 0.001]$ depending on model depth. Each model is trained to minimize the mean square error between $\hat{T}$ and $\mathbf{T}$ as mentioned in §2.1. We use the default validation set (14 tracks) as defined in the *MUSDB* package, and use the Mean Squared Error (MSE) between target and estimated signal (waveform) as the validation metric for validation. Data augmentation [20] was done on the fly to obtain fixed-length mixture audio clips comprised of the source audio clips from different tracks.

We use the official evaluation tool[2] provided by the organizers of the SiSEC2018 [21] to measure Source-to-Distortion Ratio (SDR) [14]. We use the median SDR value over all the test set tracks to obtain the overall SDR performance for each run, as done in the SiSEC2018. We report the average of 'median SDR values' over three runs for each model.

---

[2] https://github.com/sigsep/sigsep-mus-eval

block type	# blocks	# params	SDR
TDC (w/ sampling)	17	0.54M	**4.86**
TDC (w/o sampling)	17	0.52M	3.78
TDC (w/o sampling)	3	0.09M	3.56
TDF (w/o hidden layer)	17	2.83M	4.75
TDF (w/ hidden layer)	17	1.44M	4.05
TDF (w/ hidden layer)	3	1.19M	4.01

Table 1. Evaluation results of Time-Distributed Blocks.

model	sampling	# blocks	# params	SDR
TFC	O	17	1.56M	6.89
TFC	X	17	1.56M	6.75
TDC-RNN	O	17	2.08M	6.69
TFC-TDF	O	7	0.99M	7.07
TFC-TDF	O	17	1.93M	**7.12**

Table 2. Evaluation results of Time-Frequency Blocks.

4.2 U-Nets with Time-Distributed Blocks

We implemented and trained U-nets with TDC and TDF blocks. We also implemented models with TDC blocks that do not use down/up-sampling to investigate the effect of down/up-sampling in the frequency axis. The other models use 1-D convolution/transposed-convolution layers with stride 2 for down/up-sampling. Every TDC block is a dense block with 5 composite layers with the growth rate 24 (used in dense blocks [18]). The kernel size of each convolution layer in a dense block is 3. Each TDF block is either single-layered or two-layered. The bottleneck factor bf of each TDF block is set to be 4. All models have 17 intermediate blocks except for two shallow models.

We summarize evaluation results in Table 1. The TDC block-based U-Net with sampling achieves an SDR of 4.86, the highest among the three models. Results show that the use of down/up-sampling in TDC-based U-Nets was significant, although the model without sampling can exploit higher resolution of internal representations. It may indicate that enlarging receptive fields via sampling may help the model to capture long-term dependencies better, and long-term dependencies are preferred over local features when distinguishing unique time-independent frequency patterns. (at least for these configurations).

Although FCNs can capture long-ranged patterns along the frequency domain, as mentioned in [8], TDF-based U-Nets did not perform well enough compared to the TDC-based models in a deep architecture. Among TDF-based models, the U-Net equipping single-layered TDFs (the fourth row of Table 1) outperforms the other models. However, it is notable that we can reduce parameters when we use two-layered TDFs. Also, we found that the TDF blocks can outperform TDC blocks in a shallow architecture (the third and sixth row of Table 1). The reason is that the U-Nets with few TDC blocks has a small receptive field, while a single TDF block has a full receptive field in the frequency dimension, which has led us to inject it in a time-frequency block instead of TDC (see §3.2.2).

4.3 U-Nets with Time-Frequency Blocks

We implemented U-Nets with time-frequency blocks. All models are trained on 3 seconds (128 STFT frames) of music. Since the number of frequency bins is much larger than the number of frames, models with more than 7 neural transforms use both 2×2 or 2×1 sized down/up-sampling layers to scale the frequency axis more than 3

times while maintaining the number of scales in the temporal axis to 3. Exceptionally, we use different down/up-sampling layers for one model to investigate the effect of down/up-sampling in the temporal axis.

We set every TFC block to have 5 convolution layers with kernel size 3×3. We set the growth rate to be 24, the same growth rate of §4.2. By using this TFC block configuration, we implemented a TFC-based U-Net (the first row of Table 2). We set the model in the second row to use different down/up-sampling layers to investigate the effect of down/up-sampling in the temporal axis. Every kernel size used in each down/up-sampling layer of this model is 2×1 to preserve the temporal resolution while scaling frequency resolution. The first two rows of Table 2 summarize the experiment results of two TFC-based models. The model that preserves the temporal resolution was slightly inferior to the other model. It is also notable that our U-Nets with TFC blocks achieve comparable results with state-of-the-art methods 4.4, even using lower frequency resolution. Compared to the frequency axis where the TDC-based U-Net with down/up-sampling outperforms the counterpart model, no significant SDR was gained by enlarging the receptive field by down/up-sampling.

We reused the configuration of TDC in of §4.2, for TDCs in TDC-RNN blocks. The RNN layers were implemented with bidirectional GRUs with a single hidden layer, which has $f/16$ hidden units, where f is the number of input frequency bins. Although having more parameters and a better potential for capturing long temporal dependencies than the two fully convolutional models, TDC-RNN performs lower than them. Increasing the number of hidden units or hidden layers could have increased SDR since many other state-of-the-art recurrent models use a hidden size that is at least 512. Increasing the number of STFT frames, thus training on longer clips of music, might have also worked. Although it performs the worst among the time-frequency blocks, it is superior to all the time-distributed blocks. It indicates that inter-frame operations are necessary for higher quality separation.

The fourth and fifth rows of the Table 2 shows promising results regarding the U-Nets with TFC-TDF blocks. We reused the same TFC setting above, and we set bf to be 16 for each TDF. The 7-blocked U-Net with TFC-TDFs outperforms the other 17-blocked models. These results show that inserting FCNs into intermediate blocks can be useful for MSS as well as for Speech Enhancement [8]. Also, results show that it is also achievable with fewer parameters by using FCNs with a bottleneck layer.

model	# parameters	SDR (vocals)
DGRU-DGConv	more than 1.9M	6.99
TAK1	1.22M	6.60
UMX	8.89M	6.32
TFC-TDF (small)	0.99M	**7.07** $\pm.08$
TFC-TDF (large)	2.24M	**7.98** $\pm.07$

Table 3. Comparison: SDR median value on test set.

esimation	n_fft	# blocks	# params	SDR
CaC	2048	7	0.99M	**7.07**
Mag	2048	7	0.99M	6.43
CaC	4096	9	2.24M	**7.98**
Mag	4096	9	2.24M	7.24

Table 4. Comparison of TFC-TDFs: CaC vs Mag

4.4 Comparison with SOTA models

We compare our models with other spectrogram-based models on the MUSDB benchmark. The first three rows of Table 3 shows the SDR performance of SOTA models, namely DGRU-DGConv [12], TAK1 [6], and UMX [22]. Their SDRs can be found in [12], SiSEC2018 repository [3], and UMX repository [4]. We estimated the lower bound of the number of parameters of DGRU-DGConv with 1-D CNN parameters without considering its GRUs.

Comparing with Table 2, we can see that our models perform comparably to or even outperform existing models even with less frequency resolution and fewer parameters. On top of that, our TFC extensions do not use recurrent layers, which is a key factor in the other previous models. It may lead to shorter forward/backward propagation time. Also, it is worth noting that previous models adopt Multi-channel Wiener Filtering as a post-processing method to further enhance SDR. Ours directly use the signal reconstruction output without such post-processing.

For a fair comparison with SOTA models, we trained an additional U-Net with 9 TFC-TDF blocks (notated as 'large' in Table 3) with the same frequency resolution as the other SOTA models (FFT window size = 4096) and achieved outstanding results with a 0.9 dB gain over DGRU-DGConv.

4.5 Spectrogram Estimation: Complex vs Magnitude

For our final experiment, we see how much SDR was gained by extending a magnitude-only model into a CaC model. Our TFC-TDF-based U-Nets in Table 4 are compared to their magnitude-only form (referred to as 'Mag'). They use the same hyperparameter set except for $c_{in}^{(0)}$, the input/output number of channels. Mag also has an additional ReLU after the final 1×1 convolution to obtain non-negative-valued output spectrograms. Results show that training with raw STFT outputs instead of magnitudes

---

[3] https://github.com/sigsep/sigsep-mus-2018
[4] https://github.com/sigsep/open-unmix-pytorch

significantly boosts SDR performance. It is also notable that the Mag model with n_fft of 4096 still outperforms all previous state-of-the-art models in Table 3.

4.6 Discussion: Developing Reusable Insights

Our work provides a practical guideline for choosing fundamental building blocks to develop an SVS or MSS model based on the U-Net architecture as follows.

- TDC-based models are sensitive to the number of blocks, compared to TDF-based models.

- Using down/up-sampling is important for CNN-based blocks, especially in the frequency dimension.

- Stacking 2-D CNNs is a simple but effective way to capture T and F features, compared to TDC-RNNs.

- Injecting a time-distributed block to a time-frequency block can improve SDR.

- A simple extension from a magnitude-only U-Net to a CaC U-Net can improve SDR.

Our work is not limited to the U-Net-architecture nor MSS. Blocks can be used as core components in more complex architectures as well. We can use different types of blocks for a single model, meaning that a lot of space remains for improvement. Also, our observations can be exploited in other MIR tasks such as Automatic Music Transcription (AMT) or Music Generation: for example, we expect that injecting TDFs to intermediate blocks for f_0 estimation model can improve performance since fully-connected layer can efficiently model long-range correlations such as harmonics.

5. CONCLUSION AND FUTURE WORKS

In this paper, we designed several types of blocks based on different design strategies. We implemented U-Net models with these blocks for SVS and evaluated their performance. Our experiments provide abundant material for future works by comparing several U-Nets with different types of blocks. Also, one of our models outperforms SOTA methods. For future work, we would like to extend this model to utilize attention networks for modeling long-term dependencies observed in both the frequency and the temporal axis.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] J. Andreas, H. Eric, M. Nicola, B. Rachel, K. Aparna, and W. Tillman, "Singing voice separation with deep u-net convolutional networks," in *18th International*

Society for Music Information Retrieval Conference, 2017, pp. 23–27.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention.* Springer, 2015, pp. 234–241.

[3] S. Park, T. Kim, K. Lee, and N. Kwak, "Music source separation using stacked hourglass networks," *arXiv preprint arXiv:1805.08559*, 2018.

[4] W. Yuan, S. Wang, X. Li, M. Unoki, and W. Wang, "A skip attention mechanism for monaural singing voice separation," *IEEE Signal Processing Letters*, vol. 26, no. 10, pp. 1481–1485, Oct 2019.

[5] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct 2017, pp. 21–25.

[6] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 106–110.

[7] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "MUSDB18 - a corpus for music separation," Dec. 2017, mUSDB18: a corpus for music source separation. [Online]. Available: https://hal.inria.fr/hal-02190845

[8] D. Yin, C. Luo, Z. Xiong, and W. Zeng, "Phasen: A phase-and-harmonics-aware speech enhancement network," *arXiv preprint arXiv:1911.04697*, 2019.

[9] N. Takahashi, P. Agrawal, N. Goswami, and Y. Mitsufuji, "Phasenet: Discretized phase modeling with deep neural networks for audio source separation." in *Interspeech*, 2018, pp. 2713–2717.

[10] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," in *International Conference on Learning Representations*, 2018.

[11] S.-W. Fu, T.-y. Hu, Y. Tsao, and X. Lu, "Complex spectrogram enhancement by convolutional neural network with multi-metrics learning," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2017, pp. 1–6.

[12] J.-Y. Liu and Y.-H. Yang, "Dilated convolution with dilated gru for music source separation," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4718–4724.

[Online]. Available: https://doi.org/10.24963/ijcai.2019/655

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[14] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[15] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[16] P. Chandna, M. Miron, J. Janer, and E. Gómez, "Monoaural audio source separation using deep convolutional neural networks," in *International conference on latent variable analysis and signal separation*. Springer, 2017, pp. 258–266.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[19] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, p. 8, 2012.

[20] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 261–265.

[21] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2018, pp. 293–305.

[22] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - A Reference Implementation for Music Source Separation," *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, Sep. 2019. [Online]. Available: https://hal.inria.fr/hal-02293689

DISCOURSE NOT DUALISM: AN INTERDISCIPLINARY DIALOGUE ON SONATA FORM IN BEETHOVEN'S EARLY PIANO SONATAS

Christof Weiß[1], Stephanie Klauk[2], Mark Gotham[2], Meinard Müller[1], Rainer Kleinertz[2]

[1] International Audio Laboratories Erlangen, Germany
[2] Institut für Musikwissenschaft, Saarland University, Germany
christof.weiss@audiolabs-erlangen.de

ABSTRACT

The computational analysis of music has traditionally seen a sharp divide between the "audio approach" relying on signal processing and the "symbolic approach" based on scores. Likewise, there has also been an unfortunate gap between any such computational endeavour and more traditional approaches as used in historical musicology. In this paper, we take a step towards ameliorating this situation through the application of a computational method for visualizing local key characteristics in audio recordings. We exploit these visualizations of diatonic scale content by discussing their musicological implications, being aware of methodological limitations as for the case of minor keys. As a proof of concept, we use this method for investigating differences between the traditional sonata-form model and selected Beethoven piano sonatas in the context of sonata theory from the end of the 18$^{\text{th}}$ century. We consider this scenario as an example for a rewarding dialogue between computer science and historical musicology.

1. INTRODUCTION

The analysis of musical works in terms of their compositional style and context is at the core of historical musicology. Scholars engage with a reasonably large body of works over the course of their career and make observations about these works through manual analysis. As valuable as this is, it is a time-consuming and individuated approach that poses difficulties for making meaningful observations at scale. In this paper, we seek to demonstrate how a question of musicological relevance could be complemented by using computational analysis methods. While such methods can never achieve the flexible and interconnected consideration of the human mind nor capture the compositional intricacies of a specific work, we aim to show that suitable visualizations can assist in the process of expert interpretation in a rewarding way.

Numerous computational methods for harmony analysis have been developed over the past decades, centered on global key detection [1–4], local key estimation [5–7], chordal analysis [8–10], and their combination into functional harmony analysis systems [11–13]. Many of these are based on *symbolic* encodings of music notation such as MIDI or MusicXML. However, symbolic music datasets are rarely available, in particular when requiring symbolic encodings of high quality covering an entire corpus of music (not just individual pieces). Optical Music Recognition (OMR) software for automatically converting graphical formats into symbolic data does not yet offer reliable results, meaning that time-consuming manual postprocessing is often required [14, 15].

Beyond such practical problems, we should remember that Western music notation is essentially "prescriptive": a set of instructions for performance that requires reading and interpretation. As an alternative to the processing of sheet music, analyses can be carried out on the basis of audio recordings [4, 7, 10]. In this paper, we make use of an existing audio-based method [16] for visualizing the diatonic scale content of a music recording over time in order to complement and facilitate the close analytical reading by human experts. We address this paper to both musicologists and computer scientists alike and thus explain the relevant background from both domains.

We first set out the musicological context (Section 2), describe the computational method (Section 3), and explain its musical implications through the example of three piano sonatas by L. v. Beethoven (Section 4), which have been subject to automatic analysis [9, 12, 17]. Using our visualizations, we discuss the implications for sonata form theories (Section 5). Finally, we summarize our findings and the rewards of an interdisciplinary dialogue between computer science and musicology (Section 6).

2. MUSICOLOGICAL BACKGROUND

Our focus in this paper is on large-scale tonal structures that are attested to play a crucial role in sonata form. This section provides a short, simplified introduction to sonata form for readers unfamiliar with it and serves to introduce the main musicological question addressed in this paper.

Sonata form is a central model for describing the first movement of most multi-movement works in classical music from about 1770 far into the 19th century. The term is applicable not only to *sonatas* but also to symphonies, string quartets, and other genres, and thus has a wide explanatory potential for the music of the time, with a focus on these works' first movements.

The definition of sonata form widely adopted by musicians and musicologists seems to stem primarily from the writings of A. B. Marx [18], which were driven mainly by an effort to understand the music of then recently deceased Ludwig van Beethoven (1770–1827). According to this model, the sonata is divided into three main sections: a first section, later denoted as *exposition* (in two or more keys, often repeated), a central *development*, and a *recapitulation* of the exposition (set mainly in one key). This is the core structure which may be framed by an introduction at the start, and/or a coda at the end [19]. Automatically detecting these large-scale segments and their tonal relations has been approached both for Mozart's string quartets [20, 21] and Beethoven's piano sonatas [17].

The exposition is tasked with setting out the melodic material (e. g., themes or motives) and the main key relationships. It is usually divided into a first subject area, followed by a transition into a second subject area, and a short cadential passage (*Schlussgruppe*). Crucially, the two focal areas are defined by contrasts both in theme (melody) and key (tonality). The typical tonal pairing is of a major key and its dominant key (the major key one fifth higher, e. g., C major and G major), or of a minor key with its relative major key (e. g., A minor and C major).

Our primary focus will be on key, which we approximate in the broader sense with our visualizations of diatonic pitch class content. With this computational method, we elucidate the tonal relations in specific early Beethoven piano sonatas. In this paper, we focus on works in major keys overall and on the exposition sections of those works.

3. COMPUTATIONAL APPROACH

In this work, we employ a computational method for visualizing local keys or, more precisely, the *diatonic pitch class content* over the course of a piece, closely following the approach proposed in [16]. The method operates on audio recordings, i. e., *performances* of the pieces, thus allowing for scalability to a wide repertoire (see Section 1).

3.1 Chroma-based Scale Estimation

Our method is based on the measurement of spectral energies over time. These energies are summarized into twelve *chroma bands* irrespective of their octave, according to the pitch classes of the twelve-tone equal-tempered scale. The resulting *chroma features* can be represented as twelve-dimensional vectors whose entries refer to the pitch classes C, C♯, ..., B in chromatic order. For chroma extraction, we use the filter-bank approach provided by the chroma toolbox [22] with a feature rate of 10 Hz (i. e., ten chroma vectors per second). For each frame, we match the chroma

vector with binary diatonic scale templates using the inner product (cosine similarity). For instance, the template for the "0 diatonic scale" (corresponding to the pitch classes of the C major and A natural minor scales) is given by

$$\mathbf{t}_0 = (1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1)^\top . \quad (1)$$

Assuming enharmonic equivalence (C♯=D♭), there are a total of 12 diatonic scales, whose templates are obtained by circularly shifting the template $\mathbf{t}_0$ shown above. Thus, we obtain for each frame an analysis given by a twelve-dimensional vector. We will discuss our choice of using binary diatonic templates in Section 3.4.

3.2 Pre-processing

Before the template matching step described above, we apply several pre-processing steps. Since local keys or scales refer to the pitch class content of larger sections of music, we smooth the chromagrams (with an initial feature rate of 10 Hz) using a window of size $w \in \mathbb{N}$ in frames and a hopsize of 10 frames (one second). The musical implications of the window size parameter will be discussed in Section 4.3. Additionally, we normalize the smoothed chroma features according to the ℓ_2-norm.
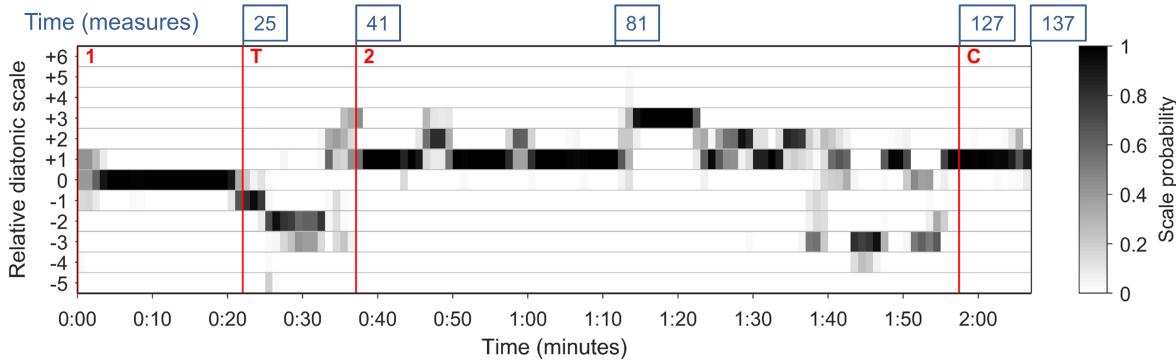
3.3 Post-processing and Visualisation

In this interdisciplinary work, we do not aim for a fully automatic "key detection," which locally decides on the most likely key or scale. Instead, following [16], we propose the use of suitable visualization techniques allowing for a direct interpretation of the continuous-valued diatonic scale probabilities in the musicological discussion. For generating this visualization, we obtain re-scaled local analyses by using the *softmax* function, thus suppressing weak components and enhancing large ones. Using a normalization with respect to the ℓ_1-norm, we can interpret the analysis as pseudo-probabilities of diatonic scales, which we then visualize in grayscale, where darker gray corresponds to higher probabilities (see Figure 1 for an example).

We adopt a musical criterion for arranging the order of scales in this visualisation. There are two main options of ordering the scales, either chromatically (C, C♯, ...) or according to the circle of fifths (C, G, ...). Motivated by [23, 24], we prefer the latter arrangement accounting for the similarity of fifth-related scales, which have six out of seven pitch classes in common. We set the diatonic scale corresponding to the piece's global key in the center of the visualization, with upper-fifth-related scales (more sharps) above and lower-fifth-related scales (more flats) below that center scale. This "global key normalization" facilitates the comparison between movements in different keys.

3.4 Scale versus Local Key

Our analytical approach directly maps the locally predominant pitch class content (as measured from the audio recordings) to probabilities for diatonic scales. While there have been many such template-based approaches based on psychological and empirical studies [25, 26], our choice

Figure 1. L. v. Beethoven, piano sonata Op. 7 in E♭ major, 1st mvmt. *Allegro molto e con brio*, exposition. Computational tonal analysis with a window size of $w = 4$ seconds.

of straightforward diatonic templates allows for an interpretable and objective investigation of the tonal content.

There are certainly limitations to this approach. Most importantly, it relates to the notion of local key only in a loose way since there is a methodological gap between "scale" and "key." In particular, pitch class content is not the only determinant of musical key. Furthermore, the pitch class content is rarely exclusively diatonic, which is especially the case for pieces in minor, where scale degrees outside the natural minor scale ($\sharp\hat{6}$ and $\sharp\hat{7}$) play a crucial role. Moreover, our method cannot resolve relative key differences such as C major – A minor, which would be relevant particularly for minor key movements.[1] On the other hand, it provides an easier overview (only 12 scales) and is not susceptible to relative key confusions.

Leaving aside these methodological gaps, even identifying a scale can be problematic. For example, certain chords such as the (relatively rare) augmented triad and the (not at all rare) diminished seventh chord pose challenges because they cannot be unambiguously assigned to a diatonic scale. Moreover, frequent modulations constitute a problem for assigning a single scale to a specified time window. That being said, this computational issue reflects a genuine musical problem: hearing a tonal moment in isolation would leave the allocation of a key highly ambiguous—and composers (Beethoven very much included) exploit this potential for ambiguity. It is only through context that we are able to make clear assertions. We remain mindful of these limitations as we proceed to consider how these visualizations may yet be useful in support of a better understanding of sonata form in general and, more precisely, in Beethoven's early piano sonatas.

4. APPLICATION TO SONATA EXPOSITIONS

Consequently, we apply the method described above to several exposition sections of Beethoven's early sonatas (first movement, respectively). The time stamps are given in MM:SS and refer to Daniel Barenboim's 1984 set of recordings for Deutsche Grammophon.

4.1 Sonata Op. 7 in E♭ major

Figure 1 provides an example of our visualization method for the exposition from Beethoven's piano sonata Op. 7 in E♭ major, first movement. As E♭ major is the global key of this movement, the relative diatonic level 0 on the y-axis of this figure refers to an absolute scale of –3, corresponding to the key signature of E♭ major and its relative minor (C minor, both with a key signature of three ♭s). Likewise, +1 refers to the key signature of B♭ major (and G minor), –1 stands for A♭ major (and F minor), and so on.

In addition to the time stamps, the figure (and all subsequent figures) also provide vertical lines (in red) to divide the sections on the basis of Donald F. Tovey's iconic guide to Beethoven's piano sonatas [27]. These lines add additional score-related timestamps given in measure numbers (in blue) and are paired with the formal labels that Tovey used, following the standard terms for the sonata form's main parts discussed above. The vertical lines on Figure 1 use the following abbreviations: "1" stands for the *first group*, "T" for the *transition*, "2" for the *second group*, and "C" for the *cadence group*. The following section takes a closer look at the example with a view to both the visualization method and the implications for understanding sonata form in this specific repertoire.

4.2 A Closer Look

The visualization in Figure 1 provides a straightforward, easily readable overview of the diatonic scale content in the exposition of the first movement. The movement begins with a moment of ambiguity, centered on the central (0) scale (corresponding to E♭ major), and quickly settles much more emphatically into that tonal region for the first 22 seconds (until m. 24). A phase of tonal instability follows, befitting the transition phase of a sonata form exposition, which traverses at least the scales –1 (4♭ or A♭ major / F minor) to –2 (5♭), and then +1 (2♭) to +3 (no accidentals).

At 0:37 (m. 41), a longer section starts in the +1 area, neatly corresponding to what is traditionally called the "Second Theme" or "Second Group." At 1:11 (m. 79), the proportions (with a second subject equal in length to the first) might lead us to expect the exposition to close. Instead, about a minute more music follows, with only the last approx. 10 seconds being displayed in the anticipated

+1 area (mm. 127–136, largely corresponding to the final group). The course of the movement from 1:11 to 1:57 (mm. 79–127), by contrast, seems to be characterized by greater harmonic mobility, centered on the +1 axis, but not restricted to it. Regardless of any designation of the formal parts, the graphic shows a clear distinction in the course of the exposition from 0:38 (m. 41) into a tonally stable part until 1:11 (mm. 78–79), a more flexible part until 1:57 (m. 127), and again a stable one until the end.
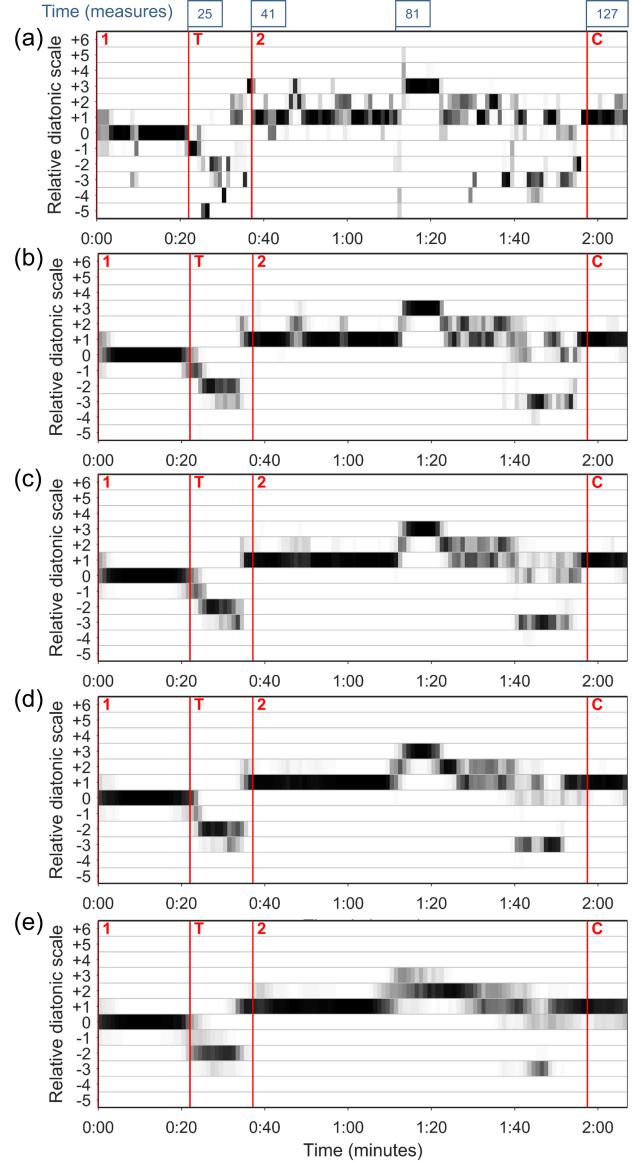
4.3 Effects of the Window Size

As one of the essential properties of our methodology, we have to specify the window size parameter w, which defines the temporal context of the local scale analysis. The use of a window size of four seconds in Figure 1 affects the result we see in that visualization. In order to demonstrate the effect of this parameter, Figure 2a–e shows five different visualizations of the same example with window sizes varying from 2–20 seconds.

Naturally, shorter time windows portray a more "atomized" harmonic course, emphasizing the moment-to-moment details, while longer windows diminish the individual moments in favor of the "bigger picture." Any gain in uniformity and clarity comes at the cost of a corresponding loss in detail. In this respect, a computational analysis does not differ from a manual one, in which the analyst also has to decide whether to focus on fine-grained details or to favour better readability and clarity. This speaks to music theory's attention to the "level" of reductive analysis [28]—an approach to which the variable window size on offer here may be highly suitable. Multi-scale approaches for simultaneously using several window sizes suggest an alternative [29]. However, these visualizations require a third dimension (usually color-coded), which complicates readability in our application scenario. Moreover, interactive visualizations with a flexible adjustment of the window size can be realized with user interfaces or websites.

To better illustrate the behaviour of our method, we now proceed to discuss the effects of the window size at the example of several specific passages. With a window size of four seconds (Figure 1), the dominant seventh chord G-Bb-Db-Eb of Ab major in m. 10 (at 0:08) does not lead to any deviation from the level 0. With a resolution of two seconds (Figure 2a), it causes a much stronger gray coloration, which reaches down to the –3 level (corresponding to Gb major / Eb minor). Similarly, the diminished seventh chord F♯-A-C-Eb in mm. 79–89 (1:11–1:22) triggers a whole range of possible interpretations. First, the chord eludes the assignment to a single diatonic scale, and second, through the pitch class F♯, it does not fit into the previous Bb major context (–2), nor through F♯ and Eb into the subsequent C major context (0). In the case of longer windows, these uncertainties are smoothed out and therefore no longer catch the eye (see Figure 2d–e).
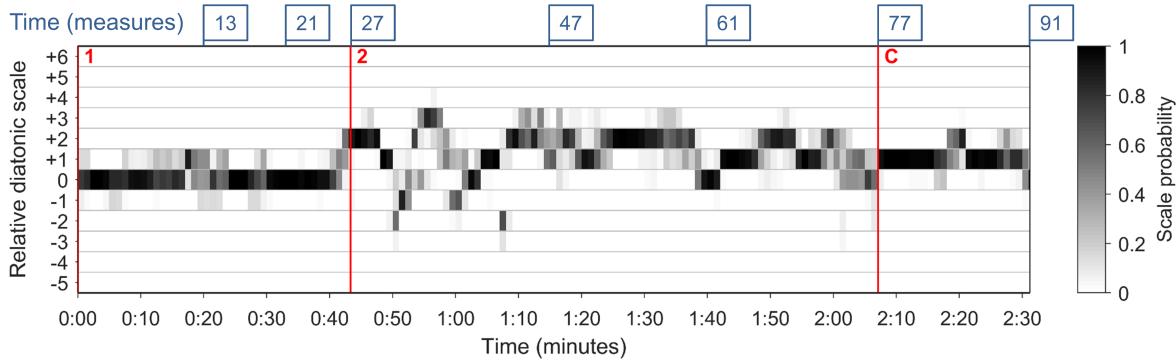
An even larger window size, e. g. of 20 seconds (Figure 2e), shows that only certain resolutions make sense for a specific work. Such large windows neither increase clarity nor do all the gray shadings of ambiguity disappear. In

Figure 2. L. v. Beethoven, piano sonata Op. 7 in Eb major, 1st mvmt. *Allegro molto e con brio*, exposition. Computational tonal analysis with a window size w of (a) 2 sec., (b) 6 sec., (c) 8 sec., (d) 12 sec., (e) 20 sec.

Figure 2e, the C major section in mm. 81ff. (1:12–1:22) is no longer visible as +3 (C major / A minor); instead, it blurs with the +2 level. At the same time, the tonally stable final group (+1), which only lasts about 10 seconds (mm. 127–136), shows considerably more shades of gray on the 0 level than visualizations with shorter windows, which is caused by the previous measures and the subsequent repetition of the exposition. The choice of very large windows not only leads to an increased smoothing of the visualization but also reaches limits beyond which the results are no longer meaningful.

In the present sonata, window sizes of 4–12 seconds have proven to be useful, illuminating most of the relevant phenomena. In particular, these visualizations clearly highlight phases of tonal stability and instability as well as diatonic regions, which are of high importance for the for-

Figure 3. L. v. Beethoven, piano sonata Op. 2 No. 3 in C major, 1st mvmt. *Allegro con brio*, exposition. Computational tonal analysis with a window size of $w = 4$ seconds.

Figure 4. L. v. Beethoven, piano sonata Op. 2 No. 3 in C major, 1st mvmt. *Allegro con brio*, mm. 25–28.

mal organization of expositions. Based on these observations, we now examine two further sonata expositions that have received frequent attention in the literature: sonatas Op. 2 No. 3 in C major and Op. 10 No. 3 in D major.

4.4 Sonata Op. 2 No. 3 in C major

Spanning 90 measures, the exposition of the sonata Op. 2 No. 3 in C major is even more extensive than that of Op. 7. The computational analysis (Figure 3) shows that the piece remains in a C major context for an unusually long time. These first 26 measures include the main subject (mm. 1–13) followed by playful figurations, which surprisingly do not modulate (mm. 13–21), and a cadence passage (mm. 21–26), which ends after a G major scale on the single tone G without having modulated (see Figure 4). [2] From m. 27 (0:43) on, this is followed by the melodic motif in G minor mentioned above, which is repeated in D minor (m. 33) and continued towards A minor (m. 39).

The visualization makes it clear that this G minor passage is by no means the beginning of the "Second Group (or Transition and Second Group) in Dominant"—as marked in Figure 3 after Tovey—but the beginning of the modulation to the upper-fifth key, i. e., the "transition." First of all, the abrupt tonal change in m. 27 (0:43) is visible. [3] However, the visualization does not show G minor (–2) but D major (+2), which may be caused by the frequent occurrence of the leading notes F♯ and C♯. In the following modulation, we observe level 0 at 1:03 (mm. 39ff., pointing to in A minor), then again level +2 at 1:09-–1:14 (mm. 43–45, pointing to D major), thereby terminating the transition to the second group (mm. 47–61; 1:15–1:40). The second group starts at level +1, then from m. 53 (1:26) on mainly represented as +2 due to several neighbor notes

C♯ and the secondary dominant chord A major sounding for a whole measure. This is followed by a longer cadence section, initially at the 0 level (C major) from 1:40 (mm. 61ff.) and then confirms level +1 with smaller swings to the levels +2 and 0. The largely stable level +1 in the final group is then clearly visible (mm. 77–90).

4.5 Sonata Op. 10 No. 3 in D major

The visualization of the sonata Op. 10 No. 3 gives a completely different picture (Figure 5). After about 20 seconds (m. 22), the initial tonality (D major, level 0) is left, followed by a longer section on the level +1 (mm. 23–45). In fact, we find here a lyrical motif in B minor, which actually corresponds to level 0. Due to the frequent occurrence of the leading note A♯, however, there seems to be a kind of "statistical averaging" between B minor (0) and B major (+3). The occurrence of C♯ major and F♯ minor in mm. 31–34 leads to a small shade of gray in the +4 level at 0:31. The second group in A major is then reached at 0:48 (m. 54). The repetition of the second subject in –2 (pointing to A minor) is briefly visible at 0:54 (mm. 60–63). It is then striking that the second group again ends more or less in the middle of the movement and is followed by a longer developmental passage at 1:00–1:18 (mm. 67–87).
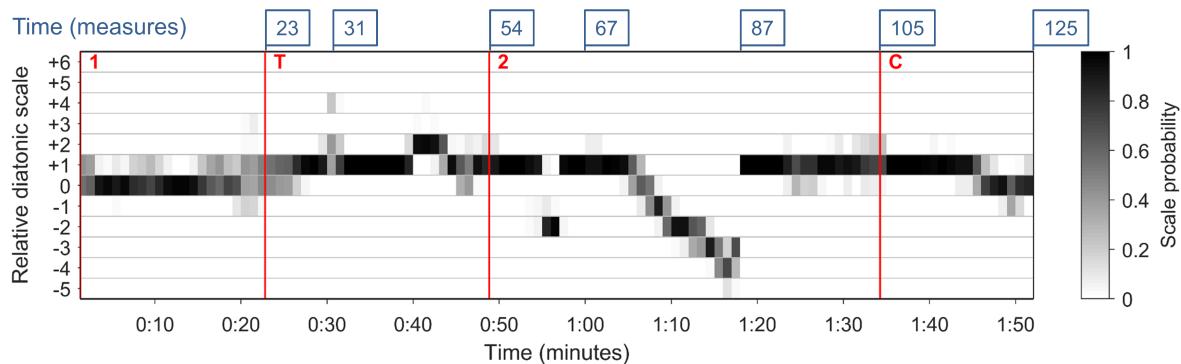
5. MUSICOLOGICAL DISCUSSION

Section 4 examined the use of this visualization method for illuminating the details of three exemplary cases. We now broaden the scope to consider its potential contribution to the wider question of sonata form itself.

The visualizations discussed above have shown a multifaceted structure within the tonal course of the exposition. The plots point out that the musical course cannot be unproblematically reconciled with the traditional sonata form schema of *first group – transition – second group –*

---

[2] This ending constitutes a clear example of a so-called "bifocal close" [30].

[3] See also [17] for a method to highlight such changes.

Figure 5. L. v. Beethoven, piano sonata Op. 10 No. 3 in D major, 1st mvmt. *Presto*, exposition. Computational tonal analysis with a window size of $w = 4$ seconds.

cadence group [19], a problem raised by Carl Dahlhaus decades ago [31, p. 101–103]. This leads us to reconsider other, earlier theories of sonata form from the late 18th century which emphasize the idea of a musical *discourse* as opposed to a thematic *dualism*.

A detailed description of such an 18th-century form model is given by Francesco Galeazzi in the second volume of his *Elementi teorico-pratici di musica* (Rome 1796) [32]. Galeazzi's model differs from the theory more familiar today in two principal respects: the presence of a contrasting *second motif* before or at the beginning of the transition and a *cadential period* ahead of what he calls the *coda* (i. e., a codetta or final group).

For the section we now denote as *exposition* (which Galeazzi simply calls *prima parte*), Galeazzi sets out an alternative schema of seven elements [32, p. 324]:

1. Prelude (*preludio*)
2. Principal motive (*motivo*)
3. Second motive (*secondo motivo*)
4. Departure to [...] related keys (*uscita di tono*)
5. Characteristic passage / middle passage (*passo caratteristico / passo di mezzo*)
6. Cadential period (*periodo di cadenza*)
7. Codetta (*coda*)

Of these seven parts, according to Galeazzi [32], parts 2, 4, and 6 (in modern terms: *first key subject*, *transition*, and *cadential confirmation of the second key*) are compulsory, the remainder are optional. Accordingly, Galeazzi bases his model on a main motif that dominates the musical discourse like the topic of a speech and that can be followed by a whole series of new, partly related thoughts:

> "The motive [...] must be very conspicuous and perceptible because inasmuch as it is the theme of the discourse, if it is not well understood, neither will the consecutive discourse be understood." [32, pp. 326–327] [4]

If one starts from such a discursive form idea, the occurrence of a new thought before or in the transition (e. g.

Op. 2 No. 3 mm. 27–39, Op. 10 No. 3 mm. 23–30) and of a longer form part after the "middle sentence" (Op. 2 No. 3 mm. 61–77, Op. 7 mm. 81–127, Op. 10 No. 3 mm. 67–105) in no way leads into "insoluble theoretical difficulties" [31]. Even though some detailed structures might differ from a human analysis, the computational visualizations show something that was taken for granted for composers and audiences of the late 18th century: the individual formal parts are not opposed in a *dualistic* tension, but rather formed a series, where uniformity and diversity, tonal stability and modulation are combined into a living, *discursive* whole. Beethoven's early piano sonatas—differing considerably from the traditional model as codified by Marx—fit perfectly in this context [33]. Beethoven as well as Mozart before him [30] seem to have been influenced by Italian music. Galeazzi's account of sonata form—based on Italian composers of the 1770s and 1780s (Mozart and Beethoven were unknown to him)—points towards such an understanding as reflected in our visualizations.

6. CONCLUSIONS AND OUTLOOK

In this paper, we have demonstrated the use of a computational analysis system for shedding new light on a research question at the heart of historical musicology. The method relies on audio recordings and visualizes the diatonic scale content of a piece in an objective and interpretable way, providing an easy, at-a-glance insight into the phases of stability, instability, and tonal transition. Being aware of several alternative analysis strategies, we plan to work on a closer interrogation of the method, its relation to local key analysis, and the comparison of the output to systematic human analyses as provided by [12, 34, 35].

Even though any type of automated approach can never achieve the flexibility of human analysis, we have shown that it can provide an overview of large-scale structures, thus aiding the research process of historical musicology. Since this approach can be scaled up easily without requiring human annotations [36], it allows for corpus studies in a novel order of magnitude, which can enrich musicological research. In future work, we thus intend to apply this method to a wider range of musical contexts involving extensive corpora and individual large-scale works, both of which would benefit from these "at-a-glance" reductions.

---

[4] Galeazzi does not at all favor a monothematic structure as found sometimes in the works of Joseph Haydn. The model described by him is open to new motives in any part of the exposition, not just in the sense of two contrasting themes as modelled by Marx [18].

7. REFERENCES

[1] S. R. Holtzman, "A program for key determination," *Interface*, vol. 6, no. 1, pp. 29–56, 1977.

[2] D. Temperley, "An algorithm for harmonic analysis," *Music Perception*, vol. 15, no. 1, pp. 31–68, 1997.

[3] S. T. Madsen and G. Widmer, "Key-finding with interval profiles," in *Proceedings of the International Computer Music Conference (ICMC)*, Copenhagen, Denmark, 2007, pp. 212–215.

[4] B. Schuller and B. Gollan, "Music theoretic and perception-based features for audio key determination," *Journal of New Music Research*, vol. 41, no. 2, pp. 175–193, 2012.

[5] C. L. Krumhansl and E. J. Kessler, "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys," *Psychological Review*, vol. 89, no. 4, pp. 334–368, 1982.

[6] T. Rocher, M. Robine, P. Hanna, and L. Oudre, "Concurrent estimation of chords and keys from audio," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 141–146.

[7] H. Papadopoulos and G. Peeters, "Local key estimation from an audio signal relying on harmonic and metrical structures," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1297–1312, 2012.

[8] T. Fujishima, "Realtime chord recognition of musical sound: A system using common lisp music," in *Proceedings of the International Computer Music Conference (ICMC)*, Beijing, China, 1999, pp. 464–467.

[9] V. Konz, M. Müller, and R. Kleinertz, "A cross-version chord labelling approach for exploring harmonic structures—a case study on Beethoven's Appassionata," *Journal of New Music Research*, vol. 42, no. 1, pp. 61–77, 2013.

[10] J. Pauwels, K. O'Hanlon, E. Gómez, and M. B. Sandler, "20 years of automatic chord recognition from audio," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 54–63.

[11] P. R. Illescas, D. Rizo, and J. M. Iñesta, "Harmonic, melodic, and functional automatic analysis," in *Proceedings of the International Computer Music Conference (ICMC)*, Copenhagen, Denmark, 2007, pp. 165–168.

[12] T. Chen and L. Su, "Functional harmony recognition of symbolic music data with multi-task recurrent neural networks," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 90–97.

[13] G. Micchi, M. Gotham, and M. Giraud, "Not all roads lead to rome: Pitch representation and model architecture for automatic harmonic analysis," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 42–54, 2020.

[14] D. Byrd and J. G. Simonsen, "Towards a standard testbed for optical music recognition: Definitions, metrics, and page images," *Journal of New Music Research*, vol. 44, no. 3, pp. 169–195, 2015.

[15] I. Fujinaga, A. Hankinson, and L. Pugin, "Automatic score extraction with Optical Music Recognition (OMR)," in *Springer Handbook of Systematic Musicology*. Springer, 2018, pp. 299–311.

[16] C. Weiß and J. Habryka, "Chroma-based scale matching for audio tonality analysis," in *Proceedings of the Conference on Interdisciplinary Musicology (CIM)*, Berlin, Germany, 2014, pp. 168–173.

[17] N. Jiang and M. Müller, "Automated methods for analyzing music recordings in sonata form," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Curitiba, Brazil, 2013, pp. 595–600.

[18] A. B. Marx, *Die Lehre von der musikalischen Komposition [The School of Musical Composition]*. Leipzig, Germany: Breitkopf und Härtel, 1837–1847.

[19] J. Hepokoski and W. Darcy, *Elements of Sonata Theory. Norms, Types, and Deformations in the Late-Eighteenth-Century Sonata*. Oxford University Press, 2006.

[20] L. Bigo, M. Giraud, R. Groult, N. Guiomard-Kagan, and F. Levé, "Sketching sonata form structure in selected classical string quartets," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 752–759.

[21] P. Allegraud, L. Bigo, L. Feisthauer, M. Giraud, R. Groult, E. Leguy, and F. Levé, "Learning sonata form structure on Mozart's string quartets," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 2, no. 1, pp. 82–96, 2019.

[22] M. Müller and S. Ewert, "Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Miami, Florida, USA, 2011, pp. 215–220.

[23] Z. Gárdonyi and H. Nordhoff, *Harmonik*, 2nd ed. Wolfenbüttel: Möseler, 2002.

[24] C. Weiß, "Computational methods for tonality-based style analysis of classical music audio recordings," Ph.D. dissertation, Ilmenau University of Technology, Ilmenau, Germany, 2017.

[25] C. L. Krumhansl, *Cognitive Foundations of Musical Pitch*. Oxford, UK: Oxford University Press, 1990.

[26] D. Temperley, "What's key for key? The Krumhansl-Schmuckler key-finding algorithm reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.

[27] D. F. Tovey, *A Companion to Beethoven's Pianoforte Sonatas*. The Associated Board of the Royal Schools of Music, 1931.

[28] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. MIT Press, 1983.

[29] C. S. Sapp, "Visual hierarchical key analysis," *ACM Computers in Entertainment*, vol. 3, no. 4, pp. 1–19, 2005.

[30] S. Klauk and R. Kleinertz, "Mozart's italianate response to Haydn's opus 33," *Music & Letters*, vol. 97, pp. 575–621, 2016.

[31] C. Dahlhaus, *Ludwig van Beethoven. Approaches to his music*. Oxford University Press, 1991.

[32] F. Galeazzi, *Theoretical-Practical Elements of Music, Parts III & IV, a translation of Elementi teorico-pratici di musica*, D. Burton and G. W. Harwood, Eds. Edited by Deborah Burton and Gregory W. Harwood, University of Illinois Press, 2012.

[33] S. Klauk, "Über Ausdruckscharakter und Programmmusik: Ludwig van Beethovens opp. 13 und 18 im Kontext italienischer Kammermusik," in *Nineteenth-Century Programme Music. Creation, Negotiations, Reception*, J. Kregor, Ed. Turnhout: Speculum musicae 32, Brepols, 2018, pp. 3–23.

[34] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, "The Annotated Beethoven Corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets," *Frontiers in Digital Humanities*, vol. 5, p. 16, 2018.

[35] M. Gotham and M. Ireland, "Taking form: A representation standard, conversion code, and example corpora for recording, visualizing, and studying analyses of musical form," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 693–699.

[36] C. Weiß and M. Müller, "Computergestützte Visualisierung von Tonalitätsverläufen in Musikaufnahmen. Möglichkeiten für die Korpusanalyse," in *Instrumentalmusik neben Haydn und Mozart. Analyse, Aufführungspraxis und Edition*, S. Klauk, Ed. Saarbrücker Studien zur Musikwissenschaft 20, Königshausen & Neumann, 2020, pp. 107–130.

THE JAZZ HARMONY TREEBANK

Daniel Harasim[1]   **Christoph Finkensiep**[1]   **Petter Ericson**[1]
Timothy J. O'Donnell[2]   **Martin Rohrmeier**[1]

[1] Digital and Cognitive Musicology Lab, École Polytechnique Fédérale de Lausanne, Switzerland
[2] Department of Linguistics, McGill University, Canada

daniel.harasim@epfl.ch

ABSTRACT

Grammatical models which represent the hierarchical structure of chord sequences have proven very useful in recent analyses of Jazz harmony. A critical resource for building and evaluating such models is a ground-truth database of syntax trees that encode hierarchical analyses of chord sequences. In this paper, we introduce the Jazz Harmony Treebank (JHT), a dataset of hierarchical analyses of complete Jazz standards. The analyses were created and checked by experts, based on lead sheets from the open iRealPro collection. The JHT is publicly available in JavaScript Object Notation (JSON), a human-understandable and machine-readable format for structured data. We additionally discuss statistical properties of the corpus and present a simple open-source web application for the graphical creation and editing of trees which was developed during the creation of the dataset.

1. INTRODUCTION

Jazz music exhibits hierarchical relations between chords. This is particularly apparent in the fact that virtually any chord of a Jazz standard can be prepared by an applied dominant or subdominant. In fact, many chord sequences can be explained as the recursive application of such preparations [41]. Chords that are far apart in time can therefore be directly related, establishing long-range dependencies that can span whole formal sections of pieces. Such hierarchical structures also correlate with empirical findings from music perception research [25]. This is by no means to say that hierarchies are the only relevant relations between chords. Hierarchical chord relations are, however, underrepresented in computational models of harmony to date; the here presented dataset is intended to ease the development of hierarchical models.

Inspired by Schenkerian theory [3, 45] and generative syntax formalisms for natural language, generative theories of harmonic syntax model the hierarchical relations in chord sequences based on formal grammatical

devices such as context-free grammars. Recent research uses formal grammars to represent hierarchical relations in melodies [1, 10, 13, 16, 24, 34], chord sequences [15, 19, 43], and rhythms [18, 29]. The fields of application include music theory [37, 40], music psychology [25, 42], automatic harmonic analysis [7, 8], and automatic music transcription [11, 30, 35].

The aim of this article is to present the Jazz Harmony Treebank (JHT), a dataset of hierarchical harmonic analyses of Jazz standards by music experts in a human-understandable and machine-readable format. We report on the creation of the treebank, describe the musical interpretation of the syntax trees, and explain the decisions that were made to meet the challenges of the annotation procedure. The dataset is available on GitHub. [1]

Treebanks are of particular importance for the study of hierarchical models and their applications. In linguistics, they have been and remain instrumental for many natural language processing tasks. The well-known Penn Treebank [28], first published in the early nineties, is an instructive example since it has been used as an object of study in and of itself [12], as a basis for publishing additional treebanks with different paradigms [21] and for different languages [27], and–most prominently–as a dataset for training and evaluating machine-learning methods [22, 31, 44].

The present article describes the creation process of the JHT. We take this as an opportunity to study the details of harmonic syntax using several concrete examples of Jazz standards. The major challenge of this application lies in the many individual decisions analysts have to take to address the ambiguity of music. Importantly, our goal is not to create uniform syntax trees of Jazz chord sequences, but to describe individual and subjective listening experiences in an unambiguous formal representation. Harmonic relations in sufficiently long chord sequences can be perceived in several ways, without one interpretation being clearly preferable. Therefore, the syntax trees of the JHT are best understood as proposals with a clear interpretation. The trees provide a basis for further analytical discussions, sophisticated computational models, and for education.

1.1 Related Symbolic Datasets

Many existing collections of symbolic data about chord sequences concentrate on providing chord labels for harmonic entities. Two prominent datasets of time-aligned

---

[1] https://github.com/DCMLab/JazzHarmonyTreebank

chord symbols were created by Harte et al. [20] and Burgoyne et al. [2] to study automatic chord transcription from audio. Neuwirth et al. [36] take a more music-theoretically motivated approach by proposing a chord-symbol representation for Western classical music and apply it to scale degree analyses of Beethoven's string quartets. Chen and Su [5] and Devaney et al. [9] similarly label excerpts from common-practice tonality. Micchi et al. [32] combine existing Roman numeral analyses into a meta-dataset.

The datasets just mentioned use chord labels to analyze music given as audio data or in a symbolic representation. Since we analyze the relations between the chords of such sequences, this study is located at a higher level of abstraction. Only a few datasets of hierarchical analyses of sequential musical data are available in divergent formats [39]. Hamanaka et al. [17] and Kirlin [23] created two datasets of tree analyses of melodies of Western Classical Music. Gotham and Ireland [14] study musical form by the creation of datasets in a hierarchical representation. Moss et al. [33] study Brazilian Choro using a dataset with hierarchical form encoding. Granroth-Wilding and Steedman [15] provide a dataset of 76 sub-sequences of Jazz standards with partial harmonic grouping labels. In contrast to previous research that analyzed snippets of musical pieces, the JHT consists of 150 full chord sequences of Jazz standards with complete harmonic syntax trees.

2. HARMONIC SYNTAX

A harmonic syntax tree, as shown in Figure 1a, denotes a mental representation of a musical piece as a whole. Unlike sequential models that describe how, for instance, a sequence of chord symbols is generated chord by chord from the start to the end, hierarchical models describe how the skeleton of a piece is generated and recursively elaborated [43]. In Jazz, the most prominent of those elaboration operations are the duplication of chords and the preparation of a chord by an applied dominant. Each application of an operation establishes a direct relation between two chords. A syntax tree consists exactly of the sum of all those relations. It is therefore not directly a model for first-time listening of a musical piece, but rather for the abstract representation of musicians or listeners who are (implicitly or explicitly) aware of a piece's harmonic relations. This usage of the word *syntax* is closely related to generative syntax formalisms of natural language that address the question of which relations between words a listener must notice to understand the meaning of a sentence [6].

The scope of this paper is limited to tonal Jazz, including Swing, Bossa Nova, Jazz Blues, Bebop, Cool Jazz, and Hard Bop, and excluding parts of traditional Blues, Modal Jazz, Free Jazz, and Modern Jazz. We furthermore excluded tunes such as *Groovin' High* whose harmonic structure requires even more expressive representations than trees. [2] The general idea of harmonic syntax is, however, also applicable to other musical styles such as Western classical music.

2.1 Prolongation and Preparation as Fundamental Principles

In the following, we present the syntactic formalism with a particular emphasis on its musical interpretation. The concept of functional harmony describes an expectation-realization structure between musical objects such as notes, chords, and keys. Consider for example the chords of the final cadence of the Jazz standard *Birk's Works*, Fm6 Abm7 Db7 G%7 C7 Fm6, where G%7 denotes a half-diminished seventh chord with root G. Figure 1b shows the expectation-realization structure of this chord sequence. The first Fm6 establishes the tonic and as such creates the expectation that the progression ends with Fm6. The chords Abm7 and Db7 function as the tritone-substituted subdominant and dominant of C7, respectively. They therefore create expectation that resolves in the (temporally distant) chord C7. The chord G%7 functions as a subdominant chord in F minor. It therefore creates expectation that resolves with the dominant chord C7 which itself resolves into the last tonic chord Fm6. We say that the tonic chords constitute a *prolongation*. The subdominant chords *prepare* the dominant chords and the dominant chords *prepare* the tonic chord. Abstractly, we say that a chord X *refers* to a chord Y if X either prolongs or prepares Y. [3]

Prolongation and preparation are the two fundamental principles of functional harmonic syntax [41]. They can be formalized as rules of a context-free grammar with chord symbols both as terminals and nonterminals. In the formalization, *strong prolongations* that prolong chords of the same root and chord form are distinguished from *weak prolongations* that prolong a chord with a functionally equivalent chord (e.g., prolongation of C with Am). Note that this concept of weak prolongation is more general than in the GTTM where prolonging chords are for instance required to have the same root [26]. Strong prolongation is represented by rules of the form $X \longrightarrow X\ X$ for chord symbols X (e.g., Fm6 $\longrightarrow$ Fm6 Fm6). For chord symbols X and Y, rules of the form $X \longrightarrow Y\ X$ and $X \longrightarrow X\ Y$ represent weak prolongations if X and Y are functionally equivalent (e.g., Fm6 $\longrightarrow$ Ab Fm6). If otherwise X and Y are not functionally equivalent, $X \longrightarrow Y\ X$ represents a preparation (e.g., Fm6 $\longrightarrow$ C7 Fm6).

The practise of having no separate alphabet of nonterminal symbols, and requiring each binary rule to have a left-hand side symbol also on the right-hand side, is related to dependency grammars [38] and categorical grammars [47] which are well-known in computational linguistics and natural language processing. The symbol that appears both on the left-hand side and the right-hand side is called the *head* of the rule. In our setting of prolongation and preparation, the prolonged (resp. prepared) chord is the head. Therefore, weak prolongation rules may be left- or right-headed, while preparation rules are always right-headed. In sum, our harmony grammar consists of the following rules which model strong prolongation, weak pro-

---

[2] *Groovin' High* exhibits crossing harmonic dependencies between a tonic prolongation from m1 to m5 and a dominant preparation from m4 to m7. A similar tune is *Out of Nowhere*.

[3] In contrast to models based on the *Generative Theory of Tonal Music* [26], we exclude the concept of departure as a primitive relation, because it is not consistent with our formalization of the expectation-realization structure.

longation, and preparation, respectively,
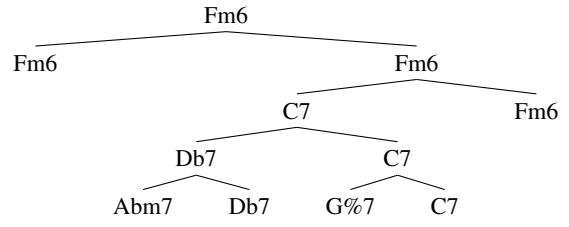
$X \longrightarrow X\ X$ for any chord X (strong prol.)

$X \longrightarrow Y\ X \mid X\ Y$ for any chord X and a   (weak prol.)
functionally equivalent
chord Y

$X \longrightarrow Y\ X$ for any chord X and a   (preparation)
chord Y that prepares X

The tree in Figure 1a is a parse tree of the chord sequence Fm6 Abm7 Db7 G%7 C7 Fm6 under such a grammar of harmonic structure. Those parse trees represent exactly the same information as expectation-realization structures such as shown in Figure 1b: Undirected edges correspond to strong prolongations and directed edges correspond to either weak prolongations or preparations. This short example is unambiguous–it has only one plausible syntactic structure. In general, however, there are many syntax trees possible for a chord sequence. Grammar rules and syntax trees can then be weighted by probabilities that capture the plausibility of an analysis [1, 19, 24]. To identify the syntax tree that most accurately describes one's perception of the harmonic structure, other dimensions such as rhythm, form, and melody must also be taken into account. Even the artistic interpretation of a musical performance and the individual musical background of listeners have the potential to influence the perceived harmonic structure of a piece. A formal grammar that purely models chord symbols can therefore only answer the question "Is this a plausible syntax tree for a Jazz standard?", but not the question "Is this tree a good analysis of that particular tune in a particular context?". Until more complete models of musical structure are developed that integrate all relevant musical dimensions, the second question can only be answered by humans.
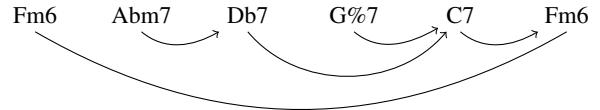
2.2 Complete Constituents and Open Constituents

Constituents formalize the notion of a musical unit such as a chord or a phrase. In the syntax tree shown in Figure 1a, the complete constituents are exactly the subsequences that are leafs of single subtrees, such as the subsequence Abm7 Db7 G%7 C7. Formally, we call a subsequence a *complete constituent* if it contains a chord, called the *head*, that is transitively referred to by all other chords of the sequence.[4] For instance, the chord C7 is the head of the phrase Abm7 Db7 G%7 C7 and Fm6 is the head of the whole sequence Fm6 Abm7 Db7 G%7 C7 Fm6. In cases in which a constituent is constituted by a strong prolongation (e.g., for the whole sequence of this example), we use the convention that the head is the right chord symbol. Since only the head of a complete constituent is allowed to refer to a chord outside the constituent, the concept of expectation-realization references is generalizable to complete constituents: we say that a complete constituent refers to a chord X if its head refers to X.

---
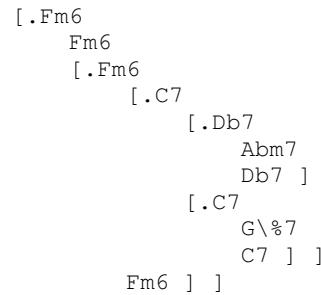
[4] Note that the word head is used both for rules and constituents. This is not a problem since the head of a constituent is always the head of the top-most rule of its (sub-)tree analysis.
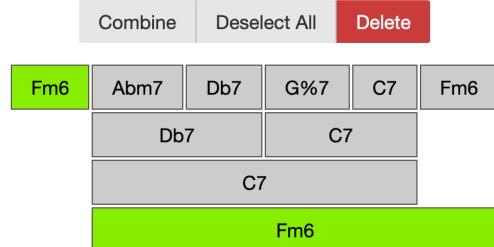
(a) Part of the harmonic syntax tree of *Birks's Works* from the treebank.

(b) Harmonic expectation-realization structure. This graph stands in 1-to-1 relation to the syntax tree shown in (a). Directed and undirected edges denote preparations and prolongations, respectively.

```
[.Fm6
    Fm6
    [.Fm6
        [.C7
            [.Db7
                Abm7
                Db7 ]
            [.C7
                G\%7
                C7 ] ]
        Fm6 ] ]
```

(c) String representation of the syntax tree in tikz-qtree format. This string is created using the tree annotation app shown in (d). The tree plot is shown in (a).
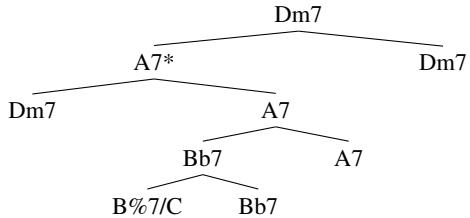
(d) Screenshot of tree annotation app. Each button represents a tree node. The user is selecting the green buttons to combine them to the full tree.
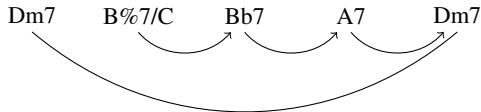
```
{"label": Fm6, "children": [
  {"label": "Fm6", "children": []},
  {"label": "Fm6", "children": [
    {"label": "C7", "children": [
      {"label": "Db7", "children": [
        {"label": "Abm7", "children": []},
        {"label": "Db7", "children": []}]},
      {"label: "C7", "children": [
        {"label: "G%7", "children": []},
        {"label: "C7", "children": []}]}]},
    {"label": "Fm6", "children": []}]}]}
```

(e) Tree string in JSON format, automatically converted from tikz-qtree format shown in (c).
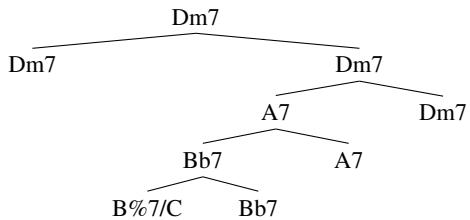
Figure 1: Syntax tree of the final chords of the Jazz standard *Birk's works* in different representations.

Dm7
A7*
Dm7
A7
Bb7
A7
B%7/C
Bb7

(a) Syntax tree using an open constituent that is marked with an asterisk.

Dm7   B%7/C   Bb7   A7   Dm7

(b) Harmonic expectation-realization structure of the syntax tree in (a). Since that tree contains an open constituent, the syntax tree and the expectation structure do not stand in 1-to-1 relation.

Dm7
Dm7
Dm7
A7
Dm7
Bb7
A7
B%7/C
Bb7

(c) Resolution of the open constituent in the syntax tree shown in (a). This tree stands in 1-to-1 relation to the expectation-realization structure in (b).

Figure 2: Hierarchical analysis of the initial chords of the Jazz standard *Why Don't You Do Right?* using open constituents (marked with asterisks).

In addition to complete constituents, one other constituent type is used in the JHT analyses. Consider for example the first four measures of the Jazz standard *Why Don't You Do Right?*,

| Dm7 B%7/C | Bb7 A7 | Dm7 B%7/C | Bb7 A7 |,

where B%7/C denotes a half-diminished seventh chord with root B and a C in the bass. The first two measures constitute a phrase following the *Lamento* schema (a stepwise descending movement of the bass from scale degree I to scale degree V [4]) that is repeated multiple times in the song. Since the transition from A7 to Dm7 does not sound like a resolution but more like a jump or an interruption (partly because of the repetition of the first two measures), we assume that A7 does not resolve into the following tonic Dm7, but into a tonic later in the song. Therefore, the phrase Dm7 B%7/C Bb7 A7 does constitute some kind of unit as shown in Figure 2a.

Since Dm7 and A7 both refer to a chord outside the phrase (see Figure 2b), the phrase does not have a head. It is therefore not a complete constituent. We call such constituents, in which multiple chords refer to a chord outside of the phrase, *open constituents*. The chords of an open constituent that refer to a chord outside of the constituent are called *chords with open references*. In the example of

Why Don't You Do Right?, the chords Dm7 and A7 are the chords with open references of the open constituent Dm7 B%7/C Bb7 A7. Both chords Dm7 and A7 refer to the same tonic chord Dm7.

The JHT allows a single type of open constituent, called *restricted* open constituent, which consists of two adjacent constituents that refer to the same chord later in the piece. Since all constituents considered in the JHT are restricted in that way, we simple refer to them as open constituents. The restriction enables a further generalization of expectation-realization references to open constituents: We say that an open constituent refers to the chord to which all of its chords with open references refer. As shown in Figure 2a, the topmost node of an open constituent is labeled by the chord symbol of the right child of the node and additionally marked with an asterisk.

Other examples of open constituents are (i) I-VI-II-V-like phrases in *I Got Rhythm* and *I Can't Give You Anything But Love* and, in particular, (ii) tunes of form ABAC in which the B-part ends in a half cadence such as *All of Me*, *How High the Moon*, and *A Fine Romance*. *Summertime*, shown in Figure 3, is a prototypical example of a song with a ABAC form and a half cadence at the end of the B section. The interruption after the half cadence is supported by the movement from scale degree 3 to scale degree 2 in the melody and denoted using an open constituent.

2.3 Interpretation of Open Constituents as Prolongation-Preparation Structures
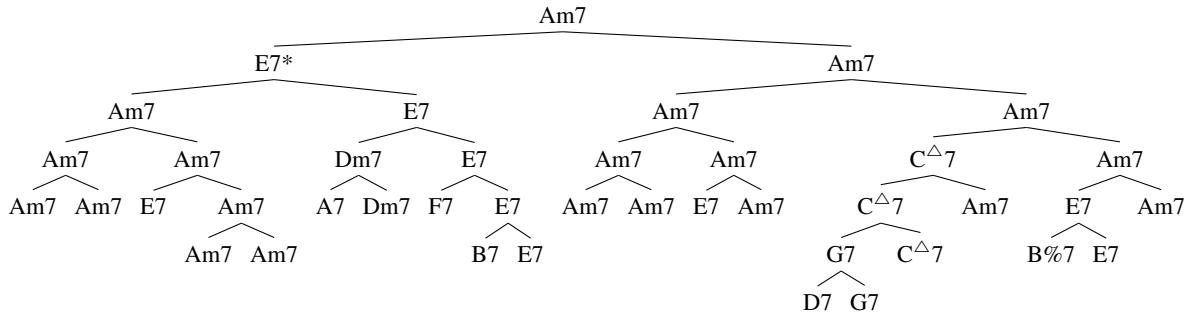
Syntax trees containing open constituents are interpretable as expectation-realization structures as shown in Figure 2. The interpretation procedure transforms a syntax tree that contains open constituents (e.g., Figure 2a) in to a tree that only represents prolongation and preparation operations (e.g., Figure 2c). This transformed tree then characterizes the expectation-realization structure (e.g., Figure 2b). Since open constituents are explicitly marked with asterisks, their interpretation is unambiguous.

To formalize the interpretation of open constituents, let Y^* be the chord symbol labeling an open constituent consisting of two constituents labeled with chord symbols X and Y. Let further be Z the chord symbol that is referenced by both X and Y. The reference is expressed by Z being the right sibling of the open constituent. The conversion then transforms

Z
Y^*
Z
X
Y
into
Z
X
Z
Y
Z

In the more general case of nested open constituents, the conversion is recursively applied from the root to the leaves of the tree (i.e., top-down).

The JHT contains trees for both representations, with open subtrees and in pure preparation-prolongation form. A python script was used to automatically transform the former into the letter. The script and additional utilities such as for tree traversal and drawing are provided with the treebank.

Figure 3: Complete syntax tree of the Jazz standard *Summertime* (turnaround omitted). The top levels of the tree reflect the ABAC form the song using an open constituent.

3. TREE ANNOTATION TOOL

The trees of the JHT are created using a graphical interface implemented as a simple web application, which was developed during the creation of the treebank. The source code of the application is written in ClojureScript (which compiles to JavaScript) and publicly available on GitHub. The application itself is hosted on GitHub pages and can be used independently of this dataset.[5] A screenshot of the application is shown in Figure 1d. The main part of the user interface displays a syntax tree that is represented by a hierarchical button layout. The user interface also contains an input-output section and buttons for creating, deleting, and deselecting tree nodes.

To create a syntax tree, the user inputs a sequence of space-separated strings such as chord symbols. To create an inner node of the tree, the nodes that become the child nodes of the new inner node are selected and combined by pressing a button or a key shortcut. Since the trees are mostly right-headed, the label of the rightmost child is used for the new node by default, but the label of a node can be changed arbitrarily. The output of the application is given as a string representation of the tree in tikz-qtree format as shown in Figure 1c and in JSON format as shown in Figure 1e.[6] Existing trees can be edited by loading them in any of these two formats. Since the application is designed to be agnostic to annotation conventions, it allows arbitrary labels and rule arities.

4. ANNOTATION PROCEDURE

All analyses in the dataset begin from chord sequences drawn from the iRealPro collection of Jazz standards. This collection was created by the user community of the iReal-Pro app[7] and transferred into kern format by Shanahan et al. [46].[8] We transformed the data into a JSON-like format and occasionally corrected individual chord symbols when we noticed serious differences between the iRealPro data and publicly available *Real Books* (i.e., collections of lead sheets.). Annotations of bass notes and optional chord
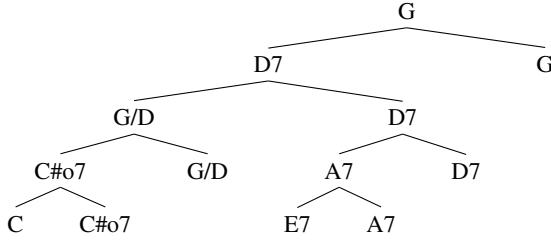
tones such as ninths and elevenths were excluded from the chord symbols. Chord symbols with a duration of more than one measure were split into multiple chord symbols. 150 Jazz standards were selected for analysis (i) by filtering pieces that are within the scope of the theory of harmonic syntax described in Section 2 and (ii) by preferring shorter pieces. If applicable, turnarounds at the end of a lead sheet were deleted or a final tonic chord not contained in the lead sheet was added. All repetitions were unfolded and codas were appended at the positions indicated in the lead sheet. The selected Jazz standards were initially analyzed by the first author and a student assistant. The analyses were then reviewed by the second and the third author and discussed in the group. To ensure consistent analyses across all 150 Jazz standards, all final tree editing was performed by the first author.

Every hierarchical analysis denotes at least one author's mental representation of the harmonic structure of a Jazz standard. Each analysis is therefore also influenced by other musical features such as harmonic rhythm, phrasing, musical form, and melody. In ambiguous cases, the analyst chose the option that he seemed most important. These choices were necessary, because a single syntax tree can only encode one harmonic function for each chord. For example in the key C major, a C major triad can act as a tonic or as a preparation of a following F major chord. For five particularly ambiguous tunes, we provide alternative analyses in the treebank.

Since the iRealPro lead sheets were created and collected by the community of the application, the chord symbol usage is not fully consistent across the pieces. For instance, a Fm6 chord symbol can denote a tonic chord in F minor over a Dorian scale or a Bb9 chord with omitted root and fifth in the bass. Another example is that fourth-voicings are commonly denoted as suspension chords while actual suspensions of the scale degree V (e.g., suspension of C and E by B and D in a G major triad) are sometimes denoted as chords over the scale degree I (with or without explicitly mentioning the second inversion).

Furthermore, some chords do not have a proper harmonic function, but are better explained as voice-leading connections between two chords. The chords C C#o7 G/D at the beginning of the final 8 measures of *Bill Bailey* are an example of such a voice-leading connection (see Figure 4). Moreover, these final measures are an example of

---

[5] Link to tree annotation app: `https://dcmlab.github.io/tree-annotation-code/`

[6] `https://www.ctan.org/pkg/tikz-qtree`

[7] `https://irealpro.com/`

[8] The iRealPro dataset is available in kern format at `http://doi.org/10.5281/zenodo.3546040`.

Figure 4: Syntax tree of the final 8 measures of *Bill Bailey* (turnaround omitted).

a common closing pattern. This pattern starts on the scale degree IV in its first measure, then transitions to a suspension of the scale degree V in measure 3, jumps away, and finally approaches the tonic through the cycle of fifths.

5. DATASET SUMMARY
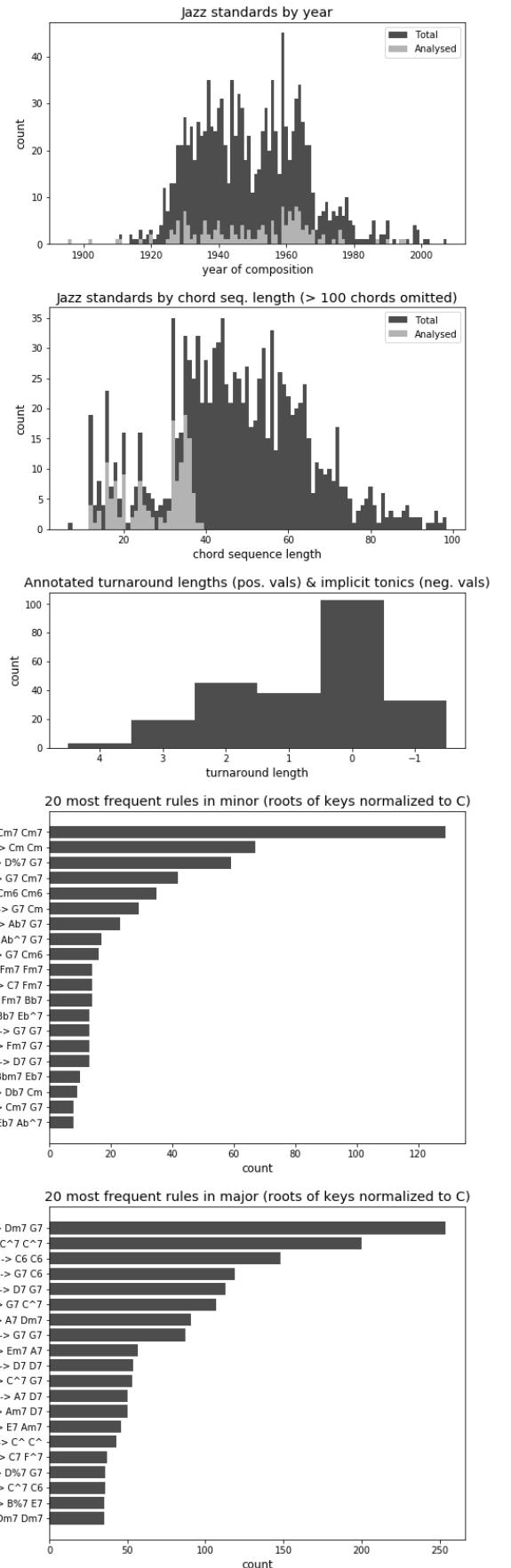
The JHT is provided as a single file in JavaScript Object Notation (JSON) format. For each Jazz standard, this file contains the chord sequence with rhythmical information (measures and beats), metadata about title, composer(s), year of composition, time signature, and key (root & major/minor) as well as the tree analyses. [9]

In addition to the hierarchical analyses, some pieces contain a turnaround annotation represented as an integer. A value of zero means that the Jazz standard ends with a tonic chord. A positive value n means that the lead sheet of the piece ends with a turnaround of length n. For example, the chord sequence of *I love Paris* (in C major) has a turnaround length of $n = 2$, because it ends with the chords Dm7 G7 C6 D%7 G7. A negative turnaround annotation means that the tonic of the piece is not at the end of the piece, but at the beginning. A value of -1 indicates, for example, that the first chord of the chord sequence is the tonic of the piece, like in *Solar*. In rare cases, the tonic is not the first chord but the n-th chord which is represented by a turnaround annotation of $-n$.

The 150 chord sequences analysed in the treebank have an average length of 27.75 and consist of 11697 chords in total with 92 unique chord symbols. The syntax trees consist in total of 3899 binary rule applications with 512 unique rules and 268 open constituents. The average tree height is 7.57.

Further descriptive statistics of the JHT are visualized in Figure 5. The first plot shows that the subset of the analyzed pieces is chosen relatively independently from the year of composition. The second plot shows the bias for short pieces in this subset. The third plot shows that the length of turnarounds, if present, usually ranges between 1 and 3. The two last plots show separately for major and minor keys how often a context-free grammar rule is used in the hierarchical analyses. For these plots, all chord sequences were transposed to C major or to C minor, respectively. Prolongations of the tonic, preparations of the tonic by the fifth scale degree, and preparations of the fifth scale degree by the second are by far the most common rules.

---

[9] The metadata was copied from the iRealPro dataset without detailed validity checking. It is provided for convenience.

Figure 5: Plots of summary statistics of the tree analyses. See the main text for further explanation.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] S. Abdallah, N. Gold, and A. Marsden. Analysing Symbolic Music with Probabilistic Grammars. In David Meredith, editor, *Computational Music Analysis*, pages 157–189. Springer International Publishing, Cham, 2016.

[2] J. A. Burgoyne, J. Wild, and I. Fujinaga. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, Miami, Florida, 2011.

[3] A. Cadwallader and D. Gagné. *Analysis of Tonal Music: A Schenkerian Approach*. Oxford University Press, Oxford, 2nd edition, 2007.

[4] W. E. Caplin. Topics and formal functions: The case of the lament. *The Oxford Handbook of Topic Theory*, page 415, 2014.

[5] T. Chen and L. Su. Functional Harmony Recognition of Symbolic Music Data with Multi-task Recurrent Neural Networks. In *ISMIR*, pages 90–97, 2018.

[6] N. Chomsky. *Aspects of the Theory of Syntax*, volume 11. MIT press, 1965.

[7] W. B. De Haas, J. P. Magalhães, and F. Wiering. Improving Audio Chord Transcription by Exploiting Harmonic and Metric Knowledge. *International Society for Music Information Retrieval Conference (ISMIR)*, pages 295–300, 2012.

[8] W. B. De Haas, J. P. Magalhães, F. Wiering, and R. C. Veltkamp. Automatic functional harmonic analysis. *Computer Music Journal*, 37(4):37–53, 2013. Publisher: MIT Press.

[9] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula. Theme And Variation Encodings with Roman Numerals (TAVERN). In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, Malaga, Spain, 2015.

[10] C. Finkensiep, R. Widdess, and M. Rohrmeier. Modelling the Syntax of North Indian Melodies with a Generalized Graph Grammar. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pages 462–269. ISMIR, 2019.

[11] F. Foscarin, F. Jacquemard, P. Rigaux, and S. Masahiko. A parse-based framework for coupled rhythm quantization and score structuring. In *Mathematics and Computation in Music*, pages 248–260, Cham, 2019. Springer International Publishing.

[12] R. Gaizauskas. Investigations into the grammar underlying the Penn Treebank II. *Research Memorandum CS-95-25, Department of Computer Science, Univeristy of Sheffield*, pages 185–189, 1995.

[13] É. Gilbert and D. Conklin. A probabilistic context-free grammar for melodic reduction. In *Proceedings of the International Workshop on Artificial Intelligence and Music, 20th International Joint Conference on Artificial Intelligence*, pages 83–94, 2007.

[14] M. Gotham and M. T. Ireland. Taking Form: A Representation Standard, Conversion Code, and Example Corpus for Recording, Visualizing, and Studying Analyses of Musical Form. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019.

[15] M. Granroth-Wilding and M. Steedman. A Robust Parser-Interpreter for Jazz Chord Sequences. *Journal of New Music Research*, 43(4):355–374, October 2014.

[16] R. Groves. Automatic Melodic Reduction Using a Supervised Probabilistic Context-free Grammar. *Proceedings of the 17th International Society for Music Information Retrieval Conference*, 2016.

[17] M. Hamanaka, K. Hirata, and S. Tojo. Musical Structural Analysis Database based on GTTM. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014.

[18] D. Harasim, T. J. O'Donnell, and M. Rohrmeier. Harmonic Syntax in Time: Rhythm Improves Grammatical Models of Harmony. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, 2019.

[19] D. Harasim, M. Rohrmeier, and T. J. O'Donnell. A Generalized Parsing Framework for Generative Models of Harmonic Syntax. *19th International Society for Music Information Retrieval Conference*, 2018.

[20] C. Harte, M. Sandler, S. Abdallah, and E. Gomez. Symbolic Representation of Musical Chords: A Proposed Syntax for Text Annotations. In *Proceedings of the 6th International Society for Music Information Retrieval Conference*, London, UK, 2005.

[21] J. Hockenmaier and M. Steedman. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396, 2007.

[22] J. Katz-Brown, S. Petrov, R. McDonald, F. Och, D. Talbot, H. Ichikawa, M. Seno, and H. Kazawa. Training a parser for machine translation reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, page 183–192, USA, 2011. Association for Computational Linguistics.

[23] P. B. Kirlin. *A Probabilistic Model of Hierarchical Music Analysis*. PhD thesis, University of Massachusetts Amherst, 2014.

[24] P. B. Kirlin and D. D. Jensen. Probabilistic Modeling of Hierarchical Music Analysis. In *ISMIR*, pages 393–398, 2011.

[25] S. Koelsch, M. Rohrmeier, R. Torrecuso, and S. Jentschke. Processing of hierarchical syntactic structure in music. *Proceedings of the National Academy of Sciences*, 110(38):15443–15448, 2013.

[26] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. Cambridge, MA, 1983.

[27] M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, pages 466–467. Cairo, 2004.

[28] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The Penn Treebank. 1993.

[29] A. McLeod and M. Steedman. Meter Detection in Symbolic Music Using a Lexicalized PCFG. *Proceedings of the 14th Sound and Music Computing Conference*, 2017.

[30] A. McLeod and M. Steedman. Meter Detection and Alignment of MIDI Performance. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[31] G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

[32] G. Micchi, M. Gotham, and M. Giraud. Not All Roads Lead to Rome: Pitch Representation and Model Architecture for Automatic Harmonic Analysis. *Transactions of the International Society for Music Information Retrieval*, 3(1):42–54, May 2020.

[33] F. C. Moss, W. F. Souza, and M. Rohrmeier. Harmony and form in Brazilian Choro: A corpus-driven approach to musical style analysis. *Journal of New Music Research*, 2020.

[34] E. Nakamura, M. Hamanaka, K. Hirata, and K. Yoshii. Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[35] E. Nakamura, K. Itoyama, and K. Yoshii. Rhythm transcription of MIDI performances based on hierarchical Bayesian modelling of repetition and modification of musical note patterns. In *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016.

[36] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier. The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets. *Frontiers in Digital Humanities*, 5, July 2018.

[37] M. Neuwirth and M. Rohrmeier. Towards a syntax of the Classical cadence. In *What is a Cadence*, pages 287–338. Leuven University Press, 2015.

[38] J. Nivre. Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32, 2005.

[39] D. Rizo and A. Marsden. A standard format proposal for hierarchical analyses and representations. In *Proceedings of the 3rd International workshop on Digital Libraries for Musicology*, pages 25–32, 2016.

[40] M. Rohrmeier. Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, 5(1):35–53, 2011.

[41] M. Rohrmeier. The syntax of jazz harmony: Diatonic tonality, phrase structure, and form. *Music Theory and Analysis*, 7(1):1–63, 2020. Publisher: Leuven University Press.

[42] M. Rohrmeier and I. Cross. Tacit tonality-Implicit learning of context-free harmonic structure. In *ESCOM 2009: 7th Triennial Conference of European Society for the Cognitive Sciences of Music*, 2009.

[43] M. Rohrmeier and M. Pearce. Musical syntax I: theoretical perspectives. In *Springer handbook of systematic musicology*, pages 473–486. Springer, 2018.

[44] A. Sarkar. Applying co-training methods to statistical parsing. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001.

[45] H. Schenker. *Der Freie Satz. Neue musikalische Theories and Phantasien*. Universal edition edition, 1935.

[46] D. Shanahan, Y. Broze, and R. Rodgers. A Diachronic Analysis of Harmonic Schemata in Jazz. In *Proceedings of the 12th International Conference on Music*

Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music, pages 909–917, 2012.

[47] M. Steedman and J. Baldridge. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and explicit models of grammar*, pages 181–224, 2011.

DOWNBEAT TRACKING WITH TEMPO-INVARIANT CONVOLUTIONAL NEURAL NETWORKS

Bruno Di Giorgi
Apple Inc.
bdigiorgi@apple.com

Matthias Mauch
Apple Inc.
mmauch@apple.com

Mark Levy
Apple Inc.
mark_levy@apple.com

ABSTRACT

The human ability to track musical downbeats is robust to changes in tempo, and it extends to tempi never previously encountered. We propose a deterministic time-warping operation that enables this skill in a convolutional neural network (CNN) by allowing the network to learn rhythmic patterns independently of tempo. Unlike conventional deep learning approaches, which learn rhythmic patterns at the tempi present in the training dataset, the patterns learned in our model are tempo-invariant, leading to better tempo generalisation and more efficient usage of the network capacity.

We test the generalisation property on a synthetic dataset created by rendering the Groove MIDI Dataset using FluidSynth, split into a training set containing the original performances and a test set containing tempo-scaled versions rendered with different SoundFonts (test-time augmentation). The proposed model generalises nearly perfectly to unseen tempi (F-measure of 0.89 on both training and test sets), whereas a comparable conventional CNN achieves similar accuracy only for the training set (0.89) and drops to 0.54 on the test set. The generalisation advantage of the proposed model extends to real music, as shown by results on the GTZAN and Ballroom datasets.

1. INTRODUCTION

Human musicians easily identify the downbeat (the first beat of each bar) in a piece of music and will effortlessly adjust to a variety of tempi, even ones never before encountered. This ability is the likely result of patterns and tempi being processed at distinct locations in the human brain [1].

We argue that factorising rhythm into tempo and tempo-invariant rhythmic patterns is desirable for a machine-learned downbeat detection system as much as it is for the human brain. First, factorised representations generally reduce the number of parameters that need to be learned. Second, having disentangled tempo from pattern we can

transfer information learned for one tempo to all others, eliminating the need for training datasets to cover all combinations of tempo and pattern.

Identifying invariances to disentangle representations has proven useful in other domains [2]: translation invariance was the main motivation behind CNNs [3] — the identity of a face should not depend on its position in an image. Similarly, voices retain many of their characteristics as pitch and level change, which can be exploited to predict pitch [4] and vocal activity [5]. Crucially, methods exploiting such invariances don't only generalise better than non-invariant models, they also perform better overall.

Some beat and downbeat trackers first estimate tempo (or make use of a tempo oracle) and use the pre-calculated tempo information in the final tracking step [6–15]. Doing so disentangles tempo and tempo-independent representations at the cost of propagating errors from the tempo estimation step to the final result. It is therefore desirable to estimate tempo and phase simultaneously [16–20], which however leads to a much larger parameter space. Factorising this space to make it amenable for machine learning is the core aim of this paper.

In recent years, many beat and downbeat tracking methods changed their front-end audio processing from hand-engineered onset detection functions towards beat-activation signals generated by neural networks [21–23]. Deep learning architectures such as convolutional and recurrent neural networks are trained to directly classify the beat and downbeat frames, and therefore the resulting signal is usually cleaner.

By extending the receptive field to several seconds, such architectures are able to identify rhythmic patterns at longer time scales, a prerequisite for predicting the downbeat. But conventional CNN implementations learn rhythmic patterns separately for each tempo, which introduces two problems. First, since datasets are biased towards mid-tempo songs, it introduces a tempo-bias that no post-processing stage can correct. Second, it stores similar rhythms redundantly, once for every relevant tempo, i.e. it makes inefficient use of network capacity. Our proposed approach resolves these issues by learning rhythmic patterns that apply to all tempi.

The two technical contributions are as follows:

1. the introduction of a scale-invariant convolutional layer that learns temporal patterns irrespective of their scale.

2. the application of the scale-invariant convolutional layer to CNN-based downbeat tracking to explicitly learn tempo-invariant rhythmic patterns.

Similar approaches to achieve scale-invariant CNNs, have been developed in the field of computer vision [24, 25], while no previous application exists for musical signal analysis, to the best of our knowledge.

We demonstrate that the proposed method generalises better over unseen tempi and requires lower capacity with respect to a standard CNN-based downbeat tracker. The method also achieves good results against academic test sets.

2. MODEL

The proposed downbeat tracking model has two components: a neural network to estimate the joint probability of downbeat presence and tempo for each time frame, using tempo-invariant convolution, and a hidden Markov model (HMM) to infer a globally optimal sequence of downbeat locations from the probability estimate.

We discuss the proposed scale-invariant convolution in Sec. 2.1 and its tempo-invariant application in Sec. 2.2. The entire neural network is described in Sec. 2.3 and the post-processing HMM in Sec. 2.4.

2.1 Scale-invariant convolutional layer

In order to achieve scale invariance we generalise the conventional convolutional neural network layer.

2.1.1 Single-channel

We explain this first in terms of a one-dimensional input tensor $x \in \mathbb{R}^N$ and only one kernel $h \in \mathbb{R}^{N^*}$, and later generalise the explanation to multiple channels in Sec. 2.1.2. Conventional convolutional layers convolve x with h to obtain the output tensor $y \in \mathbb{R}^{N-N^*+1}$
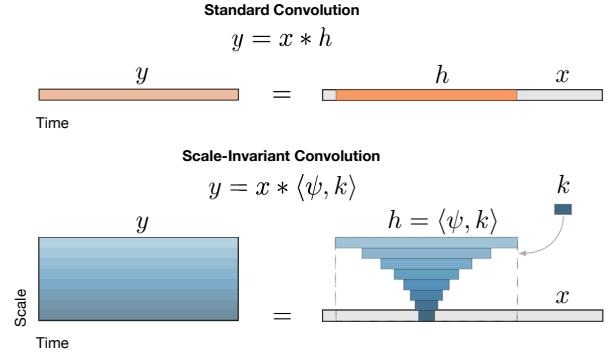
$$y = x * h, \tag{1}$$

where $*$ refers to the discrete convolution operation. Here, the kernel h is updated directly during back-propagation, and there is no concept of scale. Any two patterns that are identical in all but scale (e.g. one is a "stretched" version of the other) cannot be represented by the same kernel.

To address this shortcoming, we factorise the kernel representation into scale and pattern by parametrising the kernel as the dot product $h_j = \langle \psi_j, k \rangle$ between a fixed scaling tensor $\psi_j \in \mathbb{R}^{N^* \times M}$ and a scale-invariant pattern $k \in \mathbb{R}^M$. Only the pattern is updated during network training, and the scaling tensor, corresponding to S scaling matrices, is pre-calculated (Sec. 2.1.3). The operation adds an explicit scale dimension to the convolution output

$$y_j = x * h_j = x * \langle \psi_j, k \rangle. \tag{2}$$

The convolution kernel is thus factorised into a constant scaling tensor ψ and trainable weights k that learn a scale-invariant pattern. A representation of a scale-invariant convolution is shown in Figure 1.

Figure 1. The figure shows a representation of the standard and scale-invariant convolution operations with input/output channel dimensions removed for simplicity. In order to achieve scale invariance, we parametrise the kernel as the dot product of two tensors ψ and k, where ψ is a deterministic scaling tensor and k is the trained part that will learn scale-invariant patterns. The resulting kernel h contains multiple scaled versions of k.

	layer input	
variable	**single-channel**	**multi-channel**
# frames	N	
# pattern frames	M	
# scales	S	
# input channels	1	C_x
# kernels	1	H
signal x	$\mathbb{R}^N$	$\mathbb{R}^{N \times C_x}$
patterns k	$\mathbb{R}^M$	$\mathbb{R}^{M \times C_x \times H}$
kernel h	$\mathbb{R}^{N^* \times S}$	$\mathbb{R}^{N^* \times C_x \times S \times H}$
output y	$\mathbb{R}^{(N-N^*+1) \times S}$	$\mathbb{R}^{(N-N^*+1) \times S \times H}$
scaling tensor ψ	$\mathbb{R}^{N^* \times M \times S}$	
scale indices	$j = 0, \ldots, S-1$	

Table 1. Variables and dimensions.

2.1.2 Multi-channel

Usually the input to the convolutional layer has $C_x > 1$ input channels and there are $H > 1$ kernels. The formulas in Section 2.1 can easily be extended by the channel dimension, as illustrated in Table 1.

2.1.3 Scaling tensor

The scaling tensor ψ contains S scaling matrices from size M to $s_j M$ where s_j are the scale factors.

$$\psi_{n,m,j} = \int_{\tilde{s}} \int_{\tilde{n}} \delta(\tilde{n} - \tilde{s}m)\kappa_n(n - \tilde{n})\kappa_s(s_j - \tilde{s}) d\tilde{n} d\tilde{s}, \tag{3}$$

where δ is the Dirac delta function and κ_n, κ_s are defined as follows:

$$\kappa_n(d) = \sin(\pi d)/(\pi d)$$
$$\kappa_s(d) = \alpha \cos^2(\alpha d\pi/2)\mathcal{H}(1 - \alpha|d|),$$

where $\mathcal{H}$ is the Heaviside step function. The inner integral can be interpreted as computing a resampling matrix for a given scale factor and the outer integral as smoothing along the scale dimension, with the parameter α of the function κ_s controlling the amount of smoothing applied. The size N^* of the scaling tensor ψ (and the resulting convolutional kernel h) is derived from the most stretched version of k:

$$N^* = \max_j s_j M. \tag{4}$$

2.1.4 Stacking scale-invariant layers

After the first scale-invariant layer, the tensor has an additional dimension representing scale. In order to add further scale invariant convolutional layers without losing scale invariance, subsequent operations are applied scale-wise:

$$y_j = x_j * \langle \psi_j, k \rangle. \tag{5}$$

The only difference with Eq. (2) is that the input tensor x of Eq. (5) already contains S scales, hence the added subscript j.

2.2 Tempo invariance

In the context of the downbeat tracking task, tempo behaves as a scale factor and the tempo-invariant patterns are rhythmic patterns. We construct the sequence of scale factors s as

$$s_j = \frac{r\tau_j B}{M}, \quad \tau_j = \tau_0 2^{\frac{j}{T}} \tag{6}$$

where τ_j are the beat periods, r is the frame rate of the input feature, B is the number of beats spanned by the convolution kernel factor k, τ_0 is the shortest beat period, and T is the desired number of tempo samples per octave. The matrix k has a simple interpretation as a set of rhythm fragments in musical time with M samples spanning B beats.
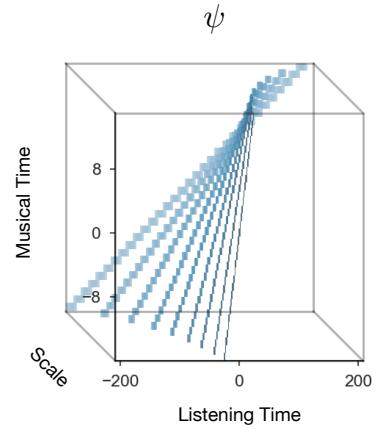
To mimic our perception of tempo, the scale factors in Eq. (6) are log-spaced, therefore the integral in Eq. (3) becomes:

$$\psi_{n,m,j} = \int_{\tilde{j}} \int_{\tilde{n}} \delta(\tilde{n} - s_{\tilde{j}} m) \kappa_n(n - \tilde{n}) \kappa_s(j - \tilde{j}) d\tilde{n} d\tilde{j}, \tag{7}$$
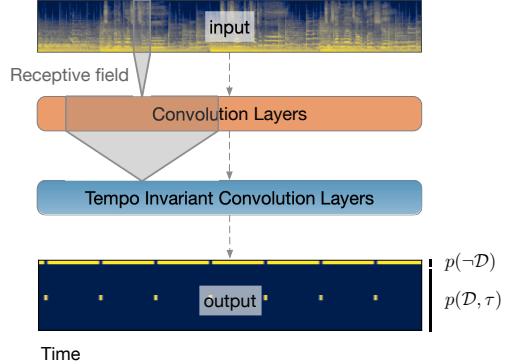
where the parameter α of the function κ_s has been set to 1. A representation of the scaling tensor used in the tempo-invariant convolution is shown in Figure 2.

2.3 Network

The tempo-invariant network (Fig. 3) is a fully convolutional deep neural network, where the layers are conceptually divided into two groups. The first group of layers are regular one-dimensional convolutional layers and act as onset detectors. The receptive field is constrained in order to preserve the tempo-invariance property of the model: if even short rhythmic fragments are learned at a specific tempo, the invariance assumption would be violated. We limit the maximum size of the receptive field to 0.25 seconds, i.e. the period of a beat at 240 BPM.

Figure 2. The scaling tensor ψ is a sparse 3-dimensional constant tensor. In the figure ψ is represented as a cube where the 0 bins are rendered transparent. ψ transforms the rhythm patterns contained in the kernel k from musical time (e.g. 16th notes) to listening time (e.g. frames) over multiple scales.

Figure 3. A global view of the neural network. The first group of layers are regular convolutional layers and act as onset detectors. They have a small receptive field, in order to focus on acoustic features and avoid learning rhythmic patterns, which will be learned by the successive tempo-invariant layers. The output tensor represents joint probabilities of downbeat presence $\mathcal{D}$ and tempo τ.

The second group is a stack of tempo-invariant convolutional layers (as described in Sec. 2.1, 2.2). The receptive field is measured in musical-time, with each layer spanning $B = 4$ beats. The last layer outputs only one channel, producing a 2-dimensional (frame and tempo) output tensor.

The activations of the last layer represent the scores (logits) of having a downbeat at a specific tempo. An additional constant zero bin [1] is concatenated to these activations for each frame to model the score of having no downbeat. After applying the softmax, the output o represents the joint probability of the downbeat presence $\mathcal{D}$ at a specific tempo τ

$$o_j = \begin{cases} p(\mathcal{D}, \tau_j) & j = 0, \ldots, S-1 \\ p(\neg\mathcal{D}) & j = S \end{cases} \tag{8}$$

---

[1] We can keep this constant because the other output values will adapt automatically.

The categorical cross-entropy loss is then applied frame-wise, with a weighting scheme that balances the loss contribution on downbeat versus non-downbeat frames. [2]

The target tensors are generated from the downbeat annotations by spreading the downbeat locations to the neighbouring time frames and tempi using a rectangular window (0.1 seconds wide) for time and a raised cosine window ($2/T$ octaves wide) for tempo. The network is trained with stochastic gradient descent using RMSprop, early stopping and learning rate reduction when the validation loss reaches a plateau.

2.4 Post-processing

In order to transform the output activations of the network into a sequence of downbeat locations, we use a frame-wise HMM with the state-space [26].

In its original form, this post-processing method uses a network activation that only encodes beat probability at each position. In the proposed tempo-invariant neural network the output activation models the joint probability of downbeat presence *and tempo*, enabling a more explicit connection to the post-processing HMM, via a slightly modified observation model:

$$
P(o_j | q) = \begin{cases} c(\tau_j, \tau_q) o_j & q \in \mathcal{D}, \quad j < S \\ o_S / (\sigma S) & q \in \neg \mathcal{D} \end{cases} \quad (9)
$$

where q is the state variable having tempo τ_q, $\mathcal{D}$ is the set of downbeat states, $c(\tau_j, \tau_q)$ is the interpolation coefficient from the tempi modeled by the network τ_j to the tempi modeled by the HMM τ_q and σ approximates the proportion of non-downbeat and downbeat states ($|\neg\mathcal{D}|/|\mathcal{D}|$).
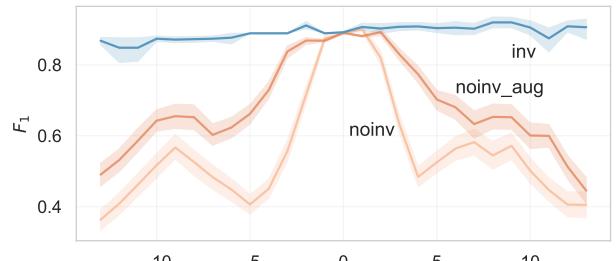
3. EXPERIMENTS

In this section we describe the two experiments conducted in order to test the tempo-invariance property of the proposed architecture with respect to a regular CNN. The first experiment, described in Sec. 3.1, uses a synthetic dataset of drum MIDI recordings. The second experiment, outlined in Sec. 3.2, evaluates the potential of the proposed algorithm on real music.
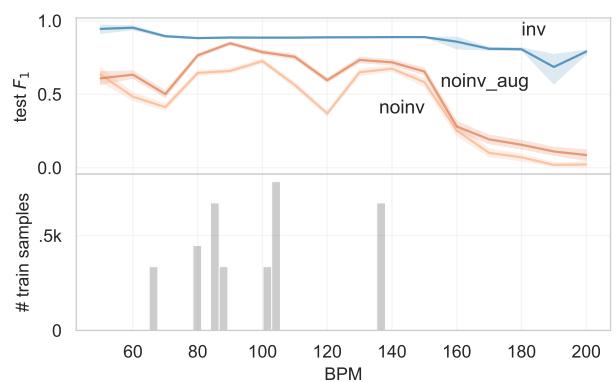
3.1 Tempo-invariance

We test the robustness of our model by training a regular CNN and a tempo-invariant CNN on a tempo-biased training dataset and evaluating on a tempo-unbiased test set. In order to control the tempo distribution of the dataset, we start with a set of MIDI drum patterns from the magenta-groove dataset [27], randomly selecting 4 bars from each of the 40 `eval-sessions`, resulting in 160 patterns. These rhythms were then synthesised at 27 scaled tempi, with scale factors $\varepsilon_i = 2^{i/26}$ ($-13 \leq i \leq 13$) with respect to the original tempo of the recording. Each track starts with a short silence, the duration of which is randomly chosen within a bar length, after which the rhythm is repeated

(a) accuracy with respect to relative tempo change

(b) accuracy with respect to absolute tempo

Figure 4. Tempo invariance experiment using a dataset of 27 time scaled versions of a set of drum patterns. The scale factors $\varepsilon_i = 2^{i/26}$ range from 0.707 to 1.414. A tempo-invariant CNN (`inv`) and a standard CNN (`noinv`) are trained on the non scaled versions (scale=0) and tested on all others. A standard CNN trained on scales $[-1, 1]$ (`noinv_aug`) simulates the effect of data augmentation. Figure (a) shows that the invariant model is able to generalise on seen patterns at unseen tempi. Figure (b) shows that the effect of the tempo-biased training set: for non-invariant models the benefit is localised, while the invariant model distributes the rhythmic information across the entire tempo spectrum.

4 times. Audio samples are rendered using FluidSynth [3] with a set of 40 combinations of SoundFonts [4] and instruments, resulting in 172800 audio files. The synthesised audio is pre-processed to obtain a log-amplitude mel-spectrogram with 64 frequency bins and $r = 50$ frames per second.

The tempo-biased training set contains the original tempi (scale factor: $\varepsilon_0 = 1$), while the tempo-unbiased test set contains all scaled versions. The two sets were rendered with different SoundFonts.

We compared a tempo-invariant architecture (`inv`) with a regular CNN (`noinv`). The hyper-parameter configurations are shown in Table 2 and were selected maximising the accuracy on the validation set.

The results of the experiment are shown in Fig. 4 in terms of F_1 score, using the standard distance threshold

group	architecture	
	inv	noinv
1	CNN 3×32	CNN 3×32
2	TI-CNN 2×16 1×1	dil-CNN 3×64 1×1
#params	60k	80k

Table 2. Architectures used in the experiment. Groups of layers are expressed as (number of layers × output channels). All layers in group 1 have kernel size equal to 3 frames. dil-CNN is a stack of dilated convolution layers with kernel size equal to 7 frames and exponentially increasing dilation factors: $2, 4, 8, 16$. The specific hyperparameters of the tempo-invariant network TI-CNN are configured as follows: $T = 8, \tau_0 = 0.25, S = 25, M = 64, B = 4$. ReLU non-linearities are used on both architectures.
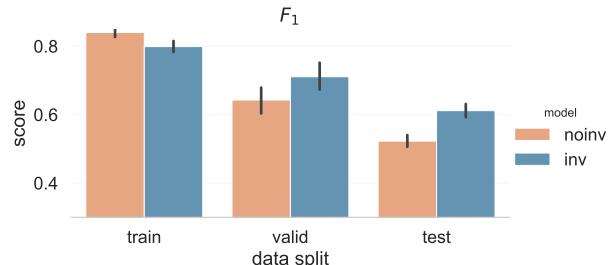
of 70 ms on both sides of the annotated downbeats [28]. Despite the tempo bias of the training set, the accuracy of the proposed tempo-invariant architecture is approximately constant across the tempo spectrum. Conversely, the non-invariant CNN performs better on the tempi that are present in the training and validation set. Specifically, Fig. 4a shows that the two architectures perform equally well on the training set containing the rhythms at their original tempo (scale equal to 0 in the figure), while the accuracy of the non-invariant network drops for the scaled versions. A different view of the same results on Fig. 4b highlights how the test set accuracy depends on the scaled tempo. The accuracy of the regular CNN peaks around the tempi that are present in the training set, showing that the contribution of the training samples is localised in tempo. The proposed architecture performs better (even at the tempi that are present in the training set) because it efficiently distributes the benefit of all training samples over all tempi.

In order to simulate the effect of data augmentation on the non-invariant model, we also trained an instance of the non-invariant model (`noinv_aug`) including two scaled versions (ε_i with $|i| \leq 1$) in the training set. As shown in the figure, data-augmentation improves generalisation, but has similar tempo dependency effects.

3.2 Music data

In this experiment we used real music recordings. We trained on an internal dataset (1368 excerpts from a variety of genres, summing up to 10 hours of music) and the RWC dataset [29] (Popular, Genre and Jazz subsets) and tested on Ballroom [30,31] and GTZAN [32] datasets. With respect to the previous experiment we used the same input features, but larger networks [5] because of the higher amount of information contained in fully arranged record-

---

[5] In terms of number of channels, layers and convolution kernel sizes, optimized on the validation set.

Figure 5. Results of the experiment on music data in terms of F-measure. Track scores are used to compute the average and the confidence intervals at 95% (using bootstrapping). The proposed tempo-invariant architecture is able to better generalise over unseen data with respect to its standard CNN counterpart.

ings, with `inv` having 170k trainable parameters and `noinv` 340k.

The results in Fig. 5 show that the proposed tempo-invariant architecture is performing worse on the training set, but better on the validation and test set, with the comparisons on train and test set being statistically significant ($p < 0.001$). Here the tempo-invariant architecture seems to act as a regularisation, allocating the network capacity to learning patterns that better generalise on unseen data, instead of fitting to the training set.

4. DISCUSSION

Since musicians are relentlessly creative, previously unseen rhythmic patterns keep being invented, much like "out-of-vocabulary" words in natural language processing [33]. As a result, the generalisation power of tempo-invariant approaches is likely to remain useful. Once tuned for optimal input representation and network capacity we expect tempo-invariant models to have an edge particularly on new, non-public test datasets.

Disentangling timbral pattern and tempo may also be useful to tasks such as auto-tagging: models can learn that some classes have a single precise tempo (e.g. ballroom dances [30]), some have varying tempos within a range (e.g. broader genres or moods), and others still are completely invariant to tempo (e.g. instrumentation).

5. CONCLUSIONS

We introduced a scale-invariant convolution layer and used it as the main component of our tempo-invariant neural network architecture for downbeat tracking. We experimented on drum grooves and real music data, showing that the proposed architecture generalises to unseen tempi by design and achieves higher accuracy with lower capacity compared to a standard CNN.

6. REFERENCES

[1] M. Thaut, P. Trimarchi, and L. Parsons, "Human brain basis of musical rhythm perception: common and dis-

tinct neural substrates for meter, tempo, and pattern," *Brain sciences*, vol. 4, no. 2, pp. 428–452, 2014.

[2] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner, "Towards a definition of disentangled representations," *arXiv preprint arXiv:1812.02230*, 2018.

[3] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, 1995.

[4] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for F0 estimation in polyphonic music," in *Proc. of the International Society for Music Information Retrieval Conference (IS-MIR)*, 2016, pp. 63–70.

[5] J. Schlüter and B. Lehner, "Zero-mean convolutions for level-invariant singing voice detection," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 321–326.

[6] M. E. Davies and M. D. Plumbley, "Beat tracking with a two state model [music applications]," in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 3. IEEE, 2005, pp. iii–241.

[7] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[8] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 342–355, 2005.

[9] N. Degara, E. A. Rúa, A. Pena, S. Torres-Guijarro, M. E. Davies, and M. D. Plumbley, "Reliability-informed beat tracking of musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 290–301, 2011.

[10] B. Di Giorgi, M. Zanoni, S. Böck, and A. Sarti, "Multipath beat tracking," *Journal of the Audio Engineering Society*, vol. 64, no. 7/8, pp. 493–502, 2016.

[11] H. Papadopoulos and G. Peeters, "Joint estimation of chords and downbeats from an audio signal," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 138–152, 2010.

[12] F. Krebs, S. Böck, M. Dorfer, and G. Widmer, "Downbeat tracking using beat synchronous features with recurrent neural networks," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 129–135.

[13] M. E. Davies and M. D. Plumbley, "A spectral difference approach to downbeat extraction in musical audio," in *2006 14th European Signal Processing Conference*. IEEE, 2006, pp. 1–4.

[14] S. Durand, J. P. Bello, B. David, and G. Richard, "Downbeat tracking with multiple features and deep neural networks," in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 409–413.

[15] ——, "Feature adapted convolutional neural networks for downbeat tracking," in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 296–300.

[16] M. Goto, "An audio-based real-time beat tracking system for music with or without drum-sounds," *Journal of New Music Research*, vol. 30, no. 2, pp. 159–171, 2001.

[17] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[18] D. Eck, "Beat tracking using an autocorrelation phase matrix," in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4. IEEE, 2007, pp. IV–1313.

[19] M. Goto, K. Yoshii, H. Fujihara, M. Mauch, and T. Nakano, "Songle: A web service for active music listening improved by user contributions," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 311–316.

[20] F. Krebs, A. Holzapfel, A. T. Cemgil, and G. Widmer, "Inferring metrical structure in music using particle filters," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 5, pp. 817–827, 2015.

[21] S. Böck and M. Schedl, "Enhanced beat tracking with context-aware neural networks," in *Proc. of the International Conference on Digital Audio Effects (DAFx)*, 2011, pp. 135–139.

[22] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 255–261.

[23] F. Korzeniowski, S. Böck, and G. Widmer, "Probabilistic extraction of beat positions from a beat activation function," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 513–518.

[24] Y. Xu, T. Xiao, J. Zhang, K. Yang, and Z. Zhang, "Scale-invariant convolutional neural networks," *arXiv preprint arXiv:1411.6369*, 2014.

[25] A. Kanazawa, A. Sharma, and D. Jacobs, "Locally scale-invariant convolutional neural networks," in *Deep Learning and Representation Learning Workshop, Neural Information Processing Systems (NIPS)*, 2014.

[26] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 72–78.

[27] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman, "Learning to groove with inverse sequence transformations," in *Proc. of the International Conference on Machine Learning (ICML)*, 2019.

[28] M. E. Davies, N. Degara, and M. D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," *Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06*, 2009.

[29] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical and jazz music databases," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, vol. 2, 2002, pp. 287–288.

[30] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.

[31] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 227–232.

[32] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[33] T. Schick and H. Schütze, "Learning semantic representations for novel words: Leveraging both form and context," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6965–6973.

A SIMPLE METHOD FOR USER-DRIVEN MUSIC THUMBNAILING

Arianne N. van Nieuwenhuijsen[1]   **John Ashley Burgoyne**[2]   **Frans Wiering**[1]   **Mick Sneekes**[1]

[1] Utrecht University, The Netherlands   [2] University of Amsterdam, The Netherlands

anvannieuwenhuijsen@gmail.com, j.a.burgoyne@uva.nl, f.wiering@uu.nl, me@micksneekes.nl

ABSTRACT

More and more music is becoming available digitally, increasing the need to navigate through large numbers of audio tracks easily. One approach for improving the browsing experience is *music thumbnailing*: the procedure of finding a continuous fragment that can represent the whole musical piece. This paper proposes a human-centred approach to creating thumbnails based on listeners' perception, directly asking listeners to identify the most characteristic fragment. We carried out a user study to assign representativeness scores to multiple fragments from a selection of popular music tracks. To strengthen the results, we performed a replication of the same user study with new participants and a different set of music. Thereafter, we used audio features, a segmentation algorithm, and participants' overall familiarity with the songs to predict representativeness scores. The results suggest that neither segmentation nor familiarity have a significant impact on users' thumbnail preferences: even segments with starting points that pay no regard to song structure can be suitable thumbnails. Three high-level audio characteristics, however, do impact the perceived representativeness of a fragment: Raw Intensity, Melodic Conventionality, and Conventionality of Intensity. Based on these findings, we propose a new, easy-to-apply method for music thumbnailing.

1. INTRODUCTION

With the rise of the digital age, more and more music is becoming available; streaming services and websites make music readily accessible to the public. The availability of so much music increases the need to navigate through large numbers of audio tracks easily, e.g., the results of search queries or long playlists. One approach to improve the browsing experience is to create music thumbnails. Music *thumbnailing*, or audio thumbnailing, is the procedure of finding a continuous segment within a musical piece which represents the whole piece [1–4]. By using these shorter fragments of audio, music thumbnails allow users to explore large quantities of music without spending too
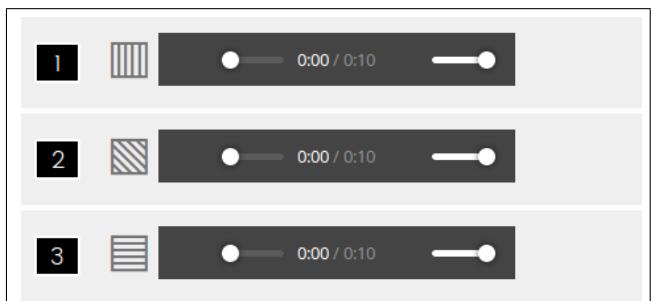
much time listening to or seeking within complete musical pieces [2, 5]. Audio thumbnailing should not be confused with music *summarisation*, which combines snippets of different parts of the song [2, 6], or audio *fingerprinting*, which creates a simpler representation of musical piece in the form of a vector or sequence [7, 8].

One practical example of music thumbnails even outside the major streaming services is Muziekweb, a Dutch music library that aims to make music and information about music available to everyone. [1] On their website, excerpts can be played to get a sense of the musical pieces on offer. To be able to assess the musical pieces, representative music thumbnails are a must. Currently, however, Muziekweb simply chooses its thumbnails randomly, which makes it likely that these excerpts do not represent the musical pieces very well.

There is no consensus about what approach works best to create good music thumbnails, and even the concept of music thumbnails is ambiguous [3–5, 9–11]. Approaches in previous studies include identification of the most repeated part [1, 4], finding the segment which is the most similar to the average sound [2], chorus detection [3–5, 9, 11], and structural identification of the "main part" [10]. Nonetheless, there is overlap between these approaches as the chorus is often the most repeated part in pop music [4] and is also likely to be the most memorable [3].

This paper proposes a user-driven approach by using the listeners' perception to improve upon Muziekweb's current thumbnailing method. Previous research has dis-

---

[1] https://www.muziekweb.nl/

Figure 1. Example of three playable audio fragments of the same tune as displayed in the user study. To be able to distinguish the fragments, the players are displayed with differently filled squares. The numbers show the ranking chosen by the participant.

cussed that the best thumbnail could be the segment containing the most memorable and distinguishable part of the musical piece [1, 3]. This aligns with the cognitive definition of *hooks*: hooks are the most salient segments in a musical piece, making them the most recognisable part of a song [12]. Suggestions have already been made about the potential of hooks and catchiness for music search engines [13]. Therefore, the method here is inspired by previous studies on catchiness to identify the most representative part as a music thumbnail.

To use listeners' perception for thumbnailing, we set up a user study to gain information on the representativeness of different fragments of the same tunes. The main task in this user study asked participants to rank three segments of the same song with respect to how well they conveyed a general idea of the song (see Figure 1). Thereafter, features were extracted from the audio fragments with the CATCHY toolbox [14]. To increase the interpretability of these features, we conducted an exploratory factor analysis for dimensionality reduction. These factors, the segmentation method, and the participant's familiarity with the songs were used to create an approximation of the scores from the user study with a linear model. Finally, we combined the feature loadings of the factors and the parameters of the linear model to create a function that can rate the relative representativeness of fragments within a song. The best-rated fragment in any set of candidates would be chosen as the audio thumbnail. To confirm our findings, we repeated the user study with new participants and a different music set and found a very similar result. Based on the findings, we propose a new user-driven method for music thumbnailing. Although we are certainly not the first researchers to test a thumbnailing algorithm on users, to our knowledge, this is the first published study to derive an algorithm for music thumbnailing algorithm directly from user preferences.

2. METHOD

2.1 Music Selection

Consistent with previous studies on catchiness [12, 14, 15], our study focused on popular music. The music came from lists of the 100 most-played songs on Muziekweb's website in 2017 and 2018, to ensure the data consisted of well-known music. Where the lists contained more than one song in languages other than English or Dutch (the two languages that would be most familiar to Muziekweb users), we retained only the most-played song. The resulting list was further reduced by removing songs with low play counts from artists or albums that appeared multiple times on the lists, in order to keep the music as diverse as possible. This resulted in a set of 60 songs, which Muziekweb provided to us as FLAC files. [2]

---

[2] The song list, segment start times, computed features, and analysis code can be found at `https://github.com/arianne-n/ISMIR-2020-User-Driven-Music-Thumbnailing`

2.2 Segmentation

Because hooks mostly occur at the start of structural sections [12, 13], we used a boundary detection algorithm to identify the start of these structural sections. Specifically, we used an algorithm that identifies boundaries based on structural features and time series similarity [16], as implemented in the Python package MSAF [17]. [3] We used Pitch Class Profiles (PCPs) as the underlying time series for segmentation, as the audio features we use to analyse the results are also mostly harmonic. The other harmonic time series available in MSAF were either too slow or resulted in too few boundaries to be feasible. Moreover, using PCPs aligns with previous thumbnailing studies that describe the importance of chroma [1, 11].

Thumbnails are by their nature short, and as such, our user study used only short excerpts from the original audio: 9.95 seconds, starting from one of the detected boundaries. Muziekweb is only allowed to make 29.9 seconds of music per song available on their site due to copyright, and excerpts of this length allow users to compare three segments from each song without causing copyright violations. Previous studies have assumed the middle of the song to be the most characteristic [3, 5], while others have noted the intro can also serve as a hook [18, 19]; given the conflicting opinions in the literature, we simply chose four segments at random among all the detected boundaries.

To check whether the segmentation method impacts the representativeness of fragments, we also created two extra baseline segments per song. The first is based on Muziekweb's current method: it picks any random point in a song as the start of the segment. The second baseline segment starts at the 1-minute mark in the song, skipping the intro, but staying away from the end. This resulted in six segments for each of the 60 songs.

2.3 User Study Design

The aim of the user study was to provide scores of the representative power for each of the six segments of the 60 songs. The study was carried out as a web-based survey, accessible between 3 April 2019 and 27 May 2019. Participants were recruited via social media and the Muziekweb website. Consent from the institutional ethics committee was acquired prior to collecting any data.

The main task in the survey was similar to the prediction task in the Hooked on Music study of catchiness [12], but rather than asking participants to make a binary choice, participants needed to provide an ordered ranking. Each question would display the title and artist of a song along with three audio fragments (see Figure 1). The participants were asked to rank the fragments on how well they helped them to get a sense of what the song is about (*"een idee van het nummer"*). This phrasing was intended to trigger participants to follow their gut feeling about the song, without thinking too much; asking for a ranking was intended to encourage participants to provide finer-grained distinctions than we might have obtained from a traditional rating

---

[3] `https://msaf.readthedocs.io/`

scale. The participants were also asked whether they were familiar with the song with a simple yes–no question.

The survey was implemented in the online survey platform Qualtrics. [4] Qualtrics offers a built-in drag-and-drop option whereby users can drag alternatives and place them in their preferred order. To help users distinguish among the different options visually, we gave each fragment one of six differently filled squares as a drag handle. These fill patterns have no apparent ordering in and of themselves, so as to avoid any bias during the ranking.

The survey started with a short explanation of the task, an informed consent form, and two practice songs. Then, the 60 songs were presented to each participant in random order, with a random selection of three of its segments also initially presented in random order. We elected for three segments instead of all six in order to keep the task manageable for participants. Participants were allowed to participate only once to reduce chances of bias. Participants were not required to complete all 60 songs and could end the survey whenever they wished.

2.4 Measures

Based on the data from the user study, we compute a representativeness score for each fragment. The data are in the form of partial rankings: for each song, we know each participant's relative ordering of three segments, but we have no information about their perception of the other three segments. The easiest way to model data in this form is to use a type of discrete-choice model known as the *Plackett–Luce* or *exploded logit model* [20]. We used the variant of the model implemented in the R package **PlackettLuce** [21]. The model is similar to a softmax function or a sort of logistic regression for rankings: specifically, it estimates the probability that, were participants given a choice among *all* six segments of a song, they would choose a particular segment as the most representative thumbnail. This probability is called the segment's *worth*.

In the user study, participants were also asked whether they were familiar with the songs they ranked. We convert these ratings to a continuous familiarity score by dividing the number of responses that indicated that a participant was familiar with the song by the number of responses where a participant was not familiar. As a continuity correction, we add one extra count to the numerator and to the denominator. Finally, we take the log of this ratio, and the standard score (z) of the result:

$$\text{familiarity} = z\left\{\log\frac{\text{known} + 1}{\text{unknown} + 1}\right\}. \tag{1}$$

2.5 Audio Features

We evaluate the core measures from the user study with the help of audio features from the CATCHY toolbox [14]. This toolbox can compute psychoacoustic features such as loudness, roughness, and sharpness as well as more common MIR features such as MFCCs, melodic pitch height estimates, and chroma based on HPCPs. Additionally,

---

[4] https://www.qualtrics.com

the CATCHY toolbox introduces three higher-dimensional harmonic and melodic features that attempt to bring some of the concepts available in symbolic music processing to audio. The first is the *Harmonic Interval Co-occurrence* (HIC), which describes the distribution of triads based on their interval representation. The *Melodic Interval Bigram* (MIB) indicates how often triples of successive melodic pitches occur in the melody. Lastly, the *Harmonic Interval* (HI) measures how often pitches in the melody are accompanied by harmonic pitches measured in the chroma.

The last feature of the toolbox is the implementation of *first-order* and *second-order* features for audio. First-order features are computed using the intrinsic content of the music or audio itself, such as the average note duration within the melody [14, 15, 22]. Second-order features reflect the characteristics of the music in context of a corpus. This means that corpus-based second-order features describe the *commonality* of a segment as it describes the segment in the context of the complete corpus. Song-based second-order features outline the *recurrence* of the segment within the song as it measures characteristics of a segment in relation to the whole song.

Like most MIR toolboxes, CATCHY creates a larger set of features than desirable for interpretability, and there is substantial overlap among some subsets of features. We conducted an Exploratory Factor Analysis (EFA) on all the features as a means of dimensionality reduction similar to [14]. Closely related to PCA, EFA looks for shared variance to identify a smaller underlying latent structure responsible for a larger set of observed features [23]. We used Spearman rank correlations instead of Pearson correlations as the basis for our EFA to avoid problems with the non-normality of some CATCHY features. Given a correlation matrix, several algorithms for EFA are in wide use; we chose the standard minimum residual method, which is commonly used for exploratory and descriptive analyses [24]. In order to maximise interpretability, we then rotated the latent factor space using Varimax, a common orthogonal rotation that pushes as many loadings as possible either toward 0 or toward the extremes (correlation of −1 or 1 with a latent factor) [23].

2.6 Regression

The last step is to combine the features to obtain insights into what contributes to the representativeness of segments. We model a segment's representativeness with a log-linear regression implemented as a generalised linear model (GLM) [25, 26]. In this case, the independent variables are the features derived from the audio, the familiarity score, and the segmentation method; the non-linear dependent variable is the Plackett–Luce worth. Although more complex models would be possible, given the applied nature of this research, we are aiming for simplicity as much as accuracy: linear relations among independent variables make the models both easy to interpret and easy to implement for non-experts. By using the resulting model to assign a worth to an unseen fragment, new fragments can be evaluated and the fragment of a song with the

highest value can thereafter be used as a music thumbnail.

2.7 Replication Study

As a final check, we ran a bilingual replication study with a new set of data from Muziekweb. The data set for this study consisted of an arbitrary list of 32 songs derived from the Dutch "Top 2000" of 2019. Although it is less directly connected to Muziekweb users, it should nonetheless also represent music that would be well known to its users. The only substantial difference in the replication was that we did not ask participants explicitly whether they were familiar with the songs. Results of the original study indicated that familiarity had no impact on how segments were perceived, and we hoped to encourage participants to rate more songs by reducing the number of extraneous questions. Ethical consent was also acquired for the replication study and the survey was available from 24 March 2020 until 30 April 2020.

3. RESULTS

3.1 Number of Responses

The original survey received 148 responses, of which 76 participants quit without completing more than the example questions, 14 participants completed the survey completely (i.e., all 60 songs), and 58 partially. The mean number of songs ranked per participant was 25 ($SD = 21$). This resulted in each segment being ranked by a mean of 15 participants ($SD = 3$). Segments in the replication study were ranked 17 times on average ($SD = 3$).

3.2 Dimensionality Reduction

To aid interpretability, we conducted an EFA based on all the CATCHY features computed for the two studies combined. As there is much disagreement in the literature about how to choose the optimal number of factors in EFA, we used a simple heuristic that each factor had to have at least three features with high loadings (correlation higher in magnitude than 0.4) to facilitate easier interpretation of factors [27]. This led to a maximum of five factors. Underfactoring is more harmful than overfactoring, and as four factors started to have more overlap between factors, we retained all five factors to improve identifiability.

Table 1 shows the CATCHY features which had loadings of magnitude greater than 0.4 for one of the five latent factors. To get a sense of what these factors are measuring, we consider the features with the highest loadings per factor.

3.2.1 Harmonic and Melodic Entropy

The first factor consists of second-order features describing harmony and melody. High absolute entropy is combined with high cross-entropy with respect to segments' own songs and with respect to the entire corpus of songs we considered. Sharpness also positively influences the factor, but does so with a far lower loading. This factor

Feature	Factors				
	1	2	3	4	5
HI Entropy	**0.90**	0.20	0.05	-0.01	0.05
MIB Entropy	**0.90**	0.35	-0.03	0.00	0.04
HI × Song Entropy	**0.89**	-0.29	0.13	-0.02	-0.01
MIB × Corpus Entropy	**0.88**	0.26	-0.05	0.00	0.04
MIB × Song Entropy	**0.88**	0.31	-0.05	-0.01	0.01
HI × Corpus Entropy	**0.87**	-0.33	0.13	-0.01	0.01
HIC Entropy	**0.86**	-0.29	0.09	0.02	0.02
HIC × Corpus Entropy	**0.85**	-0.31	0.14	0.00	0.01
HIC × Song Entropy	**0.85**	-0.28	0.14	-0.01	0.00
Sharpness	**0.46**	0.17	0.23	0.16	0.31
HI | Song	-0.33	**0.52**	0.01	0.15	0.06
HIC | Corpus	-0.20	**0.47**	0.19	0.17	0.13
HI | Song	-0.29	**0.47**	0.00	0.18	0.13
MIB | Corpus	0.11	**0.45**	0.01	0.25	0.09
HIC | Song	-0.21	**0.41**	0.15	0.16	0.15
Loudness	0.08	-0.01	**0.92**	0.00	-0.03
Roughness	0.30	0.01	**0.82**	0.07	0.05
Melodic Pitch Height	0.12	-0.08	**0.50**	-0.02	-0.10
MFCC Variance	0.21	-0.13	**-0.49**	0.02	0.00
MFCC Mean | Corpus	0.16	0.21	**0.45**	0.14	0.25
Loudness SD	0.32	-0.07	**0.43**	0.10	0.07
MIB Entropy | Corpus	-0.02	0.10	0.04	**0.79**	-0.02
HI Entropy | Corpus	-0.03	0.09	0.04	**0.77**	0.03
HI Entropy | Song	-0.01	-0.04	0.03	**0.55**	0.15
MIB Entropy | Song	-0.03	-0.04	0.01	**0.53**	0.08
Loudness | Corpus	0.10	0.09	-0.25	0.02	**0.55**
Loudness | Song	0.08	0.08	-0.05	0.08	**0.50**
Roughness | Song	0.08	0.08	0.28	0.06	**0.50**
Roughness | Corpus	0.13	0.05	0.41	0.02	**0.42**

Table 1. Factor loadings for Minimum Residual EFA for the features with loadings above 0.4 for one of the factors. The factors group features together that explain the same variance. 1. Harmonic and Melodic Entropy; 2. Harmonic Conventionality; 3. Raw Intensity; 4. Melodic Conventionality; 5. Conventionality of Intensity.

thus describes unpredictability or lack of motivic repetition in the harmony and melody; we call it Harmonic and Melodic Entropy.

3.2.2 Harmonic Conventionality

The second factor also consists of second-order features for harmony and melody. For this factor, however, the loadings prefer higher values for the commonality and recurrence of these features rather than entropy calculations. Note that the commonality of HIC and HI is of high importance both with within songs and across the entire corpus, whereas the melody-based MIB loads only against the full corpus. There is, of course, a high correlation between melody and harmony, and so we call this factor Harmonic Conventionality, while acknowledging that it may also have some melodic aspects. This factor can indicate repetition within a song itself, as well as tonal language that does not stray too far from our corpus norm.

3.2.3 Raw Intensity

The third factor mostly relies on high positive values for loudness (mean and standard deviation) and roughness. It also prefers a lower MFCC variance, which means a fragment is more consistent, a high MFCC mean in comparison to the complete corpus, which could be caused due to

Feature	Overall Result			Original – Replication		
	b	SE	p	b	SE	p
Intercept	0.15	0.06	0.006	−0.22	0.11	0.044
Audio Factors						
Harmonic and Melodic Entropy	0.03	0.04	0.448	−0.32	0.09	<0.001
Harmonic Conventionality	0.11	0.05	0.014	0.01	0.09	0.912
Raw Intensity	0.20	0.04	<0.001	−0.12	0.09	0.149
Melodic Conventionality	0.19	0.04	<0.001	0.19	0.09	0.036
Conventionality of Intensity	0.27	0.05	<0.001	0.16	0.09	0.082
Segmentation Strategies						
MSAF	−0.21	0.06	<0.001	0.14	0.13	0.249
Random	0.13	0.11	0.238	0.06	0.15	0.585
1-minute	0.10	0.08	0.205	−0.43	0.16	0.009

Table 2. GLM results showing how features contribute to perceived representativeness of thumbnails ($R^2 = 0.09$). The left-most part shows estimates using the data from both the original and replication study. The right-most results shows the differences in estimates between the two studies. For each of these results, the estimate or coefficient (b), the standard error (SE), and p-value are given.

MFCCs also measuring loudness, and a high melodic pitch height. We call this factor the Raw Intensity of a fragment, as fragments that score high on this factor sound noticeably more "aggressive" than those that do not.

3.2.4 Melodic Conventionality

The fourth factor is heavily based on corpus as well as song-based second-order features for the MIB and HI entropy. This means that the values for this factor rise when the dispersion of MIB and HI is typical for a song or the corpus. Remember also that HI is a measure that explicitly incorporates melodic information. Thus, this factor primarily describes the commonality and recurrence of the dispersion of melodic bigrams and the melody aligning with the harmony. We call it Melodic Conventionality, although it is somewhat less directly linked to conventionality than the Harmonic Conventionality factor.

3.2.5 Conventionality of Intensity

The last factor comprises corpus- and song-based second-order features for the most important components of Raw Intensity. It is easy to understand but hard to name; we call it Conventionality of Intensity.

3.3 Log-Linear Model

A GLM based solely on the data of the original user study showed that familiarity had no significant impact on how participants ranked the segments ($b = -0.08$, $SE = 0.07$, $p = .27$). As mentioned above, we excluded familiarity from the replication study in order to lessen the burden on our participants. We also exclude it from further analysis.

Table 2 showcases the results of a larger GLM with both the original and replication studies combined ($R^2 = .09$). The left part shows how each variable contributes to the representative worth of a fragment overall. The right side shows the differences in these parameter estimates between the original and replication studies. The results indicate that the Raw Intensity, Melodic Conventionality, and Conventionality of Intensity are the most important factors to approximate a segment's worth, each having a positive effect. Although there is a statistically significant difference between the two studies with respect to the *size* of the effect of Melodic Conventionality, the *direction* of the effect is the same in both studies. The role of Harmonic and Melodic Entropy is less clear: its effect on worth goes in opposite directions between the original study and the replication. Harmonic Conventionality has a small positive effect in each study. The effects of segmentation are less consistent (there is a significant Segmentation × Experiment interaction, $\chi^2(2) = 6.80$, $p = .03$) but with one surprising finding: in both studies, choosing thumbnails that line up with (estimated) structural boundaries seems to make users' opinions *worse*.

4. DISCUSSION

The results show that the most significant features that could contribute to the representative worth of a fragment are the Raw Intensity, Melodic Conventionality, and Conventionality of Intensity. Conventionality of Intensity has the highest impact on the representative worth: users prefer typical levels of intensity, neither too "hard" nor too "soft", for thumbnails. In addition to Conventionality of Intensity, higher-intensity thumbnails are preferred, as well as thumbnails with typical, familiar melodic patterns. The effect of Harmonic Conventionality is statistically significant, but its effect size is quite small; if anything, it may have a small positive effect on the perceived quality of a thumbnail.

Our results also show that the effect of Harmonic and Melodic Entropy seems to differ between the original and replication study. As both data sets had the same data format and were used to create the factors, the difference is most likely caused by the songs themselves. The replica-

tion study contained primarily pop-rock songs, whereas the original study also contained a broader of popular styles, e.g., rap and trance. Harmonic and melodic entropy are fundamental and sometimes genre-defining musical characteristics, and as such, it is not surprising that the effect of this factor would differ. This possibly genre-dependent aspect of thumbnails could be an interesting area for future work.

The impact of the segmentation method on the representative worth shows that in contrast to our hypothesis, segments chosen by a segmentation method do not outperform the base cases: in fact, boundary-aligned thumbnails seem to perform worse. While a thumbnail may benefit from containing the most memorable and recognisable part of a song, it does not necessarily need to start at that point. In practice, an algorithm for selecting thumbnails is going to be more successful if it simply has many candidate thumbnails to choose from, without worrying about where they start.

Altogether, users most prefer music thumbnails with high intensity and conventional, frequently recurring intensities and melodic patterns. This aligns with previous automatic thumbnailing studies, which have mostly focused on detecting the most repeated section or chorus [1,3–5,9,11]. Moreover, previous research shows that the chorus is generally louder, has a higher and more salient pitch, and has less dynamic diversity [28], which overlaps with the factor for Raw Intensity in this study.

A similarity can also be found with research on catchiness, which shows that the most memorable parts of a song have a more typical sound, more conventional melodies, more recurrence in the timbral aspects, as well as a prominent vocal line [14]. Earworms, which are related to catchiness, also seem to appear more in often recurring fragments with a faster tempo and a common melodic contour [22, 29]. In short, our findings about listeners' thumbnail preferences are consistent with previous studies on thumbnails, choruses, and catchiness.

4.1 Proposed Thumbnailing Method

Based on these results, we propose a new method for music thumbnailing. First, several fragments should be obtained from the song. The results of this study show that there is no preferred segmentation method and therefore that any method that results in a reasonable amount of fragments suffices. Then, the CATCHY features for each of these fragments need to be computed. An approximation of the factors in this study can be computed by multiplying standardised feature values by the highest factor loadings for the Raw Intensity, Melodic Conventionality, and Intensity Conventionality. Thereafter, these approximations are multiplied by the estimates of the GLM of the combined results to gain a representative score. The fragment of a song with the highest score can be selected as the music thumbnail.

4.2 Limitations

Like any user study, our research has some limitations. First, this study only focuses on pop music; the results cannot necessarily be transferred to other musical genres [11]. Apart from the musical genre, the choice of a linear model might also have been too simplistic to grasp fully how audio features are related to perceived representativeness. More insights might be gained by also considering nonlinear models that could pick up more intricate relationships. While this study does consider features for psychoacoustics and harmony, rhythm is not considered. Further research might look into the effects of rhythm features on representativeness. Lastly, the segmentation method used here had a negative impact on the representativeness score; perhaps a different algorithm might have yielded better results. Nonetheless, it is clear from our findings that simple heuristics like starting at a fixed time point or even a fully random starting point can also yield effective thumbnails.

5. CONCLUSION

This study aimed to create a user-driven music thumbnailing method based on easily computable audio features and an easy-to-implement scoring strategy. Segments of well-known pop songs were obtained and audio features of these segments were derived with the CATCHY toolbox. Thereafter, the segments were presented in two user studies where participants could rank segments on their representativeness. Using the data from the user studies, we used a log-linear model to understand how audio features might explain the perceived worth of a potential thumbnail. The results were significant: representativeness seems to be positively influenced by a higher intensity, and a higher commonality and recurrence of intensity and melodic dispersion. Based on these findings, we propose a new, easy-to-apply method for music thumbnailing.

6. ACKNOWLEDGEMENT

7. REFERENCES

[1] W. Chai and B. Vercoe, "Music thumbnailing via structural analysis," in *Proceedings of the 11th Association for Computing Machinery (ACM) International Conference on Multimedia*, 2003, pp. 223–226, https://doi.org/10.1145/957013.957057.

[2] M. L. Cooper and J. Foote, "Automatic music summarization via similarity analysis," in *Proceedings of the 3rd International Society of Music Information Retrieval Conference (ISMIR)*, Paris, France, 2002.

[3] Y.-S. Huang, S.-Y. Chou, and Y.-H. Yang, "Music thumbnailing via neural attention modeling of music emotion," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2017, pp. 347–350, https://doi.org/10.1109/apsipa.2017.8282049.

[4] M. Müller, N. Jiang, and P. Grosche, "A robust fitness measure for capturing repetitions in music recordings with applications to audio thumbnailing," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 3, pp. 531–543, 2013, https://doi.org/10.1109/tasl.2012.2227732.

[5] M. Levy and M. Sandler, "Application of segmentation and thumbnailing to music browsing and searching," in *Proceedings of the Audio Engineering Society Convention 120*, 2006.

[6] D. F. Silva, F. V. Falcao, and N. Andrade, "Summarizing and comparing music data and its application on cover song identification," in *Proceedings of the 19th International Society of Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[7] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A review of audio fingerprinting," *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 41, no. 3, pp. 271–284, 2005.

[8] J. Van Balen, F. Wiering, and R. Veltkamp, "Audio bi-grams as a unifying model of pitch-based song description," in *Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, Plymouth, United Kingdom, 2015.

[9] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 96–104, 2005, https://doi.org/10.1109/tmm.2004.840597.

[10] H. Nawata, N. Kamado, H. Saruwatari, and K. Shikano, "Automatic musical thumbnailing based on audio object localization and its evaluation," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, https://doi.org/10.1109/icassp.2011.5946323.

[11] B. Schuller, F. Dibiasi, F. Eyben, and G. Rigoll, "Music thumbnailing incorporating harmony- and rhythm structure," in *International Workshop on Adaptive Multimedia Retrieval*, 2008, pp. 78–88.

[12] J. A. Burgoyne, D. Bountouridis, J. Van Balen, and H. Honing, "Hooked: A game for discovering what makes music catchy," in *Proceedings of the 14th International Society of Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, 2013, pp. 245–250.

[13] H. J. Honing, "Lure(d) into listening: The potential of cognition-based music information retrieval," *Empirical Musicology Review*, vol. 5, no. 4, pp. 146–151, 2010, https://doi.org/10.18061/1811/48549.

[14] J. Van Balen, J. A. Burgoyne, D. Bountouridis, D. Müllensiefen, and R. C. Veltkamp, "Corpus analysis tools for computational hook discovery," in *Proceedings of the 16th International Society on Music Information Retrieval (ISMIR)*, Malaga, Spain, 2015, pp. 227–233.

[15] D. Müllensiefen and A. R. Halpern, "The role of features and context in recognition of novel melodies," *Music Perception: An Interdisciplinary Journal*, vol. 31, no. 5, pp. 418–435, 2014, https://doi.org/10.1525/mp.2014.31.5.418.

[16] J. Serra, M. Müller, P. Grosche, and J. Ll. Arcos, "Unsupervised music structure annotation by time series structure features and segment similarity," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1229–1240, 2014, https://doi.org/10.1109/tmm.2014.2310701.

[17] O. Nieto and J. P. Bello, "MSAF: Music structure analysis framework," in *Proceedings of the 16th International Society on Music Information Retrieval (ISMIR)*, 10 2015.

[18] G. Burns, "A typology of 'hooks' in popular records," *Popular music*, vol. 6, no. 1, pp. 1–20, 1987, https://doi.org/10.1017/s0261143000006577.

[19] C. Kronengold, "Accidents, hooks and theory," *Popular Music*, vol. 24, no. 3, pp. 381–397, 2005.

[20] S. Beggs, S. Cardell, and J. Hausman, "Assessing the potential demand for electric cars," *Journal of Econometrics*, vol. 17, no. 1, pp. 1–19, 1981, https://doi.org/10.1016/0304-4076(81)90056-7.

[21] H. L. Turner, J. van Etten, D. Firth, and I. Kosmidis, "Modelling rankings in R: The PlackettLuce package," *arXiv preprint arXiv:1810.12068*, 2018.

[22] K. Jakubowski, S. Finkel, L. Stewart, and D. Müllensiefen, "Dissecting an earworm: Melodic features and song popularity predict involuntary musical imagery," *Psychology of Aesthetics, Creativity, and the Arts*, vol. 11, no. 2, pp. 122–135, 2017, https://doi.org/10.1037/aca0000090.

[23] J. W. Osborne, A. B. Costello, and J. T. Kellow, "Best practices in exploratory factor analysis," *Best Practices in Quantitative Methods*, pp. 86–99, 2008, https://doi.org/10.4135/9781412995627.d8.

[24] H. E. Tinsley and D. J. Tinsley, "Uses of factor analysis in counseling psychology research," *Journal of Counseling Psychology*, vol. 34, no. 4, pp. 414–424, 1987, https://doi.org/10.1037/0022-0167.34.4.414.

[25] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[26] J. A. Nelder and R. W. Wedderburn, "Generalized linear models," *Journal of the Royal Statistical Society: Series A (General)*, vol. 135, no. 3, pp. 370–384, 1972.

[27] T. A. Brown, *Confirmatory Factor Analysis for Applied Research*, 2nd ed.　New York: Guilford, 2015.

[28] J. Van Balen, J. A. Burgoyne, F. Wiering, and R. C. Veltkamp, "An analysis of chorus features in popular song," in *Proceedings of the 14th International Society of Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, 2013.

[29] V. J. Williamson, S. R. Jilka, J. Fry, S. Finkel, D. Müllensiefen, and L. Stewart, "How do 'earworms' start? classifying the everyday circumstances of involuntary musical imagery," *Psychology of Music*, vol. 40, no. 3, pp. 259–284, 2012, https://doi.org/10.1177/0305735611418553.

SCORE-INFORMED SOURCE SEPARATION OF CHORAL MUSIC

Matan Gover
Schulich School of Music, McGill University
`matan.gover@mail.mcgill.ca`

Philippe Depalle
Schulich School of Music, McGill University
`philippe.depalle@mcgill.ca`

ABSTRACT

Choral music recordings are a particularly challenging target for source separation due to the choral blend and the inherent acoustical complexity of the 'choral timbre'. Due to the scarcity of publicly available multi-track choir recordings, we create a dataset of synthesized Bach chorales. We apply data augmentation to alter the chorales so that they more faithfully represent music from a broader range of choral genres. For separation we employ Wave-U-Net, a time-domain convolutional neural network (CNN) originally proposed for vocals and accompaniment separation. We show that Wave-U-Net outperforms a baseline implemented using score-informed NMF (non-negative matrix factorization). We introduce *score-informed Wave-U-Net* to incorporate the musical score into the separation process. We experiment with different score conditioning methods and show that conditioning on the score leads to improved separation results. We propose a 'score-guided' model variant in which separation is guided by the score alone, bypassing the need to specify the identity of the extracted source. Finally, we evaluate our models (trained on synthetic data only) on real choir recordings and find that in the absence of a large training set of real recordings, NMF still performs better than Wave-U-Net in this setting. To our knowledge, this paper is the first to study source separation of choral music.

1. INTRODUCTION

In this paper, we set out to investigate the application of source separation to choral music. We aim to take a recording of choral music and extract from it individual recordings for each of the choir sections (normally soprano, alto, tenor, and bass).

Audio source separation refers to extracting one or more sound sources of interest from a recording that involves multiple sound sources [1]. The musical applications of audio source separation include separating instruments in a recording and generating 'karaoke' tracks of songs by separating the accompaniment and the lead vocals [2]. To the best of our knowledge, this paper is the first to attempt separation of choral music. Separation of choral music en-

ables applications such as fine-grained editing, analysis, and automatic creation of practice tracks (recordings of individual choir parts used by singers as an aid for learning new music) from professional choir recordings.

At the outset, choral music separation would seem a challenging task. Every choir section is composed of multiple singers singing simultaneously with slight variations in pitch and in timing, and every singer has a unique voice timbre. It follows that the resulting 'choral timbre' has extremely varied acoustical characteristics. Furthermore, an important goal in choral performance is achieving *blend* between singers, so that the choir is perceived by listeners as one coherent sound source [3]. This blend can naturally hinder the operation of an algorithm wishing to separate the choir. Choral music is often recorded in highly reverberant spaces such as churches, and the reverberations constitute yet another hurdle for separation. Finally, choirs are seldom recorded in a 'one voice per track' setting [4], and this lack of multi-track recordings makes it harder to design and validate source separation systems.

The rest of this paper is structured as follows. In Section 2 we review related work. In Section 3, we present a dataset of synthesized Bach chorale harmonizations. In Section 4, we establish baseline separation performance for choral music using NMF [5]. In Section 5, we apply a deep learning separation technique called Wave-U-Net [6] to choral music and in Section 6 we extend it to incorporate musical scores into the separation process. In Section 7, we present the results of several experiments conducted to determine the effectiveness of the proposed techniques.

2. RELATED WORK

Recently, the state of the art in source separation has advanced considerably, with some applications in speech even surpassing ideal time-frequency magnitude masking [7]. In music, one of the most common applications is vocals and accompaniment separation [8]. In this task, deep learning methods show the best performance among separation techniques [9]. Some state-of-the-art techniques operate on spectrograms [10, 11] while others operate directly on the signal [6, 7, 12–14]. The reader is referred to [15] for a review of deep learning for speech separation and to [2, 8] for overviews of music separation.

U-Net [16] is a prominent deep learning separation technique. Originally used for semantic segmentation of biomedical images [17], U-Net employs an encoder-decoder CNN architecture with skip connections to process the input on multiple scales. Wave-U-Net [6] extends

U-Net but instead of processing spectrograms it is applied directly to the signal. Demucs [12] is also based on a U-Net architecture and operates on the time-domain signal, with added features such as gated linear units and a recurrent layer between the encoder and the decoder.

2.1 Score-Informed Source Separation

The musical score, when available, is an invaluable source of detailed information on the mixture, such as instrumentation, pitch, and timing. Score-informed separation techniques use this information to guide the separation process [18]. One of the earliest techniques is synthesizing a signal from the target source's score and then using this signal as a reference [19–21]. Another technique is creating harmonicity-based masks or constraints driven by the note pitches and timings specified in the score [22–24]. Scores have also been used extensively as factorization constraints in the framework of NMF and its extensions [25–30].

More recently, scores have also been integrated into deep learning-based separation techniques. In [31], an autoencoder network was trained while imposing score-based constraints on the latent representation so that each latent unit represents a single note. Separation was then performed on a note-by-note basis. A technique for orchestral music separation [32] used a CNN that operates on 'score-filtered' spectrograms.

3. SYNTHESIZED BACH CHORALES DATASET

For training source separation techniques based on supervised learning, a large dataset of multi-track recordings is required. For example, the MUSDB18 dataset [9] for vocals and accompaniment separation contains 150 songs with a total duration of about 10 hours. Unfortunately, such a dataset of choir recordings does not currently exist. The Mixing Secrets dataset [1] contains some multi-microphone choral recordings, but there is significant leakage between the microphones. Choral Singing Dataset [33] is a good multi-track dataset, but it consists of only three songs.

In the absence of a large choral music dataset, we opt to use a synthesized dataset. Recently, a method for choir synthesis was proposed based on voice cloning [34], but unfortunately the implementation and the dataset are not publicly available. Choir audio tracks are often produced using commercial sample libraries [2] that contain thousands of professionally recorded choir samples. Unfortunately, these sample libraries are prohibitively expensive.

Previous work has shown that synthetic training data does not have to sound realistic for a model to generalize well [35, 36]. In light of this, we choose a relatively simple and cheap approach. We use the FluidSynth software synthesizer [37], which converts MIDI messages to audio by using audio samples and synthesis rules stored in a SoundFont file. We use the 'Choir Aahs' preset from the `MuseScore_General` SoundFont [3]. Each sample

in this preset is a short recording of a single choir section singing a sustained note on an 'aah' vowel with a single pitch. To synthesize a pitch that does not have an associated sample, FluidSynth pitch-shifts the sample that has the closest pitch. To synthesize a note that is longer than the corresponding sample, a predefined segment of the sample is looped.

3.1 Bach Chorale Harmonizations

We construct our dataset from a well-known corpus of chorale harmonizations by J. S. Bach. A chorale is a Lutheran church hymn [38]. Bach harmonized around 400 chorales as part of large-scale vocal compositions as well as shorter works [39]. Bach's chorales are highly structured and this makes them good candidates to serve as a coherent dataset for source separation. They are written for four voices in homorhythmic texture [39]. The rhythm consists mainly of quarter notes and eighth notes. Structurally, the chorales are built as a sequence of short phrases, each ending with a fermata (musical pause).

3.2 Data Augmentation

Real-world choir recordings possess many sources of variability that are absent from Bach chorales. In order to make our dataset more closely resemble real-world recordings, we augment it with three added features: simulated breaths, random omitted notes, and tempo variations.

To simulate breaths between phrases, we insert a one-beat-long rest in all voices simultaneously every eight beats. To simulate sections in which one or more voices are silent while the other voices continue to sing, we randomly choose 10% of the notes in each voice and change them into rests. To add tempo variation, we synthesize each chorale at a random tempo between 70 and 100 BPM.

3.3 Synthesis Procedure

To synthesize our dataset we read the corpus of Bach chorales in MusicXML format using the music21 library [40]. From the 371 chorales in the Riemenschneider edition we exclude 20 chorales that contain instrumental parts or more than four vocal parts. The 351 remaining chorales are split into three partitions: training (270 chorales), validation (50), and test (31). For each chorale we export four MIDI files, one file per voice, and synthesize them using FluidSynth. [4] The total duration of the dataset is 3h 48m.

4. BASELINE: SCORE-INFORMED NMF

We establish a baseline for separation performance on our dataset using a classic separation technique: non-negative matrix factorization (NMF) [5, 41]. NMF factorizes a mixture spectrogram into two matrices: basis signals and temporal activations. To constrain the NMF separation process we use a score-based initialization scheme for the basis

---

signals and activations matrices [25]. We dub this technique SI-NMF. Our implementation is available online.[5] We use the variant dubbed I_{WH} in the original paper, which imposes constraints on both basis signals and activations. For the STFT we use a Hann window with a size of 2,048 samples (the dataset sampling rate is 22,050 Hz). The SI-NMF score-based constraints allow some tolerance to account for slight pitch and timing variations in the mixture. Since in our dataset the scores are perfectly aligned to the mixture, we use onset tolerance of 0 and offset tolerance of 0.2 seconds (to account for note decay). We use pitch tolerance of 1 semitone. These parameters were found to give the best results after comparing several alternatives.

5. WAVE-U-NET FOR CHORAL MUSIC

To improve on the SI-NMF baseline, we propose to apply Wave-U-Net (described in Section 2). Wave-U-Net attained good results in the SiSEC 2018 evaluation campaign [9] and its code is publicly available. Since Wave-U-Net operates in the time domain, it may be well suited for separating sources with overlapping partials, which are ubiquitous in choral music and may pose a challenge for methods that rely on spectrogram masking [12].

We follow the training procedure used in the original Wave-U-Net paper. Every training batch consists of 16 short (6-second) segments extracted from the training set at random positions. The Adam optimizer [42] is used with the mean squared error loss and an initial learning rate of 0.0001. The validation set is used for early stopping.

In the original implementation of Wave-U-Net, a single model is trained to extract all sources at the same time. This is economical in terms of model weights and training time, but it forces the latent representations to be generic enough to fit all sources. Instead, we propose to train a separate model for each extracted source. This way the model can be specifically geared to extract each of the sources.

6. SCORE-INFORMED WAVE-U-NET

We propose to condition Wave-U-Net on the musical score of the separated sources to improve separation quality.[6] The pitch and timing information contained in the score can help overcome the challenges of separating choral music. Timbre is generally a useful differentiating factor for separation, but the timbres of the women's voices (soprano and alto) are similar to each other, and so are the men's (tenor and bass). Relying on the pitch range of each choir part is also not sufficient for separation because the ranges have considerable overlap. For example, an F4 note (F above middle C) could easily be sung by the soprano, alto, or tenor, and in rare cases also by the bass [3, p. 234]. The standard SATB (soprano-alto-tenor-bass) ordering of the voices could sometimes be used for separation, but this ordering is not always kept, and in any case it could only be used in sections where all voices sing at the same time.

Hence, in many cases the musical score may be the only way to associate notes to a specific voice in choral music.

6.1 Score Representations

Our dataset provides the score for each part as a monophonic MIDI file indicating each note's onset time, offset time, and pitch. We transform the MIDI note sequence into a representation that can be efficiently processed by Wave-U-Net. In choral music, every part sings at most one note at a time. (In the case of *divisi*, such as when soprano is split into soprano 1 and soprano 2, we can treat every divisi section as a distinct source.) Therefore, we represent a part's score as a time series that indicates the active pitch (if any) at every time point. To keep the score aligned with the network's audio input, we use the same sampling rate for the audio and the score representation. We investigate four different score representations [43].

normalized pitch. A part's score is represented as a vector in which every element indicates the active pitch at the corresponding time instant. Since the range of MIDI note numbers (0 to 127) is radically different from the range of the audio input (-1 to 1), we normalize the note number to the range $[0, 1]$, and use the special value -1 to indicate no note is active. Given a MIDI note number M, the normalized pitch S_{n} is computed as:

$$S_{\text{n}}(M) = \frac{M - M_{\text{min}}}{M_{\text{max}} - M_{\text{min}}}, \qquad (1)$$

where M_{min} and M_{max} are the minimum and maximum expected note pitches, respectively. We set $M_{\text{min}} = 36$ and $M_{\text{max}} = 84$, based on the normal choral voice ranges: from C2 (very low bass note) to C6 (very high soprano note).

pitch and amplitude. In order to better encode the difference between sung notes and silence, we introduce a two-channel representation, in which one channel represents pitch and the other represents amplitude. The pitch channel S_{p} is normalized to the range $[-1, 1]$, as given by: $S_{\text{p}}(M) = 2S_{\text{n}}(M) - 1$. The amplitude channel is boolean: its value is 1 when a note is active and 0 otherwise. When no note is active the pitch channel is set to -1.

piano roll. The score is represented as a one-hot matrix of size $p \times n$ where p is the number of available pitches ($p = M_{\text{max}} - M_{\text{min}} + 1$) and n is the length of the network's audio input. The matrix element at row p_i and column n_j is set to 1 if a note with pitch p_i is active at time n_j. Otherwise, the element is set to 0.
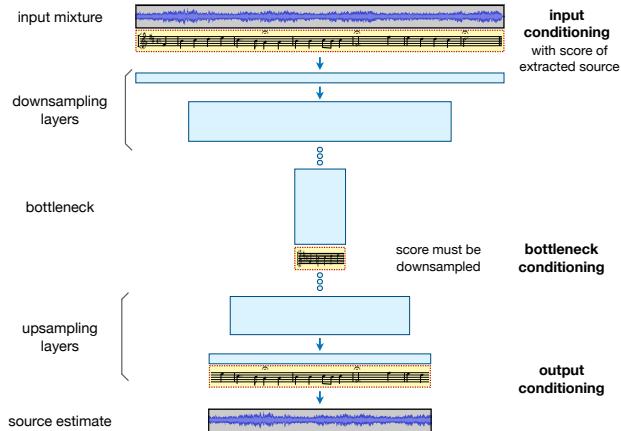
pure tone. Since the model inputs are audio, we propose to represent the score in a simplistic audio-like form. We use a pure tone signal constructed as a piecewise sine function where the frequency is controlled by the active note's pitch. For simplicity, we do not create smooth note transitions, so any note onset will result in a discontinuity. The pure tone frequency f is determined by the standard MIDI note number to frequency mapping:

$$f(M) = 440 \cdot 2^{\frac{M-69}{12}}. \qquad (2)$$

When there is no active note, f is set to 0. The score vector

---

[5] https://git.io/si-nmf
[6] https://git.io/si-Wave-U-Net

Figure 1. Conditioning locations for Wave-U-Net (showing a model that extracts a single source, see Section 5)

then receives the following value at each sample index i:

$$S_\text{t}(M, i) = \sin\left(2\pi f(M) * \frac{i}{F_s}\right), \qquad (3)$$

where F_s is the sample rate of the model's audio input.

All four score representations do not differentiate between a sustained note and a repeated note: consecutive notes with the same pitch are represented the same as one note with a longer duration. Devising a score representation that does encode this difference is left to future work.

6.2 Conditioning Method

Common methods for conditioning neural networks include *concatenation*, in which a conditioning tensor is concatenated to the input tensor; *biasing*, in which a conditioning tensor is added to the input tensor; and *scaling*, in which the input tensor is multiplied element-wise by a conditioning tensor [44]. In this work we use concatenation, which is equivalent to biasing with a linear transformation applied to the conditioning [44].

We investigate three conditioning locations in the Wave-U-Net architecture (see Figure 1): *input conditioning* (score is concatenated to the input audio before the decoder), *output conditioning* (score is concatenated to the decoder's output before the output layer), and *input-output conditioning* (a combination of both). Other conditioning locations are also possible, but they would require a transformed score representation. Conditioning at the bottleneck, for example, would necessitate resampling the score information to the bottleneck's much lower temporal resolution, thus discarding important timing information contained in the score. Conditioning at the bottleneck could work well when the conditioning has no temporal dimension, such as instrument labels [45].

6.3 Multi-Source Training

In addition to the standard method of training the network to extract specific voices, we propose a *multi-source* model variant which can separate any one of the four voices given only that voice's score. To achieve this, we train a model

to extract a single voice from the mixture, where every training example consists of a mixture segment (used as input to the model), the score of one random voice out of the four voices (used to condition the model), and the corresponding audio for that voice as the target to extract (used to compute the loss). Since training examples do not explicitly specify which voice they correspond to, the model learns to extract the desired voice based on its score alone. Whereas a normal score-informed model *could* use the score to improve separation results, the multi-source model *must* make use of the score. In this sense, the separation is not only score-informed, it is *score-guided*. A multi-source model also gives greater flexibility by allowing users to choose individual notes to extract, possibly alternating between voices. Furthermore, multi-source training can enable a model trained only on four-voice mixtures to be used on recordings with any number of voices.

7. EXPERIMENTS AND RESULTS

To evaluate model performance, we use the SDR metric [46] provided by the BSS Eval library (version 4) [9] with its default settings [7]. Like SiSEC 2018 and subsequent works, we report median SDR rather than mean in order to reduce the effect of outliers. We compare all proposed model variants in 6 experiments, listed in Table 1. Audio examples are available online. [8] We assess whether certain methods perform better than others by reporting p-values from pairwise Conover–Iman tests [47] (also used by [9]; we adjust for multiple comparisons using the Bonferroni method [48]), always after rejecting the Kruskal–Wallis [49] null hypothesis with $P < 0.001$.

Experiment	Method	Score-Informed	Model Type
1	SI-NMF	yes	-
2	Wave-U-Net	no	one model for all voices
3	Wave-U-Net	no	one model per voice
4	Wave-U-Net	yes	one model for all voices
5	Wave-U-Net	yes	one model per voice
6	Wave-U-Net	yes	one model: multi-source

Table 1. List of experiments
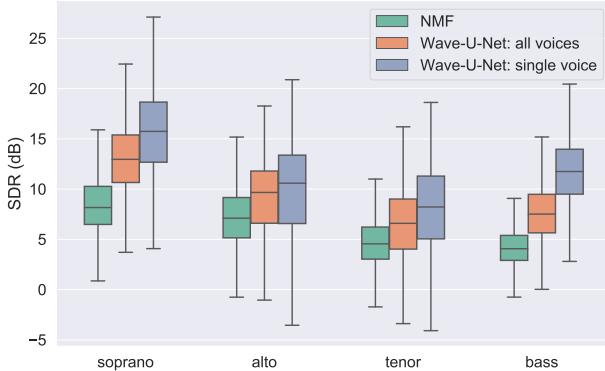
7.1 Experiments 1–3: SI-NMF and Wave-U-Net

A comparison of separation performance of SI-NMF and Wave-U-Net on the test set is shown in Figure 2. While SI-NMF achieves decent results, Wave-U-Net consistently outperforms it in all voices by a large margin ($P < 0.001$).
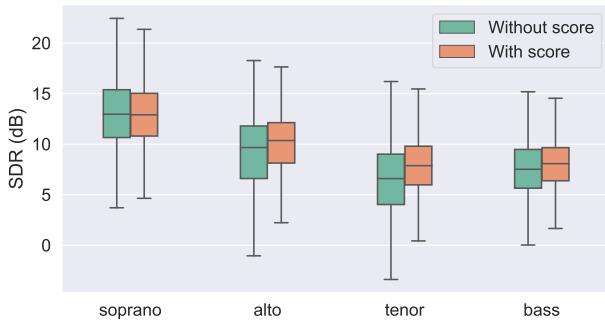
In SI-NMF, interferences between estimated sources are very low due to the hard constraints imposed using the score. However, estimated sources contain noticeable amplitude modulation artifacts. These are likely caused by the use of static spectral templates, which cannot effectively model the continuous evolution of spectral parameters in choral music. Source-filter signal models can be integrated into NMF to improve its performance in such cases [50–52]. The effectiveness of such models for choral

---

[7] We also provide supplementary SIR, SAR, and ISR evaluations. [8]
[8] https://www.matangover.com/choirsep-ismir

Figure 2. Results for SI-NMF and Wave-U-Net

Figure 3. Results from Experiment 4 (with score; score type: normalized pitch, conditioning location: input) compared to Experiment 2 (without score)

Figure 4. Results from Experiment 4 by voice, score type, and conditioning location
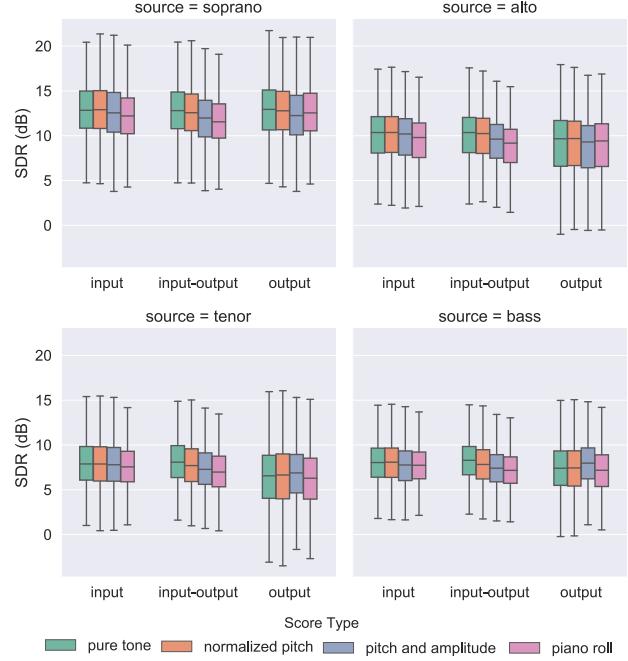
music may be limited, however, as a choir section is actually composed of multiple sound sources (singers). Figure 2 further shows that single-voice Wave-U-Net (Experiment 3) is superior to all-voice Wave-U-Net (Experiment 2) ($P < 0.001$).

Examination of segments in which the model achieved a particularly low SDR reveals that the most common source of errors is misclassified notes (that is, when the model assigns notes to the wrong voice) [43]. One cause for misclassified notes is voice crossings, which occur when the normal voice ordering is violated. The fact that voice crossings cause misclassification shows that the model has learned to rely on the standard ordering of the voices. Misclassified notes also occur in segments in which one voice is silent while the other voices continue to sing. In such segments the model cannot always infer which voice is silent due to the overlap between voice ranges.

7.2 Experiment 4: Score-Informed, Extract All Voices

In Experiment 4 we examine the effect of adding score conditioning to the model from Experiment 2. We train 12 score-informed model variants: all combinations of 4 score representations and 3 conditioning locations. Figure 3 shows that adding the score improves median SDR in all voices ($P < 0.001$) except for soprano ($P > 0.05$).

Figure 4 compares all score conditioning methods. Conditioning location has no consistent effect on soprano

and bass separation. For alto and tenor, however, output conditioning is overall worse than both input and input-output conditioning ($P < 0.001$). We suspect that output conditioning performs poorly because the Wave-U-Net output layer is a simple sample by sample dot-product (convolutional layer with kernel of size 1).

7.3 Experiment 5: Score-Informed, Extract Single

Figure 5 compares the performance of score conditioning methods for tenor extraction in Experiment 5. We compare results for tenor specifically because it is the most challenging to separate (it achieved the lowest median SDR in most experiments). Output conditioning gives the worst performance and has no significant effect compared to no score at all ($P > 0.05$). It appears the models conditioned at the output have learned to simply ignore the score. For input and input-output conditioning, the choice of score type has no effect, and all score types perform considerably better than no score at all ($P < 0.001$), with an improvement of up to 2.7 dB in median SDR.

7.4 Experiment 6: Score-Informed, Multi-Source

This experiment tested the effect of score conditioning method on multi-source training (described in Section 6.3). We do not include a figure due to limited space, see website [8] for results. Models using output conditioning perform very poorly, confirming the results of Experiments 4 and 5. Other than that, conditioning method does not have an effect in this experiment. The difference in median SDR between the best method (pitch and amplitude, input) and the worst method (piano roll, input-output) is only 0.4 dB.

Figure 5. Comparison of score conditioning methods in Experiment 5 (on tenor only), with the non-score-informed counterpart (Experiment 3) shown for reference

7.5 Overall Comparison

In Figure 6 we compare results from all experiments. For the score-informed models we use the conditioning method that performed best, taking into account all experiments and all voices (score type: pitch and amplitude, conditioning location: input). As expected, using the score improves performance mainly for the inner voices (alto and tenor), as they are more prone to induce misclassified notes due to voice crossings and vocal range overlap (see Section 7.1). Examination of frames with misclassified notes confirms that using the score eliminates this problem [43, p. 95].
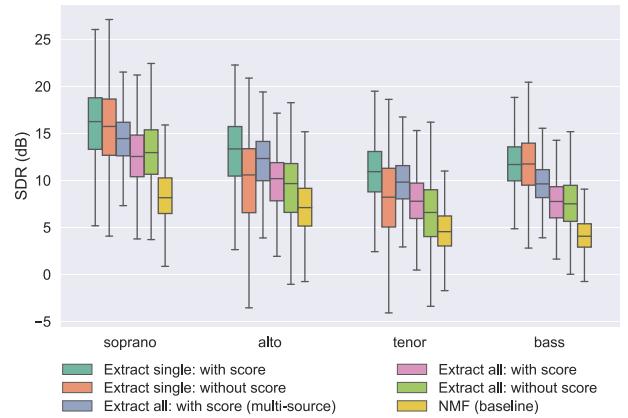
The score-informed single-source model has the best performance overall. For alto and tenor, this model achieves a 2.7 dB improvement in median SDR compared to the best non-score-informed model ($P < 0.001$). For soprano, the improvement is only 0.5 dB ($P < 0.001$) and for bass performance is degraded by 0.06 SDR ($P < 0.01$). Compared to the NMF baseline, score-informed Wave-U-Net improves median SDR by 6.2 to 8.1 dB ($P < 0.001$).

Interestingly, for tenor and alto, the multi-source model outperforms the non-score-informed single-source model ($P < 0.001$), even though the multi-source model uses only a quarter of the parameters (because it uses a single model for all four voices).

Listening to audio results of score-informed models,[8] we notice that most score conditioning methods result in audible clicks at note boundaries. This is likely caused by the discontinuity of the score representations at these locations. These clicks hardly affect the SDR evaluations because they are highly localized. Using the pure tone score representation eliminates these clicks almost completely.

7.6 Evaluation on Real-World Recordings

Although our models have only been trained on synthesized data, we also evaluate using real choir recordings from the Choral Singing Dataset [33]. In this evaluation, non-score-informed Wave-U-Net (Experiment 3 model) performs poorly with a median SDR of 0 dB (for all voices

Figure 6. Comparison of results from all experiments

combined). Score-informed Wave-U-Net performs better with SDR of 1.4 and 1.5 dB (models from Experiments 5 and 6, respectively). SI-NMF outperforms Wave-U-Net by a large margin with SDR of 5.6 dB ($P < 0.001$).

Listening to estimated sources we notice that score-informed Wave-U-Net predicts all the right notes, but cannot faithfully generate the lyrics and unique timbre of the specific choir, likely due to it being trained on a dataset containing only a single choir without any lyrics. SI-NMF predictions also omit many of the lyrics and timbre variations, but are nonetheless better than Wave-U-Net in this case. This shows that to be effective on real-world recordings, Wave-U-Net needs to be trained on a more representative dataset. We postulate that if score-informed Wave-U-Net (or similar methods) could be trained on a diverse dataset of choral recordings, it would achieve an improvement over SI-NMF that is comparable to the improvement that it has achieved on the synthesized dataset.

8. CONCLUSIONS

In this paper we investigated source separation of choral music. Due to the lack of publicly available datasets, we developed a dataset of synthesized Bach chorales. We established baseline separation performance using a score-informed NMF method. We then showed that NMF is outperformed by Wave-U-Net, a deep learning separation technique. We further proposed to condition Wave-U-Net on musical scores. Our experiments with several conditioning methods showed that using the score improves separation quality. We introduced multi-source training, in which a single model separates any of the four choir voices using only the score as a guide. We found that multi-source training performs comparably to single-source training, even though it requires much less resources.

When evaluated on real choir recordings, SI-NMF still outperforms Wave-U-Net. Hence, a major challenge that remains is compiling a multi-track choir recording dataset to be used for training. Until such a dataset is available, better choir synthesis methods could be used. Another avenue for improvement would be to consider more versatile conditioning methods, such as FiLM layers [53, 54].

9. ACKNOWLEDGEMENTS

10. REFERENCES

[1] E. Vincent, T. Virtanen, and S. Gannot, *Audio Source Separation and Speech Enhancement*. John Wiley & Sons, 2018.

[2] E. Cano, D. FitzGerald, A. Liutkus, M. D. Plumbley, and F.-R. Stöter, "Musical source separation: An introduction," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 31–40, 2019.

[3] B. Smith and R. T. Sataloff, *Choral Pedagogy*, 3rd ed. Plural Publishing, 2013.

[4] K. Ihalainen, "Methods of choir recording for an audio engineer," Bachelor's thesis, Tampere University of Applied Sciences, 2008.

[5] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[6] D. Stoller, S. Ewert, and S. Dixon, "Wave-U-Net: A multi-scale neural network for end-to-end audio source separation," in *Proc. of the International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 334–340.

[7] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2018.

[8] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, D. FitzGerald, and B. Pardo, "An overview of lead and accompaniment separation in music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307–1335, 2018.

[9] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *14th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, Guildford, UK, 2018, pp. 293–305.

[10] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - a reference implementation for music source separation," *Journal of Open Source Software*, 2019. [Online]. Available: https://doi.org/10.21105/joss.01667

[11] N. Takahashi, N. Goswami, and Y. Mitsufuji, "MM-DenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, Tokyo, Japan, 2018, pp. 106–110.

[12] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," *arXiv preprint arXiv:1911.13254*, 2019.

[13] F. Lluís, J. Pons, and X. Serra, "End-to-end music source separation: Is it possible in the waveform domain?" *arXiv:1810.12187 [cs, eess]*, 2018.

[14] E. M. Grais, D. Ward, and M. D. Plumbley, "Raw multi-channel audio source separation using multi-resolution convolutional auto-encoders," in *Proc. of the 26th European Signal Processing Conference (EUSIPCO)*, Rome, Italy, 2018, pp. 1577–1581.

[15] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1702–1726, 2018.

[16] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep U-Net convolutional networks," in *Proc. of the International Society for Music Information Retrieval Conference*, Suzhou, China, 2017.

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Cham, Switzerland, 2015, pp. 234–241.

[18] S. Ewert, B. Pardo, M. Müller, and M. D. Plumbley, "Score-informed source separation for musical audio recordings: An overview," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 116–124, 2014.

[19] Y. Meron and K. Hirose, "Separation of singing and piano sounds," in *5th International Conference on Spoken Language Processing*, Sydney, Australia, 1998.

[20] C. Raphael, "A classifier-based approach to score-guided source separation of musical audio," *Computer Music Journal*, vol. 32, no. 1, pp. 51–59, 2008.

[21] J. Ganseman, G. J. Mysore, J. S. Abel, and P. Scheunders, "Source separation by score synthesis," in *Proc. of the International Computer Music Conference (ICMC)*, New York, NY, 2010, pp. 462–465.

[22] A. Ben-Shalom and S. Dubnov, "Optimal filtering of an instrument sound in a mixed recording given approximate pitch prior," in *Proc. of the International Computer Music Conference (ICMC)*, San Francisco, CA, 2004.

[23] Y. Li, J. Woodruff, and D. Wang, "Monaural musical sound separation based on pitch and common amplitude modulation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 7, pp. 1361–1371, 2009.

[24] Z. Duan and B. Pardo, "Soundprism: An online system for score-informed source separation of music audio," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1205–1215, 2011.

[25] S. Ewert and M. Müller, "Using score-informed constraints for NMF-based source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, 2012, pp. 129–132.

[26] R. Hennequin, B. David, and R. Badeau, "Score informed audio source separation using a parametric model of non-negative spectrogram," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 45–48.

[27] U. Şimşekli and A. T. Cemgil, "Score guided musical source separation using Generalized Coupled Tensor Factorization," in *Proc. of the 20th European Signal Processing Conference (EUSIPCO)*, Bucharest, Romania, 2012, pp. 2639–2643.

[28] F. J. Rodriguez-Serrano, Z. Duan, P. Vera-Candeas, B. Pardo, and J. J. Carabias-Orti, "Online score-informed source separation with adaptive instrument models," *Journal of New Music Research*, vol. 44, no. 2, pp. 83–96, 2015.

[29] F. J. Rodriguez-Serrano, S. Ewert, P. Vera-Candeas, and M. Sandler, "A score-informed shift-invariant extension of complex matrix factorization for improving the separation of overlapped partials in music recordings," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, 2016, pp. 61–65.

[30] M. Miron, J. J. Carabias-Orti, J. J. Bosch, E. Gómez, and J. Janer, "Score-informed source separation for multichannel orchestral recordings," *Journal of Electrical and Computer Engineering*, vol. 2016, 2016.

[31] S. Ewert and M. B. Sandler, "Structured dropout for weak label and multi-instance learning and its application to score-informed source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017, pp. 2277–2281.

[32] M. Miron, J. Janer, and E. Gómez, "Monaural score-informed source separation for classical music using convolutional neural networks," in *Proc. of the International Society for Music Information Retrieval Conference*, Suzhou, China, 2017, pp. 55–62.

[33] H. Cuesta, E. Gómez, A. Martorell, and F. Loáiciga, "Analysis of intonation in unison choir singing," in *Proc. of the International Conference on Music Perception and Cognition (ICMPC)*, Graz, Austria, 2018.

[34] M. Blaauw, J. Bonada, and R. Daido, "Data efficient voice cloning for neural singing synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019, pp. 6840–6844.

[35] J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gómez, and J. P. Bello, "An analysis/synthesis framework for automatic f0 annotation of multitrack datasets," in *Proc. of the International Society for Music Information Retrieval Conference*, Suzhou, China, 2017, pp. 71–78.

[36] M. Miron, J. Janer, and E. Gómez, "Generating data to train convolutional neural networks for classical music source separation," in *Proc. of the Sound and Music Computing Conference*, Espoo, Finland, 2017, pp. 227–233.

[37] D. Henningsson and F. D. Team, "FluidSynth real-time and thread safety challenges," in *Proc. of the 9th International Linux Audio Conference*, Maynooth, Ireland, 2011, pp. 123–128.

[38] R. L. Marshall and R. A. Leaver, "Chorale," in *Grove Music Online*. Oxford, UK: Oxford University Press, 2001.

[39] ——, "Chorale settings," in *Grove Music Online*. Oxford, UK: Oxford University Press, 2001.

[40] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," in *Proc. of the International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010, pp. 637–642.

[41] A. Cichocki, R. Zdunek, and S.-i. Amari, "New algorithms for non-negative matrix factorization in applications to blind source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, Toulouse, France, 2006, pp. 621–624.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, 2014.

[43] M. Gover, "Score-informed source separation of choral music," Master's thesis, McGill University, 2019.

[44] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. de Vries, A. Courville, and Y. Bengio, "Feature-wise transformations," *Distill*, vol. 3, no. 7, p. e11, 2018.

[45] O. Slizovskaia, L. Kim, G. Haro, and E. Gómez, "End-to-end sound source separation conditioned on instrument labels," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019, pp. 306–310.

[46] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[47] W. J. Conover and R. L. Iman, "On multiple-comparisons procedures," Los Alamos Scientific Laboratory, Tech. Rep. LA-7677-MS, 1979.

[48] C. Bonferroni, "Teoria statistica delle classi e calcolo delle probabilità," *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, vol. 8, pp. 3–62, 1936.

[49] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

[50] T. Heittola, A. Klapuri, and T. Virtanen, "Musical instrument recognition in polyphonic audio using source-filter model for sound separation," in *Proc. of the International Society for Music Information Retrieval Conference*, Kobe, Japan, 2009, pp. 327–332.

[51] J.-L. Durrieu, A. Ozerov, C. Févotte, G. Richard, and B. David, "Main instrument separation from stereophonic audio signals using a source/filter model," in *17th European Signal Processing Conference*, Glasgow, UK, 2009, pp. 15–19.

[52] T. Nakamura and H. Kameoka, "Shifted and convolutive source-filter non-negative matrix factorization for monaural audio source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 489–493.

[53] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *32nd Conference on Artificial Intelligence (AAAI-18)*, New Orleans, LA, 2018.

[54] G. Meseguer-Brocal and G. Peeters, "Conditioned-U-Net: Introducing a control mechanism in the U-Net for multiple source separations," in *Proc. of the International Society for Music Information Retrieval Conference*, Delft, Netherlands, 2019.

CAN'T TRUST THE FEELING? HOW OPEN DATA REVEALS UNEXPECTED BEHAVIOR OF HIGH-LEVEL MUSIC DESCRIPTORS

Cynthia C. S. Liem
Delft University of Technology
Delft, The Netherlands
c.c.s.liem@tudelft.nl

Chris Mostert
Delft University of Technology
Delft, The Netherlands
chrismostert@outlook.com

ABSTRACT

Copyright restrictions prevent the widespread sharing of commercial music audio. Therefore, the availability of resharable pre-computed music audio features has become critical. In line with this, the AcousticBrainz platform offers a dynamically growing, open and community-contributed large-scale resource of locally computed low-level and high-level music descriptors. Beyond enabling research reuse, the availability of such an open resource allows for renewed reflection on the music descriptors we have at hand: while they were validated to perform successfully under lab conditions, they now are being run 'in the wild'. Their response to these more ecological conditions can shed light on the degree to which they truly had construct validity. In this work, we seek to gain further understanding into this, by analyzing high-level classifier-based music descriptor output in AcousticBrainz. While no hard ground truth is available on what the true value of these descriptors should be, some oracle information can still be derived, relying on semantic redundancies between several descriptors, and multiple feature submissions being available for the same recording. We report on multiple unexpected patterns found in the data, indicating that the descriptor values should not be taken as absolute truth, and hinting at directions for more comprehensive descriptor testing that are overlooked in common machine learning evaluation and quality assurance setups.

1. INTRODUCTION

In many music information retrieval (MIR) applications, it is useful to include information related to music content. However, many large-scale music audio collections of interest cannot legally be shared as-is. As a compromise, efforts have been undertaken to locally pre-compute music audio descriptors and make these available through APIs or as part of research datasets. Parties without in-house access to large audio corpora need to rely on such data for

subsequent use. Indeed, large-scale pre-computed descriptor corpora have been feeding into further machine learning pipelines, empowering music applications, facilitating benchmarking initiatives [1, 2], and leading to inferences and statements about the nature of music preferences and listening behavior at an unprecedented scale [3–6].

Audio-based music descriptors are commonly divided into low- and high-level descriptors. Low-level descriptors can closely be related to the audio signal, while high-level descriptors are more semantically understandable to humans. This does not make high-level descriptors easier to extract; many of them cannot objectively and directly be measured in the physical world, and thus consider *constructs* rather than physically measurable phenomena.

The performance of automated music descriptor extraction procedures is reported according to the common evaluation methodologies in the field. For descriptors based on supervised machine learning, this normally includes a performance report on a test set that was partitioned out of the original dataset and not seen during training, or on cross-validation outcomes. However, descriptors that are reported and assumed to be successful may still be prone to sensitivities not explicitly accounted for in their design and evaluation. In lower-level music descriptors, implementations of MFCC and chroma descriptors showed sensitivities to different audio encoding formats [7], while common textual descriptions of audio extractor pipelines turned out insufficiently specific to yield reproducible results [8]. For higher-level descriptors, seemingly well-performing trained music genre classifiers turned out to be unexpectedly sensitive to subtle, humanly interpretable audio transformations [9]. Such sensitivities are not restricted to music genre classification; for example, trade-offs between accuracy and semantic robustness have also been observed in deep music representations [10]. Generally, in many MIR tasks, ground truth relies on human judgement and labeling. This may be imprecise and subjective, leading to low inter-rater agreement. In its turn, this leads to questions on whether a clear-cut ground truth exists at all, while this often is fundamental to machine learning techniques and their evaluation [11–14].

Can we tell whether automated descriptors are as trustable as initially assumed? Do they truly measure what they are intended to measure? Do they match broader, less explicitly encoded assumptions we have on them? These are important questions to ask: in case of negative an-

swers, the descriptors may not provide a valid basis for subsequent work to build upon. However, finding sensitivities that were unnoticed in original evaluation contexts is non-trivial, requiring a broader, more meta-analytic perspective. In this work, we focus on this, by providing an analysis of music descriptor values obtained through the AcousticBrainz [15] platform. By soliciting community-contributed submissions of locally run, but largely standardized music feature extractors, the platform offers a large-scale perspective on music that 'people felt worth the upload'. As such, it offers a more ecological 'in-the-wild' data perspective than what was studied in the lab, when the descriptors were originally designed. Indeed, through cross-collection evaluation procedures employing independent ground truth validation sets, several well-known genre classification models were shown not to generalize well beyond their original evaluation datasets [16].

The AcousticBrainz data is unusually transparent and rich: more so than e.g. the popular Million Song Dataset [17]. Many descriptor fields are available for each submission, multiple submissions can be added for the same MusicBrainz recording, each submission is encoded with additional metadata on characteristics of the input audio and the extractor software, and the extractor software is open source [18]. We use this richness to comprehensively analyze existing computed descriptor values in AcousticBrainz. Rather than relying on explicit and clear-cut ground truth, we look at the data through a meta-scientific lens, and impose more general assumptions on descriptor behavior, inspired by psychological and software testing techniques. This way, we will reveal several unexpected patterns in the descriptor values. As original music audio is not attached to the descriptor entries, we will not (yet) be able to fully replicate how descriptors were computed, nor will we be able to recreate experimental conditions on this data, in which possible reasons for unexpected behavior can cleanly be statistically controlled. Still, our analysis will help in pinpointing concrete directions towards future controlled studies.

In the remainder of this paper, we will discuss related work in Section 2. Then, we will introduce the data used for our analyses in Section 3, after which we will present analyses into intra-dataset correlations (Section 4), descriptor stability (Section 5), and descriptor value distributions (Section 6), followed by the conclusion and an outlook towards future work.

2. RELATED WORK

In conducting science, it is non-trivial to assess whether the outcomes we are observing, the inferences we are making and the conclusions we are drawing are truly correct. These questions of *validity* were first acknowledged in the domain of psychological testing, where the focus was on measuring psychological constructs: abstracted human characteristics (e.g. 'conscientiousness') that are not directly and physically observable, but that can still be measured (e.g. through well-designed surveys). Various subcategories of validity exist [19]. Among these, one of the

most intuitive to understand, yet hardest to pinpoint, is the notion of *construct validity*: the question whether a measurement procedure can indeed be considered to yield a *"measure of some attribute or quality which is not "operationally defined""* [20].

The traditional viewpoint on ways to assess construct validity, is to consider a measure procedure as part of a *nomological network*, and relate its outcomes to those of other procedures, that have previously been shown to be valid [20]; in practice, in much of psychological research, this is done by assessing correlations between construct measurements that are theorized to have an interpretable relation to one another. This does create dependencies under uncertainty, still boiling down to a philosophical question of 'what the first truth is to start with'—something that may be disproven during the research process, as more evidence will come in and further comparisons are being made. It has therefore been argued that comprehensive inquiry into construct validity will not only lead to better assessments, but also leads to fundamental questionings and improvements of the complete scientific process [21].

Within MIR, while comprehensive meta-scientific questions on this have not been asked, criticisms of current evaluation practices, referring to the notions of both validity and reliability and the way in which they have been used in the Information Retrieval field, have been presented by Urbano et al. [22]. In addition, Sturm's criticisms of 'horse systems' in MIR [9] (machine learning-based systems that performance-wise appear to make humanly intelligent decisions, but that turn out to pick up on irrelevant confounds in data) can again be related to construct validity.

As a method to assess whether a system is a 'horse system', Sturm proposes to investigate how systems react to input data transformations that are considered 'irrelevant' (i.e. imperceptible) to humans. Interestingly, this technique has been used in another research field focused on 'testing': the field of software testing, in which it would be called *metamorphic testing* [23]. While software testing appears to be a much more objective and precise procedure than psychological testing, from a formal, logical perspective, many real-life programs may actually be considered non-testable, and the problem of determining whether a software artefact is bug-free is undecidable [24]. While one cannot pinpoint one exact oracle truth, it still may be possible to *derive* partial oracle truth through transformations based on known data relationships [25], e.g. by applying input transformations that should not change a system's output, which is done in metamorphic testing.

3. ACOUSTICBRAINZ

In our studies, we study descriptor values as found through the AcousticBrainz platform. More specifically, we will depart from the most recent high-level descriptor data dump obtained through the AcousticBrainz website [1]. We are interested in the high-level descriptors, as they should

mimic humanly understandable semantic concepts, which should be relatable in humanly interpretable ways.

The data dump considers 1,805,912 entries of community-contributed high-level descriptor values, that can be broken down into genres, moods, and other categories (e.g. danceability); a full overview can be found in ([15], Table 4). Unless indicated otherwise, our analyses will consider this full data dump. In all cases, descriptor values consider classification outputs, obtained through machine learning; for each possible class label within a descriptor (e.g., `jazz` in the genre_dortmund classifier), the classifier confidence for that class label is given as a float value. The performance of each of the classifiers is documented on the AcousticBrainz website; where possible, performance is reported on publicly available datasets [2].

4. INTRA-DATASET CORRELATIONS
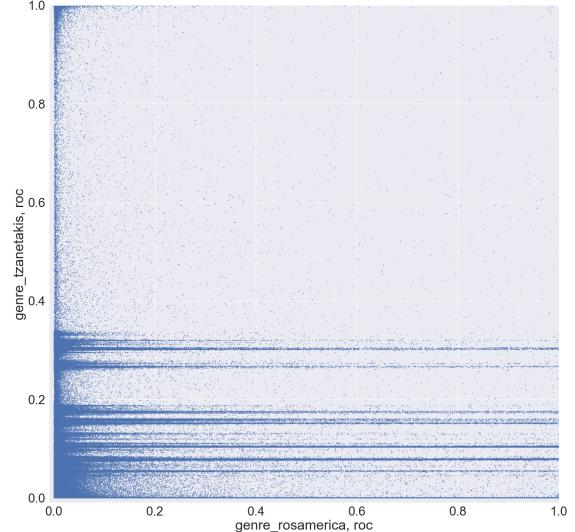
Following the psychological concept of the nomological network, one way to assess validity is to assess how the outcomes of related measurement procedures correlate with each other. For this, we take advantage of *semantic redundancy* within the AcousticBrainz high-level descriptors. For example, several musical genres literally re-occur as class labels within the various genre classifiers. Then, it is not unrealistic to assume that, given the same audio input, the output of alternative jazz classifiers should positively correlate. Furthermore, some 'softer' assumptions on meaningful relationships can be made: e.g., aggressive music is likely not relaxed, and happy music is likely not sad. We defined multiple of these relationships for which we would expect to observe (strong) positive correlations between classifier label predictions, and computed their Pearson correlations. The results are displayed in Table 1.

The found correlations were unexpected; we were especially surprised by the very low correlations found for the genre classifiers, while they should target the same concepts. A scatter plot of `rock` classifier confidences in genre_rosamerica and genre_tzanetakis (which yielded a negative correlation) is given in Figure 1. It appears that confidences outcomes do not uniformly distribute over the full [0.0, 1.0] confidence range; we will investigate this further in the following sections.

Out of all 'softer' assumptions that were compared, the lowest correlation (.13) is between `happy` and `not sad`, implying that music classified as happy could be sad at the same time. The classifiers used in AcousticBrainz indeed allow for this, as separate binary classifiers exist for happy and sad moods; however, this contradicts Russell's 2D circumplex model of affect [26], in which happiness and sadness would have opposite scores on the valence dimension.

5. STABILITY

Our correlation analyses showed unexpected results. However, as different classifiers were trained on different datasets, they may have considered different characteristics of the input data. Inspired by the idea of derived oracles,

---

[2] https://AcousticBrainz.org/datasets/accuracy

Figure 1: Scatter plot of classifier confidences. Each point indicates an AcousticBrainz submission, with confidences for genre_rosamerica, `roc` and genre_tzanetakis, `roc`.

we can however also consider relationships that should be closer to the identity, and thus should lead to (nearly) identical outcomes.

In AcousticBrainz, multiple submissions can be made for the same MusicBrainz recording ID (MBID). Semantically, a MusicBrainz recording really references one and the same recording. So while users may have encoded the recording audio in different ways, and may be using different versions of the feature extractor, we should intuitively be able to assume that re-submissions of one and the same recording should yield descriptor values that are very close to one another. In other words, we wish for re-submissions for the same MBID to display *stability*.

For this, we need to consider the MBIDs in our data dump that have more than one associated submission. Filtering for this led to a corpus of 941,018 submissions for 299,097 different MBIDs. If n submissions are available for a given MBID, a given classifier c and a given classifier label l, the corresponding classifier confidences for these submissions can now be grouped into a population $(MBID, c, l)$ of size n. Considering we have k unique MBIDs in our dataset (in our case, $k = 299,097$), we can then enumerate the populations as $[(MBID_1, c, l), (MBID_2, c, l), ..., (MBID_k, c, l)]$, and operate within and/or across them when calculating instability metrics.

We consider two alternative ways to quantify instability. First, for each of the submission populations, we can compute the variance observed for classifier confidences, for each label l in classifier c. As there may be a varying amount of submissions within a population, we normalize for this by computing the *pooled variance* $\overline{var}(c, l)$ over our filtered corpus as follows:

$$\overline{var}(c, l) = \frac{\Sigma_{i=1}^{k}(n_i \times var((MBID_i, c, l)))}{\Sigma_{i=1}^{k} n_i} \quad (1)$$

Classifier, label A	Classifier, label B	Pearson's r	p
genre_rosamerica, `cla`	genre_tzanetakis, `cla`	.29	<.001
genre_dortmund, `rock`	genre_rosamerica, `roc`	.24	<.001
genre_dortmund, `jazz`	genre_rosamerica, `jaz`	.22	<.001
genre_dortmund, `pop`	genre_rosamerica, `pop`	.11	<.001
genre_dortmund, `jazz`	genre_tzanetakis, `jaz`	.08	<.001
genre_rosamerica, `pop`	genre_tzanetakis, `pop`	.06	<.001
genre_rosamerica, `hip`	genre_tzanetakis, `hip`	.05	<.001
genre_rosamerica, `jaz`	genre_tzanetakis, `jaz`	.02	<.001
genre_dortmund, `blues`	genre_tzanetakis, `blu`	.01	<.001
genre_dortmund, `pop`	genre_tzanetakis, `pop`	-.05	<.001
genre_dortmund, `rock`	genre_tzanetakis, `roc`	-.06	<.001
genre_rosamerica, `roc`	genre_tzanetakis, `roc`	-.07	<.001
mood_aggressive, `aggressive`	mood_relaxed, `not_relaxed`	.59	<.001
mood_acoustic, `acoustic`	mood_electronic, `not_electronic`	.58	<.001
danceability, `danceable`	mood_party, `party`	.53	<.001
mood_electronic, `electronic`	genre_dortmund, `electronic`	.48	<.001
danceability, `danceable`	genre_rosamerica, `dan`	.33	<.001
mood_happy, `happy`	mood_party, `party`	.20	<.001
mood_happy, `happy`	mood_sad, `not_sad`	.13	<.001

Table 1: Pearson correlations between high-level classifier outcomes, theorized to positively correlate with another.

where n_i is the sample size of the ith population in our enumeration.

As there are multiple possible labels within the same classifier, but we want to discuss outcomes at the classifier level, we then take the mean pooled variance, $\overline{\overline{\mathrm{var}(c)}}$, over all possible labels $l \in L_c$ for classifier c.

When using variances, classifier confidences are considered to be informative. Alternatively, one could choose to rather consider each classifier label as a binary label. To reflect this perspective, for each population and for each classifier, we can compute the normalized information entropy $\hat{H}(\mathrm{MBID}_i, c)$, which uses the Shannon entropy [27], but normalizes by the amount of possible labels $|L_c|$ for c:

$$
\begin{aligned}
&\hat{H}(\mathrm{MBID}_i, c) \\
&= -\Sigma_{l \in L_c} \frac{P((\mathrm{MBID}_i, c, l)) \log_2 P((\mathrm{MBID}_i, c, l))}{\log_2 |L_c|} \\
&= -\Sigma_{l \in L_c} P((\mathrm{MBID}_i, c, l)) log_{|L_c|} P((\mathrm{MBID}_i, c, l))
\end{aligned}
$$
(2)

where $P((\mathrm{MBID}_i, c, l))$ is the probability of label l in classifier c, following the observed empirical distribution within the population corresponding to MBID_i. Then, to have a weighted measure per classifier over the whole filtered corpus, we calculate the pooled normalized entropy $\overline{\hat{H}}(c)$, similarly to how we computed the pooled variance.

While we want for descriptor values to be stable within a submission, it is usually not the intention that for a given descriptor, the classifier would be so stable that it always predicts a single l throughout the whole corpus. This e.g. happens for the genre_dortmund classifier, which unrightfully classifies many AcousticBrainz submissions as electronic music, as also noticed in [16]. To quantify the unbiasedness of a classifier, we compute the normalized entropy for each classifier over our complete (unfiltered) corpus, denoted as $\hat{H}(c)_{all}$. A higher $\hat{H}(c)_{all}$ denotes a more uniform distribution over the different possible class labels for c across the corpus, and thus lower classifier bias.

Plots in which we illustrate $\overline{\overline{\mathrm{var}(c)}}$ and $\overline{\hat{H}}(c)$ (pooled with regard to recordings with multiple submissions) vs. $\hat{H}(c)_{all}$ (taken across the whole, unfiltered corpus) are shown in Figure 2. As we can see, indeed, the genre classifiers turn out stable but highly biased. While in most cases, observed trends are comparable for the two possible instability measures, some exceptions are found, most notably on the gender classifier, which is considered stable when using $\overline{\overline{\mathrm{var}(c)}}$, but unstable when using $\overline{\hat{H}}(c)$. Seemingly, confidences for this classifier are close to 0.5, meaning that male/female classifications easily flip within a submission.
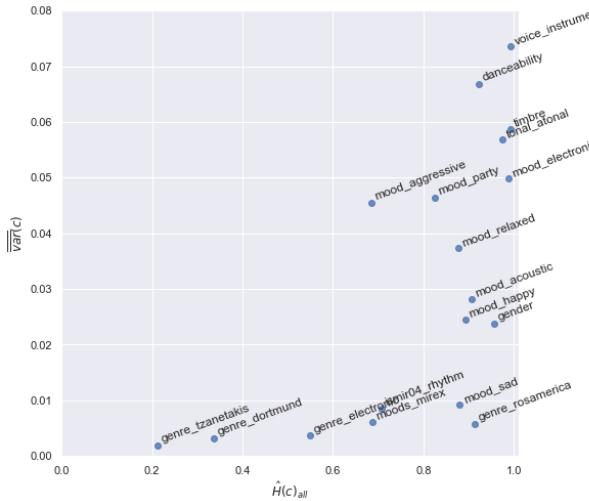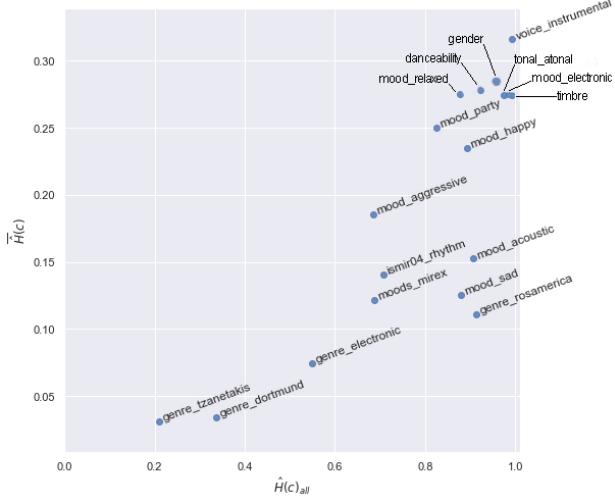
6. VALUE DISTRIBUTIONS

From Figure 1, it was observed that descriptor values clustered together in small bands. This behavior occurs for several genre and mood classifiers. To illustrate this, Figure 3 displays a histogram of descriptor values for the mood_acoustic, mood_relaxed, mood_electronic and mood_sad classifiers, as observed across the complete AcousticBrainz corpus. Some confidence values seem disproportionally represented: in the histogram, sharp spikes occur for mood_acoustic, mood_relaxed, mood_electronic, and a minor spike for mood_sad.
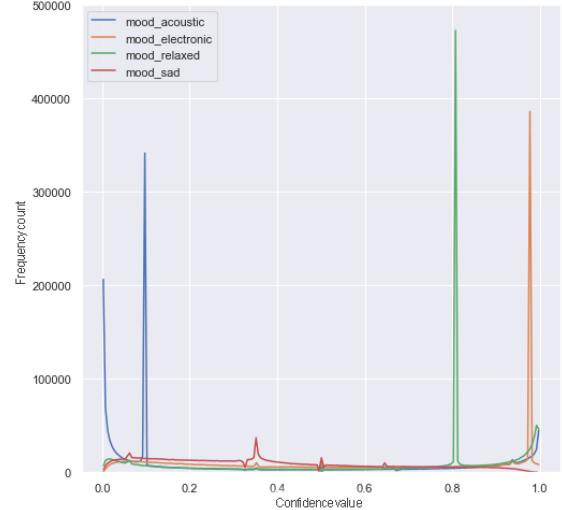
There are various reasons why this may be the case. Possibly, the community may have fed skewed data to the classifier. Alternatively, the feature extractor may have shown anomalous responses to specific inputs. For each submission, we have rich metadata, which e.g. includes information about audio codecs, bit rates, song lengths, and software library versions that were used when the submission was created. While, in the absence of a conscious experimental design underlying the data, we cannot cleanly test for contributions of individual facets, we still can examine *whether major distributional differences occur for submissions with scores within the anomalous-looking spikes, when comparing these to submissions with scores outside of these.*

For this, for each of the classifiers, we manually define range intervals for the classifier confidences, within which

(a) Instability based on mean pooled variance $\overline{\overline{\mathrm{var}}}(c)$.

(b) Instability based on pooled normalized entropy $\overline{\overline{\hat{H}}}(c)$.

Figure 2: Submission instability vs. corpus-wide unbiasedness ($\hat{H}(c)_{all}$).

we consider a submission to belong to an anomalous classifier confidence value spike. We then compare the metadata value distributions of submissions within each classifier spike to those of submissions that do not occur in any of the four anomalous spikes (1,239,882 submissions for 855,266 unique MBID recordings).

To investigate whether the observed anomalies may have been skewed towards any particular genre, we also study a subset of our corpus, which was cross-matched against the AcousticBrainz genre dataset [28]. More specifically, we only kept MBIDs which also occurred in all three publicly available ground truth sets (Discogs, last.fm and tagtraum) of the AcousticBrainz genre dataset, reducing the corpus to 402,279 submissions for 164,826 unique MBID recordings. Examining confidence value distributions for this filtered dataset, we still observed the same anomalous spikes for the same range intervals. Therefore, we will apply the same range intervals as before to select values associated to anomaly spikes, and will

Figure 3: Histogram of descriptor values for several classifiers, considered across the whole corpus.

again compare distributional differences between these and non-anomalous submissions (now amounting to 267,394 submissions for 128,687 unique MBID recordings), in this case to see whether certain genres are overrepresented in the anomalous spikes. For each classifier of interest, an overview of anomalous spike interval ranges and counts of corresponding unique recording MBIDs and submissions is given in Table 2.

To quantify distributional differences, we use the Jensen-Shannon (JS) distance metric:

$$JS\_distance(p,q) = \sqrt{\frac{D(p\|m) + D(q\|m)}{2}} \quad (3)$$

where m is the pointwise mean of p and q and D is the Kullback-Leibler (KL) divergence [29]. The JS distance is based on the JS divergence [30]; as advantages over the KL divergence, the JS divergence is symmetric and always has a finite value within the [0, 1] range [31].

For each metadata category in our overall corpus, and for each genre category in our genre-filtered corpus, we calculate the JS distance between the frequency occurrence profiles of category values, counted over all submissions within an anomalous spike, vs. all submissions without any anomalous spike. As some categories can assume many different values (e.g. replay_gain), we only do comparisons for values that occur at least 10 times in both frequency profiles. JS distance values for the metadata comparisons are listed in Table 3, while JS distance for the genre comparisons are listed in Table 4.

As can be observed in Table 3, comparing submissions within and outside of the anomalous spikes, major distributional differences are found for used extractor software versions. These go up to the level of Essentia Git commit and build versions that were used for low-level feature extraction. In addition, we also observe distributional differences for bit_rate and codec, likely confirming earlier observations [7] that low-level feature extractors may display sensitivities with regard to different audio codecs and

Classifier	Anomalous range	Full		Genre	
		#MBIDs	#submissions	#MBIDs	#submissions
mood_acoustic, `acoustic`	[0.09, 0.10]	282,605	358,747	60,261	94,268
mood_relaxed, `relaxed`	[0.805, 0.815]	373,555	485,184	72,739	119,050
mood_electronic, `electronic`	[0.972, 0.982]	315,626	401,151	64,944	101,915
mood_sad, `sad`	[0.346, 0.362]	57,697	75,688	8,854	14,242

Table 2: Details of anomalous spike data slices used for distributional comparisons. For each classifier of interest, we indicate the classifier confidence range for which a submission was considered to be anomalous. We also list the counts of unique MBID recordings and overall submissions, both for the full corpus and our genre-filtered corpus.

	`acoustic`	`relaxed`	`electronic`	`sad`
bit_rate	.42	.32	.39	.17
codec	.34	.26	.32	.06
length	.15	.15	.15	.32
lossless	.28	.21	.27	.02
essentia_low	.61	.52	.59	.15
essentia_git_sha_low	.67	.58	.66	.23
essentia_build_sha_low	.70	.62	.69	.24

Table 3: JS distances between frequency profiles over metadata categories, for anomalous vs. non-anomalous submissions considering the four classifiers of interest. For metadata categories that are not listed, found JS distances were always 0.

	`acoustic`	`relaxed`	`electronic`	`sad`
Discogs	.12	.09	.11	.11
last.fm	.14	.12	.13	.14
tagtraum	.14	.11	.13	.14

Table 4: JS distances between frequency profiles over genre categories, for anomalous vs. non-anomalous submissions considering the four classifiers of interest.

compression rates. In contrast, Table 4 shows that JS distances are equivalent and low across genre taxonomies and types of anomalies: from this, it seems more likely that the anomalies were caused by submission extraction contexts, rather than the inclusion of anomalous data.

7. CONCLUSIONS AND FUTURE WORK

In this work, we analyzed patterns in high-level descriptor values in AcousticBrainz. As we showed, while the descriptors were successfully validated under lab conditions, they show unexpected behavior in the wild, raising questions on the extent to which they have construct validity.

The unexpected behavior could have two potential causes. First of all, **the construct underlying several high-level descriptors may be conceptually problematic by itself**. For example, the concept of genre [32], as well as its use in machine learning classification tasks [33] has been criticized by musicologists and musicians. Furthermore, within music psychology, there have been findings that sad music does not necessarily elicit sad emotions [34, 35]. Further interdisciplinary research will be needed to better understand these phenomena.

Our current analyses also accumulated evidence that **the AcousticBrainz community confronted the descriptors with audio and extraction contexts that were too different from the contexts on which classifiers originally were trained**. It should be noted that original training datasets for the classifiers were far smaller in size (several hundreds to thousands of data points) than the current scale of AcousticBrainz, and that this logically may not have managed capturing all intricacies of larger-scale, ecologically valid data. However, our analyses suggest that anomalous behavior may also be due to audio codecs, compression rates and different versions of software implementations and builds that were used during extraction, which are rarely explicitly considered and reported in evaluation setups. As for the software versions, it should further be noted that, while we focused on high-level descriptors, all found differences occurred in the extraction procedures of low-level descriptors (feature representations), while the high-level machine learning models stayed constant. Thus, low-level descriptor performance should explicitly stay in scope when studying high-level descriptors.

With this work, we wished to shed light on current challenges regarding the reproducibility and generalizability of research outcomes, and on elements of processing pipelines that are under-represented in applied machine learning and signal processing literature, yet play a critical role for the pipeline's performance [8, 36]. Inspired by literature in both psychological and software testing, we also offered several possible strategies to assess descriptor validity, even in the absence of a clear ground truth.

While we exposed several potentially problematic patterns, we explicitly do not wish for this work to be seen as a criticism of AcousticBrainz and/or Essentia. No other MIR resource or API currently offers similar levels of transparency that allow for analyses like we performed here, and we would like to explicitly thank the teams behind these initiatives for their openness. It also is this openness that will allow for us to perform further research in the near future—with more systematic testing strategies and experimental designs—towards more holistic quality assurance procedures for applied machine learning procedures in the context of humanly-interpretable signal data.

8. REFERENCES

[1] A. Schindler and R. Mayer and A. Rauber, "Facilitating Comprehensive Benchmarking Experiments on the Million Song Dataset," in *Proceedings of the 13th Conference of the International Society for Music Information Retrieval (ISMIR 2012)*, 2012.

[2] D. Bogdanov, A. Porter, J. Urbano, and H. Schreiber, "The MediaEval 2018 AcousticBrainz Genre Task: Content-based Music Genre Recognition from Multiple Sources," in *MediaEval Benchmark Workshop*, 2018.

[3] J. Serrà, A. Corral, M. Boguñá, M. Haro, and J. L. Arcos, "Measuring the Evolution of Contemporary Western Popular Music," *Scientific Reports*, vol. 2, 2012.

[4] M. Interiano, K. Kazemi, L. Wang, J. Yang, Z. Yu, and N. L. Komarova, "Musical trends and predictability of success in contemporary songs in and out of the top charts," *Royal Society Open Science*, vol. 5, no. 171274, 2018.

[5] M. Park, J. Thom, S. Mennicken, H. Cramer, and M. Macy, "Global music streaming data reveal diurnal and seasonal patterns of affective preference," *Nature Human Behaviour*, vol. 3, no. 3, pp. 230–236, 2019.

[6] E. Zangerle, R. Huber, M. Vötter, and Y.-H. Yang, "Hit Song Prediction: Leveraging Low-and High-Level Audio Features," in *Proceedings of the 20th Conference of the International Society for Music Information Retrieval (ISMIR 2019)*, 2019.

[7] J. Urbano, D. Bogdanov, P. Herrera, E. Gómez, and X. Serra, "What is the Effect of Audio Quality on the Robustness of MFCCs and Chroma Features?" in *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.

[8] B. McFee, J. W. Kim, M. Cartwright, J. Salamon, R. M. Bittner, and J. P. Bello, "Open-Source Practices for Music Signal Processing Research: Recommendations for Transparent, Sustainable, and Reproducible Audio Research," *IEEE Signal Processing Magazine*, vol. 36, 2019.

[9] B. L. Sturm, "A Simple Method to Determine if a Music Information Retrieval System is a "Horse"," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1636–1644, 2014.

[10] J. Kim, J. Urbano, C. C. S. Liem, and A. Hanjalic, "Are Nearby Neighbors Relatives? Testing Deep Music Embeddings," *Frontiers in Applied Mathematics and Statistics*, vol. 5, p. 53, 2019.

[11] A. Flexer and T. Grill, "The Problem of Limited Inter-rater Agreement in Modelling Music Similarity," *Journal of New Music Research*, vol. 45, no. 3, pp. 239–251, 2016.

[12] A. Flexer and T. Lallai, "Can we increase inter- and intra-rater agreement in modeling general music similarity?" in *Proceedings of the 20th Conference of the International Society for Music Information Retrieval (ISMIR 2019)*, 2019.

[13] H. V. Koops, W. B. de Haas, J. A. Burgoyne, J. Bransen, A. Kent-Muller, and A. Volk, "Annotator subjectivity in harmony annotations of popular music," *Journal of New Music Research*, vol. 48, no. 3, pp. 232–252, 2019.

[14] S. Balke, J. Abeßer, J. Driedger, C. Dittmar, and M. Müller., "Towards evaluating multiple predominant melody annotations in jazz recordings," in *Proceedings of the 17th Conference of the International Society for Music Information Retrieval (ISMIR 2016)*, 2016.

[15] A. Porter, D. Bogdanov, R. Kaye, R. Tsukanov, and X. Serra, "AcousticBrainz: A Community Platform for Gathering Music Information Obtained from Audio," in *Proceedings of the 16th Conference of the International Society for Music Information Retrieval (ISMIR 2015)*, 2015.

[16] D. Bogdanov, A. Porter, P. Herrera, and X. Serra, "Cross-collection evaluation for music classification tasks," in *Proceedings of the 17th Conference of the International Society for Music Information Retrieval (ISMIR 2016)*, 2016.

[17] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *Proceedings of the 12th Conference of the International Society for Music Information Retrieval (ISMIR 2011)*, 2011.

[18] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, "Essentia: An Audio Analysis Library for Music Information Retrieval," in *Proceedings of the 14th Conference of the International Society for Music Information Retrieval (ISMIR 2013)*, 2013.

[19] W. R. Shadish, T. D. Cook, and D. T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin, 2002.

[20] L. J. Cronbach and P. E. Meehl, "Construct Validity in Psychological Tests," *Psychological Bulletin*, vol. 52, p. 281–302, 1955.

[21] G. T. Smith, "On Construct Validity: Issues of Method and Measurement," *Psychological Assessment*, vol. 17, no. 4, pp. 396–408, 2005.

[22] J. Urbano, M. Schedl, and X. Serra, "Evaluation in Music Information Retrieval," *Journal of Intelligent Information Systems*, vol. 31, pp. 345–369, 2013.

[23] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic Testing: A New Approach for Generating Next Test Cases," Hong Kong University of Science and Technology, Tech. Rep., 1998.

[24] E. J. Weyuker, "On Testing Non-testable Programs," *The Computer Journal*, vol. 25, no. 4, pp. 465–470, 1982.

[25] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The Oracle Problem in Software Testing: A Survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[26] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, pp. 1161–1178, 1980.

[27] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[28] D. Bogdanov, A. Porter, H. Schreiber, J. Urbano, and S. Oramas, "The AcousticBrainz genre dataset: Multi-source, multi-level, multi-label, and large-scale," in *Proceedings of the 20th Conference of the International Society for Music Information Retrieval (ISMIR 2019)*, 2019.

[29] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[30] D. Endres and J. Schindelin, "A new metric for probability distributions," *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1858–1860, 2003.

[31] J. Lin, "Divergence measures based on the Shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[32] C. C. S. Liem, A. Rauber, T. Lidy, R. Lewis, C. Raphael, J. D. Reiss, T. Crawford, and A. Hanjalic, "Music Information Technology and Professional Stakeholder Audiences: Mind the Adoption Gap," in *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012, vol. 3.

[33] B. L. Sturm, "Classification accuracy is not enough: On the evaluation of music genre recognition systems," *Journal of Intelligent Information Systems*, vol. 41, pp. 371–406, 2013.

[34] J. K. Vuoskoski and T. Eerola, "Can sad music really make you sad? Indirect measures of affective states induced by music and autobiographical memories." *Psychology of Aesthetics, Creativity, and the Arts*, vol. 6, no. 3, pp. 204–213, 2012.

[35] A. Kawakami, K. Furukawa, K. Katahira, and K. Okanoya, "Sad music induces pleasant emotion," *Frontiers in Psychology*, vol. 4, 2013.

[36] M. F. Dacrema, S. Boglio, P. Cremonesi, and D. Jannach, "A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research," *arXiv preprint arXiv:1911.07698*, 2019.

ARTIST GENDER REPRESENTATION IN MUSIC STREAMING

Avriel Epps-Darling
Harvard University
avrielepps@g.harvard.edu

Romain Takeo Bouyer
Spotify
romaintakeo@spotify.com

Henriette Cramer
Spotify
henriette@spotify.com

ABSTRACT

This study examines gender representation in current music streaming, utilizing one of the world's largest streaming services. First, we found listeners generally stream fewer female or mixed-gender creator groups than male artists, with differences per genre. Second, while still relatively low, we found that recommendation-based streaming has a slightly higher proportion of female creators than "organic" listening (i.e., tracks that are not recommended by editors or algorithms). Third, we examined streaming data from 200,000 US users to determine the proportion of female artists in organic and recommended streams over a 28-day period and the relationship between recommended streams and users' future organic listening. The proportion of female artists in recommended streaming appears predictive of the proportion of female artists in organic streaming; these effects are moderated by gender and age. Fourth, this study also samples creators across different popularity levels, seeing more female and multi-gender groups at lower levels than in the middle tiers. However, (solo) female artists are better represented again in the superstars category, suggesting influence of selected superstars and genres. We conclude by discussing potential avenues in algorithmic auditing.

1. INTRODUCTION

Music has long presented barriers to success for underrepresented groups, including female artists [4, 22]. While gender inequities existed before the advent of streaming, the 7.4 billion dollar streaming industry [1] operates at a scale that merits critical examination. In particular, we examine whether music streaming presents similar imbalances or instead presents opportunities for greater gender parity. Music streaming services recommend tracks using a combination of human editorial and algorithmic decisions. Services learn users' musical taste and make predictions on tracks that may suit a given users' current activity, mood, or curiosity for new artists. Such recommendations

---

[1] http://wwwriaa.com/wp-content/uploads/2019/02/RIAA-2018-Year-End-Music-Industry-Revenue-Report.pdf

may amplify or counter existing inequities. Research on music consumption suggests that online consumers of music tend to have more diverse listening than consumers who primarily discover their music through radio and TV [10] and personalized recommendations can introduce users to new and potentially more diverse content [17]. However, prior research has also indicated that personalized recommendations may funnel consumers into narrower content [7, 18]. Such conflicting results suggest that the impact of algorithmic recommendation may depend on specific data, models used, and the context in which they are applied. This study seeks to understand how one streaming service's recommendations reflect existing gender representation in the music industry as well as different approaches to making such assessments.

2. BACKGROUND & LITERATURE

2.1 Gender Representation in the Music Industry

Women have historically been underrepresented in the music industry relative to society as a whole. This is reflected in industry charts and awards. Smith et al, [22] found that 10.4% of Grammy nominees between 2013 and 2019 were female. In 2018's Hot 100 year-end Billboard Chart, 17.1% were female; a m:f ratio of 4.8 to one, lowest of the 7 years prior evaluated.

Women throughout history have been music role models and artists [11, 20], but barriers have limited their proportional representation in industry. Historically, women for example were not always allowed to be hired as musicians, or to play certain instruments at all [4]. Contemporary barriers reported by female artists include discounting of their abilities, lack of connections, unwanted stereotyping or sexualization, uncomfortable studio cultures, financial instability and lack of female role models [22].

Artists have to contend with expectations of genres and subcultures, including gendered trends and themes. In a content analysis of US music videos, Emerson [9] describes how black female artists appear to navigate both empowerment themes and aesthetic and social expectations. More specific genre (sub)cultures can play a role as well. In country music, Watson [24] found a decline between 1996 and 2016 in individual female artists played on country radio, and cites explicitly asserted beliefs by decision makers that playing more female artists would lead to less advertising revenue. In electronic dance music, Gavanas & Reitsamer [12] report female DJs navigating an environment where male entrepreneurs and DJs are much more visible and networked. In rock, the "groupie"

description is a distinctly lower status label almost exclusively applied to women, even sometimes to those working in the industry, reinforcing a consumer rather than creative or production role [16].

Many music scenes however also explicitly provide space to explore non-conforming identities and roles [2, 13, 25]. Previous research has hypothesized that increased access to music afforded by the Internet disrupts barriers. For example, Epps & Dixon's [10] study on the consumption of hip hop music suggests that listeners who find the majority of their rap online consume more diverse tracks than listeners who consume most of their rap music on traditional media outlets. The same study also found that hip hop music on the Billboard charts (before streaming was included in these rankings) was less diverse on measures of lyrical themes, artist gender, and artist race than hip hop music shared online. While this work suggests that an increase in choice afforded by the internet is related to an increase in diversity of music consumption, few studies have been extended to evaluate the impact of music recommendation systems.

2.2 Bias in Algorithmic Recommendation Systems

A growing body of research examines biases in algorithmic systems. Often, these biases are extensions of biases that exist in broader society [6,23]. We might expect to see biases in algorithmic systems that mirror those in the music industry. However, impact also depends on objectives set, which can include a variety of metrics designed to broaden content consumption and diversity [15]. Choices within algorithmic models, which features to include, types of models used, also influence their output [5]. For gender and book recommendations, Ekstrand et al. [8], for example, found that when using skewed input, most collaborative filtering algorithms reflected user's profile tendencies, but that this effect was substantially stronger for implicit feedback recommendations (behavioral, e.g. clicks or reading) than explicit feedback (e.g. ratings).

In industry practice, a multitude of models build on top of each other. Algorithms designed to recommend music on streaming platforms utilize predetermined content and meta-data traits (e.g., tempo, genre, artist, historical period, etc.), as well as collaborative filtering techniques based on listening behavior by similar users. Some 'biases' are by design, such as when recommending only new releases on a new artists playlist, a playlist focused on women in rock only featuring women, or playlists focused on mood that may not include genres less suitable to that context. Other biases may be unintended, but can still be examined. For music, Aguiar et al. [1] examined gender imbalances on Spotify. They had insufficient evidence to conclude that female underrepresentation in streams was due to platform bias, they found pro-female bias in some playlists, and asserted a potential supply imbalance.

However, their work raises questions on the availability of baselines of streaming as a whole, the comparison at scale of programmed vs. non-programmed streams, and the influence of both streaming services and artist supply into these services. In this article, we build on their work by analyzing a larger data set of streams, providing insight into streaming behavior, as well as a hand-labeled sample of 'supply' in the hopes to provides the research community with baselines for further research.

We show that recommendation-based streaming has a slightly higher proportion of female artists than "organic", non-programmed listening. However, listeners generally stream fewer female or mixed-gender creator groups than male artists, making the proportion as a whole much lower than representation of women in society. We identify differences per genre that merit further investigation. Third, we find that indeed there is a relationship between recommended streams and users' future organic listening, moderated by gender and age. Fourth, we find that while supply of starting female artists plays a role, differing patterns of female representation at different popularity levels suggest differing investment patterns and again differences between genres. We conclude by discussing potential avenues in algorithmic auditing.

3. ORGANIC VS. PROGRAMMED STREAMS

Streams can be either programmed or non-programmed. Programmed streams originate from recommendations such as in algorithmic or editorial playlists, whereas non-programmed, 'organic' streams are explicitly asked-for through user-initiated actions such as search, or picking a playlist from a user's personal library.

Programmed streams include editorial playlists (curated by professional editors), and algorithmic playlists (those that are primarily created by machine learning models). Note that in practice, the latter distinction can be hard to make; editors manually selecting tracks for playlists still have algorithmic tools at their disposal. Similarly, algorithmic playlists are still human-designed with a specific purpose in mind (e.g. to discover new music), or may combine approaches using both editorial pools and algorithmic ranking, as discussed in Bonini & Gandini [3].

4. RESEARCH QUESTIONS

This study addresses the following questions through an analysis of data from a global music streaming service:

- RQ1: What is the current distribution of artist gender in music streaming, and how do recommended and user-initiated streams differ?

- RQ2: Does the proportion of female artist streams in recommended playlists predict the proportion of organic streams?

- RQ3: How do these results relate to gender distribution in creator 'supply' at different popularity levels?

For the first, we analyze a sample of a month of streams from a popular streaming service, and existing commercially available gender metadata. For the second, we take a sample of users, and investigate the relationship between their programmed and non-programmed (self-selected) streams, and the impact of listener characteristics. For the third, to counter the inherent popularity

biases in our large-scale (meta)data, we take a random sample of creators at different levels of popularity, hand-label these creators and investigate the 'supply' proportion of female, male, non-binary and multi-gender artists and groups. Each of these will be discussed in their own section below.

5. REPRESENTATION IN STREAMING PATTERNS (RQ1)

We start this study by comparing programmed and non-programmed streams, and understanding the proportion of female-artist streams within this setting.

5.1 Methods

To obtain a baseline understanding of the artist-gender makeup of streaming, we present a sample containing 30 days of streams starting in early April 2020, from Spotify, a music streaming service with Millions of worldwide users [2] .

For purposes of this study, a *stream* is defined as a 30 second or longer play of a *track recording*. This time threshold minimizes the impact of skipped songs on our analysis. Note that some tracks may be streamed never, while others may get millions of streams. This means that popular artists and their streams will have a large impact on the analysis presented here, which is why we investigate representation at different levels of popularity in the section addressing RQ3.
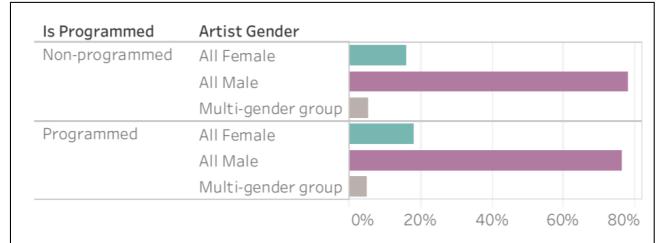
For our analyses of streams, artist characteristics are supplied from commercially available metadata. Our focus here is on the main performing artist, not potential featured artists, songwriters, composers or producers. This data set has coverage on gender for around 86% of all streams sampled. For each artist entity in the data set, a gender entry states whether they are female, male, a mixed multi-gender creator group (e.g. a band, duo), or unknown/other. The latter covers both non-binary as well as unknown gender artists, meaning that we cannot distinguish between other gender identities in our at-scale analysis than male, female and multi-gender groups. This means this analysis is not inclusive to non-binary gender artists, even though binary conceptualization of gender is an inaccurate conceptualization [14, 21].

5.2 Results
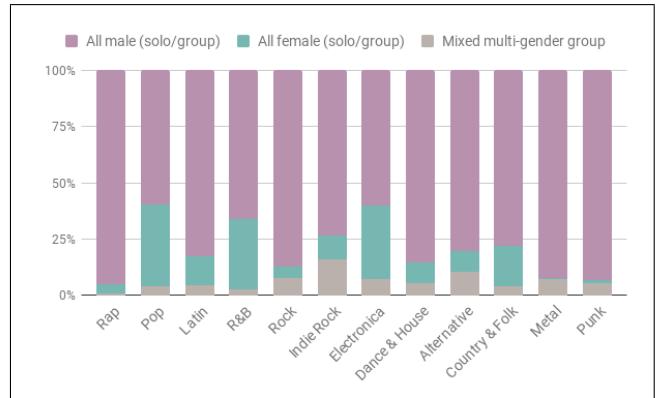
In the 30 days analyzed, for all streams where gender information is available, around 1 in 5 have a female performing artist associated with them, see Fig 1. Of particular relevance to RQ1 was the comparison between programmed and organic streams. Female artists receive slightly more streams in programmed content than in organic streams (Pearson's $\chi^2 = 8e07, df = 2, p < 2.2e - 16$, see Fig 1).

Streams with either a female artist or multi-gender group comprised respectively 21.75% of non-programmed (e.g. user search or library) streams, and 23.55% for programmed (recommended) streams.

Figure 1. 'Programmed' vs. 'organic' streams, stream %. As discussed in section 5.1, non-binary gender not included due to data limitations.

Figure 2. Proportions of streams for most popular genre groupings (cut-off for inclusion: 2% of streaming). Combined programmed and non-programmed streams.

We found considerable differences between genre streams (Fig 2), suggesting that subcultures can impact representation. For example, 95% of rap/hip hop streams were associated with male-only performing artists. For pop, around 40% of performers included a female artist or at least one female group member. For metal, all-female performer streams were rare (0.7%), with 7.0% mixed-gender groups.

This suggests the need for not only industry-wide, genre-agnostic follow up studies, but also genre-specific deep-dives that take into account sub-cultural processes, networks and industry structures.

6. PREDICTING ORGANIC CONSUMPTION (RQ2)

To better understand the relationship between programmed and non-programmed 'organic' listening, and the potential influence of recommendations, we then conducted an analysis centering user-level listening. This analysis uses a random US sample, and investigates whether the proportion of female artist streams in recommended playlists predict the proportion of organic streams listened to with various controls.

6.1 Methods

6.1.1 Sample

We limited our sample to US users who had a paid subscription and were between the self-reported ages of 13

and 90. We limited our sample to US users under the assertion that gender preferences in musical taste (whether explicit or implicit) are culturally dependent, and thus a cross-national analysis would present additional complexities beyond the scope of this project. Because of this, we decided to focus on US listeners because it was the largest population of users in our data set, and the market with which our research team was most familiar. One obvious alternative to this choice would be an international stratified sample; we hope future research will consider this approach. Gender was also self reported by users. From this larger population, we randomly sampled 200,000 active users for whom we had organic, editorial, and algorithmic streaming data in a 28-day period ending on September 30th, 2018. We chose a fall month to avoid seasonal and holiday-based differences in listening patterns, which are most pronounced at the end of the calendar year [19]. We allocated 60% of these data for training ($n = 120,000$), 20% for testing ($n = 40,000$), and 20% for validation ($n = 40,000$).

User characteristics, including gender and age, are gathered through the sampled streaming service's on-boarding process, during which new users set up their profile. Within our total sample ($N = 200,000$), 46% of listeners were female and 0.06% identified as non-binary. We also grouped participants into age categories.

For the purpose of this research, a user's *"top genre"* is defined as the highest-ranking genre when dividing their total streams by the number of streams in each genre. In total, there were 30 top genre categories. For 51% of listeners, pop was the most listened-to genre. Rock was the second, with a distant 15.6% of listeners.

6.1.2 Statistical Analysis

Our outcome variable of interest was the proportion of female artists in tracks streamed organically. In first assessing the data, we modeled the proportion of female artists in organically streamed tracks using ordinary least squares regression. We then introduced controls shown to be important in the larger literature. Finally, we included interaction terms between all main effects features and control features in the OLS regression. The final model equation is:

$$\hat{Y} = \beta_1 X + \beta_2 Z + \beta_3 X \cdot Z + \epsilon \quad (1)$$

In this equation, $\hat{Y}$ is the predicted proportion of female artists streamed organically over a 28-day period, X represents the matrix of the main effects features plus the constant, Z represents the matrix of control variables, and ϵ is the error term.

In the end, we retained five dependent variables and the interactions between them, given that they were theoretically significant, had a reasonable amount of predictive power, and showed no collinearity with other variables. Table 1 summarizes the features selected for our final analysis without their interaction terms.

Thereafter, we applied several basis functions to see whether the model could be improved by including higher order polynomial features. The best fitting basis function was $\phi(X) = (x_1^1, x_1^2, ..., x_1^6, ...x_D^1, x_D^2, ..., x_D^6)$ ($\alpha =$

0.01, R-Squared = .40). However, the small increase in R-squared statistic did not seem to justify increased model complexity and decreased interpretability. Five-fold cross validation was used to ensure the final model was not overfit to the data.
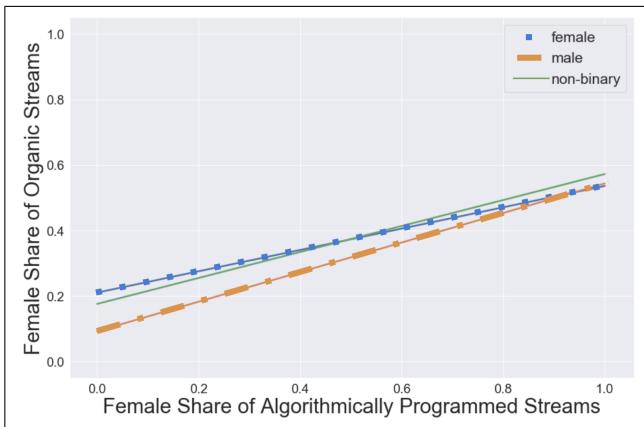
Variable	Description
Outcome Variable	
organic	Share of female artists streamed organically for longer than 30 seconds during a 28-day period
Main Effects Features	
algo	Share of female artists streamed via algorithmically programmed playlists for longer than 30 seconds during a 28-day period
editor	Share of female artists streamed via editor programmed playlists for longer than 30 seconds during a 28-day period
Control Features	
gender	Gender of user. One-hot encoded into female, male, and non-binary
age	Age of user. Self-reported age bucketed and one-hot encoded as categorical variables: 0-17, 18-24, 25-29, 30-34, 35-44, 45-54, and 55+
top genre	User's most listened to genre. One-hot encoded variable categorizes a user's most listened to genre. Such as afropop, atmospheric, blues, brazil, children, christian, classical, comedy, country, edm, hip hop, etc.

Table 1. Feature descriptions for RQ2

6.2 Results

For addressing RQ2, we used a randomly sampled dataset of US users and their streaming behavior. We iteratively built three models [3] to predict the effect of female artist

---

[3] Ideally, we would include all variables and their coefficients for the iterative models, considering space limitations we have limited the description to the final and best fitting model.

Figure 3. Plotting proportion of female artists in algorithmically recommended content on organic streaming of female artists when controlling for user age, gender, and top genre. Line style indicates moderation by listener gender.

share in algorithmic and editor programmed content on organic streaming of female artists. Additionally, we controlled for user age, gender, and top genre, and moderated by user age and gender. This final model's results are discussed below.

6.2.1 User Demographic Differences in Listening

Chi-squared tests revealed that there are no statistically significant differences between male and female users with regard to the share of female artists they stream (Pearson's $\chi^2 = 0.04, p = 0.98$). Additionally, Chi-squared tests revealed that there were no statistically significant differences between users of different age categories with regard to the female artist stream share (Pearson's $\chi^2 = 0.01, p = 1.0$). With this, we conclude that gender and age are independent of female artist stream share.

6.2.2 Linear Regression

In fitting our model, our null hypotheses were that the (1) there is no effect of programmed female artist share on organic female artist share and (2) effect of programmed female artist share on organic female artist share is not moderated by any of our demographic variables. With regard to listener gender, we found that the estimated effects for men were larger than corresponding effects for women ($\beta_{algoXmale} = .076, p < .001$). That is, compared to the women in our sample, men who streamed more female artists in algorithmically-programmed playlists were also more likely to listen to female artists organically. Figure 3 illustrates the moderated effects of algorithmic female share on organic female share by gender.

There are similar, yet weaker, associations for the interaction between gender and editor programmed content ($\beta_{editorXmale} = 0.014, p < .001$), as well as age and algorithmically programmed content. Notably, we found that the estimated effects for 18-24 year-olds ($\beta_{algoX18-24} = .026, p < .001$) and 25-29 year-olds ($\beta_{algoX25-29} = .047, p < .001$) were larger than corresponding effects for 45-54 ($\beta_{algoX45-54} = -.054, p < .001$) and 55+ year-olds ($\beta_{algoX45-54} = -.060, p < .001$). That is, compared

to the 18-29 year-olds in our sample, those over the age of 45 who streamed more female artists in algorithmically-programmed playlists were less likely to listen to female artists organically. The interaction terms for age and algorithmically programmed content for 30-44 year-olds were not statistically significant.

Further, the strength of the model, as evaluated with the R-squared (r-squared = .374) and Root Mean Square Error (rmse =.152) statistics, was moderate. When evaluating this model's fit using the test set ($n = 40,000$), we found the r-squared statistic of the validation set was .361, meaning the model was not overfit to the training set and was the best performing model we built.

While main effects are often not interpretable in the presence of an interaction term, we can relax this guideline in this model because both features are captured by dichotomous variables where 0 is a meaningful value and within the range of the variable. For example, where a listener self-identifies as a woman, the interaction term is 0. In any cases when the interaction term is equal to 0, we can interpret the main effects. However, additional post-hoc tests were needed to conclude if the difference we have observed is, in fact, statistically significant, and can be inferred at the population level. When we conducted a GLH test of their joint population equality ($F(1, 59) = 1285, p < .001$), we found that we could reject this null hypothesis.

We conclude that we have sufficient evidence that there is a moderate, positive relationship between the proportion of female artists streamed on programmed playlists and the proportion of female artists listened to organically. Additionally, this relationship is moderated by both user gender identity and age in the population.
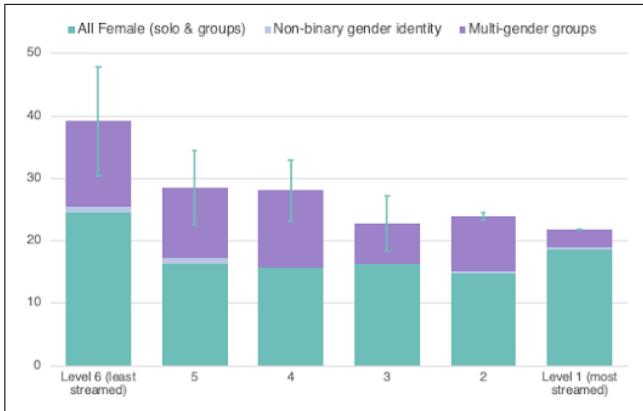
7. SUPPLY SIDE ANALYSIS (RQ3)

Large-scale analyses may offer insight in the proportion of streams that go to female artists or multi-gender groups being lower than male artists, but do not provide insight whether this reflects the 'supply' of female creators and multi-gender groups.

There is a long tail of less popular artists for whom data is scarce. Self-identification at this scale is not feasible for all artists who are streamed, not in the least for those deceased or without direct service access. This means that, for example, playlists focused on discovery of new artists, or those highlighting historic artists who are less well-known will have less complete, and potentially less accurate, associated metadata. To further investigate the presence, or supply, of female creators at different levels of popularity, we followed up with a manual sample across a wider range of creators.

7.1 Method

For our analyses of creator supply, we randomly sampled artists from six different levels of popularity. This, in an effort to reflect a spectrum of the artist community, from early projects to global superstars. Levels of popularity are defined as such: artists in the first level have 10 times more

Figure 4. Percentage of Female, non-binary and multi-gender group 'supply' from least to most popular artists. Error bars: confidence levels of total (female + non-binary + multi-gender) % due to sample size vs. large (Millions) creator population at lower popularity levels

streams than the ones in the second level, who have 10 times more streams than the next, and so on. These were sampled in Feb 2020, and based on streams within the last 90 days. A professional team of data curators labeled 1330 creators in a similar manner to [22] (who manually sampled 800 chart entries). We here focused on a wider sample beyond charts, as well as including non-binary artists and multi-gender groups. It is noteworthy that information could not be found for at least 300 more creators, even by the expert data curation team.

7.2 Results

Representation of the aggregate of female + non-binary + multi-gender groups appears to differ at different levels of popularity (χ^2 = 12.865, df = 5, p-value = 0.02468). At entry-level, female representation is higher than at the middle levels, where it goes down slightly (see Figure 4). However, there is an uptick of female artists *better* in the superstars category, while less multi-gender groups are present. This suggests success of selected superstars, and influence of popular genres with higher female representation such as pop and R&B.

Even though more data collection would be necessary at lower popularity levels to get to results with higher confidence levels, we do now have a clear indication that both supply and demand matter. This suggests that the research community and services should address representation in streams overall, but that we as a community should especially also pay attention to how certain artists climb -or not- in popularity across platforms, and what factors lead to that climb.

8. DISCUSSION & CONCLUSION

In summary, this study resulted in several key findings. First, we found listeners generally stream fewer female or mixed creator groups than male artists. Second, we found that recommendation-based streaming has a slightly higher proportion of female creators than organic listening, but

this proportion is still relatively low. Third, we found that gender and age of listener are independent of female artist stream share. Fourth, higher proportions of female artists in recommended streaming is predictive of higher proportions of female artists in organic streaming; these effects are moderated by gender and age. Younger age groups exhibited larger effect sizes, which may indicate that younger listeners are more open to taking (new) recommendations, or potentially more influenced by them. An alternative explanation may be that outside factors, such as terrestrial radio exposure, may be more salient for groups with smaller effect sizes. Future research should investigate the role of age, gender, and other identity markers in more depth. Finally, we find that in lower popularity levels, more multi-gender groups and more female creators appear to exist than in the middle - while at the top level (solo) female artists appear more present again. We have also highlighted the influence of hits on high-level stream numbers, as well as genre.

It is noteworthy that while examining gender representation is important, gender labeling in itself can be problematic. Performing labeling without self-identification can cause errors, and demographic data collection in itself presents significant risks. This results in a dilemma between inclusive representation vs. data minimization. In addition, some data ambiguity will always remain. People's expressed gender identities are not necessarily static; artists may come out as non-binary mid-career. Challenges also especially apply for historical as well as international art, and large collectives. Backing bands may or may not be taken into account in credits, orchestras and bands change and add or remove members. Information is scarce for lesser known artists, may be in other languages or terms than data curation or research teams may understand. Thus, striving for comparisons and repeated sampling rather than exact numbers and 'completeness' may be more productive tasks.

In this study, we primarily looked at streaming outcomes in aggregate, rather than who is 'shown' as a recommendation in a specific product context. Results may be skewed by top-level streaming outcomes, and higher popularity genres such as pop which have higher female representation than other genres. Although the results in this study are not causal, they do suggest that further work on the ability of content recommendations to diversify user listening habits are warranted. We primarily discussed descriptive baselines; future studies should explore alternative models and sampling approaches, potentially consider causal inference methods that do not require experimentation, or experimental designs that thoughtfully contend with the ethical concerns of manipulating user experiences on a commercial platform. Future work should also study how gender intersects with genre and subculture, as well as other factors such as race/ethnicity, locale and congruence with existing cultural expectations.

We conclude that there are barriers to entry, and to climbing to the top, but that streaming services may be able to challenge structural inequities by spotlighting underrepresented artists in their recommendations.

9. REFERENCES

[1] Luis Aguiar Wicht, Joel Waldfogel, and Sarah Waldfogel. Playlisting favorites: Is spotify gender-biased? Technical report, Joint Research Centre (Seville site), 2018.

[2] Andy Bennett. *Music, space and place: popular music and cultural identity*. Routledge, 2017.

[3] Tiziano Bonini and Alessandro Gandini. "first week is editorial, second week is algorithmic": Platform gatekeepers and the platformization of music curation. *Social Media+ Society*, 5(4):2056305119880006, 2019.

[4] Jane M Bowers and Judith Tick. *Women making music: the Western art tradition, 1150-1950*. University of Illinois Press, 1987.

[5] Engin Bozdag. Bias in algorithmic filtering and personalization. *Ethics and information technology*, 15(3):209–227, 2013.

[6] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018.

[7] Jörg Claussen, Christian Peukert, and Ananya Sen. The editor vs. the algorithm: Targeting, data and externalities in online news. *Data and Externalities in Online News (June 5, 2019)*, 2019.

[8] Michael D Ekstrand, Mucun Tian, Mohammed R Imran Kazi, Hoda Mehrpouyan, and Daniel Kluver. Exploring author gender in book rating and recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 242–250, 2018.

[9] Rana A Emerson. "Where my girls at?"Negotiating black womanhood in music videos. *Gender & Society*, 16(1):115–135, 2002.

[10] Avriel C Epps and Travis L Dixon. A comparative content analysis of anti-and prosocial rap lyrical themes found on traditional and new media outlets. *Journal of Broadcasting & Electronic Media*, 61(2):467–498, 2017.

[11] Leslie Gaston-Bird. *Women in Audio*. Routledge, 2019.

[12] Anna Gavanas and Rosa Reitsamer. DJ technologies, social networks and gendered trajectories in European DJ cultures. *DJ culture in the mix: power, technology and social change in electronic dance music*, pages 51–78, 2013.

[13] Paul Hodkinson. *Goth. Identity, style and subculture*. Berg Publishers, 2002.

[14] Os Keyes. The misgendering machines: Trans/HCI implications of automatic gender recognition. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–22, 2018.

[15] Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-Based Systems*, 123:154–162, 2017.

[16] Gretchen Larsen. 'it's a man's man's man's world': Music groupies and the othering of women in the world of rock. *Organization*, 24(3):397–417, 2017.

[17] Judith Möller, Damian Trilling, Natali Helberger, and Bram van Es. Do not blame it on the algorithm: an empirical assessment of multiple recommender systems and their impact on content diversity. *Information, Communication & Society*, 21(7):959–977, 2018.

[18] Eli Pariser. *The filter bubble: How the new personalized web is changing what we read and how we think*. Penguin, 2011.

[19] Minsu Park, Jennifer Thom, Sarah Mennicken, Henriette Cramer, and Michael Macy. Global music streaming data reveal diurnal and seasonal patterns of affective preference. *Nature Human Behaviour*, 3(3):230–236, 2019.

[20] Karin Pendle. *Women & music: a history*. Indiana University Press, 2001.

[21] Ari Schlesinger, W Keith Edwards, and Rebecca E Grinter. Intersectional hci: Engaging identity through gender, race, and class. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5412–5427, 2017.

[22] Stacy L Smith, Marc Choueiti, and Katherine Pieper. Inclusion in the recording studio? Gender and race/ethnicity of artists, songwriters and producers across 600 popular songs from 2012-2017. *Annenberg Inclusion Initiative*, 2018.

[23] Latanya Sweeney. Discrimination in online ad delivery. *Queue*, 11(3):10–29, 2013.

[24] Jada Watson. Gender on the Billboard hot country songs chart, 1996–2016. *Popular Music and Society*, 42(5):538–560, 2019.

[25] Sheila Whiteley. *Women and popular music: Sexuality, identity and subjectivity*. Routledge, 2013.

DATA CLEANSING WITH CONTRASTIVE LEARNING FOR VOCAL NOTE EVENT ANNOTATIONS

Gabriel Meseguer-Brocal[1] **Rachel Bittner**[2] **Simon Durand**[2] **Brian Brost**[2]

[1] STMS UMR9912, Ircam/CNRS/SU, Paris. [2] Spotify, USA.

`gabriel.meseguerbrocal@ircam.fr, {rachelbittner, durand, brianbrost}@spotify.com`

ABSTRACT

Data cleansing is a well studied strategy for cleaning erroneous labels in datasets, which has not yet been widely adopted in Music Information Retrieval. Previously proposed data cleansing models do not consider structured (e.g. time varying) labels, such as those common to music data. We propose a novel data cleansing model for time-varying, structured labels which exploits the local structure of the labels, and demonstrate its usefulness for vocal note event annotations in music. We frame the problem as an instance of contrastive learning, where we train a model to predict if an audio-annotation pair is a match or not. We generate training data for this model by automatically deforming known correct annotations to form incorrect annotations. We demonstrate that the accuracy of a transcription model improves greatly when trained using our proposed strategy compared with the accuracy when trained using the original dataset. Additionally we use our model to estimate the annotation error rates in the DALI dataset, and highlight other potential uses for this type of model.

1. INTRODUCTION

Labeled data is necessary for training and evaluating supervised models, but the process of creating labeled data is often error prone. Labels may be created by human experts, by multiple human non-experts (e.g. via crowd sourcing), semi-automatically, or fully automatically. For many problem settings, even in the best case scenario where data is labeled manually by experts, labels will almost inevitably have inconsistencies and errors. The presence of label noise is problematic both for training and for evaluation [1]. During training, it can cause models to converge slower and to require much more data, or overfit the noise thus resulting in poor generalization. During evaluation, it can lead to unreliable metrics with artificially low scores for models with good generalization and artificially high scores for models which overfit noisy data. This issue is particularly timely and relevant for the music informa-

tion retrieval (MIR) community as recent datasets such as LakhMIDI [2], DALI [3] or the Free Music Archive [4] take advantage of large music collections accessible from the Internet but often rely on noisy annotations. Additionally, many common annotation tasks are particularly costly as they have to be aligned in time and require a participant with musical expertise to be done accurately.

Data cleansing is a well studied technique in the machine learning community for mitigating the effects of label noise, with a focus on improving model generalization when training on noisy datasets [1]. A common and effective approach is to build a model to identify and discard data points with incorrect labels. Most methods taking this approach do not assume any structure or correlation between different labels. This is appropriate for many common tasks, such as image recognition. However, in music, labels are often highly structured and time-varying, and the label noise is not random. For example, musical note-annotations, which we focus on in this work, are locally stable in time and follow certain common patterns. Typical noise for note events include incorrect pitch values, shifted start times, and incorrect durations, among others.

Our contributions are as follows. We propose a novel contrastive learning-based [5, 6] data cleansing model which can exploit sequential dependencies between labels to predict incorrectly labeled time-frames trained using likely correct labels pairs as positive examples and and local deformations of correct pairs as negative examples. We focus our experiments on a model for detecting errors in vocal note event annotations, which we believe extends easily to other types of music transcription labels. We then demonstrate the usefulness of this data cleansing approach by training a transcription model on the original and cleaned versions of the DALI [3] dataset. Further, we use the model to estimate the error rates in the DALI dataset, and highlight other potential uses for this type of model, including for reducing manual labeling efforts. Finally, the code used in this work, including the pre-trained error detection model, is made freely available [1] along with the outputs of the model for the DALI dataset [2].

2. BACKGROUND AND RELATED WORK

We first introduce prior work on learning in the presence of label noise, and conclude by summarizing the work rel-

---

[1] https://github.com/gabolsgabs/contrastive-data-cleansing
[2] https://zenodo.org/record/3576083

event to our example use case of note event annotations.

2.1 Classification in the presence of label noise

We consider the problem of training a classifier on a dataset where some of the labels are incorrect. One class of solutions attempts to solve the problem by mitigating the effect of label noise, rather than modifying the data used for training. This is commonly done by modifying the loss function to directly model the distribution of label noise, for example, by creating a noise-robust loss with an additional softmax layer to predict correct labels during training [7], or with a generalized cross-entropy that discards predictions that are not confident enough while training, looking at convergence time and test accuracy [8], or by inferring the probability of each class being corrupted into another [9]. However, these approaches are restricted to specific types of loss functions or make restrictive assumptions about the statistical distribution of noise.

Data cleansing [10] and **outlier detection** [11] based approaches aim to identify the correctly labeled data points and train only on them. The vast majority of data cleansing methods are model prediction-based [1]. In their simplest form, model prediction-based methods train a model to remove items from the dataset where the label predicted by the model disagrees with the dataset label. Note that in many cases, the choice of model for data cleansing is often the same as the choice of model used after data cleansing.

These data cleansing approaches have several advantages over learning directly with noisy labels. First, the filtering does not depend on the downstream inference task, thus a cleansing method can be applied to filter data used to train many different models. Second, we can train less complex downstream models, as they do not need to account for label noise. To the best of our knowledge however, prior data cleansing approaches do not exploit the structured nature of labels often seen in MIR tasks.

In addition to developing data cleansing methods, or learning methods that are robust to label noise, there are a variety of less closely related paradigms for dealing with data quality issues. In **semi-supervised learning** reliably labeled data is combined with a large amount of unlabeled data [12]. In **weakly supervised learning** [13–15] low-quality or insufficiently granular labels are used to infer the desired target information. Finally, **active learning** estimates the most valuable unlabeled points for which to solicit additional labels [16, 17].
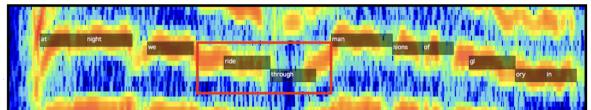
2.2 Note Event Annotations

Automatic music transcription, one of the core tasks in MIR, involves converting acoustic music signals into some form of music notation [18]. *Musical note events* are a common intermediate representation, where a note event consists of a start time, end time and pitch. They are useful for a number of applications that bridge between the audio and symbolic domain, including symbolic music generation and melodic similarity. Instruments such as the piano produce relatively well-defined note events, where

each key press defines the start of a note. Other instruments, such as the singing voice, produce more abstract note events, where the time boundaries are often related with changes in lyrics or simply as a function of our perception [19], and are therefore harder to annotate correctly.

Datasets providing note event annotations are created in a variety of ways, all of which are error prone. Notes may be manually labeled by music experts, requiring the annotator to specify the start time, end time and pitch of every note event manually, aided by software such as Tony [20]. MIDI files from the Internet can in some cases be aligned automatically as in the LakhMIDI [2] and DALI [3] datasets, with varying degrees of accuracy and completeness. Note data has also been collected automatically using instruments which "record" notes while being played, such as a Disklavier piano in the MAPS [21] and MAESTRO [22] datasets, or a hexaphonic guitar in the GuitarSet dataset [23]. Data collected in this way is typically quite accurate, but may suffer from global alignment issues [22] and can only be achieved for these special types of instruments. Another approach is to play a MIDI keyboard in time with a musical recording, and use the played MIDI events as note annotations [24] but this requires a highly skilled player to create accurate annotations.

Figure 1 shows an example of correct and incorrect note annotations in the DALI dataset. The types of errors produced by the previous methods can vary. A single note can be imprecise in time, resulting in an incorrect start time or duration, or the pitch value can be annotated wrong. Additionally, notes can be annotated where there are no actual notes in the audio, and conversely, notes in the audio can be missed all together. Systematic errors include global shifts and stretches in time and shifts in key/octave. Local errors are difficult to detect, and systematic errors can cause every note event to be wrong in some way. At the individual time-frame level, notes with incorrect start/end times will have errors at the beginning/ending frames, but can still be correct in the central frames.

Figure 1. Example of two incorrect note annotations from the DALI dataset, outlined in red, figure from [25].

3. DATA CLEANSING FOR NOTE EVENTS

Given an input space $\mathcal{X}$ and a label space $\mathcal{Y}$, we propose a model prediction based approach, but rather than training a classifier $h : \mathcal{X} \to \mathcal{Y}$, we directly train a model $g : (\mathcal{X} \times \mathcal{Y}) \to [0, 1]$ which approximates the probability that the label is incorrect. We denote this probability by $g(x, \hat{y})$.

Note that this is mathematically equivalent in the ideal case to the previous model prediction based approaches: Given a perfect estimator h which always predicts a correct label y, $g(x, y) = \mathbb{1}_{h(x) \neq y}$ where $\mathbb{1}$ is the indicator

function. However, for complex classification tasks with high numbers of classes and structured labels, modeling h can be much more complex than modeling g. For instance, consider the complexity of a system for automatic speech recognition, versus the complexity needed to estimate if a predicted word-speech pair is incorrect. Intuitively, you don't need to know the right answer to know if something is right or wrong.

This idea is similar to the "look listen and learn" [26] concept of predicting the "correspondence" between video frames and short audio clips – two types of structured data. It is also similar to CleanNet [27], where a dedicated model predicts if the label of an image is right or wrong by comparing its features with a class embedding vector. However, this approach operates on global, rather than position-dependent labels.

In our approach, we generate training data for g by directly taking pairs (x, y) from the original dataset as positive examples and creating artificial distortions of y to generate negative examples. In this section, we study the use of an estimator $g(x, \hat{y})$ for detecting local errors in noisy note-event annotations. See Figure 2 for an overview of the system.

3.1 Input Representations

As our input representation, instead of using the raw audio signal itself, we compute the Constant-Q Transform (CQT) [28] as a matrix X, where X_{ij} is a time-frequency bin. The time index i corresponds to the time stamp $r_i = \upsilon \cdot i$ where υ is a constant defining the spacing between time stamps, and the frequency index j corresponds to a frequency q_j in Hz. The CQT is a bank of filters transformation centered at geometrically spaced frequencies. We use a frequency bin resolution with 6 octaves, 1 bin per semitone, a sample rate of 22050 Hz and a hop size of 256, resulting in a time resolution of $\upsilon = 11.6$ ms. We compute the CQT from the original mixture and from the isolated vocal version derived from the mixture using a source separation technique [29]. We include the CQT of the isolated vocals to boost the information in the signal related to the singing voice, and couple it with the CQT of the mixture to include information we may have lost in the separation process.

We define the annotated label $\hat{Y}$ as a binary matrix created from the original note-event annotations. For a given track, let K be the set of note annotations, let t_k^0 and t_k^1 be the start and end time of note k in seconds, and f_k be its frequency in Hz. Then $\hat{Y}$ is defined as:

$$\hat{Y}_{ij} = \begin{cases} 1, & \text{if } t_k^0 \leq r_i \leq t_k^1, \, q_{j-1} < f_k \leq q_j, \, k \in K \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

and has the same time and frequency resolution as X.

3.2 Learning Setup

We estimate the label noise at the ***time frame*** level. Let $\ell \in L$ be an index over the set of tracks L, I^ℓ be the index of all time frames for track ℓ, and X^ℓ and $\hat{Y}^\ell$ be its CQT

and label matrices respectively. Let X_i^ℓ and $\hat{Y}_i^\ell$ indicate a time frame of X^ℓ and $\hat{Y}^\ell$ that contains all its frequency bins j, so we index the data points according to their time index only. Finally, let $X_{a:b}^\ell$ and $\hat{Y}_{a:b}^\ell$ denote the sequence of time frames of X and $\hat{Y}$ between time indices a and b.

Our goal is to identify the subset of time frames $i \in I^\ell$ which have errors in their annotation for each track in a dataset by training a binary data cleansing model. Our data cleansing model is a simple estimator that can be seen as a binary supervised classification problem that produces an error detection model. Given a datapoint centered at time index i, g predicts the likelihood that the label $\hat{Y}_i$ is wrong. Critically, we take advantage of the structured labels (i.e. the temporal context); as input to g we use $X_{a:b}$ and $Y_{a:b}$ to predict if the center frame $\hat{Y}_{(a+b)/2}$ of $\hat{Y}_{a:b}$ is incorrect. That is, we aim to learn g such that:

$$g(X_{a:b}, \hat{Y}_{a:b}) = \begin{cases} 0, & \text{if } \hat{Y}_{(a+b)/2} \text{ is correct} \\ 1, & \text{if } \hat{Y}_{(a+b)/2} \text{ is incorrect} \end{cases} \quad (2)$$

Thus, in order to evaluate if a label $\hat{Y}_i$ is correct using n frames of context, we can compute $g(X_{i-n:i+n}, \hat{Y}_{i-n:i+n})$. In the remainder of this work, we will define

$$g_n(X_i, Y_i) := g(X_{i-n:i+n}, \hat{Y}_{i-n:i+n}) \quad (3)$$

as a shorthand. In this work, we use $n = 40$.

Let

$$D = \bigcup_{\ell \in L} I^\ell \quad (4)$$
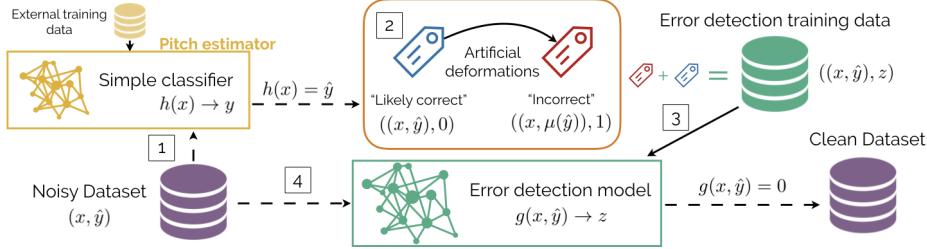
be the set of all time indices of all tracks in L. Our aim is to use g to create a filtered index F, where:

$$F = \{i \in D : g_n(X_i, \hat{Y}_i) = 0\} \quad (5)$$

3.3 Training data generation

Let z_i be a binary label indicating whether the center frame $\hat{Y}_i$ for an input/output pair $(X_{i-n:i+n}, \hat{Y}_{i-n:i+n})$ is incorrect. To train g, we need to generate examples of correct and incorrect data-label pairs $\left((X_{i-n:i+n}, \hat{Y}_{i-n:i+n}), z_i \right)$. We will again introduce a shorthand $((X_i, Y_i), z_i)$ to refer to data points of the form $\left((X_{i-n:i+n}, \hat{Y}_{i-n:i+n}), z_i \right)$.

Incorrect Data To generate "incorrect" data points (where $z_i = 1$), we can simply randomly distort any of the existing labels in the dataset by applying a modification function $\mu(\hat{Y}_i)$. These modifications $\mu(\hat{Y}_i)$ are not random but rather specific to match the characteristics of typical note-event errors (issues in the positions of the start or end times, incorrect frequencies, or the incorrect absence/presence of a note). These distorted $\hat{Y}_i$ should be contextually realistic, meaning that notes should have a realistic duration and should not overlap with the previous or next note. To do this, we modify the original note events (t_k^0, t_k^1, f_k) described in Section 3.1 by randomly shifting start and end times, frequency values, and by randomly deleting or adding notes. Given these new note events, we generate a new label matrix $\mu(\hat{Y}_i)$ as described in Eqn (1).

Figure 2. Overview of our data cleansing system. The training data for our error detection model g is generated automatically. 1- We predict the f_0 for the whole noisy dataset and compare it with the annotated label. 2- We select as "likely correct" examples those where the prediction is similar to the label, distorting them to generate the incorrect examples. 3- We train our g using this new training set. 4- We filter the noisy dataset obtaining the clean version.

Some note-level modifications may still result in the center frame being "correct" – for example, if a note begins a few frames late, the following frames will still be correct – thus, after modifying creating $\mu(\hat{Y})$, we sample examples from frames where the center frame $\mu(\hat{Y}_i) \neq \hat{Y}_i$.

Likely Correct Data Since we do not have direct access to the true label Y_i (indeed this is what we aim to discover), we first use a "simple classifier" proxy for selecting likely correct data points. There are many possible choices for this proxy – for example, if there is a manually verified subset of a dataset, it can be used directly as the set of likely correct data points – in this work we outline one specific example. We first compute the output of a pretrained f_0 estimation model $s(X_i)$ that given X_i outputs a matrix with the likelihood that each frequency bin contains a note [30]. $s(X_i)$ is trained on a different dataset than we use in the subsequent experiments and has been proven to achieve state-of-the-art results for this task [30]. $s(X_i)$ produces f_0 sequences, rather than note events, which vary much more in time than note events, so we define an agreement function in order to determine when the labels agree. $s(X_i)$ is not a perfect classifier, and while its predictions are not always correct, we have observed that when the agreement is high, $\hat{Y}_i$ is usually correct. However, we cannot use low agreement to find incorrect examples, because there are many cases with low agreement even though $\hat{Y}_i$ is correct. Therefore, we only use κ to select a subset of "likely correct" data points.

We compute both "local" (single-frame) and "patch-level" (multi-frame) agreement, and use thresholds on both to select time frames which are likely correct. The local agreement, κ_l is computed as:

$$\kappa_l(\hat{Y}_i, s(X_i)) = \max_j \left(\hat{Y}_{ij} \cdot s(X_i)_j \right) \qquad (6)$$

and the patch-level agreement κ_p is a k point moving average over time of κ_l. For the test set of g, we use very strict thresholds and select $(X_i, \hat{Y}_i)$ pair to be a likely correct if $\kappa_l > .999$ and $\kappa_p > .85$. For the training set, we use more relaxed thresholds, and select points with $0.9 < \kappa_l \leq 0.999$ and $0.7 < \kappa_p \leq 0.85$. These values have been found manually and assure the selection of good "likely correct" examples. This procedure gives us a set of positive examples in non-silent regions, but does

not take into account the silent areas. In order to select correctly labeled points from silent regions, we take additional $(X_i, \hat{Y}_i)$ points from regions with low energy in the isolated vocals and no annotations in a window of length v. In this work we use $v = 200 \ (\approx 2, 32$ s$)$.
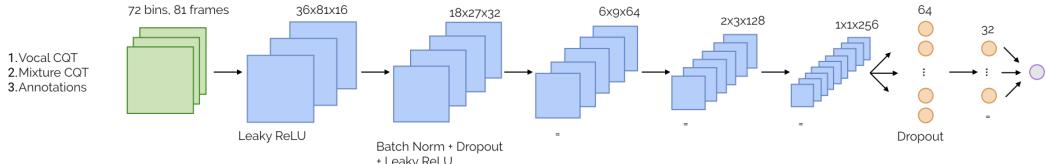
Finally, the combination of "likely correct" and "incorrect" data results in a dataset of $\{((X_i, \hat{Y}_i), z_i)\}$ with which we can train g.

3.4 Error Detection Model Architecture
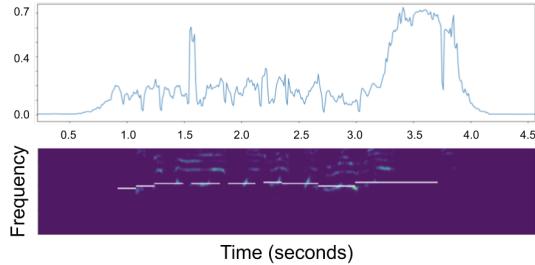
We propose a standard convolutional architecture for our error detection model $g_n(X_i, \hat{Y}_i) = z_i$, as shown in Figure 3. The input of the model is a matrix with 72 frequency bins, 81 time frames (0.94 seconds) and three channels: the two CQTs (mixture and vocals) $\{X_{i-n:i+n}\}$ and the label matrix $\{\hat{Y}_{i-n:i+n}\}$. It has five convolutional blocks with 3×3 kernels, 'same' mode convolutions, with leaky ReLU activations for the first block and batch normalization, dropout and leaky ReLU for the rest. The strides are $[(2, 1), (2, 3), (3, 3), (3, 3), (2, 3)]$ and the number of filters $[16, 32, 64, 128, 256]$ generating features maps of dimensions $(36 \times 81 \times 16)$, $(18 \times 27 \times 32)$, $(6 \times 9 \times 64)$, $(2 \times 3 \times 128)$, $(1 \times 1 \times 256)$. Then, we have two fully-connected layers with 64 and 32 neurons, a ReLU activation and dropout and a last fully-connected layer with one neuron and a sigmoid activation. The model is trained using a binary cross entropy loss function.

4. EXPERIMENTS

We test our approach using the DALI dataset, version 2 [25], which contains 7756 user-submitted vocal MIDI annotations from karaoke websites, automatically matched and aligned to polyphonic audio files. The MIDI annotations are crowd sourced and thus have highly varying quality, while the alignment is done automatically and likely to contain mistakes. This dataset is then particularly relevant for this work. The contrastive learning-based error detection model g is trained as described in Section 3.4 using the data generation method described in Section 3.3. The trained model g had a frame-level accuracy of 72.1% on the holdout set, and 76.8% on the training set.

Figure 3. Error detection model architecture.

4.1 Training with Cleaned Data

Figure 4. (Top) The output of the error detection model for a short segment. (Bottom) the corresponding CQT and annotated notes (in white). The error is high at the beginning of the fourth note because it starts late, and at end of the last note because it is too long.

Validating the performance of g is challenging, as we only have likely correct and artificially created wrong examples, but we do not have any "real" ground truth correct and incorrect examples. Thus, we first manually verified the predictions of the error detection model in many random examples (see Figure 4), and found that they appeared to be strongly correlated with errors in y_i. However, a manual perceptual evaluation of the error detection model is both infeasible and defeats the purpose of automating the process of correcting errors. Instead, we validate the usefulness of this approach by applying it to model training. In this section, we address the question: **Is the error detection model useful? How much?**

The ultimate goal of a data cleansing technique such as this one is to identify incorrect labels and remove them from the dataset in order to better train a classifier. Thus, one way to demonstrate the effectiveness of the data cleansing method is to see if training a model using the filtered dataset results in better generalization than training on the full dataset.

To validate the usefulness of $g_n(X_i, \hat{Y}_i)$ for improving training, we train the Deep Salience vocal pitch model [30] three times [3] using three different training sets. The training sets are subsets of DALI, and contain the $\hat{Y}_i$ of all the songs that have a Normalized Cross-Correlation $> .9$ [3]. This results in a training set of 1837 songs. The three sets are defined as follows:
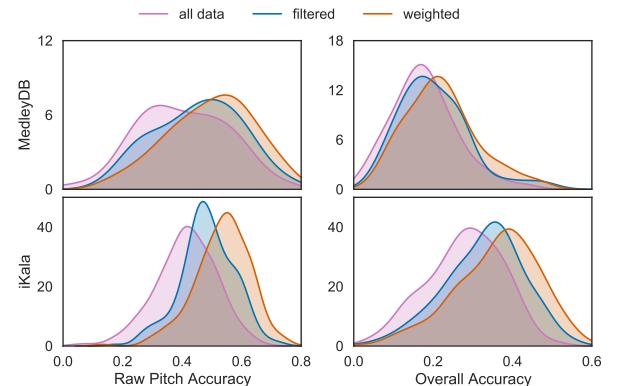
1. **All** data. Trained using all time frames, D (Eqn (4)).

2. **Filtered** data. Trained using the filtered, "non-error" time frames, F (Eqn (5)), where the output of g has

been binarized with a threshold of 0.5. With this data we tell the model to skip all the estimated noisy labels.

3. **Weighted** data. Trained using all time frames, D, but during training, the loss for each sample is weighted by $1 - g_n(X_i, \hat{Y}_i)$. This scales the contribution of each data point in the loss function according to how likely it is to be correct.
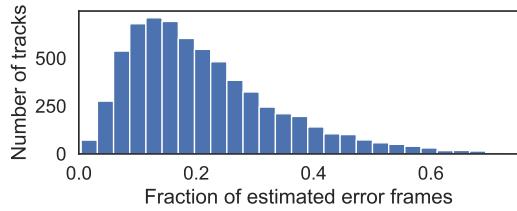
We test the performance of each model on two polyphonic music datasets that contain vocal fundamental frequency annotations: 61 full tracks with vocal annotations from MedleyDB [31] and 252 30-second excerpts from iKala [32]. We compute the the generalized [4] Overall Accuracy (OA) which measures the percentage of correctly estimated frames, and the Raw Pitch Accuracy (RPA) which measures the percentage of correctly estimated frames where a pitch is present, which are standard metrics for this task [33, 34]. The distribution of scores for each dataset and metric are shown in Figure 5.

Figure 5. Distribution of scores for the three training conditions. Each condition is plotted in a different color. Scores for MedleyDB are shown in row 1 and scores for iKala are in row 2. Raw pitch accuracy is shown in column 1 and overall accuracy is shown in column 2. The y-axis in all plots indicates the number of tracks.

While the scale of the results are below the current state of the art [30] – likely due to the noisiness of the training data! – we see a clear positive impact on performance when data cleansing is applied. The overall trend we see is that training using filtered data outperforms the baseline of training using all the data with statistical significance

---

[3] We train each new model from scratch, not using transfer learning

[4] using continuous voicing from the model output and binary voicing from the annotations
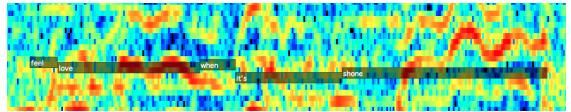
Figure 6. Histogram of the estimated error rate per track.

Figure 7. The CQT of an excerpt of the track in DALI with the lowest percentage error ($< 1\%$ error), with it's annotations overlaid. The audio for this excerpt can be found at `https://youtu.be/Wq4tyDRhU_4?t=40`.

($p < 0.001$ in a paired t-test) for all cases, indicating that our error detection model is successfully removing time frames which are detrimental to the model. We also see that, overall, training using all the data but using the error detection model to weigh samples according to their likelihood of being correct is even more beneficial than simply filtering. This suggests that the likelihoods produced by our error detection model are well-correlated with the occurrence of real errors in the data. These trends are more prominent for the iKala dataset than for the MedleyDB dataset – in particular, the difference between training on filtered vs. weighted data is statistically insignificant for MedleyDB while it is statistically significant ($p < 0.001$ in a paired t-test) for the iKala dataset. The iKala dataset has much higher proportion of *voiced* frames (frames with a pitch annotation) than MedleyDB. This suggests that the weighted data is beneficial for improving pitch accuracy, but may not bring any improvement over filtering for detecting whether a frame should have a pitch or not (voicing). Nevertheless, both conditions which used the error detection model to aid the training process see consistently improved results compared with the baseline.

4.2 Estimated Quality of The DALI Dataset

As a final experiment, we ran the error detection model on the full DALI dataset (version 2) in order to estimate the prevalence of errors. We compute the percentage of frames per-track where the likelihood of being an error is ≥ 0.5. A histogram of the results is shown in Figure 6. We estimated that on average, 21.3% of the frames of a track in DALI will have an error in the note annotation, with a standard deviation of 12.7%. 31.1% of tracks in DALI have more than 25% errors, while 18.2% of tracks have less than 10% errors. We also measure the relationship between the percentage of estimated errors per track and the normalized cross correlation from the original DALI dataset [3], and found no clear correlation. This indicates that while the normalized cross correlation is a useful indication of the global alignment, it does not reliably capture the prevalence of local errors.

We manually inspected the tracks with a very high percentage of estimated errors ($> 70\%$) and found that all of them were the result of the annotation file being matched to the incorrect audio file (see [25] for details on the matching process). On the other hand, we found that the tracks with a very low percentage of estimated errors ($< 1\%$) had qualitatively very high quality annotations. For example, Figure 7 shows an excerpt of the track with the lowest error

rate along with a link to listen to the corresponding audio. While this is only qualitative evidence, it is an additional indicator that the scores produced by the error detection model are meaningful. The outputs of our model on DALI are made publicly available.

An error detection model that estimates the quality of a dataset can be used in several ways to improve the quality of a dataset. For one, we can use it to direct manual annotation efforts both to the most problematic tracks in a dataset, but also to specific incorrect instances within a track. Additionally, one of the major challenges regarding automatic annotation is knowing how well the automatic annotation is working. For example, the creation of the DALI dataset used an automatic method for aligning the annotations with the audio, and it was very difficult for the creators to evaluate the quality of the annotations for different variations of the method. This issue can now be overcome by using an error detection model to estimate the overall quality of the annotations for different variations of an automatic annotation method.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented a novel data cleansing technique which considers the time-varying structure of labels. We showed that it can be successfully applied to a dataset of vocal note event annotations, improving Raw Pitch Accuracy by over 10 percentage points simply by filtering the training dataset using our data cleansing model. Our approach is particularly useful when training on very noisy datasets such as those collected from the Internet and automatically aligned. We also used our proposed error detection model to estimate the error rate in the DALI dataset.

For future work, while our experiments focused on vocal note event annotations, we believe this technique could be directly applied to any kind of note event annotation, as well as extended for other types of time-varying annotations such as chords or beats. We also believe the error detection model could be applied to scenarios other than training. First, a natural use of such a model is to streamline manual annotation efforts by using the model to select time regions that are likely wrong and send them to an expert for correction. Similarly, it could be used as an objective measure to guide the design of automatic annotation methods, which are otherwise forced to rely on manual evaluation. We would also like to explore how this idea can be generalized to other domains beyond music and to test the contribution of different factors including the amount of noise in a dataset and the nature of the noise.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2014.

[2] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Columbia University, 2016.

[3] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "DALI: a large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm." in *International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[4] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[5] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3733–3742.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020.

[7] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *International Conference on Learning Representations (ICLR)*, 2017.

[8] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *International Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[9] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *Journal of artificial intelligence research*, vol. 11, pp. 131–167, 1999.

[11] S. Guo, W. Huang, H. Zhang, C. Zhuang, D. Dong, M. R. Scott, and D. Huang, "CurriculumNet: Weakly supervised learning from large-scale web images," in *European Conference on Computer Vision (ECCV)*, 2018.

[12] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[13] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009.

[14] V. Mnih and G. Hinton, "Learning to label aerial images from noisy data," in *International Conference on Machine Learning (ICML)*, 2012.

[15] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[16] B. Settles, "Curious machines: Active learning with structured instances table of contents," Ph.D. dissertation, Stanford Music Department, 2008.

[17] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei, "The unreasonable effectiveness of noisy data for fine-grained recognition," in *European Conference on Computer Vision (ECCV)*, 2016.

[18] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[19] S. Fürniss and M. Castellengo, "Ecoute musicale et acoustique," *Cahiers d'ethnomusicologie. Anciennement Cahiers de musiques traditionnelles*, no. 29, pp. 223–226, 2016.

[20] M. Mauch, C. Cannam, R. M. Bittner, G. Fazekas, J. Salamon, J. Dai, J. P. Bello, and S. Dixon, "Computer-aided melody note transcription using the Tony software: Accuracy and efficiency," in *International Conference on Technologies for Music Notation and Representation (TENOR)*, 2015.

[21] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2009.

[22] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations (ICLR)*, 2019.

[23] Q. Xi, R. M. Bittner, X. Ye, J. Pauwels, and J. P. Bello, "GuitarSet: A dataset for guitar transcription," in *International Society of Music Information Retrieval (ISMIR)*, 2018.

[24] L. Su and Y.-H. Yang, "Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription," in *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2015.

[25] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Creating DALI, a large dataset of synchronized audio, lyrics, and notes," *Transactions of the International Society for Music Information Retrieval*, 2020.

[26] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *International Conference on Computer Vision (ICCV)*, 2017.

[27] K.-H. Lee, X. He, L. Zhang, and L. Yang, "CleanNet: Transfer learning for scalable image classifier training with label noise," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[28] J. Brown, "Calculation of a constant q spectral transform," *Journal of the Acoustical Society of America*, vol. 89, pp. 425–434, 1991.

[29] A. Jansson, E. J. Humphrey, N. Montecchio, R. M. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep U-net convolutional networks," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[30] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for f0 estimation in polyphonic music," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[31] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[32] T.-S. Chan, T.-C. Yeh, Z.-C. Fan, H.-W. Chen, L. Su, Y.-H. Yang, and J.-S. R. Jang, "Vocal activity informed singing voice separation with the ikala dataset," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[33] R. M. Bittner and J. J. Bosch, "Generalized metrics for single-f0 estimation evaluation," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[34] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

MULTIDIMENSIONAL SIMILARITY MODELLING OF COMPLEX DRUM LOOPS USING THE GROOVETOOLBOX

Fred Bruford[1]   **Olivier Lartillot**[2]   **SKoT McDonald**[3]   **Mark Sandler**[1]

[1] Centre for Digital Music, Queen Mary University, United Kingdom

[2] RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion, University of Oslo, Norway

[3] inMusic Brands inc, United States

`fred.bruford@gmail.com, olivier.lartillot@imv.uio.no, smcdonald@inmusicbrands.com, mark.sandler@qmul.ac.uk`

ABSTRACT

The *GrooveToolbox* is a new Python toolbox implementing various algorithms, new and pre-existing, for the analysis and comparison of symbolic drum loops, including rhythm features, similarity metrics and microtiming features. As part of the *GrooveToolbox* we introduce two new metrics of rhythm similarity and four features for describing the significant properties of microtiming deviations in drum loops. Based on a two-part perceptual evaluation, we show these four new microtiming features can each correlate to similarity perception, and be used with rhythm similarity metrics to improve personalized similarity models for drum loops. A new measure of structural rhythmic similarity is also shown to correlate more strongly to similarity perception of drum loops than the more commonly used Hamming distance. These results point to the potential application of the *GrooveToolbox* and its new features in drum loop analysis for intelligent music production tools. The *GrooveToolbox* may be found at: `https://github.com/fredbru/GrooveToolbox`

1. INTRODUCTION

Growing attention has been drawn to the applications of Music Information Retrieval (MIR) within the realm of music creation to improve upon conventional workflows and enhance creativity [13]. Due to their popularity in contemporary music, research into the analysis of drum loops is a field with strong potential to provide genuine value in real-world music production applications.

The problem of similarity modelling is a key element of this research. The ability to compare drum loops according to perceptually relevant qualities is an essential enabling factor in many plausible systems, such as drum loop recommendation systems, automatic drum loop generation systems and interfaces for navigating drum loop libraries.

For the purposes of this paper, one example use case for drum loop similarity modelling is to enable intelligent

drum loop searching tools within *BFD3*, a virtual drum kit plugin [8]. *BFD3* generates realistic drum sounds based on audio renderings of expressive and unquantized symbolic sequences recorded by real drummers on an electronic drum kit. Including third-party expansions, over 7000 of these symbolic loops are available, providing rich potential for intelligent navigation or recommendation tools.

In **Section 2**, we give an overview of work related to intelligent drum production tools (IDPTs) and discuss possible improvements to their methods of drum loop analysis. In **Section 3** we introduce the *GrooveToolbox*, a Python toolbox primarily aimed towards use in drum loop analysis research. It contains implementations of many existing rhythm features and similarity measures for the analysis and comparison of symbolic drum loops with fixed tempo and metre. New algorithms are also provided: models of rhythmic structural similarity, and models of microtiming accounting for timing styles and mixtures of metrical subdivisions. This section gives an overview of the algorithms implemented in the *GrooveToolbox*.

In **Section 4**, we investigate the effectiveness of the algorithms contained within the *GrooveToolbox* via application to modelling similarity for drum loops. We test the effectiveness of our new polyphonic rhythm similarity measures against the commonly used Hamming distance, and test to what extent high-level rhythm feature-based similarity models can be improved with low-level rhythm similarity metrics and microtiming features to build individualized predictive models that could be used in user-aware IDPTs. In **Section 5** we summarize conclusions of this work, and detail further work required on the *GrooveToolbox* itself and in drum loop analysis generally.

2. BACKGROUND

2.1 Intelligent tools for drum loop production

The primary application of the *GrooveToolbox* is towards research into IDPTs. Much of this centres on the symbolic level rather than audio, and it largely centres on two applications: automatic generation of drum loops, and intelligent interfaces for exploring libraries of drum loops.

In general the goals of automatic drum loop generation in a music production context can be both to speed up the production process and help stimulate new ideas in the producer [17]. Much of the work in this area aims to

generate variations on existing patterns to help producers create evolving drum parts. Genetic algorithms have been applied to achieve this, with the target vector derived from similarity metrics like the Hamming distance [21] or a feature set [16]. Also using genetic algorithms, [15] presents a system that interpolates between two existing drum patterns via a feature space. Variational autoencoders have also been used, generating loop variations that adapt to fit structural changes in an existing song [32] or to fit within a musical trio alongside melody and bass instruments [23].

A second area is in intelligent interfaces for exploring drum loop libraries. The design of intelligent user interfaces for improved music collection exploration is a well researched area with many successful applications [18]. Similarly, the mapping of drum loops on a 2D space via a similarity measure or dimensionality reduction of a feature space has potential to enable improved navigation of a large library. In [3], the authors map a large library of drum loops via the Self-Organizing Map, using a modified version of the Hamming distance as the similarity measure. In [11], the authors present a continuous, generative 2D space for drum patterns based on applying Multi-dimensional Scaling (MDS) to a set of rhythm features.

2.2 Improving drum loop analysis

In the IDPTs described above, symbolic drum loops are analyzed using rhythm features, such as density and syncopation, or rhythmic similarity measures such as the Hamming distance [30]. These may not capture all the important characteristics of complex loops. Two possible areas of improvement, informed by recent musicological research, are in the analysis of microtiming and rhythmic structure.

2.2.1 Microtiming

Microtiming can be defined as sub-rhythmic quasi-random or systematic timing deviations from a metrical grid in human performance. Representations of rhythms that are fit to a metrical grid are a requirement for many rhythm similarity measures, such as the Hamming distance, or features relying on metrical profiles like syncopation [20]. However, fitting rhythms to a grid removes microtiming information, which can be musically significant.

The timing 'feel' or 'groove' of a performance may be an important perceptual factor of drum loops; it has been shown that drummers can control the 'pushed' or 'laid-back' feel of their performance [5]. In [25], timing strategies in drumming for 'laid-back', 'ontop' and 'pushed' styles are measured for a group of drummers based on the typical back-beat rhythmic structure. It was found that their strategies can be formalized as specific timing interactions occurring on downbeat metrical positions. These are between the kick or snare and the hi-hat, or the metrical grid (or metronome) when there is no hi-hat present. The reference to hi-hat as well as grid was based on the understanding that when present the hi-hat usually acts as the time-keeper of the pattern. Detecting these timing interactions may be important for analysing groove in human-performed (or human-imitating) drum loops.

Secondly, gridded representations of rhythms do not account for swung rhythms, or loops where multiple subdivisions of a beat occur. Calculating a pattern-based rhythm similarity metric may still be desirable in these cases however. For example, a swung and unswung version of the same rhythm will be somewhat similar. Or, a loop in 4/4 time may have one instance of a triplet rhythm, in a fill for example. By incorporating a measure of deviation outside of a grid, we can manage these cases, whilst keeping the metrical reference required by other rhythm features.

2.2.2 Rhythmic Structure
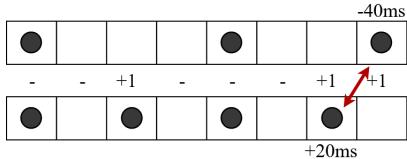
The Hamming distance for rhythm similarity works by stepping through each metrical position in two rhythms, and counting the distance (difference) as the number of instances where rhythms contain different values (one a rest and the other an onset) in the same positions. Hence, in **Figure 1**, the Hamming distance would be 3. This measure is adapted to variable dynamics by using the difference in intensity or velocity as a weighting factor.

Though possibly the most popular rhythm similarity metric, the Hamming distance's stepwise nature fails to pick up regional rhythmic similarities. Onsets in similar but non-identical positions do not register as similar in the Hamming distance, even though perceptually they may be.

A measure of structural similarity may pick this up but requires the derivation of a structural representation. The recent rhythmic transformation model of [26] provides a means of doing so. In [26], an algorithm is described for characterizing rhythms as combinations of three types of ornamentation: syncopation, pickup (anacrusis) and density. Each of these is classified in terms of its position within a metrical profile and surrounding onsets. Syncopations are onsets placed in weak metrical positions, not followed immediately by an onset in a stronger position. Density ornamentations are placed in weak positions between two events in stronger positions. Pickups are placed in weak positions but followed immediately by an onset on a stronger position. Any rhythm can be defined as an ordered combination of these ornamentations against metrical profile, and any rhythm can be decomposed to a 'metronome' - an onset on each downbeat - in the same manner by reversing (removing) these ornamentations. This decomposition process can be used to simplify rhythms in a musicologically sound manner, where the simplified rhythmic representation is analogous to a structure.

3. GROOVETOOLBOX

GrooveToolbox is a Python toolbox for analysing and comparing rhythmic and microtiming qualities of drum loops in various formats. The toolbox contains functions for a variety of pre-existing features and new ones that account for rhythmic structure and microtiming. They fall within three groups: rhythm features, microtiming features and similarity measures. As the toolbox is designed to work with loops of fixed tempo, it does not contain features for tempo tracking or conventional metre detection. Nor does it provide timbral analysis as it works on a symbolic level.

Figure 1. Measuring similarity of two short rhythms. Hamming distance = 3. With 16th note steps at 120BPM fuzzy Hamming distance = 2.26

Two Python toolboxes in the public domain relate to the *GrooveToolbox*. The *Rhythm Toolbox* [10] presented as part of [9] provides a starting point for drum loop analysis, with functions for two different types of syncopation features plus density features. However the feature set is limited, and it only supports MIDI files. The *Groove-Toolbox* adds many algorithms to this set. It also supports BFD3 [8] format Groove files as an alternative to MIDI, MIREX format [1] and audio files through the integrated *ADTLib* drum transcription library [29].

The *SynPy* toolbox [28] also provides related functionality, consisting of implementations of seven syncopation models. While useful, the toolkit is designed for use on monophonic rhythms, and as such do not immediately apply to drum loops. In the *GrooveToolbox*, we implement one syncopation model [20] also found in *SynPy*, and add another designed specifically for drum patterns [34].

The algorithms implemented in the *GrooveToolbox* are collected from a range of research, with the aim of enabling comprehensive modelling of the perceptual qualities of drum loops. For re-implemented algorithms, we chose those which were experimentally verified as perceptually relevant and ensured they could account for variable onset velocities. The algorithms currently provided in the *GrooveToolbox* are listed in **Table 1**. In this section, we will describe the two new similarity models and four microtiming features. More details may be found at: `https://github.com/fredbru/GrooveToolbox`.

3.1 New Rhythm Similarity Models

3.1.1 Fuzzy Hamming distance

The fuzzy Hamming distance extends the Hamming distance by incorporating one metrical step of displacement along with the microtiming deviations of each onset. Where there is an onset in one rhythm but not the other, the algorithm looks ahead one step to look for a nearby onset. An instance of this is shown in **Figure 1**. If there is a nearby onset, the distance is reduced depending on how close that onset is. The microtiming deviations are also considered when two onsets occur at the same metrical position. In each case the timing difference between the two nearby onsets is incorporated in the similarity calculation. If present, the difference in onset velocity may also be used as a weighting factor as with the Hamming distance.

For the example of **Figure 1**, the value 1 in position 3 is the same as for the Hamming distance. The final two positions add 1 each to the Hamming distance, but the fuzzy Hamming counts the first of these as the timing difference

between the two onsets, divided by the time of two metrical steps. At 120BPM, one 16th note lasts 125ms, so the similarity at this position $= (125 - 20 - 40)/250 = 0.26$.

With this proximity accounted for, the final position is calculated as usual, adding 1 to the distance. The overall distance is therefore 2.26 - close to the Hamming distance (3) but scored as more similar due to the proximity of the last two onsets. In a different case where the microtiming deviations in the two nearby onsets were removed, the fuzzy Hamming distance would be 2.50, higher to reflect the increased distance between the two onsets.

By accounting for possible similarity between nearby onsets, this method reduces the Hamming distance's limitation in detecting regional similarities. It also accounts for rhythms with microtiming deviations, such as swung rhythms, by not discarding them. Accounting for microtiming differences between onsets in the same position may also capture the overall difference in microtiming feel.

3.1.2 Structural Similarity

The structural similarity metric measures the similarity of a structural representation of two loops, derived following [26]'s transformation model (see **Section 2.2.2**). First, we remove any 'ghost notes', below a loudness threshold. Ornamentations are then found and reversed (removed) until any onsets only occur on downbeats. This results in representations of rhythmic structure at the downbeat level upon which the Hamming distance can be calculated.

3.2 Microtiming features

To develop features that describe the perceptual properties of microtiming deviations in drum loops, the first stage is to represent them in a form from which features can be extracted. In the context of drum pattern analysis, a sparse matrix format has been used by [12] to express microtiming. Here a matrix shows the timing deviation from the grid in milliseconds, positive (behind the beat) or negative (in front of the beat), for each onset. **Figure 2** shows this for a simple 2 beat pattern. The features extracted from this representation measure two types of microtiming effect: swing or metrical feel and performance styles.

	1	2	3	4	5	6	7	8
K	5	-	-	-	-	-	-	-
S	-	-	-	-	21	-	-	-
HH	-4	-	-8	-	-10	-	3	-

Figure 2. Matrix representation of timing deviations (ms) from 16th note positions in a 2 beat 120BPM kick, snare and hihat pattern (Laid-back event highlighted).

3.2.1 Swing and metrical feel

Using a sparse matrix representation of microtiming deviations alongside a rhythmic representation, swing can be detected along with the presence of triplets in a quadruple-time pattern. Swung onsets are detected as significantly delayed second eighth-notes, approximating the typically

Type	Feature Name	Description
Syncopation	Monophonic	Comparing onset pattern with hierarchical metrical profile [20]
	Polyphonic	Comparing interaction between instruments with metrical profile [34]
	Weak-Strong Ratio	Number of onsets not occurring on downbeats vs on downbeats [15]
Density	Absolute Density	Total onsets divided by number of possible onsets for any number of parts
	Relative Density	Density of one part divided by total density (cf. hiness feature [9])
	Syncopation Density	Syncopation divided by total number of onsets (cf. hisyness feature [9])
Complexity	Rhythmic Complexity	Quadratic mean of density and syncopation [27]
Periodicity	Autocorrelation Skewness	Skewness of autocorrelation curve [22]
	Autocorrelation Max Amplitude	Maximum amplitude of autocorrelation curve [22]
	Harmonicity	Harmonicity of autocorrelation curve, (primarily pulse clarity) [19]
	Symmetry	Proportion of onsets at the same position in 1st and 2st half of pattern. [22]
Intensity	Average	Mean of all velocity values in pattern [15]
	Standard Deviation	Standard deviation of all velocity values in pattern [15]
Swing and metrical feel	**Swing-ness**	Whether loop is swung, weighted by number of swung notes
	Triplet-ness	Whether loop contains any triplets, weighted by number of triplet notes
Microtiming style	**Laidback-ness**	Microtiming style as number of push/laid-back events
	Timing accuracy	Mean of absolute timing deviation from grid of all onsets
Similarity measures	Hamming distance	Counting number of metrical positions where values (onset/rest)' are different
	Fuzzy Hamming distance	Hamming with 1 step lookahead, distance weighted by microtiming
	Structural similarity	Similarity of patterns simplified using [26]'s transformation algorithm

Table 1. List of features and similarity measures currently implemented in the *GrooveToolbox*. New features are in **bold**.

Figure 3. Matrix representation of timing deviations (ms) for 16th note metrical positions in a 2 beat 120BPM rhythm. Red = swung events, green = triplet events.

understood 2:1 eighth-note swing ratio. Although musically these are considered as eighth notes, they fall into sixteenth note positions when quantized, with significant negative (ahead of the position) deviations. The 'swing-ness' feature first records whether these timing deviations occur or not, returning 0 for no swing or 1 for swing. This is then weighted by the number of swung onsets to model perceptual salience of the swing. The deviation threshold for swing is calculated dependent on the tempo of the rhythm. The 'triplet-ness' feature is calculated in the same way, but also records the second note of a triplet as significantly delayed second eighth note. A triplet note detected from a microtiming matrix is shown in green in **Figure 3**.

3.2.2 Microtiming style

Overall timing accuracy is calculated as the mean of all absolute timing deviations per onset in a loop. For onsets classed as swung or triplets, the deviation is calculated from the 'ideal' triplet or swung note position.

Following the timing interaction classification of [25] as described in **Section 2.2.1** the laidback-ness feature counts the number of laid-back timing events subtracted from pushed events, with a negative score meaning an overall 'pushed' loop and positive a 'laid-back' loop. Thus a pattern's feel is calculated based on the detection of specific timing discrepancies above a perceptual threshold, known to impart a given feel from drumming performance analysis. We modified [25] by also counting for ride cymbal in place of hihat due to their similar musical roles.

Based on analysis of timing accuracy in BFD3's library, we chose a threshold of 12ms as a one that would disregard

performance noise. However, ideally this would be calculated per drummer as in [25]. An example 'laid-back' pattern is shown in **Figure 2**. The highlighted event is 'laid-back', as there is a discrepancy on the downbeat between (in this case) snare and hihat that is above the threshold. For this pattern, the timing accuracy value would be 8.5, and laidback-ness 1.

4. EVALUATION

To evaluate our new algorithms and address open questions in drum loop analysis, a two-part experiment was carried out into modelling similarity for BFD3's drum loops, using perceptual data from humans collected via listening test. Three research questions were addressed:

1. *Should models of similarity for drum loops rely on rhythm similarity metrics, feature sets or both?*
   One approach to modelling drum loop similarity is to adapt rhythm similarity metrics [30] that measure distances between onset patterns, used for example in [3, 21, 31]. An alternative is to model similarity as a combination of higher-level rhythm features [11, 16]. While the two are not usually combined, both may be important as they emphasize different information.

2. *Can the models of microtiming proposed be used in modelling similarity of drum loops?*
   Existing IDPTs, described in **Section 2.1**, tend to assume simpler quantized and unswung rhythms. To apply this work to complex loops that are unquantized or swung, or include multiple subdivisions of the metre, models that can account for microtiming deviations could be important.

3. *How do the new rhythm similarity models compare to the Hamming distance?*
   We investigate alternate ways of measuring rhythmic similarity by testing the two new rhythm similarity measures proposed against the Hamming distance.

4.1 Data collection

The dataset consisted of similarity ratings for 80 pairs of BFD3's drum loops (160 total) as provided by 21 participants in a listening test, first collected in [4]. Loops were generated through the same virtual kit, BFD3's 70s Rock kit, chosen due to its generic timbre. Loops were collated equally from 8 genre groups: Blues/Country, Rock, Metal, Jazz, Funk, Reggae/Latin, Pop and Dance/Hiphop. Tempo, metre and loop length were fixed at 120bpm, 2 bars and 4/4 time. Some were swung, and a small number contained triplet rhythms. The test was distributed online using the Web Audio Evaluation Tool [14], with participants representing a range of musical and technical experience.

The listening test used a pairwise comparison methodology. Participants rated how similar two loops were on a continuous scale with five equally spaced markers (*Completely Different, Different, Slightly Different/Slightly Similar, Similar, Identical*). This was to maximize the number of loops in the study, whilst keeping the test length reasonable (30-40 minutes), ensuring multi-genre validity but giving more fine similarity ratings than in the common triadic comparison test design [2]. 5 training pairs were given at the start of the test, and 10 pairs were repeated at the start and end to test participants' internal consistency. The inter-rater reliability (IRR) calculated for all raters across all comparisons using the intraclass correlation coefficient (ICC) in (2,1) form, was 0.73, a moderate-to-good agreement. This is expected, as it is known that musical similarity perception is very individualized [33], with low IRR a challenge in musical similarity studies [7]. The average internal consistency of all participants, calculated as the median ICC (2,1) between ratings for the 10 repeated pairs, was 0.85, equal to good consistency.

4.2 Evaluation Design

Based on this perceptual data, our evaluation was formed in two parts to address the three research questions. The evaluation was designed with consideration of the individualized nature of similarity perception.

4.2.1 Overall perceptual relevance of new models

First we evaluated the new similarity and microtiming models based on their correlation to listeners' similarity ratings. The extent to which the models relate to perceived similarity was measured as the Pearson correlation between the feature score and median similarity rating of the 21 participants. The spread of ratings per pair was approximately normally distributed (D'Agostino-Pearson test $p > 0.01$) so the median across raters is used. While this indicates if the features proposed relate to overall perceived similarity, limited IRR means that more precise analysis of the performance of these features against an average of ratings may not be valid. The second part of the experiment therefore uses individuals' ratings instead.

4.2.2 Building Individualized Similarity Models

In this part, the third research question is addressed. This experiment will find to what extent rhythm feature-based

models can be adapted for use in complex drum loops when combined with microtiming features, and whether a combination of rhythm similarity metric and feature set can offer a better similarity model than either alone can. The aim here is to build similarity models that are predictive, investigating the utility of the models in practical use-cases that may require precise, fine-grained models. Due to low IRR, this requires we develop models for individuals' ratings separately, echoing the concept of 'user-aware' MIR [24]. The models are tested against individual ratings of 7 participants, chosen as those with highest internal consistency (for repeated pairs median ICC = 0.92), and whose ratings were normally distributed (D'Agostino and Pearson test $p > 0.01$), meaning regression models were applicable. Not all participants' individual ratings fit a normal distribution, and not all had high enough internal consistency for precise prediction, so not all could be used.

For each participant we tested seven feature combinations: rhythm features, microtiming features, the best similarity metric from **Section 4.3.1** and each combination of the three. We used all features in **Table 1**. The density features were calculated across three instrument groups separately: low (kick), mid (snare and toms) and high (cymbals) as in [11]. All other features were calculated with all instruments combined. Fitting a regression model to the 7 participants for 7 conditions, we measured the predictive power of the models as the explained variance (R-squared) for each feature combination and participant.

For the single similarity measure we used linear regression. For the feature sets, Partial Least-Squares (PLS) regression was chosen [35]. This was chosen because the features exhibit a high degree of colinearity, due to the large number of features and the inherent co-dependence of musical qualities. This combined with the high predictor-cases ratio means that linear modelling does not work. PLS regression combines elements from multiple linear and principal component regression, and has been found to work well in multidimensional musical emotion prediction [6]. Its use of principal component analysis to create latent prediction variables alleviates the problems of colinearity and high predictor-cases ratio, but limits interpretability of the significance of specific features. This is partly why we evaluate features differently in **Section 4.3.1**. To choose the best number of principal components for each model consistently, we used the Bayesian Information Criterion (BIC), which accounts for overfitting by penalising model complexity against model performance.

4.3 Results & Discussion

4.3.1 Overall perceptual relevance of new models

The correlation between the features and median similarity ratings is shown in **Table 2**. All models exhibited statistically significant correlation ($p < 0.05$).

The structural similarity measure exhibits higher correlation to the median similarity ratings than the ordinary velocity-weighted Hamming distance (t=1.69, p=0.046). It could therefore be the case for complex drum loops that listeners compare rhythms at a more global structural level

Feature/Model	Correlation r	p
Hamming Distance	0.59	9.7e-9
Structural Similarity	0.65	6.1e-11
Fuzzy Hamming Distance	0.56	6.1e-8
Swing-ness	0.46	1.9e-5
Triplet-ness	0.49	3.4e-6
Timing accuracy	0.33	2.9e-3
Laidback-ness	0.22	0.046

Table 2. Pearson r and *p-value* between median similarity ratings and model difference values.
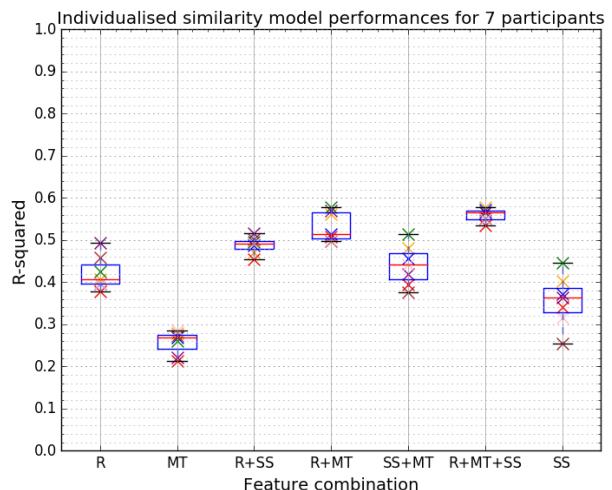
rather than a precise low level. Alternatively, the structural representation could indicate shared genre between loops by showing the approximate locations of events. Given this seemingly strong performance, using the structural similarity measure over Hamming distance may be advantageous.

For the laidback-ness feature, an issue was that values were typically low, as there is often not a significant difference in microtiming styles between loops. Based on the 12ms deviation threshold, in only 21 of the 80 comparisons was there a difference between timing styles, with the remaining three quarters of the comparisons returning 0. Looking at the correlation between the laidback-ness feature and similarity ratings just for these 21 comparisons where timing style is different, correlation is stronger ($r = 0.47$, $p = 0.0033$). One interpretation is that a feature like this, which models a precise low-level quality, is only relevant for a similarity comparison when there is a significant difference in this quality. While further investigation is required, this may point to the use of adaptive feature weightings for similarity comparisons that select features based on the relevance to a particular comparison.

The other microtiming features exhibited moderate to good correlation, with the swing-ness and triplet-ness features being approximately the same. This may be because loops in our dataset with onsets matching to second triplet positions likely have notes in swung positions too, so there is little practical difference in their values. For a larger dataset this difference may be more significant. The fuzzy Hamming distance did not differ significantly from the regular Hamming distance. The correlations of both the swing and microtiming style features indicate that a better way to incorporate microtiming in similarity models could be as a separate set of features modelling global characteristics of microtiming, rather than being inserted into rhythmic similarity measures. Overall, it appears that these features are able to some extent to capture perceptually salient features of microtiming deviations in drum loops.

4.3.2 Building Individualized Similarity Models

The results of this part are shown in **Figure 4**. Comparing the median r^2 score for the 7 participants, it can be seen that the combination of rhythm and microtiming features with structural similarity measure results in the best predictive model ($r^2 = 0.56$), closely followed by rhythm feature and microtiming model ($r^2 = 0.51$). This confirms that both microtiming and structural similarity models can improve rhythm feature-based similarity models in this case. However, there is still improvement required before they can accurately predict similarity perception.

Figure 4. Model performance as R-squared value for combinations of rhythm **R** and microtiming **MT** feature sets with structural similarity feature **SS** for each participant.

There are a few possible reasons for this. As mentioned in **Section 4.3.1**, the system of deriving a fixed weighting of features in a multidimensional similarity model may not be the best way to model similarity; instead, adaptive weighting schemes may be required that weigh features according to their relevance in a given comparison. While the feature set seems comprehensive, there may be some qualities of drum loops that are not effectively being modelled, in particular features that explicitly detect style or genre. Similarity ratings from more listeners should be collected in the future to construct further personalized models and verify these findings for a wider range of listeners.

5. CONCLUSIONS & FURTHER WORK

We presented a new toolbox for drum loop analysis, with implementations of pre-existing algorithms and new ones for analysing and comparing rhythmic structure and microtiming. These were found to correlate to perceived similarity of drum loops. The rhythmic structural similarity metric was shown to correlate at least as well as the conventional Hamming distance to similarity perception in drum loops. It has been shown that the ideal model of similarity for complex drum loops combines rhythm and microtiming features with a rhythm similarity metric. These results all have implications in future work on IDPTs.

As further work, new algorithms should be developed to improve similarity models in the *GrooveToolbox*, in particular ones that explicitly model stylistic similarity. Here more investigation into rhythmic grouping or structure could be useful. Due to the complex nature of similarity perception it is difficult to infer the practical utility of similarity models from this evaluation. For a more ecologically valid understanding, we will next evaluate them in the context of an IDPT. An approach to combining features that accounts for the possibly attention-based nature of similarity perception could also be a valuable direction. Given the added requirement for personalized models, the next step is to investigate methods such as active learning to learn an adaptive similarity model from a user.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] 2019:Drum Transcription - MIREX Wiki. `https://www.music-ir.org/mirex/wiki/2019:Drum_Transcription`. Accessed May 2020.

[2] Hamish Allan, Daniel Müllensiefen, and Geraint A. Wiggins. Methodological considerations in studies of musical similarity. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2007.

[3] Fred Bruford, Mathieu Barthet, SKoT McDonald, and Mark Sandler. Groove Explorer: An Intelligent Visual Interface for Drum Loop Library Navigation. In *IUI Workshops*, 2019.

[4] Fred Bruford, Mathieu Barthet, SKoT McDonald, and Mark Sandler. Modelling Musical Similarity for Drum Patterns: A Perceptual Evaluation. In *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, 2019.

[5] Anne Danielsen, Carl Haakon Waadeland, Henrik G Sundt, and Maria AG Witek. Effects of instructed timing and tempo on snare drum sound in drum kit performance. *The Journal of the Acoustical Society of America*, 138(4):2301–2316, 2015.

[6] Tuomas Eerola, Olivier Lartillot, and Petri Toiviainen. Prediction of multidimensional emotional ratings in music from audio using multivariate regression models. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2009.

[7] Arthur Flexer and Taric Lallai. Can we increase inter- and intra-rater agreement in modelling general music similarity? *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[8] FXpansion. BFD3 `https://www.fxpansion.com/products/bfd3/`, Accessed May 2020.

[9] Daniel Gómez Marín. *Similarity and style in electronic dance music drum rhythms*. PhD thesis, Universitat Pompeu Fabra.

[10] Daniel Gómez Marín. Rhythmtoolbox `https://github.com/danielgomezmarin/rhythmtoolbox`, Accessed May 2020.

[11] Daniel Gómez-Marín, Sergi Jorda, and Perfecto Hererra. Drum rhythm spaces: from global models to style-specific maps. In *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2017.

[12] Kahl Hellmer and Guy Madison. Quantifying microtiming patterning and variability in drum kit recordings: A method and some data. *Music Perception*, 33(2):147–162, 2015.

[13] Eric J. Humphrey, Douglas Turnbull, and Tom Collins. A brief review of creative MIR. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2013.

[14] Nicholas Jillings, David Moffat, Brecht De Man, and Joshua D. Reiss. Web Audio Evaluation Tool: A browser-based listening test environment. In *12th Sound and Music Computing Conference*, 2015.

[15] Maximos Kaliakatsos-Papakostas. Generating drum rhythms through data-driven conceptual blending of features and genetic algorithms. In *International Conference on Computational Intelligence in Music, Sound, Art and Design*, pages 145–160. Springer, 2018.

[16] Maximos A. Kaliakatsos–Papakostas, Andreas Floros, and Michael N. Vrahatis. evoDrummer: Deriving Rhythmic Patterns through Interactive Genetic Algorithms. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, pages 25–36. Springer Berlin Heidelberg, 2013.

[17] Peter Knees, Kristina Andersen, Sergi Jordà, Michael Hlatky, Günter Geiger, Wulf Gaebele, and Roman Kaurson. Giantsteps-progress towards developing intelligent and collaborative interfaces for music production and performance. In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–4. IEEE, 2015.

[18] Peter Knees, Markus Schedl, and Masataka Goto. Intelligent user interfaces for music discovery: The past 20 years and what's to come. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[19] Olivier Lartillot, Tuomas Eerola, Petri Toiviainen, and Jose Fornari. Multi-feature modeling of pulse clarity: Design, validation and optimization. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2008.

[20] H. C. Longuet-Higgins and C. S. Lee. The Rhythmic Interpretation of Monophonic Music. *Music Perception: An Interdisciplinary Journal*, 1(4):424–441, 1984.

[21] Cárthach O. Nuanáin, Perfecto Herrera, and Sergi Jorda. Target-based rhythmic pattern generation and

variation with genetic algorithms. In *Sound and Music Computing Conference*, 2015.

[22] Maria Panteli, Bruno Rocha, Niels Bogaards, and Aline Honingh. A model for rhythm and timbre similarity in electronic dance music. *Musicae Scientiae*, 1:24, 2016.

[23] Adam Roberts, Jesse Engel, and Douglas Eck. Hierarchical variational autoencoders for music. In *NIPS Workshop on Machine Learning for Creativity and Design*, 2017.

[24] Markus Schedl, Arthur Flexer, and Julián Urbano. The neglected user in music information retrieval research. *Journal of Intelligent Information Systems*, 41(3):523–539, 2013.

[25] George Sioros, Guilherme Schmidt Câmara, and Anne Danielsen. Mapping timing strategies in drum performance. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[26] George Sioros, Matthew EP Davies, and Carlos Guedes. A generative model for the characterization of musical rhythms. *Journal of New Music Research*, 47(2):114–128, 2018.

[27] Georgios Sioros and Carlos Guedes. Complexity driven recombination of midi loops. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[28] Chunyang Song, Marcus Pearce, and Christopher Harte. Synpy: a python toolkit for syncopation modelling. In *Sound and Music Computing Conference*, 2015.

[29] Carl Southall, Nicholas Jillings, Ryan Stables, and Jason Hockman. Adtweb - an open source browser based automatic drum transcription system. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[30] Godfried T. Toussaint. A comparison of rhythmic similarity measures. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2004.

[31] Richard Vogl, Matthias Leimeister, Cárthach Ó Nuanáin, Michael Hlatky, Sergi Jordà Puig, and Peter Knees. An intelligent interface for drum pattern variation and comparative evaluation of algorithms. *Journal of the audio engineering Society. 2016; 65 (7/8): 503-13.*, 2016.

[32] I-Chieh Wei, Chih-Wei Wu, and Li Su. Generating structured drum pattern using variational autoencoder and self-similarity matrix. *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[33] Geraint A Wiggins. Models of musical similarity. *Musicae Scientiae*, 11:315–338, 2007.

[34] Maria AG Witek, Eric F Clarke, Mikkel Wallentin, Morten L Kringelbach, and Peter Vuust. Syncopation, body-movement and pleasure in groove music. *PloS one*, 9(4), 2014.

[35] Svante Wold, Michael Sjöström, and Lennart Eriksson. Pls-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, 58(2):109–130, 2001.

RULE MINING FOR LOCAL BOUNDARY DETECTION IN MELODIES

Peter van Kranenburg

Meertens Instituut, Utrecht University

`peter.van.kranenburg@meertens.knaw.nl`

ABSTRACT

The task of melodic segmentation is a long-standing MIR task that has not yet been solved. In this paper, a rule mining algorithm is employed to find rule sets that classify notes within their local context as phrase boundaries. Both the discovered rule set and a Random Forest Classifier trained on the same data set outperform previous methods on the task of melodic segmentation of melodies from the Essen Folk Song Collection, the Meertens Tune Collections, and the set of Bach Chorales. By inspecting the rules, some important clues are revealed about what constitutes a melodic phrase boundary, notably a prevalence of rhythm features over pitch features.

1. INTRODUCTION

Melody is one of the basic aspects of music. As such, it has been the object of study in numerous research projects in various fields, including music theory, ethnomusicology, music cognition, and music information retrieval. In virtually all those studies, it is generally accepted that a given melody can be analysed in terms of smaller constituents. The availability of a musically sensible segmentation facilitates various music information retrieval tasks [1]. There is, however, no coherent theoretical answer to the questions what exactly are these constituents and how to isolate them from the holistic construct of a melody.

One line of research has been to design computational models to detect segment boundaries at the surface level of the melody. Typically, these models have been tested on a corpus of melodies in which segment boundaries are annotated, mainly the Essen Folk Song Collection [2].

Computational models that have been proposed to partition a melody into a sequence of segments, basically take one of two approaches. In the first approach, which is theory-driven, a set of rules is designed based on theories of human perception and cognition of melodic information, typically drawing on a combination of Gestalt Psychology [3] and Music Theory. These rules are then formalised and quantised in such a way that they can be implemented in software to automatically detect possible segment boundaries in the melodies. The underlying assump-

tion is that these rules reflect the way humans detect patterns in sensory input.

In the other approach, which is data-driven, a model is learnt from data without strong a-priori theoretic assumptions. This approach is based on the idea that a human listener learns to recognise musical events (such as segment boundaries) by exposure.

In this article, we apply a rule mining algorithm that infers from a large corpus of segmented melodies a rule-based model of what is a phrase boundary. The choice for rule mining is motivated by the explainability of the resulting models, which consist of human readable sets of rules. By examining the discovered rules we gain a better understanding of what constitutes a melodic segment boundary, and what features play a role for detecting segment boundaries. We include many features to allow the mining algorithm to choose which features are necessary for the task. We apply the rule-mining algorithm RIPPER [27] as wel as a Random Forest classifier [29] on several subsets of features. By using other data sets next to the Essen Folk Song Collection, we broaden the information on which the models are based, and we are able to compare phrase boundaries across different melodic styles.

2. RELATED WORK

In this section, we review relevant related work. First, we present theory-driven, rule-based approaches (Section 2.1), and then data-driven approaches (Section 2.2).

2.1 Theory-Driven Approaches

The seminal book on *Emotion and Meaning in Music* by Leonard Meyer [4] was one of the first to explicitly relate music expectation to principles of gestalt theory. This publication initiated major lines of research in music cognition and music theory. Tenney and Polansky [5] were among the first to define an implementable, quantitative model for detecting segment boundaries. Their model is based on the principles of proximity (in time) and similarity (in pitch). Several other models are based on gestalt principles as well: the Local Boundary Detection Model (LBDM) by Cambouropoulos [6, 7], the Grouper model by Temperley [8], the preference rules for grouping as defined in A Generative Theory of Tonal Music (GTTM) by Lerdahl and Jackendoff [9], the quantisation of these rules by Frankland and Cohen [10], the Implication-Realization theory by Narmour [11, 12], and the partial quantisation of this theory by Schellenberg [13]. More recently, vari-

ous theory-based approaches have been proposed by Rodríguez López [14]. A rule-based model not explicitly grounded on gestalt principles was proposed by Cenkerová et al. [15].

2.2 Data-Driven Approaches

Explicitly challenging the gestalt principles, Bod [16] introduces Data Oriented Parsing (DOP). A DOP-Markov parser learns probabilities for rewrite rules from a set of examples. One of the applications of this model was the prediction of segment boundaries in the Essen Folk Song Collection. In an error analysis, Bod shows that the DOP-Markov parser is able to learn regularities in phrase-ending patterns that do not adhere to gestalt rules.

Various data-driven approaches are based on information theory. Generally, a phrase boundary is inferred either before an unexpected melodic event or after an event for which the continuation is hard to predict. Methods differ in the way of computing the conditional probability of events given their preceding context. Juhasz [17] takes this approach to segment a collection of Hungarian folk songs. The multiple viewpoint statistical modelling method by Conklin and Witten [18] has been used for many symbolic music processing tasks such as generation, classification, and pattern discovery. The IDyOM model [19] employed the multiple viewpoint method for melodic segmentation. Lattner [20] employs a Restricted Bolzmann Machine to model the probability of a melodic event. This approach outperforms IDyOM, and sets the state-of-the-art for recognising phrase boundaries in the Essen Collection.

Rodríguez López [14] also introduced a data-driven component. A part of his segmentation system needs to be trained on a corpus.

3. DATA

An often used collection of segmented melodies is the Essen Folksong Collection (EFSC). This collection contains thousands of folk song melodies mainly from Germany, but also from other parts of Europe, and a relatively small number of melodies from other continents. In the process of creating this collection, the melodies have been segmented into phrases. Therefore, it offers a large amount of data on melodic segmentation which allows for statistical evaluation. Following earlier work, we use the database Erk, a subset of EFSC consisting of c. 1,700 melodies.

We also employ a recently published corpus from the Meertens Tune Collections (MTC), consisting of collections of thousands of instrumental and vocal songs from Dutch sources [21]. The collection we use in this paper is MTC-FS-INST-2.0, more specifically, those melodies that have lyrics, are dated after 1850, and have a time signature. This results in a selection of c. 7,500 melodies. For reasons that will be explained in section 3.1.2 we apply a further selection: from each tune family, we randomly select one melody. This results in a set of 1,323 melodies.

The third corpus we use is the collection of 371 harmonisations of chorales (CHOR) by Johann Sebastian Bach

Dataset	#songs	#boundary	#noboundary	total
MTC	1,323	7,054	63,856	70,910
ESSEN	1,632	7,703	62,490	70,193
CHOR	370	1,907	15,455	17,362

Table 1. Overview of the datasets indicating the number of songs and the sizes of the classes (number of 5-grams).

(1685–1750). [1] Since our focus is on melodic segmentation, we only use the melodies (i.e., the soprano parts).

An overview of the datasets, the number of songs, and the class sizes is included in Table 1.

3.1 Some Caveats

Employing a collection of folk song melodies has some important consequences that have often not been discussed in previous work. We focus on two problems: tune family relations, and the rest as notational device.

3.1.1 Tune Families

One defining property of folk music is that it has been in oral circulation [22]. In the process of oral transmission, changes are introduced to the melodies and texts. Therefore, in a typical collection of folk songs, several variants of the same melody are included, exhibiting minor to large differences among each other. Such a group of related melodies is often designated as a *tune family* [23]. EFSC and MTC are no exceptions to this. For the collections in MTC, the tune families have largely been identified by collection specialists at the Meertens Institute. The tune family labels are included in the metadata that comes with the collection. For the ESFC this has not been done. From the titles of the songs, which are available in the metadata, it is clear that duplicates and variants of melodies are included, but there is no account of precisely which melodies are related.

The consequence of this for a data mining approach is that the independence of the train and test sets is not guaranteed since members of the same tune family may end up in both the train and test sets. Especially when the differences are small, this is problematic.

To solve this issue, we take advantage of the tune family labels as provided in the metadata of MTC.

3.1.2 Rests

In related work, the presence of a rest appeared to be a strong indicator of a phrase boundary. For example, one of the quantised GTTM preference rules (GPR 2a) states that the boundary strength is proportional to the length of a rest. In LBDM, next to pitch and inter-onset-intervals, rests are explicitly incorporated as one of the three features that contribute to the resulting local boundary strength. Furthermore, the occurrence of a rest is the first of Narmour's six conditions of melodic closure [11, p. 11].

There is, however, a difference between the meaning of a rest in composed music and in folk song transcriptions.

---

[1] This corpus is available as part of the humdrum-data repository: https://github.com/humdrum-tools/humdrum-data

Figure 1. Transcriptions of two variants of the same melody showing different uses of the rest as notational device.

Again, this is related to the process of oral transmission. A composer typically uses common music notation as the primary device to communicate a piece of music to performers. Here, the notation of a rest is *prescriptive*, indicating the performers not to make sound. On the contrary, music notation as found in folk song collections is typically *descriptive*. The melodies have been transcribed from audio recordings, or from aural observation. Here, the rest is an indication of something a performer already has done. The example from the MTC that is shown in Figure 1 illustrates the resulting confusion that can arise if various transcribers contribute independently (or if one transcriber works inconsistently). In the upper transcription, the final note of each phrase is extended to fill the measure, while in the lower transcription, rests are included at the phrase boundaries. Crucially, inspecting the audio recordings that are the sources of these two transcriptions, [2] no noticeable differences are observable between the way the singers separate the phrases.

It appears that the rest as a symbol has a use in folk song transcription to represent a phrase boundary, rather than to indicate absence of sound. As a consequence, using the rest as a feature actually includes the ground truth in the feature set, which obviously results in an optimistic estimation of classification performance. We will therefore report results without using rests, and results including rests – and other ground truth dependent features – separately.

---

[2] These are available at http://www.liederenbank.nl/index.php?lan=en by entering the respective record numbers (73639 and 74427) in the search field.

4. METHOD

The approach in this paper largely is a feature engineering exercise. From previous studies and from general music theoretic considerations, we take inspiration of what features may contribute to the establishment of a phrase boundary. Next, we apply two machine learning algorithms: RIPPER and Random Forest. Thus, we do not take an a-priori theoretical basis, such as the gestalt principles, but we let the learning algorithm explore which features are of value and in what combination.

4.1 Objects and Features

The target of the classification is to find those notes after which a phrase ends. As object of classification we take each note in the melody with its local context of the two preceding and the two following notes, resulting in sequences of five notes, 5-grams. Since the aim is segmentation, the final phrase end, which also ends the melody, is excluded from the data set. Those 5-grams of which the third note is the final note of a phrase get the class label *boundary*, while all other 5-grams get the class label *noboundary*.

For each of the 5-grams we extract a large number of features. Each of those features can be considered a hypothesis of which information contributes to the concept of phrase boundary. We discern various groups of features. For extracting the feature values, the music21 toolkit has been used [24].

Elementary pitch features include for each of the five notes: the scale degree, the absolute pitch value in MIDI-representation, the interval with the previous note in semitones, the pitchcontour (up, down, equal), and the *Harmony* and the *Center of Gravity* as defined in [25].

Elementary rhythm features include the meter 'numerator' and 'denominator', the duration of the beat, the number of beats in the measure, and for each of the five notes: the metric weight, the duration (inter-onset-interval) in units of the beat-length, whether the note starts on or off the beat, and whether the duration increases for each of the first three notes. Furthermore, following the reasoning of Temperley [8, p. 70], we include a boolean feature that is True when the onset time of the fourth note is at the same position in the measure as the onset of the very first note of the melody. This accounts for the preference to start phrases at corresponding positions in the measures. To allow a more fine-grained version of this preference, we also include a boolean feature that is True if the onset time of the fourth note completes the time-span of a beat, starting the first time-span at the onset of the first note (which possibly is not on the beat in case of an anacrusis). The metric weight (beatstrength) and the length of the beat, both included as feature, are computed with the music21 toolkit.

Elementary lyric features (MTC only) include for each of the five notes: whether the lyric syllable is stressed, whether the lyric is a content word, whether the lyric syllable ends a content-word that rhymes with another content-word anywhere in the lyrics, whether the lyric syllable is the final syllable of a word, and whether the note is part

of a melisma. For labeling non-content words, detection of rhyme, and determining the word stress, the methods as described in [26] are used. Furthermore, we measure the distance between the third note of the 5-gram and the most recent rhyming syllable, both as number of notes and as number of beats.

Wherever applicable, we include the first-order contour of these elementary features as separate features, registering whether the value for a note is higher, equal, or lower than the value for the previous note. This provides the rule mining algorithm with relational information for the consecutive notes, which is beneficial because RIPPER is not able to include comparisons between features into the conditions that constitute the rules. Each condition consists of a single feature compared to an absolute value.

Next to these elementary features, we include features that are derived from previous models. For each of the five notes, the following values are included as feature:

- the sum of the values for the quantised GTTM GPRs 2a, 2b, 3a, and 3d, as defined in [10];

- the Local Boundary Strength as computed by the LBDM [7];

- the values for pitch proximity and pitch reversal as defined in [13];

- the prediction of Grouper [8];

- the information content as computed by the IDyOM model according to [19];

- features that are based on the conditions of closure as stated by Narmour [11, p. 11]: the metric weight contour for the third note, whether the third note is longer than the second, whether the interval between the first and second notes is larger than the interval between the second and third notes, and whether the direction of the melodic contour changes between the second and the third note.

Finally, we include several features that are not independent of earlier annotated segment boundaries. In an inspection of classification results, it appeared that often a boundary very close to the beginning of a phrase was predicted. To prevent this, we include the distance between the third note of the 5-gram and the beginning of the phrase, both as number of notes, and as number of beats. Furthermore, we include a boolean feature that is True if the onset of the fourth note is at the same position in the bar as the onset of the first note in the phrase. Lastly, we include for each of the five notes whether a rest follows the note.

In total, we have 162 features (excluding the class label), 31 of which are lyric features.[3] The lyric features are only computed for the MTC dataset, since lyrics are not present in the ESSEN and CHOR collections.

4.2 Learning Algorithms

RIPPER [27] is a rule mining algorithm that infers a set of classification rules from a data set. The basic procedure that is implemented in this algorithm is to split the training data into two folds (1/3 and 2/3), grow a rule on the 2/3 split, prune the rule using the 1/3 split, and remove the objects that are covered by the rule from the training set. This is repeated until no objects remain in the training set. Each iteration results in a rule that is added to the rule set. The algorithm starts finding rules that target the minority class, which is appropriate for the segmentation problem in which phrase boundaries are a minority class. The resulting rules are not independent. To reach a classification, the rules have to be applied in the order as provided by the mining algorithm. The advantage of a rule-set as resulting model is its interpretability. From the rules it is clear how a classification is established.

One important parameter of the RIPPER algorithm is the minimum number of objects per rule. By setting this to a low value, many rules result that might be too specific, while setting this to a high value results in less, and more general rules. We found that in general for our purpose 32 is a sensible value. Smaller values lead to much more rules, without considerably improving classification performance. Furthermore, since songs in general have much less than 32 phrase boundaries, this value forces the algorithm to generalise over songs. We use the implementation that is provided in the Weka workbench [28].

To better show the potential of the feature set, we also use a Random Forest classifier [29]. During training, a large number of decision trees are fitted to random subsets of the data. The classification is a majority vote of these individual trees. The models that result from this approach are not easily interpretable, but they generally reach a higher classification performance compared to a single decision tree or rule set. We experimentally found that the optimal number of trees in the forest is around 40. Larger forests do not considerably add to the classification performance. We use the implementation as provided in the Python module sklearn [30]. For evaluation, we employ a 5-fold cross-validation procedure, both for RIPPER and Random Forest. To further raise the independence between test and train sets in case of the Random Forest, we make the splits between the train and test sets at the level of melody. Thus, the 5-grams from the same melody all are either in the test or in the train set. For MTC, this implies that also tune families are always separated, while for EFSC and CHOR this cannot be guaranteed. The code for this paper is publicly available.[4]

5. RESULTS

Table 2 shows the classification results for the three datasets, the two classifiers, and various feature subsets.

5.1 General Remarks

The separate groups of elementary features (pitch, rhythm, lyrics) only reach moderate performance. Rhythm features consistently score better than pitch features. Lyric features clearly have considerable discriminative power,

---

[3] The full feature set is included in the supplementary material.

| | MTC | | | | | |
| | RIPPER | | | Random Forest | | |
Features	**Pr**	**Rc**	**F1**	**Pr**	**Rc**	**F1**
El. Pitch	0.58	0.17	0.26	0.43	0.26	0.32
El. Rhythm	0.75	0.53	0.62	0.72	0.57	0.63
El. Lyrics	0.64	0.38	0.48	0.56	0.43	0.49
El. NoLyr	0.73	0.61	0.67	0.80	0.58	0.68
El. All	0.77	0.73	0.75	0.85	0.69	0.76
Prev.	0.81	0.62	0.70	0.83	0.62	0.71
NoLyr	0.79	0.66	0.72	0.86	0.64	0.73
All	**0.82**	**0.76**	**0.79**	**0.89**	**0.72**	**0.80**
NoLyr+GT	0.84	0.80	0.82	0.90	0.76	0.82
All+GT	0.86	0.87	0.87	0.92	0.82	0.87

| | EFSC | | | | | |
| | RIPPER | | | Random Forest | | |
Features	**Pr**	**Rc**	**F1**	**Pr**	**Rc**	**F1**
El. Pitch	0.57	0.18	0.27	0.49	0.31	0.38
El. Rhythm	0.78	0.53	0.63	0.77	0.62	0.69
El. Lyrics	-	-	-	-	-	-
El. NoLyr	0.78	0.63	0.69	0.83	0.69	0.76
El. All	-	-	-	-	-	-
Prev.	0.81	0.66	0.73	0.88	0.64	0.74
NoLyr	**0.83**	**0.68**	**0.75**	**0.90**	**0.70**	**0.79**
All	-	-	-	-	-	-
NoLyr+GT	0.90	0.88	0.89	0.95	0.87	0.90
All+GT	-	-	-	-	-	-

| | CHOR | | | | | |
| | RIPPER | | | Random Forest | | |
Features	**Pr**	**Rc**	**F1**	**Pr**	**Rc**	**F1**
El. Pitch	0,68	0.49	0.57	0.77	0.65	0.71
El. Rhythm	0.76	0.66	0.71	0.84	0.69	0.76
El. Lyrics	-	-	-	-	-	-
El. NoLyr	**0.84**	**0.75**	**0.79**	**0.94**	**0.85**	**0.89**
El. All	-	-	-	-	-	-
Prev.	0.81	0.73	0.77	0.93	0.82	0.87
NoLyr	**0.85**	**0.77**	**0.81**	**0.95**	**0.86**	**0.90**
All	-	-	-	-	-	-
NoLyr+GT	0.94	0.84	0.89	0.98	0.91	0.94
All+GT	-	-	-	-	-	-

Table 2. Classification results (precision, recall, and F1 for the boundary class) on MTC, EFSC, and CHOR for various feature subsets, both for the rule miner (RIPPER) and for the Random Forest classifier. "El." denotes the elementary features. "NoLyr" denotes all features except for the lyrics features. "Prev." denotes the features from previous models. "GT" denotes the group of features that are not independent of the annotated phrase boundaries.

as is observable in the increase of the recall between the "El. NoLyr" and "El. All" subsets for MTC.

Comparing the performance between using elementary features only and using all features shows some improvement in the later case for MTC and EFSC, but not for CHOR. The "Prev." subset on its own consistently shows a good performance. This implies that the explainable power of the elementary features is comparable to the explainable power of the previous models. A large part of the boundaries remains unexplained with either which method.

Overall, MTC is the hardest to classify. Undoubtedly, this is a consequence of the careful compilation, ensuring only one melody per tune family. Since we have no tune family labels for EFSC and CHOR, the independence of

train and test sets cannot be fully guaranteed. Therefore, the classification results might be too optimistic.

5.2 Rule Sets

The contents of the rules as found by the RIPPER algorithm reveals which features are paramount in detecting phrase boundaries. Although all rule sets give rise to interesting observations, it is not possible to discuss them all within the scope of this article. We show for two cases the first few rules, which typically cover many objects. As these rules are not directly derived from a theory of melodic perception, we are specifically interested to see to what extent the rules confirm existing understanding of melodic closure. Furthermore, these rules have the potential to lead to new hypotheses about what establishes closure in a melody.

First, we focus on the cases in which only elementary features are used. These are the first three discovered rules for MTC using the elementary pitch and rhythm features: [5]

```
Rule 0:
   (IOIbeatfractionthirdfourth = -) and
   (completesmeasuresong = True) and
   (IOIbeatfractionthird >= 1.25) and
   (meternumerator >= 4) and
   (IOIbeatfractionfirst <= 0.666667)
   => class=boundary (739.0/54.0)
Rule 1:
   (IOIbeatfractionthirdfourth = -) and
   (completesmeasuresong = True) and
   (IOIbeatfractionthird >= 1) and
   (IOIbeatfractionsecondthird = +) and
   (beatstrengthfourth >= 1)
   => class=boundary (705.0/88.0)
Rule 2:
   (IOIbeatfractionthirdfourth = -) and
   (completesmeasuresong = True) and
   (IOIbeatfractionthird >= 1.25) and
   (IOIbeatfractionfifth <= 1.5) and
   (VosHarmonyfourth >= 4) and
   (intervalsecond <= 0) and
   (diatonicpitchthird <= 30)
   => class=boundary (272.0/15.0)
```

Rule 0 classifies 739 5-grams from the train set correctly, and additionally covers 54 false positives. `IOIbeatfraction` denotes the duration of the note in units of the beat-length. The first rule mainly states that the fourth note should be shorter than the third, the third note ends at the position in the measure that is parallel to the start of the first note of the melody, the third note is longer than the beat (>=1.25 times), the first note is fairly short, and the `meternumerator` is 4. The last condition excludes all songs in e.g., 6/8 or 3/4 meter. The conditions that have been selected for these rules confirm considerations for several previous models. One of the central properties of a phrase-closing note seems to be its length, which should be longer than the beat. Furthermore, these rules all state that the length of the fourth note should be shorter than the third. This condition is present in 22 of the 30 discovered rules in this set. It contrasts with one of Narmour's conditions of closure [11, p. 11], which states that the closing note is longer than the previous note. Apparently, the data indicates that the relation with the next

---

[5] The full rule set is included in the supplemental material.

Dataset	RIPPER	Random Forest	IDyOM	Grouper	LBDM	Rest	Always
MTC	0.73 0.61 **0.67**	0.80 0.58 **0.68**	0.65 0.51 **0.57**	0.69 0.67 **0.68**	0.60 0.51 **0.55**	0.92 0.26 **0.40**	0.10 1.00 **0.18**
EFSC	0.78 0.63 **0.69**	0.83 0.69 **0.76**	0.71 0.49 **0.58**	0.70 0.61 **0.65**	0.60 0.47 **0.53**	0.96 0.31 **0.47**	0.11 1.00 **0.20**
CHOR	0.84 0.75 **0.79**	0.94 0.85 **0.89**	0.61 0.39 **0.47**	0.64 0.59 **0.62**	0.48 0.42 **0.45**	0.99 0.09 **0.17**	0.11 1.00 **0.20**

Table 3. Classification performance of related models. For each model, precision, recall, and F1 (bold) for the boundary class are reported. Results for RIPPER and Random Forest are for the featureset El. NoLyr.

note is more indicative, instead. One could speculate that the perception of closure at the third note is reinforced in retrospective when noticing that the next note is shorter. The rules that are found for the EFSC bear a similarity to those for MTC. The rules for CHOR differ more. But for all three data sets rhythm features dominate the top rules. This confirms earlier results as reported by Weyde [31]. The pitch features that are included mainly refer to pitch contour and the level of dissonance of the melodic interval (as registered by the features based on [25]).

Next, we consider the top rules that are discovered for MTC with the feature subset of all features ("All"): [6]

```
Rule 0:
  (grouperthird = True) and
  (rhymesthird = True) and
  (lbdmthird >= 0.280929)
  => class=boundary (2413.0/149.0)
Rule 1:
  (grouperthird = True) and
  (wordendthird = True) and
  (informationcontentfourth >= 7.252784) and
  (contourthird = -) and
  (lbdmfifth <= 0.159635)
  => class=boundary (641.0/33.0)
```

It is clear that the combined models of LBDM and Grouper, and the condition of rhyme constitute a very powerful rule that covers 2,413 boundary 5-grams in the training set, and only 149 noboundary 5-grams. In the second rule, also the information content as computed by IDyOM plays a role. But also some elementary features are used.

5.3 Existing Models

Table 3 shows a comparison with the performance of several existing models. The values for the RIPPER and Random Forest classifiers are those for the set of elementary features without the lyrics. This is not the best performing feature subset, but the larger subset would include IDyOM, Grouper and LBDM as features, which would not render a fair comparison. The IDyOM segmentation is computed with the implementation of IDyOM as available on GitHub. [7] Grouper is available as part of the Melisma Music Analyzer. [8] LBDM is implemented according to [7]. The threshold for peak-picking is chosen such that the resulting F1-value is maximised. The Rest model assumes a phrase boundary wherever a rest is notated in the score. This quantifies the effect of including the rest as a feature in a segmentation model that is evaluated on a collection of folk song melodies. As can be seen, the rest model typically results in high-precision, low-recall segmentation.

---

[6] The full rule set is included in the supplemental material.
[7] http://mtpearce.github.io/idyom/
[8] https://www.link.cs.cmu.edu/music-analysis/

Dataset	MTC	ESFC	CHOR
MTC	0.80 0.58 **0.68**	0.83 0.57 **0.67**	0.85 0.49 **0.62**
ESFC	0.76 0.61 **0.68**	0.83 0.69 **0.76**	0.83 0.68 **0.74**
CHOR	0.77 0.32 **0.45**	0.80 0.37 **0.51**	0.95 0.86 **0.90**

Table 4. Performance of cross-evaluation. The rows show the train sets, the columns the test sets. The values are: precision recall F1 (bold) for the boundary class.

For both MTC and EFSC the occurrence of a rest explains around 30% of the phrase boundaries. This is a considerable effect. Finally, a baseline model is included that classifies each note as a phrase boundary. The Random Forest classifier with the elementary feature set outperforms the other methods for ESFC and CHOR, and performs comparable to Grouper on MTC, although precision and recall are more balanced with Grouper. It also outperforms Lattner's RBM approach on EFSC (0.80 0.55 **0.63**) [20]. Notably the recall is higher. However, the currently presented approach is supervised and uses more features.

5.4 Cross Relations

We now examine the performance of the classifiers on the datasets they are not trained on. We use the Random Forest classifier and the feature subset "El. NoLyr". Most notable in the results as shown in Table 4 is the comparable cross-performance between ESFC and MTC. Apparently, the phrase endings in the Dutch and German folk song styles have comparable properties. The higher self-performance of ESFC might partly be caused by the tune-family problem (Section 3.1.1). The low performances of the classifiers trained on CHOR are mainly caused by low recall. This could indicate that some types of phrase endings that occur in MTC and ESFC are absent in CHOR.

6. CONCLUSION

We presented an approach to melodic segmentation that builds on and integrates elementary melodic features and existing segmentation models in a theory-agnostic way. By deriving a rule-set using a large number of features, we get an indication of which features are crucial for detecting phrase boundaries in melodies. A notable observation is that a phrase boundary is mainly detectable with rhythm features. By employing a Random Forest classification, we get an indication of the discriminative power of the considered feature sets. The resulting classifier outperforms all earlier approaches to the problem of automatic melodic segmentation. By cross-evaluation, we detect a connection between MTC and EFSC.

7. REFERENCES

[1] D. Conklin, "Melodic analysis with segment classes," *Machine Learning*, 2006.

[2] H. Schaffrath, Ed., *The Essen Folksong Collection*. Stanford, CA: Center for Computer Assisted Research in the Humanities, 1995.

[3] K. Koffka, *Principles of Gestalt psychology*. Harcourt, Brace and Company, 1935.

[4] L. B. Meyer, *Emotion and Meaning in Music*. Chicago: University of Chicago Press, 1956.

[5] J. Tenney and L. Polansky, "Temporal gestalt perception in music," *Journal of Music Theory*, vol. 24, no. 2, pp. 205–241, 1980.

[6] E. Cambouropoulos, "Musical rhythm: A formal model for determining local boundaries, accents and metre in a melodic surface," in *Music, Gestalt, and Computing: Studies in Cognitive and Systematic Musicology*, M. Leman, Ed. Springer Verlag, 1997, pp. 277–293.

[7] ——, "The local boundary detection model (lbdm) and its application in the study of expressive timing," in *Proc. of the Intl. Computer Music Conf*, 2001.

[8] D. Temperley, *The Cognition of Basic Musical Structures*. Cambridge, MA: MIT Press, 2001.

[9] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. Cambridge, Mass.: MIT Press, 1983.

[10] B. W. Frankland and A. J. Cohen, "Parsing of melody: Quantification and testing of the local grouping rules of lerdahl and jackendoff's a generative theory of tonal music," *Music Perception*, vol. 21, no. 4, pp. 499–543, 2004.

[11] E. Narmour, *The Analysis and Cognition of Basic Melodic Structures - The Implication-Realization Model*. Chicago and Londen: The University of Chicago Press, 1990.

[12] ——, *The Analysis and Cognition of Melodic Complexity: the Implication-Realization model*. Chicago: University of Chicago Press, 1992.

[13] E. G. Schellenberg, "Expectancy in melody: tests of the implication-realization model," *Cognition*, vol. 58, no. 1, pp. 75–125, 1996.

[14] M. E. Rodríguez-Lopez, "Automatic melody segmentation," Ph.D. dissertation, Utrecht University, Utrecht, 2016.

[15] Z. Cenkerová, M. Hartmann, and P. Toiviainen, "Crossing phrase boundaries in music," in *Proceedings of the 15th Sound and Music Computing Conference 2018*, 2018, pp. 66–71.

[16] R. Bod, "Memory-based models of melodic analysis: Challenging the gestalt principles," *Journal of New Music Research*, vol. 31, no. 1, pp. 27–37, 2002.

[17] Z. Juhász, "Segmentation of hungarian folk songs using an entropy-based learning system," *Journal of New Music Research*, vol. 33, no. 1, pp. 5–15, 2004. [Online]. Available: http://dx.doi.org/10.1076/jnmr.33.1.5.35395

[18] D. Conklin and I. H. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.

[19] M. T. Pearce, D. Müllensiefen, and G. A. Wiggins, "Melodic grouping in music information retrieval: New methods and applications," in *Advances in Music Information Retrieval*, ser. Studies in Computational Intelligence, Z. W. Raś and A. A. Wieczorkowska, Eds. Berlin, Heidelberg: Springer, 2010, vol. 274, pp. 364–388.

[20] S. Lattner, C. E. Cancino Chacón, and M. Grachten, "Pseudo-supervised training improves unsupervised melody segmentation," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, Q. Yang and M. J. Wooldridge, Eds., 2015, pp. 2459–2465.

[21] P. Van Kranenburg and M. De Bruin, "The meertens tune collections: Mtc-fs-inst 2.0," Meertens Institute, Amsterdam, Meertens Online Reports 2019-1, 2019.

[22] R. Elbourne, "The question of definition," *Yearbook of the International Folk Music Council*, vol. 7, pp. 9–29, 1975.

[23] S. Bayard, "Prolegomena to a study of the principal melodic families of british-american folk song," *Journal of American Folklore*, vol. 63, no. 247, pp. 1–44, 1950.

[24] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," in *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR 2010)*, 2010, pp. 637–642.

[25] P. G. Vos and D. Pasveer, "Goodness ratings of melodic openings and closures," *Perception & Psychophysics*, vol. 64, no. 4, pp. 631–639, 2002.

[26] P. Van Kranenburg and F. Karsdorp, "Cadence detection in western traditional stanzaic songs using melodic and textual features," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, Taipei, 2014, pp. 391–396.

[27] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 115–123.

[28] E. Frank, M. A. Hall, and I. H. Witten, *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*, fourth edition ed. Morgan Kaufmann, 2016.

[29] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] T. Weyde, "On the influence of pitch on melodic segmentation," in *Proceedings of the Fifth International Conference on Music Information Re- trieval*, Barcelona, 2004.

COMBINING MUSICAL FEATURES FOR COVER DETECTION

Guillaume Doras[1,2]   **Furkan Yesiler**[3]

Joan Serrà[4]   **Emilia Gómez**[3,5]   **Geoffroy Peeters**[6]

[1]Sacem, France [2]Ircam, CNRS, Sorbonne Université, STMS Lab, France
[3]Music Technology Group, Universitat Pompeu Fabra, Spain [4]Dolby Laboratories, Spain
[5]European Commission, Joint Research Centre, Spain [6]Telecom Paris, LTCI, France

`guillaume.doras@ircam.fr, furkan.yesiler@upf.edu`

ABSTRACT

Recent work have addressed the automatic cover detection problem from a metric learning perspective. They employ different input representations, aiming to exploit melodic or harmonic characteristics of songs and yield promising performances. In this work, we propose a comparative study of these different representations and show that systems combining melodic and harmonic features drastically outperform those relying on a single input representation. We illustrate how these features complement each other with both quantitative and qualitative analyses. We finally investigate various fusion schemes and propose methods yielding state-of-the-art performances on two publicly-available large datasets.

1. INTRODUCTION

Music retrieval has come a long way in the last 25 years. Since the earlier works on symbolic music retrieval [1, 2], applications with increasing complexity have been developed. In the mid-1990's, query-by-humming aimed at retrieving songs based on melodic similarity with a short hummed or whistled audio excerpt [3,4], while fingerprinting in the early 2000's aimed at identifying a song based on one of its excerpts [5]. Music matching at large – the task of retrieving an excerpt based on its musical similarity with another – was developed in the mid-2000's, typically comparing sequences of harmonic features via dynamic programming methods [6–8].

Automatic cover detection – the task of retrieving covers of a given track from an audio corpora – emerged at the same period, and was largely inspired by the previous decade of music retrieval research. Some of the early cover detection systems were relying on dominant melody to assess musical similarity [9, 10], and one of them reached the 3rd place (out of 8 participants) at the first MIREX [1]

---

[1] https://www.music-ir.org/mirex

cover song identification contest in 2006. The same year however, 1st and 2nd ranking algorithms were relying on harmonic features – chroma vectors or estimated chords series [11–13]. These results seem to have fostered the use of harmonic representations – chroma in particular – over melodic ones for cover detection, and all algorithms submitted to the next 2007 MIREX edition were using a tonal representation [14–16]. Enhanced chroma and time series comparison via dynamic programming then became the *de facto* standard method in the field – and remained the state of the art for more than a decade [17, 18].
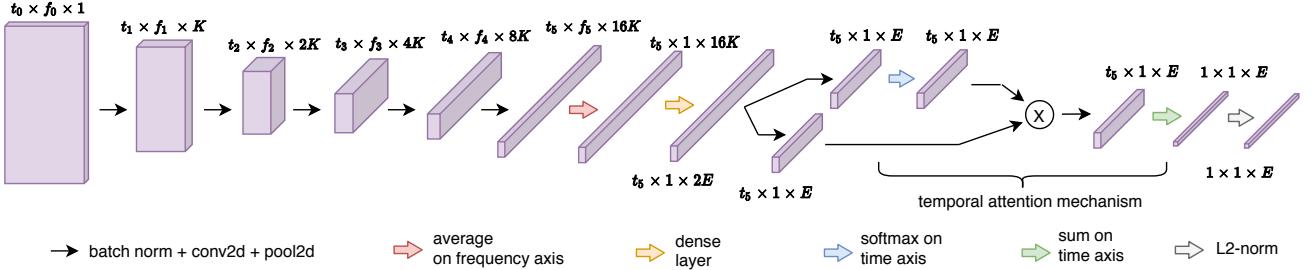
During the following years, the community focused on improving both accuracy and scalability of existing approaches. As to accuracy, it was proposed to aggregate the results obtained with different methods [19–21] or different input features [22–24]. As to scalability, several strategies were investigated to compress the original representations and to reduce the similarity comparison function to a lightweight distance computation [25–28] or a fast lookup operation in a database index [29, 30].

The advent of efficient machine learning methods in other fields – such as image recognition – encouraged the community to shift from these previous methods based on ad-hoc and handcrafted features toward a new approach based on data-driven feature learning [31–33]. Recently, promising results were obtained using the metric learning paradigm in a cover detection context. The principle is to train a neural network to represent each track as a compact vector – its embedding – so that the distance between embeddings of a cover pair is smaller than that of non-cover pairs. Features used as input data were as varied as the Constant-Q Transform [34], dominant melody or multi-pitch [35, 36] or chord-informed chroma [37].

In this work, we propose a comparative study of these input features and investigate their combinations to improve cover detection performance. In Section 2, we briefly review different works inspiring our approach. In Section 3, we present the features that we consider for this study and their respective performances. In Section 4, we discuss the results obtained using different combinations of these features with a simple averaging fusion scheme, and explain them with a qualitative analysis. We then propose in Section 5 an architecture able to learn to combine various features efficiently. We conclude with future potential improvements.

Figure 1: MICE architecture.

2. RELATED WORK

We present here the main concepts inspiring this work: input features combination and metric learning paradigm.

2.1 Combination of input features

A first attempt to combine information from various input features for cover detection was made by Foucard et al. using a source separation algorithm to obtain three inputs: the mixed original track, the dominant melody – assumed to correspond mainly to the solo singing voice – and the accompaniment [22]. In another study, Salamon et al. argued that, albeit closely related, dominant melody, bass line and harmonic progression embed different and complementary information. To prove this idea, they proposed to compare the systems that use each feature separately and their combinations with different fusion schemes [23]. More recently, Tralie et al. investigated another multi-representation approach, fusing harmonic and timbral features [24]. These studies showed that systems combining several input features outperformed those using each feature individually.

2.2 Classification vs. metric learning paradigm

Different teams recently proposed data-driven feature learning methods to address the cover detection problem. A common approach is to use a Convolutional Neural Network (CNN) to extract a compact representation – an embedding – from a low- or mid-level spectral representation of the audio, for instance Harmonic Pitch Class Profile (HPCP) [38] or Constant-Q Transform (CQT) [34, 39]. These authors considered the problem as a classification task, introducing an additional dense layer as a classifier.

Similarly, Doras et al. used dominant melody or multi-pitch representations [35, 40], while Yesiler et al. used crema-PCP [41], a chord-informed chroma representation [37] to extract the embedding. These input features were obtained with specialized neural networks [36, 41]. These authors also adopted a metric learning approach in which the CNN is trained with a triplet loss to produce embeddings whose pairwise Euclidean distance is lower for covers than for non-covers. Using these melodic or harmonic input features along with the metric learning paradigm yielded promising results and inspired this present work.

3. COMPARING INPUT FEATURES

We compare here performances obtained with a full spectral feature (CQT), two melodic features (dominant melody and multi-pitch) and two harmonic features (chroma and crema-PCP). For brevity, we denote them Cq, Dm, Mp, Ch, and Cp, respectively.

3.1 Input features

We computed Cq and Ch using Librosa v0.7 [42]. We obtained Dm and Mp with the convolutional network we previously described in [36], and we obtained Cp with the model publicly released by [41], as done in [37].

Temporal resolution All features were computed for an audio duration of 180 seconds as in [35], with a frame duration of 93ms (1937 bins). For tracks longer that 180 seconds, the beginning of the 180 seconds is taken at random, while shorter tracks were zero-padded, as in [37].

Frequency resolution All features were computed with a resolution of 1 bin per semi-tone. Cq was computed across 6 octaves. Dm and Mp are originally extracted with a resolution of 5 bins per semi-tone and their resolution is downsampled by a factor 5 via 2D-interpolation, following [35]. For each Dm, only the 3 octaves around its mean pitch are considered, as done in [35]. To account for all possible circular shifts in chroma features, we concatenate on top of the Ch and Cp their 11 lowest frequency bins, following [37, 38]. To summarize: Cq, Dm, Mp, Ch and Cp have 72, 36, 72, 23 and 23 frequency bins, respectively.

Normalization Cq and Ch are log-compressed and trimmed at -80dB. Finally, each feature is globally normalized between 0 and 1.

3.2 Model

Yesiler et al. introduced MOVE, a network containing a convolutional part specially designed to capture Cp patterns and a temporal attention part [37, 43], while Doras et al. used a plain convolutional network to capture Dm and Mp patterns [35, 40]. We introduce here a new model that reuses the plain convolutional part of the latter and the temporal attention mechanism of the earlier. The rationale behind this architecture is twofold: we need a generic model that can be used for all types of input features in order to conduct fair performance comparisons, and we observed in preliminary experiments that the temporal attention mechanism improves the results of the plain CNN. We call this model MICE (Musically Informed Cover Embeddings).

As shown on Figure 1, the first part of the model is the 5-layer CNN of [35]. Each layer block consists of a batch normalization layer, a convolution layer with 3×3 kernels and a mean-pooling layer with 2×2 kernel and 2×2 stride. The number of kernels K of the first layer is doubled at each level. Output is then averaged along the frequency axis, and a dense layer is applied to output a number of channels of $2E$, where E is the final embedding size.

The attention mechanism is then introduced: the tensor is split in 2 on its channels dimension to obtain two tensors of E channels. A softmax function is then applied on the time axis for the first tensor, and the output is multiplied element-wise with the second tensor. The resulting values are then summed along the time axis, which gives a vector of size E. The softmax followed by the multiplication and the sum implements a weighted average along the time axis per channel. The network is thus trained to give preference to the parts along the time dimension that are the most relevant to meet the objective function. The embedding vector is then L2-normalized. We used K=64 and E=512.

3.3 Experiments

In these first experiments, we train a different instantiation of MICE for each type of input feature and evaluate their cover detection performances.

Datasets We used the publicly available training set SHS_{5+} [2], containing Cq, Dm, Mp, Ch and Cp features for ~62k covers of ~7.5k works. It was split into a training/validation set with a ratio of 80/20 with respect to the works, i.e. all covers of a given work belong to one or the other set. We tested our model for each feature with two publicly available test datasets: SHS_{4-} [2], containing ~50k covers of ~20k works, and Da-TACOS [3], containing 13k covers of 1k works and 2k confusing tracks [44].

Loss We used a triplet loss to train this network [45]. Formally, if we let $\{a, p, n\}$ denote a triplet of track embeddings, where a is an anchor, and p or n is one of its covers or non-covers, respectively, the loss to minimize is expressed as $L = \max(0, d_{ap} + \alpha - d_{an})$, where α is a margin and d_{ap} and d_{an} are the distances between anchor a and p or n, respectively. We set $\alpha = 1$.

In practice, we used online semi-hard negative pairs mining [46], where triplets are built within each training batch: instead of using all possible triplets, each track in the batch is successively considered as an anchor, and compared with all its covers in the batch. For each of these positives pairs, if there are negatives such as $d_{an} < d_{ap}$, only the one with the highest d_{an} is kept. If no such negative exists, only the one with the lowest d_{an} is kept. Other negatives are not considered.

Training We train MICE with Adam optimizer [47], with an initial learning rate of $1e^{-4}$, divided by 2 each time the loss on the validation set has not decreased after 5k training steps. Training is stopped after 50k steps, or if the learning rate falls below $1e^{-7}$. The batch size is 64.

---

[2] https://gdoras.github.io/topics/coversdataset
[3] https://github.com/MTG/da-tacos

Testing For each feature, we use the corresponding trained model to compute the embeddings on the two test datasets. For SHS_{4-}, one cover per work is used as a query against the entire test set to compute a 20k×50k distance matrix. For Da-TACOS, each cover is used as a query against the entire dataset to compute a 13k×15k distance matrix. The Mean Average Precision (MAP), the mean number of correct answers in the ten first answers (MT@10) and the mean rank of first correct answer (MR1) are then computed.

3.4 Quantitative analysis

We report in Table 1 the performance scores obtained on Da-TACOS and SHS_{4-} for each type of input feature.

Input	Da-TACOS			SHS_{4-}		
	MAP	MT@10	MR1	MAP	MT@10	MR1
Cq	0.215	2.468	94	0.397	0.718	886
Dm	0.311	3.521	111	0.412	0.722	1431
Mp	0.293	3.290	**71**	0.422	0.760	**862**
Ch	0.121	1.476	117	0.174	0.371	1465
Cp	**0.375**	**4.084**	86	**0.499**	**0.842**	1169

Table 1: Results on SHS_{4-} and Da-TACOS for each feature.

Consistently, the Cp yields by far the best results, followed by the Dm and the Mp. This confirms our initial intuition that both melodic line and harmonic progression are prominent common musical facets between covers. Cq, representing the full spectrum, yields lower performance, which suggests that, albeit also embedded in the spectrum, the melodic and harmonic information is obfuscated, e.g. by percussive sounds information. Finally, the tonal information embedded in the Ch does not seem to be efficiently caught by our model.

From a practical point of view, crema-PCP is probably the best feature among those considered in this work, as it yields the best results with the lowest memory footprint.

4. COMBINING INPUT FEATURES
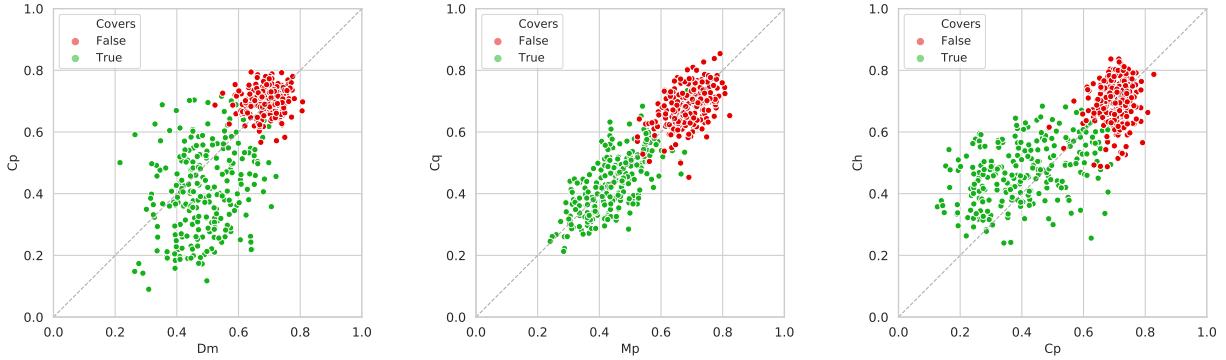
In this set of experiments, we now investigate if combining different features can improve the performance of each feature considered individually.

4.1 Are features complementary ?

We first compare pairwise embedding distances computed for the same pairs of tracks but obtained with different input features, as shown on Figure 2. The leftmost plot for instance compares the pairwise distances obtained for Dm and Cp. If each track's embeddings extracted from different input features were carrying the same information, the pairwise distance would be the same for a given pair of tracks, independently of the feature used. Figure 2 shows on the contrary that the same pair of tracks can obtain a low distance when using a given input feature, but a notably higher distance when using another one.

Figure 2: Comparison of the normalized distance obtained for the same pairs from SHS_{4-} (cover pairs in green and non-cover pairs in red) with different features: Dm vs. Cp (left), Mp vs. Cq (middle), Cp vs. Ch (right). Other combinations are not shown due to space constraints. For clarity, only 500 pairs randomly picked are drawn (250 covers and 250 non-covers).

All features seem relatively consistent when labeling non-cover pairs (red points exhibit high distances on both axes). Conversely, cover pairs (green points) are more scattered. Dm and Cp in particular seem to give very distinct results, as many pairs are spread far from the diagonal, which means that some cover pairs are more efficiently scored by one or the other feature. Intuitively, it seems logical that Dm and Cp are encoding complementary melodic and harmonic facets. This suggests that combining these features could benefit of this complementarity. We now conduct a quantitative and a qualitative analysis to confirm this intuition and to understand why certain representations yield better results for certain songs and vice-versa.

4.2 Quantitative analysis

We first experiment with a simple fusion scheme, which consists of averaging the pairwise distances obtained for the same pair with different features. We then re-compute the evaluation metrics based on this new averaged distance matrix for each possible feature combination. The rationale behind this approach is that we expect pairs incorrectly clustered with one representation to benefit from the correct clustering obtained with another representation.

The results are summarized in Table 2 for all combinations of Cq, Dm, Mp and Cp representations (we did not consider Ch here). We also computed the scores obtained by an oracle, which always picks among the distances obtained for each feature the lowest (resp. highest) distance for positive (resp. negative) pairs.

It appears clearly that any combination yields a better performance than any feature isolated (see Table 1). It also appears that the combinations where the Cp is used yield higher scores than the others, which was expected as Cp alone was already obtaining the highest scores. But more interestingly, we observe that the best improvements are obtained when combining melodic and harmonic features, i.e. Dm+Cp or Mp+Cp. The Mp probably embeds some of the information also carried by the Cp, as the improvement is lower when combining Mp+Cp than Dm+Cp.

All in all, the combination Dm+Cp yields the best performances, and an improvement of 15%-20% compared to Dm or Cp alone. Considering a third feature along Dm+Cp

Test set	Da-TACOS			SHS_{4-}		
Input	MAP	MT@10	MR1	MAP	MT@10	MR1
Cq+Dm	0.359	4.002	62	0.590	0.982	567
Cq+Mp	0.324	3.603	62	0.530	0.909	623
Cq+Cp	0.427	4.636	46	0.621	1.024	581
Dm+Mp	0.394	4.347	61	0.571	0.956	614
Dm+Cp	**0.547**	**5.861**	**37**	**0.679**	**1.098**	**529**
Mp+Cp	0.496	5.330	40	0.627	1.034	593
Cq+Dm+Mp	0.403	4.434	51	0.624	1.030	498
Cq+Dm+Cp	0.524	5.640	36	**0.713**	**1.143**	**430**
Cq+Mp+Cp	0.480	5.184	40	0.660	1.078	505
Dm+Mp+Cp	**0.553**	**5.939**	**35**	0.702	1.133	453
Dm+Cp (O)	0.800	8.360	4	0.873	1.344	115
Cq+Dm+Cp (O)	0.881	9.072	1	0.935	1.419	51
Dm+Mp+Cp (O)	0.874	9.022	2	0.924	1.405	63

Table 2: Comparison on Da-TACOS and SHS_{4-} of input feature combinations. Results obtained with the embeddings produced by MICE architecture trained for each feature (O=Oracle).

(Cq or Mp) improves the results slightly further.
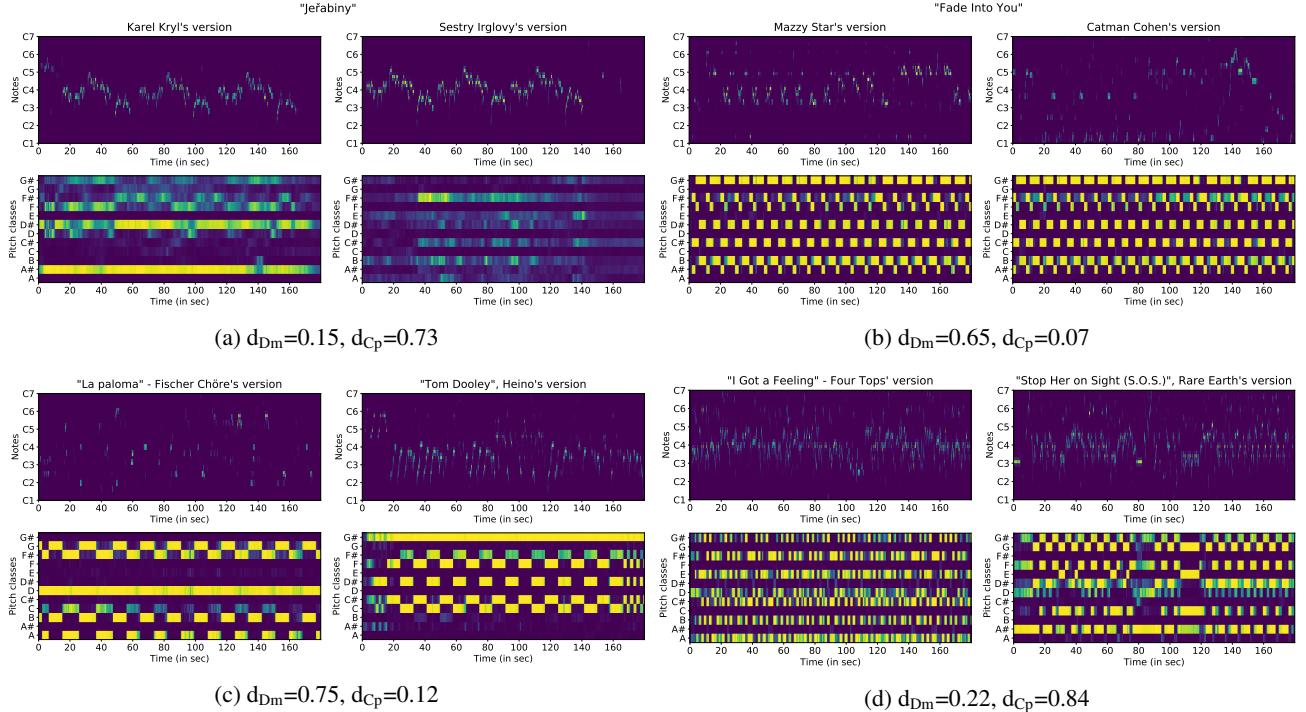
We also observe that the oracle scores about 20% above the highest scores obtained with the averaging fusion scheme, which suggests that further improvements are theoretically possible (we also experimented a minimum fusion scheme, which yielded lower performances).

From a practical perspective (e.g. memory footprint), the best trade-off seems to concentrate only on the Dm and the Cp. We will now investigate why the combination of these two features yields a better performance than others.

4.3 Qualitative analysis

To this aim, we selected the tracks where the first feature (e.g. Dm) gives particularly correct results and where the second feature (e.g. Cp) gives particularly incorrect results, or vice-versa. In other terms, we analyzed the pairs of songs for which the two features would give the most contradictory results for positive and negative pairs. The Dm and the Cp obtained for some of these cover and non-cover pairs [4] are shown on Figure 3.

---

[4] The audio of the songs described here can be listened on Youtube with the following IDs: Figure 3(a) clBw3cWgPnE and PNQeBX-tUdgc, Figure 3(b) -uJ61jgFCMM and xXvPFsoNnD4, Figure 3(c) 7nPBAiE76qY and bRrVMte9IQQ, Figure 3(d) pFrTXGEmU2Q and 3IOD9SqSfY4. Last accessed 11/5/2020.

(a) d_{Dm}=0.15, d_{Cp}=0.73

(b) d_{Dm}=0.65, d_{Cp}=0.07

(c) d_{Dm}=0.75, d_{Cp}=0.12

(d) d_{Dm}=0.22, d_{Cp}=0.84

Figure 3: Examples of cover pairs (top, (a) and (b)) and non-covers pairs (below, (c) and (d) where Dm and Cp gives contradictory results due to the melodic or harmonic content of each version. For each pair, Dm is displayed above and Cp below, and the corresponding distances d_{Dm} and d_{Cp} obtained for each feature are indicated.

Cover pairs Figure 3(a) displays two versions of "Jeřabiny", by Czech composer Karel Kryl (left, singing voice and guitar accompaniment) and Sestry Irglovy (right, purely a cappella, and poorly caught by the Cp). The pair is identified as covers thanks to the dominant melody.

Figure 3(b) displays two versions of "Fade Into You", by Mazzy Star (left) and Catman Cohen (right). The accompaniment is similar, but Catman Cohen's voice is very hoarse and rough, thus poorly caught by the dominant melody. The pair is identified as covers thanks to the Cp.

Non-cover pairs Figure 3(c) displays two different tracks: "La paloma" interpreted by a choir (left, mainly choir voices) and "Tom Dooley" by German singer Heino (right, voice and guitar accompaniment). Both songs share the same succession of two chords (but transposed), so the Cp are very similar. The pair is identified as non-covers thanks to the Dm, which are different.

Figure 3(d) displays two different tracks: "I Got a Feeling", by Four Tops (left) and "Stop Her on Sight (S.O.S.)" by Rare Earth (right). Both songs exhibit leading voice, backing voices, piano or strings section, and a brass instruments section. Both Dm appear very confused and look similar. The pair is identified as non-covers thanks to the Cp, which are different.

We could intuitively expect these results: Dm is better suited for songs where a melody is clearly prominent, while Cp is better suited for songs where no clear melody is present or is hidden by a prominent accompaniment. As such, there is no "better" feature: they simply perform differently on different tracks, and their combination performs statistically better on large corpora than separately.
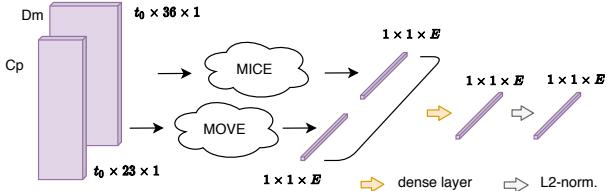
5. LEARNING TO COMBINE FEATURES

Despite its encouraging results, the simple averaging fusion scheme has two flaws. Firstly, it does not guarantee that averaging the distances is the most optimal manner to merge the information contained in different representations. Many tracks might end up scoring around the mean of all distances, which will not help to decide whether they are covers or not. Secondly, it requires to train one model per representation, and consequently to store one embedding per representation, which complicates the operational usage of the system (e.g. indexing various embeddings and combining several search results is sub-optimal). In this section, we study the possibility to train a single model to learn how to fuse several input features efficiently.

We consider here only the combination of Dm and Cp features, as they individually yielded the most promising results with the averaging fusion scheme, while remaining practical from a memory footprint perspective.

5.1 Late fusion scheme

To address these flaws, we propose a two-branch architecture, where each input feature is processed by a dedicated model into an embedding, as previously. These two embeddings are then concatenated and merged into a single one by a final dense layer. We can now use different models for each feature, as we are not comparing their individual performance as previously. In particular, we use MOVE to process the Cp, as it was specially designed for this feature, and keep MICE to process the Dm, as shown on Figure 4.

Figure 4: The late fusion architecture.

Let $\mathbf{e}_{Dm}$ and $\mathbf{e}_{Cp}$ denote the embeddings output by the Dm and Cp branches, $\mathbf{e}$ the final embedding and $\mathbf{W}$ the dense layer parameters. It comes:

$$\mathbf{e} = \mathbf{W}\begin{bmatrix}\mathbf{e}_{Dm}\\\mathbf{e}_{Cp}\end{bmatrix} = \mathbf{W}^{Dm}\mathbf{e}_{Dm} + \mathbf{W}^{Cp}\mathbf{e}_{Cp} \tag{1}$$

where $\mathbf{W}^{Dm}$ and $\mathbf{W}^{Cp}$ are the parameters of $\mathbf{W}$ that are applied to $\mathbf{e}_{Dm}$ and $\mathbf{e}_{Cp}$, respectively. Normalizing $\mathbf{e}$ to unit norm, we can interpret the embedding resulting from this fusion as a weighted mean of the initial embeddings moved to another location on the unit sphere to optimize the loss.

5.2 Experiments

We compare here three training options: **a)** each branch and the last layer are trained simultaneously with random initialization from scratch; **b)** each branch is first pre-trained individually with its corresponding input feature as previously; then their trained weights are reloaded in the late fusion architecture, and are fine tuned while training the last layer; **c)** is the same as **b)**, but the weights of each branch are frozen once reloaded in the fusion model, and only the weights of the final dense layer are learned.

For these three options, we train each architecture on the same proprietary training set that was used in [37]. This set contains 98k tracks and is much larger than the one used in features comparison experiments of Section 4.2. Models trained with this proprietary training set were evaluated with Da-TACOS in order to compare the results with the baseline established in [37]. We also conduct the same experiments for each architecture trained on SHS$_{5+}$ and evaluated with SHS$_{4-}$ as previously, in order to compare the results with the baseline established in [40].

The training procedure for all three options is the same as described in Section 3.3, except that the learning rate is initialized at $5e^{-6}$ for option **b)** and at $1e^{-1}$ for option **c)**.

5.3 Results

The results of the late fusion learning experiments are summarized on Table 3. We indicated the scores obtained for each feature (Dm and Cp) individually, as well as the corresponding distance averaging score for comparison.

For both sets, the two-branch model outperforms the ones where each feature is considered individually, which shows that it jointly learns from both input features. Late fusion with end-to-end training from scratch (option **a)**) scores below the other two options, which suggests that the model learns from each feature but does not make an optimal use of the available information.

Late fusion with fine tuning of the pre-trained branches (option **b)**) yields better results. However, it does not outperform the late fusion where only the dense layer is trained (option **c)**). A possible explanation could be that one feature tends to yield better results than the other (probably the Cp, as seen in Section 3), and allowing the update of branches might confuse the previously acquired knowledge of the weaker one. This assumption should however be investigated further in another work.

Input	Da-TACOS			SHS$_{4-}$		
	MAP	MT@10	MR1	MAP	MT@10	MR1
Dm (MICE)	0.360	4.032	94	0.412	0.722	1431
Cp (MOVE)	0.484	5.214	59	0.533	0.890	1188
Dm+Cp (A)	0.621	6.613	32	**0.697**	**1.120**	517
Dm+Cp (LF-a)	0.570	6.101	**29**	0.617	1.017	686
Dm+Cp (LF-b)	0.592	6.318	32	0.655	1.059	655
Dm+Cp (LF-c)	**0.635**	**6.744**	30	0.660	1.080	657
Doras et al. [40]	n/a	n/a	n/a	0.323	0.615	1476
Yesiler et al. [37]	0.507	-	40	n/a	n/a	n/a

Table 3: Comparison on Da-TACOS (resp. SHS$_{4-}$) of all fusion schemes trained on [37] proprietary training set (resp. SHS$_{5+}$). A=averaging, LF=Late fusion. Note that Cp and Dm+Cp (A) scores are higher here than in Table 2 because Cp is now processed by MOVE.

Training a dense layer on top of two pre-trained frozen branches (option **c)**) thus yields the best scores, similar to the ones obtained by the averaging scheme.

We finally compare these performances to the current state of the art for each set. We observe that all late fusion schemes notably outperform the results obtained in [40] and [37], for the same training and test sets.

6. CONCLUSION

We proposed in this work a comparative study of different input features that have been used in recent works addressing the cover detection problem with a metric learning approach. We observed that the best feature of the one we studied is the crema-PCP, a harmonic feature. We then showed that combining this feature with a dominant melody representation drastically improves the results compared to each feature considered alone. We showed that this can be explained by the fact that using both melodic and harmonic features helps to disambiguate pairs of tracks that don't have a clear melodic or harmonic structure. We finally proposed a late fusion scheme learning to combine input features, which yields to new state-of-the-art performances on two publicly available datasets.

This system could be improved in several ways. As suggested by the oracle results, further strategies could be developed to force the model to focus more adequately on the available features. Also, the need to maintain several dedicated branches in the late fusion scheme could be avoided with a single architecture merging the two branches earlier in the process. But perhaps more importantly, considering the variety of other features commonly shared by covers, such as lyrics, could be a fruitful strategy to build future cover detection systems.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] J. S. Downie, *Evaluating a simple approach to music information retrieval: Conceiving melodic n-grams as text*. Faculty of Graduate Studies, University of Western Ontario London, Ont., 1999.

[2] S. Doraisamy and S. M. Rüger, "An approach towards a polyphonic music retrieval system." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2001.

[3] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming," in *Proceedings of ACM Multimedia*, 1995.

[4] T. Nishimura, H. Hashiguchi, J. Takita, J. X. Zhang, M. Goto, and R. Oka, "Music signal spotting retrieval by a humming query using start frame feature dependent continuous dynamic programming." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2001.

[5] A. Wang, "An industrial strength audio search algorithm." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2003.

[6] J. P. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2005.

[7] M. Müller, F. Kurth, and M. Clausen, "Audio matching via chroma-based statistical features." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2005.

[8] F. Kurth and M. Müller, "Efficient index-based audio matching," *IEEE Transactions on Audio, Speech, and Language Processing*, 2008.

[9] W.-H. Tsai, H.-M. Yu, H.-M. Wang *et al.*, "Query-by-example technique for retrieving cover versions of popular songs with similar melodies." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2005.

[10] C. Sailer, "Using string alignment in a query-by-humming system for real world applications," *The Journal of the Acoustical Society of America*, 2005.

[11] E. Gómez and P. Herrera, "The song remains the same: identifying versions of the same piece using tonal descriptors." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2006.

[12] D. P. Ellis and G. Poliner, "Identifying 'cover songs' with beat-synchronous chroma features," in *MIREX (Music Information Retrieval Evaluation eXchange)*, 2006.

[13] K. Lee, "Identifying cover songs from audio using harmonic representation," *MIREX (Music Information Retrieval Evaluation eXchange)*, 2006.

[14] J. Serrà and E. Gómez, "A cover song identification system based on sequences of tonal descriptors," *MIREX (Music Information Retrieval Evaluation eXchange)*, 2007.

[15] D. P. Ellis and C. V. Cotton, "The 2007 labrosa cover song detection system," 2007.

[16] J. P. Bello, "Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2007.

[17] J. Serrà, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Transactions on Audio, Speech, and Language Processing*, 2008.

[18] J. Serrà, X. Serra, and R. G. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, 2009.

[19] S. Ravuri and D. P. Ellis, "Cover song detection: from high scores to general classification," in *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. IEEE, 2010.

[20] A. Degani, M. Dalai, R. Leonardi, and P. Migliorati, "A heuristic for distance fusion in cover song identification," in *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*. IEEE, 2013.

[21] J. Osmalskyj, J.-J. Embrechts, P. Foster, and S. Dixon, "Combining features for cover song identification," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2015.

[22] R. Foucard, J.-L. Durrieu, M. Lagrange, and G. Richard, "Multimodal similarity between musical streams for cover version detection," in *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. IEEE, 2010.

[23] J. Salamon, J. Serrà, and E. Gómez, "Melody, bass line, and harmony representations for music version identification," in *Proceedings of the 21st International Conference on World Wide Web*. ACM, 2012.

[24] C. J. Tralie, "Early mfcc and hpcp fusion for robust cover song identification," *arXiv preprint arXiv:1707.04680*, 2017.

[25] T. Bertin-Mahieux and D. P. Ellis, "Large-scale cover song recognition using the 2d fourier transform magnitude." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2012.

[26] D. F. Silva, C.-C. M. Yeh, G. E. d. A. P. A. Batista, and E. Keogh, "Simple: assessing music similarity using subsequences joins," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2016.

[27] D. F. Silva, F. V. Falcao, and N. Andrade, "Summarizing and comparing music data and its application on cover song identification," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2018.

[28] P. Seetharaman and Z. Rafii, "Cover song identification with 2d fourier transform sequences," in *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. IEEE, 2017.

[29] M. Marolt, "A mid-level representation for melody-based retrieval in audio collections," *IEEE Transactions on Multimedia*, 2008.

[30] P. Grosche and M. Müller, "Toward characteristic audio shingles for efficient cross-version music retrieval," in *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. IEEE, 2012.

[31] E. J. Humphrey, O. Nieto, and J. P. Bello, "Data driven and discriminative projections for large-scale cover song identification." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2013.

[32] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, 2016.

[33] T. Tsai, T. Prätzlich, and M. Müller, "Known artist live song id: A hashprint approach." in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2016.

[34] Z. Yu, X. Xu, X. Chen, and D. Yang, "Learning a representation for cover song identification using convolutional neural network," *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, 2020.

[35] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2019.

[36] G. Doras, P. Esling, and G. Peeters, "On the use of u-net for dominant melody estimation in polyphonic music," in *International Workshop on Multilayer Music Representation and Processing (MMRP)*. IEEE, 2019.

[37] F. Yesiler, J. Serrà, and E. Gómez, "Accurate and scalable version identification using musically-motivated embeddings," *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, 2020.

[38] X. Xu, X. Chen, and D. Yang, "Key-invariant convolutional neural network toward efficient cover song identification," in *Proceedings of IEEE ICME (International Conference on Multimedia and Expo)*. IEEE, 2018.

[39] Z. Yu, X. Xu, X. Chen, and D. Yang, "Temporal pyramid pooling convolutional neural network for cover song identification," *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019.

[40] G. Doras and G. Peeters, "A prototypical triplet loss for cover detection," in *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, 2020.

[41] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2017.

[42] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015.

[43] J. Serrà, S. Pascual, and A. Karatzoglou, "Towards a universal neural network encoder for time series." in *CCIA*, 2018.

[44] F. Yesiler, C. Tralie, A. A. Correya, D. F. Silva, P. Tovstogan, E. Gómez, and X. Serra, "Da-tacos: A dataset for cover song identification and understanding," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2019.

[45] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems*, 2006.

[46] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of IEEE CVPR (Conference on Computer Vision and Pattern Recognition)*, 2015.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

THE FREESOUND LOOP DATASET AND ANNOTATION TOOL

António Ramires[1] **Frederic Font**[1] **Dmitry Bogdanov**[1] **Jordan B. L. Smith**[2]
Yi-Hsuan Yang[3] **Joann Ching**[3] **Bo-Yu Chen**[3] **Yueh-Kao Wu**[3]
Hsu Wei-Han[3] **Xavier Serra**[1]

[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
[2] TikTok, Lodon, United Kingdom
[3] Research Center for IT Innovation, Academia Sinica, Taipei, Taiwan

`antonio.ramires@upf.edu`

ABSTRACT

Music loops are essential ingredients in electronic music production, and there is a high demand for pre-recorded loops in a variety of styles. Several commercial and community databases have been created to meet this demand, but most are not suitable for research due to their strict licensing. We present the Freesound Loop Dataset (FSLD), a new large-scale dataset of music loops annotated by experts. The loops originate from Freesound, a community database of audio recordings released under Creative Commons licenses, so the audio in our dataset may be redistributed. The annotations include instrument, tempo, meter, key and genre tags. We describe the methodology used to assemble and annotate the data, and report on the distribution of tags in the data and inter-annotator agreement. We also present to the community an online loop annotator tool that we developed. To illustrate the usefulness of FSLD, we present short case studies on using it to estimate tempo and key, generate music tracks, and evaluate a loop separation algorithm. We anticipate that the community will find yet more uses for the data, in applications from automatic loop characterisation to algorithmic composition.

1. INTRODUCTION

Repurposing audio material to create new music—also known as *sampling*—was a foundation of electronic music and is a fundamental component of this practice. Loops are audio excerpts, usually of short duration, that can be played repeatedly in a seamless manner [28]. These loops can serve as the basis for songs, which music makers can combine, cut and rearrange, and have been extensively used in Electronic Dance Music (EDM) tracks [4].

Audio loops have been made available for amateur and professional music makers since the early ages of elec-

tronic music. Currently, large-scale databases of audio offer huge collections of audio material for users to work with. Some databases, like Freesound [1] and Looperman [2], are community-oriented: people upload their sounds so that other users can employ them in their works. More commonly, these collections are commercially oriented: loops are available to paying costumers, either through a subscription service (*e.g.* Sounds.com, [3] Splice [4]) or by allowing customers to buy packs of loops (*e.g.* Loopmasters, [5] and Prime Loops [6]).

Despite the number of loops available on these databases, the technologies used to analyse and navigate these databases still rely on human annotations and human content curation to, for instance, group sounds into packs for specific genres or styles. Loops are being manually annotated with information like instrument, tonality (key), tempo (bpm) and music genre. This is a time-consuming task which is often unfeasible, which results in badly annotated databases and poor user experience when browsing them. In the field of Music Information Retrieval (MIR), a substantial effort has been put into automatically identifying the aforementioned characteristics for musical pieces. However, loops are inherently different from music pieces (*i.e.* with reduced instrumentation and short length). Therefore, existing MIR algorithms need to be tested and (possibly) adapted to work successfully in this scenario. Furthermore, new MIR tasks are emerging with the study of music loops including loop retrieval [12], loop detection [18], loop discovery [19] and extraction [27], loop recommendation [5], exploration of large loop databases [29], and automatic loop generation [26].

In this paper, we present FSLD, an open dataset with 9,455 music loops to support reproducible research in MIR. FSLD contains production-ready loops from Freesound which are distributed under Creative Commons licenses and can, therefore, be freely shared among the research community and industry. Part of the dataset has been manually annotated with information about rhythm, tonality, instrumentation and genre, in a similar way as

[1] `https://freesound.org/`
[2] `https://www.looperman.com/`
[3] `https://sounds.com/`
[4] `https://splice.com/`
[5] `https://www.loopmasters.com/`
[6] `https://primeloops.com/`

commercially available loop collections are annotated. The annotation service is made public[7] so that the community can work on enlarging the annotations of this collection. We expect this dataset to have an impact on the research community as it supports further research into several timely research topics which are also of great interest to the industry.

The rest of the paper is structured as follows. In Sec. 2, we present some of the datasets used in the literature for loop analysis. Sec. 3 details how the proposed dataset was collected and annotated. In Sec. 4, general statistics of the dataset are given. In Sec. 5 and 6, we present some potential applications and provide a benchmark of the dataset using some classic MIR tasks. Finally, in Sec. 7, we conclude and suggest future work directions.

2. RELATED WORK

Early work on the retrieval of loops focused on tempo extraction and transcription from drum loops [11, 15]. Gouyon et al. compared several tempo induction algorithms proposed in the ISMIR 2004 competition [15]. The loop dataset used in this work has been commonly used for evaluating tempo estimation algorithms and is divided into three subsets. One of these comprises two thousand audio loops (with tempo annotations) from Sound Effects Library.[8] These audio loops are not free and a license needs to be obtained to use them for research.

Automatic transcription of drum loops focuses on identifying when the different percussion instruments occur in a loop. Gillet and Richard used a collection of 315 drum loops for evaluating their system and provided "a compressed version of a few drum loops" [11]. The URL to the webpage the authors provide is broken and, presumably, the lower quality versions of the loops do not represent commercial-quality content. The authors also use this dataset for automatically retrieving drum loops from spoken queries [12]. This database was later used by Bello et al. for automatic rhythm modification and analysis of drum loops [1,23].

The work from Gómez-Marín et al. explores rhythmic similarity measures for audio loops [14]. The authors validate the proposed metric using 9 drum break loops from Rhythm Lab.[9] The authors do not specify which are the drum loops used.

Font et al. presented a dataset of audio loops from Freesound [9] in their work on tempo estimation and a confidence measure for audio loops [10]. The authors use two commercial datasets, loops bundled with music production software Apple's Logic Pro[10] and Acoustica's Mixcraft,[11] and two community datasets. The first one is a private collection of loops downloaded from Looperman, which was previously used for research in [24]. Looperman does not allow the re-distribution of loops "as is", and considers as misuse the automatic download of their

loops.[12] A collection of 4000 loops from Freesound, obtained by searching Freesound for sounds with the queries "loop" and "bpm" is also proposed. The sounds' filenames, tags and textual descriptions are parsed to identify tempo annotations provided by the users. However, these annotations are not always accurate, and, to enable further work on audio loops, more information besides the tempo is desired.

In short, existing academic work which employs loops resorts to commercial samples as the source of data and open datasets do not have complete and reliable annotations. This makes it difficult to reproduce existing research. To promote open and accessible research on audio loops, we propose a free and distributable database of loops from Freesound, which provides production-ready sounds with high-quality annotations.

3. DATASET CREATION

In this section the process we have followed to create the dataset is described. We show how we collected the loops to annotate, how they were pre-analysed for a faster annotation procedure and explain what was annotated and how the annotation tool was implemented. Finally, we present how the dataset is distributed and organised.

3.1 Loop Selection

To select an initial pool of candidate loops, we followed the same methodology as in [10]: i.e., we retrieved sounds with both "loop" and "bpm" keywords on Freesound, resulting in 9,490 sounds. Using the Freesound API, it was straightforward to obtain these loops and their metadata— title, tags, textual description, and author's username.

3.2 Loop Annotation

We want the loops in our dataset to be annotated in a way which is similar to commercially available loops. This way, we make sure that the loop characterisation is compatible with industry standards. For this, we decided to annotate the loops' instrumentation, tempo, time signature, key and genre, as described below. The annotation was performed by 8 MIR researchers and students, with knowledge of electronic music production. To make the annotation procedure as efficient as possible, we created a web application for the annotators with several tools at their disposal, which can be seen in Fig. 1. This application was developed using Flask,[13] a web framework for Python.

This interface provides fields for the annotators to fill in the desired information, which will be described in the following sections. The instructions are provided on tooltips for quick access by annotators.

3.2.1 Instrumentation

Instead of annotating instruments in a traditional way, which would not be straightforward in heavily processed

---

[7] http://mtg.upf.edu/fslannotator
[8] http://www.sound-effects-library.com/
[9] https://rhythm-lab.com/
[10] https://www.apple.com/logic-pro/
[11] https://acoustica.com/mixcraft

[12] https://www.looperman.com/help/terms
[13] https://flask.palletsprojects.com/

Figure 1. The user interface provided to the annotators, available online. [7]

audio or more experimental loops, we chose to annotate general *roles* which can be useful for both music makers and automatic generation of music. We asked annotators to tick all the roles that apply to each loop. Usually, specific instruments could be easily assigned to a specific role. We present the roles along with some examples in Table 1.

Role	Example Instruments
Percussion	Drums, glitches, tuned percussion
Bass	Synth bass, fingered bass
Chords	Piano chords, guitar chords, synth pads
Melody	Instrument playing a melody, arpeggiator
Sound FX	Risers, cinematic sounds, foley, scratching
Vocal	Singing voice, spoken word, vocoder

Table 1. Instrumentation roles and the examples provided for each category.

3.2.2 Rhythmic Characteristics

We asked for annotations on three rhythmic aspects:

Tempo provides an easy measure of rhythmic compatibility and is the most common information provided in commercial loop databases. We ask annotators if the loop has a clear and steady tempo, to identify loops with constant tempo and clear beat (BPM value and steady tempo), with changing tempo (BPM value of the initial tempo

and no steady tempo), and loops with no clear beat but where the tempo can be inferred (BPM value and no steady tempo).

Meter is not a feature we see annotated as often as BPM, which might be due to the common use of 4/4 meter in electronic music. This feature is relevant to annotate, for calculating the number of bars in a loop, from its meter, tempo and duration.

Finally, as sometimes the length of the audio file is not the length of the loop, we also annotate if it is **well-cut**. If there is some silence at the beginning or the end of the file or if there is a "tail" (*e.g.* a decay of a reverb effect) when the audio is exported, it might not loop correctly just by staring the loop again when it finishes playing.

3.2.3 Tonal Characteristics

We annotate if the loop has prominent tonal content and, if so, to indicate a root key and mode that matches the tonal content of the loop (*i.e.*, root note from a chromatic scale and *Major/Minor* mode). We explained "prominent tonal content" as whether it is easy to sing along to the loop or to find a meaningful root note for the loop. For root key annotations, we asked to choose a note from a dropdown with 12 notes, or "Unknown" in case the key could not be found. For annotating mode, the annotators had the choice

of "Major" or "Minor" if the loop sounded good with one of these modes; "None" if the loop could not be clearly assigned to either "Major" or "Minor" (e.g. loop contains a single note); or "Unknown" for other cases.

3.2.4 Genre

We annotate genre in non-exclusive categories, where each is assigned to a loop if it can be used to make music in that genre. To create a taxonomy which would be similar to commercially available ones, we merged the taxonomies of Sounds.com and Splice. These were chosen as they provided several examples for each genre and had similar parent categories. We present the taxonomy in Table 2.

Genre	Examples
Bass Music	Dubstep, Drum and Bass, Jungle
Live Sounds	Rock, Jazz, Disco
Cinematic	Sound FX, Filmscore, Sci-Fi
Global	Reggae, Dancehall, Indian Music
Hip Hop	Trap, Boom Bap, Lofi Hip Hop
Electronic	Ambient, IDM, Chill Out
House / Techno	Deep House, Electro, Tech House
Other Dance Music	EDM, Psy Trance, Hardstyle

Table 2. Taxonomy of genres used for the annotation and examples for each category.

3.2.5 Loop Pre-Analysis and Annotation Tools

We performed a pre-analysis on the loops to obtain tempo, key and genre suggestions. To obtain the tempo information, we followed the same approach of [10], parsing the title, description and the tags of the loop for tempo information provided by users. To propose an initial key and mode to the annotators, we analysed the loops using the algorithm proposed by Faraldo et al. [7], which is implemented in the Essentia audio analysis library [3]. Finally, by taking the genre information from the textual metadata of the loops, we were able to map some of the sounds to the genres to annotate. The checkboxes were selected for the genres which either were mentioned or had a sub-genre mentioned in the textual metadata. Our annotators were familiar with the annotation procedure and took the pre-annotations only as suggestions to speed-up the annotation process.

In the annotation tool, at the top of the display is the loop's metadata: its unique sound id, title, author's username, and the tags and textual description provided by the author (see Fig. 1). The waveform of the loop and a playhead is also shown, which are linked to an audio player. The audio player always restarts the playback of the loop when it finishes, and triggers a metronome with the BPM provided in the BPM annotation field. We provide stop, play and pause controls for the loop and metronome and volume controls for the loop. A button which restarts only the metronome is also present. To ease finding a key and mode which suits the loop, we present a synthesizer which plays the chord present in the tonal annotation section. In case the mode selected is "None" or "Unknown", the synthesizer will just play the root note of the key selected. Using the computer's keyboard, the annotator can cycle

through the options for key and mode, in several octaves. Finally, buttons are provided for submitting the annotation when it is finished, saving the sound for later and discarding the sound in case it is not a loop.

3.3 Dataset Availability

The loops and corresponding annotations (provided in a JSON file) are publicly available on Zenodo. [14] This dataset can be divided into three subsets, defined by their level of annotations. These are:

- Multiple-annotations (MA): the loops annotated by at least two researchers. It contains 1,472 loops.
- Single-annotation (SA): the loops annotated by a single researcher. Currently contains 1,464 loops.
- Automatic-annotations (AA): the loops annotated by the analysis algorithms mentioned in Section 3.2.5. Contains 9,455 loops: the loops in MA and SA and 6,519 more.

In addition to the main dataset, we provide a repository [15] with the code used for the annotation tool interface and server, the pre-analysis that generated the subset of automatic annotations, and the analysis and potential applications presented in Sections 4, 5 and 6.

4. DATASET ANALYSIS

To understand the diversity and reliability of the dataset, we investigate the distribution of annotated characteristics and inter-annotator agreement.

4.1 Annotation Distribution

The human-annotated part of the dataset contains 1,579 sounds which, in total, have been annotated 2,809 times. The distribution of genres, instrumentation, and keys are shown in Tables 3 and 4 and the tempo histogram in Figure 2. It is well-balanced in terms of instrument and genre; reasonably balanced in terms of tempo, although 120 bpm dominates; and highly imbalanced in terms of key, with C Major and Minor dominating.

Percussion	54.95%	Bass Music	32.04%
Bass	19.10%	Live Sounds	21.38%
Chords	11.90%	Cinematic	19.95%
Melody	21.31%	Global	14.26%
FX	24.80%	Hip-hop	17.29%
Vocal	2.29%	House/Techno	29.05%
		Other Dance Music	25.63%
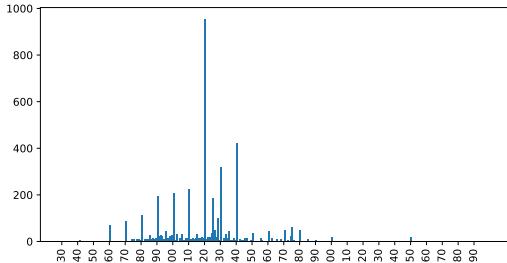
Table 3. Distribution of the instrument roles and genre in our dataset.

4.2 Inter-annotator Agreement

To measure the agreement of the annotators in our dataset, we measure the inter-annotator agreement for the MA annotations subset. To do this, we use two metrics: proportion of overall agreement (Agr.) for all the annotations,

---

[14] https://zenodo.org/record/3967852
[15] https://github.com/aframires/
freesound-loop-annotator

Figure 2. Distribution of BPMs in FSLD.

Key	Maj	Min	None	Unknown
C	9.63%	8.38%	3.65%	0.95%
C#	1.38%	2.84%	0.60%	0.43%
D	3.31%	5.37%	1.85%	0.39%
D#	1.29%	2.49%	0.73%	0.21%
E	2.28%	3.65%	1.20%	0.26%
F	4.64%	4.43%	1.16%	0.34%
F#	1.12%	2.49%	1.12%	0.13%
G	2.66%	4.25%	1.93%	0.43%
G#	1.72%	2.58%	0.90%	0.13%
A	3.01%	5.50%	1.68%	0.26%
A#	1.50%	1.89%	0.52%	0.04%
B	0.82%	3.01%	0.64%	0.21%

Table 4. Distribution of the keys in our dataset.

and positive and negative agreement (PA and NA) [8] for binary classification tasks. The proportion of overall agreement reflects the number of cases when both annotators agree on a label, and is calculated by dividing their number by the total number of annotations. This overall metric does not distinguish the agreement in positive and negative cases, so for the binary annotation tasks we also calculated the positive and negative agreement. The formulas for calculating these are given in Eq. 1, where the variables represent the annotations by the annotators (e.g., NP = first annotator answered negative, second positive).

$$PA = \frac{2PP}{2PP + NP + PN} , \qquad (1)$$

$$NA = \frac{2NN}{2NN + NP + PN} , \qquad (2)$$

Table 5 presents the results for this analysis. We can see that overall, the values for the agreement are high. Bass, melody and chords have a lower positive agreement value, despite the high negative agreement. This might indicate that annotators are not able to easily distinguish if an element should fit in one of the 3 roles, but can say when it is not present. The lower value for root key agreement indicates that several keys are used to describe the same sounds. This fits our annotating indications, where we asked annotators to select a key which sounds good with the loop and therefore, personal taste may arise in this choice. Finally, the positive agreement for genres always has values lower than 65%, which might be due to how genre might be perceived subjectively between annotators.

5. BENCHMARKING MIR TASKS

To demonstrate the usefulness of this dataset, we use it in several short case studies. To benchmark tempo, we fol-

Char.	Sub-Char.	Agr.	PA	NA
Inst.	Percussion	85.16%	86.62%	83.35%
	Bass	76.73%	45.83%	85.19%
	Melody	82.33%	60.57%	88.61%
	Chords	87.40%	47.35%	92.84%
	FX	72.04%	43.61%	81.41%
	Vocal	98.66%	71.88%	99.31%
Tempo	BPM	87.84%	NA	NA
	Signature	97.84%	NA	NA
	Well Cut	86.88%	92.73%	32.82%
Key	Root	67.56%	NA	NA
	Mode	69.80%	NA	NA
Genre	Bass Music	69.50%	53.26%	77.37%
	Live Sounds	80.09%	55.28%	87.19%
	Cinematic	81.66%	57.14%	88.33%
	Global	82.33%	51.53%	89.19%
	Hip-Hop	79.05%	31.30%	87.64%
	House/Techno	69.35%	48.56%	78.17%
	Other	73.53%	45.64%	82.50%

Table 5. Inter-annotator agreement for the MA subset.

lowed the evaluation approach of [10] and used the *Accuracy 1* and *Accuracy 2* presented in [16], together with the *Accuracy 1e* proposed in [10]. Due to space constraints, here we only report the mean of the 3 accuracies. Full results can be seen in an accompanying website. [16] The algorithms selected for the tempo benchmarking were the following (details for each algorithm can be found in respective papers):

- **Percival [21]:** We use both the original implementation and the one provided in Essentia.

- **Zapata [31]:** Implementation provided in Essentia.

- **Degara [6]:** We also use Essentia's implementation.

- **Böck [2]:** We use the 3 variants available in the Madmom library [17]: COMB, ACF and DBN.

We validate tempo estimation algorithms on the 3 proposed subsets. Key estimation is only validated on the MA and SA subsets as we do not have original uploader annotations for key. The MA subset, which has at least 2 annotations per loop, was analysed in two ways: BOTH and EITHER. In BOTH, we run the MIR algorithms exclusively on the loops which have the same labels from both annotators. In EITHER, the output of the algorithm was deemed correct if it was at least one of the annotated labels. The results are presented in Table 6

Algorithm	AA	SA	BOTH	EITHER
Percival14	58.09	62.98	65.75	84.13
Percival14e	57.82	64.00	65.49	84.98
Zapata14	51.81	58.79	58.99	77.97
Degara12	52.32	58.77	59.31	79.16
Bock15COMB	44.42	51.17	52.92	71.35
Bock15ACF	48.65	51.96	54.75	74.90
Bock15DBN	45.76	50.60	52.32	70.90

Table 6. Evaluation of tempo estimation algorithms in the proposed subsets.

---

[16] https://aframires.github.io/freesound-loop-annotator/
[17] https://github.com/CPJKU/madmom

We can see that the results are similar to the ones obtained in [10], with Percival14 having better accuracy across all the datasets. We can see that the accuracy increases from AA to SA, and from SA to BOTH. This might be due to the user-annotated loops having incorrect annotations; it may also be that when both annotators agree on a tempo, the tempo is strong and defined. The EITHER evaluation gives the largest accuracies, which may be due to its broader criteria for considering tempos correct.

For benchmarking key estimation algorithms, we used the evaluation metrics from MIREX,[18] which evaluates how *close* the estimated key and the annotated key are to provide an accuracy. The algorithms compared in the evaluation were the following:

- **EDMKey [7]:** We use the implementation in Essentia, with 4 key profiles: Krumhansl [17], Temperley [30], Shaath [25] and the one proposed in [7].
- **EssentiaBasic [3]:** Essentia's implementation of the algorithm presented by Gomez [13].
- **QMUL [20]:** We use the Key Detection implementation available in QM Vamp Plugins.[19]

In Table 7, we present part of the results of the key estimation evaluation. Due to lack of space, only the final MIREX scores for each dataset are presented. The full results can be seen in the accompanying website.[16]

Algorithm	SA	BOTH	EITHER
Edmkey	72.26	88.25	85.63
EdmkeyKrumhansl	66.99	84.85	82.98
EdmkeyTemperley	61.46	71.78	71.77
EdmkeyShaath	72.38	88.25	85.63
EssentiaBasic	71.25	88.80	85.30
QMULKeyDetector	35.09	42.15	46.25

Table 7. Evaluation of key estimation algorithms in the proposed subsets.

We see that EssentiaBasic and EDMkey are the best performing algorithms here. EDMKey has been specially tuned to be used for EDM, which might make it more suitable to the loops we are annotating. We again see that the accuracy increases from SA to BOTH, which might indicate again that when the key is clear and defined, the algorithms are also able to correctly identify it.

6. MUSIC GENERATION AND DECOMPOSITION

Another way the dataset is valuable is for creating synthetic datasets of songs for evaluating loop-extraction algorithms, such as [27]. We created 100 random songs, each using 5 random drum loops and 5 non-drum loops (chosen from a subset of 4/4, 120-bpm, 1-bar, single-instrument loops for which there was no disagreement among the annotators on the instrument role). Each song is a random arrangement of the loops, either in a *sparse* arrangement, in which one drum and one non-drum loop occurs per bar, or a *dense* one, in which 4 loops occur per bar (i.e., 2 drum

and 2 non-drum). For comparison, we also recreated the "composed" and "factorial" layouts from [27]. Examples of each layout are shown on the accompanying website.

We used the public implementation[20] of [27] to extract loops for each song, informed with the true number of loop segments (4 or 10) and the true downbeat boundaries. The metrics SDR, SIR and SAR (the signal to distortion, interference and artefacts ratios [22]) are reported in the left part of Table 8.

These are normally computed by trying all permutations of estimated sources to true sources and using that which maximises the score. This is infeasible for permutations of 10 items, so we first find the permutation that maximises the similarity between the source and true loop spectra.

Layout	SDR	SIR	SAR	F1	Acc.
Sparse	−5.2	−3.9	15.8	0.194	0.691
Dense	−7.9	−7.3	14.4	0.294	0.542
Composed	12.5	18.4	22.6	0.585	0.546
Factorial	19.8	29.2	24.1	0.560	0.551

Table 8. Evaluation of loop source quality (SDR, SIR, SAR) and estimated layouts (F-measure and accuracy) for each song layout.

This permutation is also used to evaluate the quality of the estimated layout. We binarize each row of the estimated layout, using the row's mean as threshold. We then compute the raw accuracy (as in [27]), but here we propose also using the F-measure, so as not to weight true negatives unduly. The results are in the right columns of Table 8.

SDR, SIR and SAR are all lower for the random 10-part songs than for the 4-part songs, showing that we have created a more challenging testing ground for loop extraction systems. For the layout evaluation, our evaluation makes clear that the raw accuracy gives undue weight to true negatives: the highest accuracy was obtained for the sparse layouts, despite having the lowest F-measure. This short evaluation is a proof of concept; with more space, we could study the impact of the instrumentation, number of loops, loop duration, and other factors on the separation quality. We can also generate layouts with loops of many durations and evaluate hierarchical loop extraction systems.

7. CONCLUSION

In this paper, we presented our work on addressing the lack of standard loop datasets to carry MIR tasks. We presented FSLD, a dataset of audio loops annotated at a level similar to commercial loop collections. These loops are licensed for redistribution and can be used and redistributed for research purposes. We provide a detailed analysis of the dataset and its annotations and provided several use cases for tempo and key benchmarking, music generation and loop separation. Furthermore, we present the online annotation tool used to build the dataset, and we make it available online so other researchers and the general public can contribute and extend the dataset.

---

[18] https://www.music-ir.org/mirex/wiki/2019:
Audio_Key_Detection
[19] https://vamp-plugins.org/plugin-doc/
qm-vamp-plugins.html#qm-keydetector

[20] https://github.com/jblsmith/loopextractor

8. ACKNOWLEDGEMENT

9. REFERENCES

[1] Juan P. Bello, Emmanuel Ravelli, and Mark B. Sandler. Drum sound analysis for the manipulation of rhythm in drum loops. In *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, 2006.

[2] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, 2015.

[3] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, Jose R. Zapata, and Xavier Serra. Essentia: an audio analysis library for music information retrieval. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, 2013.

[4] Mark Jonathan Butler. *Unlocking the groove: Rhythm, meter, and musical design in electronic dance music*. Indiana University Press, 2006.

[5] Bo-Yu Chen, Jordan Smith, and Yi-Hsuan Yang. Neural loop combiner: Neural network models for assessing the compatibility of loops. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020.

[6] Norberto Degara, Enrique Argones Rúa, Antonio Pena, Soledad Torres-Guijarro, Matthew EP Davies, and Mark D Plumbley. Reliability-informed beat tracking of musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):290–301, 2011.

[7] Ángel Faraldo, Emilia Gómez, Sergi Jordà, and Perfecto Herrera. Key estimation in electronic dance music. In *38th European Conference on Information Retrieval*, pages 335–347. Springer-Verlag, 2016.

[8] Alvan R Feinstein and Domenic V Cicchetti. High agreement but low kappa: I. the problems of two paradoxes. *Journal of clinical epidemiology*, 43(6):543–549, 1990.

[9] Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. In *ACM International Conference on Multimedia*, 2013.

[10] Frederic Font and Xavier Serra. Tempo estimation for music loops and a simple confidence measure. In *Proceedings of the 17th International Society for Music Information Retrieval Conference*, 2016.

[11] Olivier Gillet and Gaël Richard. Automatic transcription of drum loops. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004.

[12] Olivier Gillet and Gaël Richard. Drum loops retrieval from spoken queries. *Journal of Intelligent Information Systems*, 24(2):159–177, 2005.

[13] Emilia Gómez. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing*, 18(3):294–304, 2006.

[14] Daniel Gómez-Marín, Sergi Jordà, and Perfecto Herrera. Pad and sad: Two awareness-weighted rhythmic similarity distances. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, 2015.

[15] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[16] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[17] Carol L. Krumhans. Cognitive foundations of musical pitchl. *Music Perception: An Interdisciplinary Journal*, 9:476–492, 07 1992.

[18] Patricio López-Serrano, Christian Dittmar, Jonathan Driedger, and Meinard Müller. Towards modeling and decomposing loop-based electronic music. In *Proceedings of the 17th International Society for Music Information Retrieval Conference*, 2016.

[19] Patricio López-Serrano, Christian Dittmar, and Meinard Müller. Finding drum breaks in digital music recordings. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, 2017.

[20] Katy Noland and Mark Sandler. Signal processing parameters for tonality estimation. In *Audio Engineering Society Convention 122*, May 2007.

[21] Graham Percival and George Tzanetakis. Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1765–1776, 2014.

[22] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel PW Ellis. mir_eval: A transparent implementation of common MIR metrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, 2014.

[23] Emmanuel Ravelli, Juan P. Bello, and Mark Sandler. Automatic rhythm modification of drum loops. *IEEE Signal Processing Letters*, 14(4):228–231, 2007.

[24] Gerard Roma. *Algorithms and representations for supporting online music creation with large-scale audio databases*. PhD thesis, Universitat Pompeu Fabra, June 2015.

[25] ibrahim Sha'ath. Estimation of key in digital music recordings. Master's thesis, Birkbeck College, University of London, 2011.

[26] Z. Shi and G. J. Mysore. LoopMaker: Automatic creation of music loops from pre-recorded music. In *Proceedings of SIGCHI International Conference on Human Factors in Computing Systems*, 2018.

[27] Jordan B. L. Smith and Masataka Goto. Nonnegative tensor factorization for source separation of loops in audio. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 171–175, 2018.

[28] Glenn Stillar. Loops as genre resources. *Folia Linguistica*, 39(1-2):197 – 212, 2005.

[29] S. Streich and B. S. Ong. A music loop explorer system. In *Proceedings of International Computer Music Conference*, 2008.

[30] David Temperley. What's key for key? the Krumhansl-Schmuckler key-finding algorithm reconsidered. *Music Perception: An Interdisciplinary Journal*, 17(1):65–100, 1999.

[31] José R Zapata, Matthew EP Davies, and Emilia Gómez. Multi-feature beat tracking. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):816–825, 2014.

SHOULD WE CONSIDER THE USERS IN CONTEXTUAL MUSIC AUTO-TAGGING MODELS?

Karim M. Ibrahim[1,2]   **Elena V. Epure**[2]   **Geoffroy Peeters**[1]   **Gaël Richard**[1]

[1] LTCI, Télécom Paris, Institut Polytechnique de Paris

[2] Deezer Research

`karim.ibrahim@telecom-paris.fr`

ABSTRACT

Music tags are commonly used to describe and categorize music. Various auto-tagging models and datasets have been proposed for the automatic music annotation with tags. However, the past approaches often neglect the fact that many of these tags largely depend on the user, especially the tags related to the context of music listening. In this paper, we address this problem by proposing a user-aware music auto-tagging system and evaluation protocol. Specifically, we use both the audio content and user information extracted from the user listening history to predict contextual tags for a given user/track pair. We propose a new dataset of music tracks annotated with contextual tags per user. We compare our model to the traditional audio-based model and study the influence of user embeddings on the classification quality. Our work shows that explicitly modeling the user listening history into the automatic tagging process could lead to more accurate estimation of contextual tags.

1. INTRODUCTION

Tags are a popular way to categorise music in large catalogues in order to facilitate their exploration and music retrieval on demand [17]. Music tags include different categories such as emotions (sad, happy), genres (rock, jazz), instrumentation-related (guitar, vocals), or listening activities (dance, relax, workout). Traditionally, tags were assigned to music items by humans, either through editors or through crowdsourcing. However, with the expanding availability of online music, there have been also increasing efforts towards developing music auto-tagging models, i.e. systems that do not require to manually annotate the tracks [1]. Music auto-taggers are models trained to automatically predict the correct tags for a given music track from the track content. Several models have been proposed that use the audio content, either as raw signal [15, 20, 26] or pre-processed spectrograms [4, 5, 25, 26], to predict the appropriate tags. However, certain tags largely depend on

users and their listening preferences, in particular, the tags referring to the context of music listening such as 'running' or 'relaxing' [21]. Thus, traditional auto-tagging models that rely only on the audio content without considering the case where tags depend on users, are not ideal for describing music with user-dependent tags like contexts. Additionally, their evaluation protocol should be also adapted to account for different users.

Previous studies showed that user context has a clear influence on the user's music selection [10, 18]. Hence, context is progressively becoming the focus of music streaming services for reaching a personalized user experience [14]. The user context, e.g. activity or location, can change frequently while listening to music, which leads to changes in user preferences. Consequently, users often need different recommendations. Automatically inferring the user context is often not feasible due to privacy issues. Hence, giving users the option to select a specific context and propose him/her related personalized tracks is a potential alternative [7]. Another use case is automatic continuation or generation of context-specific playlists for each user which are made available to them to select based on their current context [2]. Thus, describing tracks with contextual tags provides a means to improve music exploration and playlist generation in a dynamic way, suitable for the frequent changes in the user context. However, previous work [13] showed that using only the audio might not be sufficient to predict the right contextual tag of a track without putting the user in the loop. Here, we investigate the impact of including user information in context auto-taggers.
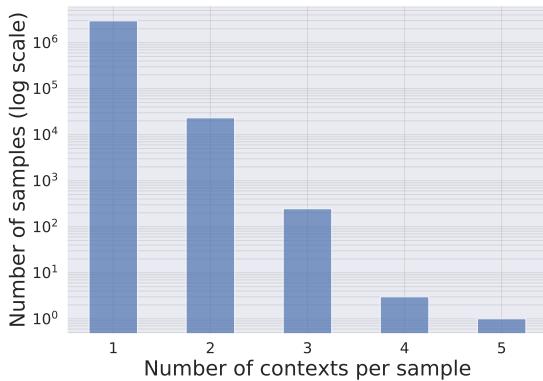
In this paper, we propose the following contributions: 1) a dataset of ∼182K user/track pairs labelled with 10 of the most common context tags based on the users' contextual preferences presented in Section 2, which we make available for future research; 2) a new evaluation procedure for music tagging which takes into account that tags are subjective, i.e. user-specific in Section 3; 3) an auto-tagging model using both audio content and user information to predict contextual tags in Section 4. Our experiments in Section 5 clearly show the advantage of including the user information in predicting contextual tags compared to traditional audio-only-based auto-tagging models presented.

2. DATASET
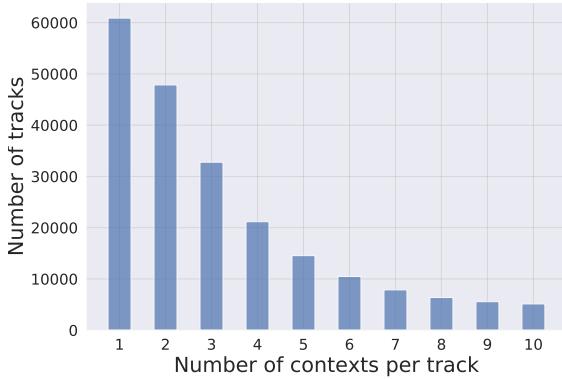
To properly study the influence of including user information in context auto-tagging models, we need a dataset of

Figure 1. Distribution of the number of contextual tags per sample (user/track pair) in the initial dataset.

Figure 2. Distribution of the number of contextual tags per track in the initial dataset.

tracks labelled with their contextual tags according to different users. For this purpose, we rely on the user-created context-related playlists. Users often create playlists for specific contexts and the titles of these playlists may convey these contexts. Thus, similar to [13, 19], we exploit the playlist titles to label tracks with their contextual use. Additionally, we put the users in the loop as playlist creators by explicitly including them in the dataset.

2.1 Dataset Collection

To retrieve contextual playlists, we used a set of contextual keywords collected from the literature [9, 18, 24, 27]. Then, we added keywords that were semantically similar. Ninety six keywords were categorized in one of four categories: location, activity, time, and mood. This is similar to the categorization proposed in [14]. To construct our dataset, we selected, out of all collected context-related keywords, 10 which were the most frequent keywords found in the playlist titles in the Deezer catalogue [1]. We selected the keywords that shared a similar number of playlists to avoid any bias due to the popularity of some contexts. The contextual tags we finally selected are: *car*, *gym*, *happy*, *night*,

	#Samples	#Track	#Users
Train	102K	15K	40K
Validation	30K	4.4K	21K
Test	50K	7.5K	16K

Table 1. Number of samples (track/user pairs), unique tracks and users in the train, validation and test datasets.

relax, *running*, *sad*, *summer*, *work*, *workout*.

We collected all the public user playlists that included any of these 10 keywords in the stemmed title and applied a series of filtering steps to consolidate the dataset, similar to our previous work in [13]. We removed all playlists that contained more than 100 tracks, to ensure that the playlists reflected a careful selection of context-related tracks, and not randomly added. We also removed all playlists where a single artist or album made up more than 25% of all tracks in the playlist, to ensure that the playlist was not intended for a specific artist, similar to [13]. Finally, to properly study the effect of the user on the contextual use of a track, we only kept the tracks that were selected by at least 5 different users in at least 3 different contexts. Hence, our dataset reflects how user preferences change the contextual use of tracks. Finally, we tagged each sample, the track/user pair, with the contextual tag found in the corresponding playlist title.

2.2 Dataset Analysis

In Figure 1, by observing the distribution of contextual tags per track/user pairs in the dataset, we noticed that most of the pairs were assigned to a unique contextual tag. Let us remind that the log scale is used and a sample represents a user/track pair labelled with the contextual tags. It appears that the majority of users tend to associate a track with a single context. Out of ∼3 millions samples, ∼2.9 millions are labelled with a single context. Nonetheless, ascertaining if this observation is generally valid requires further empirical investigation. For this study though, we limited our final dataset to track/user pairs with single context tags, i.e. we excluded users that assigned the same track to multiple contexts.

Observing the distribution of contextual tags per tracks in Figure 2, we find that tracks often have multiple contexts associated with them. This shows that the suitability of a track for a specific context varies from user to user. However, as previously outlined, given the user, the track is most frequently associated with the same unique context.

The final dataset for this study contains ∼182K samples of user/tracks pairs made of ∼28K unique tracks and ∼75K unique users. We collected the dataset such that each context is equally represented, ensuring a ratio of ∼ $\frac{1}{10}$ of all user/track pairs. We split our dataset in an iterative way to keep the balance between classes across subsets, while preventing any overlap between the users and minimising the overlap between tracks in these subsets [22]. The distribution of our final split dataset is shown in Table 2.2. The dataset is publicly available to the research community [2]

3. PROPOSED EVALUATION PROTOCOL

Previous studies on music auto-tagging [4, 20] performed the evaluation in a multi-label classification setup, therefore focusing on assessing the correctness of the tags associated with each track. This is suitable for datasets and tags that are only music-dependent. However, in the case of tags that are also user-dependent, the previous evaluation procedures are limiting.

3.1 User Satisfaction-focused Evaluation

The purpose of our study is to measure the influence of leveraging the user information on the quality of the prediction of contextual tags. Consequently, we are interested in measuring the potential satisfaction of each user when predicting contexts, instead of relying on a general evaluation approach that could be biased by highly active users or by the popularity of certain tags. Hence, we propose to compute the model performance by considering each user independently. To assess the satisfaction of each user, the evaluation metrics are computed by considering only the contextual tags specific to a user. Then, to assess the overall user satisfaction, we average the per-user results yielded by each model.

Formally, let $\mathbb{U}$ denote a finite set of users in the test set, $G_u = \{0,1\}^{n_u \times m_u}$ denote the groundtruth matrix for user u, n_u is the number of tracks associated with the user u, and m_u is the number of contextual tags employed by the user. Similarly, $P_u = \{0,1\}^{n_u \times m_u}$ denotes the matrix outputted by the model for all active tracks and contextual tags for the given user u. First, we compute each user-aware metric, hereby denoted by S, for a given user u as:

$$S_u = f(G_u, P_u) \qquad (1)$$

where f is the evaluation function. In our evaluation, we use standard classification metrics such as the area under the receiver operating characteristic curve (AUC), recall, precision, and f1-score [12]. While the protocol is defined for the general case of multi-label setting, in our current work, given the dataset, it is applied to the case of single-label. Then, we compute the final metrics, by averaging over all users in the test set:

$$S_{\mathbb{U}} = \frac{1}{N} \sum_{u \in \mathbb{U}} S_u, \text{ where } N = |\mathbb{U}|. \qquad (2)$$

3.2 Multi-label Classification Evaluation

In this work, we develop a system that takes both the audio and the user information as input. As seen in Section 2.2, for a given track and user, there is a single groundtruth context to be predicted. The problem is said to be single-label. However, if we want to compare this system with a system that only takes audio as input, we need to consider during training various possible groundtruth contextual tags for a track, each from a different user. Then, the problem becomes multi-label. The comparison of the two systems is therefore not straightforward. Indeed, for the user-agnostic case, we can train a multi-label system, i.e. a system with

a set of sigmoid output activations optimized with a sum of binary cross entropy, and estimate it either as single-label by taking the output with the largest likelihood, or as multi-label by selecting all outputs with a likelihood above a fixed threshold. For these reasons, in the current evaluation, we consider the following scenarios:

1. Multi-output / multi-groundtruth (MO-MG): This is the classical multi-label evaluation where the model outputs several predictions and each track is associated with several groundtruths. This evaluation is however independent of the user.

2. Multi-output / single-groundtruth (MO-SG): In this scenario, a model trained as multi-label (such as a user-agnostic model) is still allowed to output several predictions. However, since the groundtruth is associated with a given user, there is a single groundtruth. The obtained results are then over-optimistic because the model has several chances to obtain the correct groundtruth.

3. Single-output / single-groundtruth (SO-SG): this is the case that is directly comparable to our single-output user-aware auto-tagging model. As opposed to the MO-SG scenario, models trained as multi-label are now forced to output a single prediction, the most likely contextual tag. This prevents them from being over-optimistic as they only have one chance to obtain the correct groundtruth, as does the single-label model too.

4. PROPOSED MODEL FOR CONTEXTUAL TAG ESTIMATION

We propose to build a user-aware auto-tagging system. Given that contextual tags are interpreted differently by different users, we hypothesize that considering the user information in training a personalized user-aware contextual auto-tagging model may help. For this, we propose to add to the system, along with the audio input, a user input. We study the effectiveness of representing the user via 'user embeddings', obtained from user listening history.

4.1 Traditional Audio-based Auto-tagger

In this paper, we chose the prevalent audio-based auto-tagging model proposed by Choi et al [4]. The model is a multi-layer convolutional neural network. The input to the network is the pre-processed Mel-Spectrogram of the music track. This multi-label classification model predicts, for a given track, the set of all possible tags.

We trained the network with the Mel-spectrogram as an input of size 646 frames x 96 mel bands, which corresponds to the snippet from 30 to 60 seconds for each track. The output is the predictions for the 10 contextual tags. The input Mel-Spectrograms is passed to a batch normalization layer then to 4 pairs of convolutional and max pooling layers. The convolutional layers have a fixed filter size of (3x3) and (32, 64, 128, 256) filters respectively, each followed by a ReLu activation function. The max pooling

filters have a size (2x2) each. The flattened output of the last CNN layer is passed to a fully connected layer with 256 hidden units with ReLU activation function. We apply a dropout with 0.3 ratio for regularization. Finally, we pass the output to the final layer of 10 output units each with a Sigmoid activation function. The loss function is the sum of the binary cross entropy optimized with Adadelta and a learning rate initialized to 0.1 with an exponential decay every 1000 iterations. We applied early stopping after 10 epochs in case of no improvement on the validation set, and kept the model with the best validation loss.

4.2 Proposed Audio+User-based Auto-tagger

The Audio+User model that we propose is an extension of the Audio-based auto-tagger described above. Our model has two branches, one for the audio input and one for the user embeddings input. The audio branch is identical to the one described above, i.e. 4 pairs of convolutional and max pooling layers with ReLu activation. The input to the user branch is the user embedding of size 256. We apply batch normalization to it followed by a fully connected layer with 128 units and Relu activation. We concatenate the output of the audio branch and the user branch after applying batch normalization to each. We pass the concatenated output to a fully connected layer with 256 hidden units with ReLu activation function and apply a dropout with 0.3 ratio for regularization. The final layer is made of 10 output units with a Softmax activation function. We train the model with minimizing the categorical cross entropy using the same configuration as in the previous model, described in Section 4.1. We present the flowchart of the complete model in Figure 3
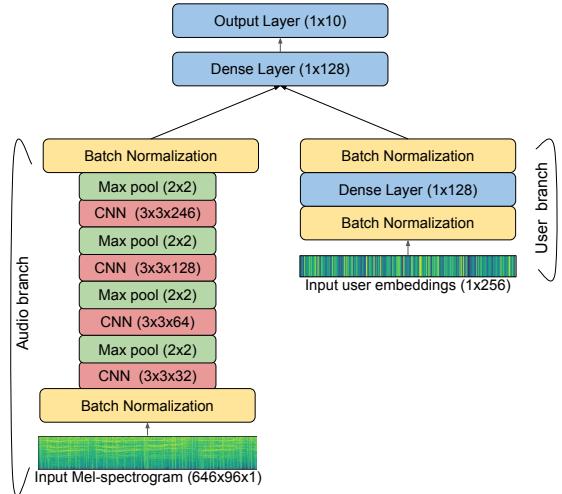
4.3 User Embeddings

The user embeddings are computed by applying implicit alternating least squares matrix factorization (ALS) [11, 16] on the users/tracks interactions matrix. The matrix represents the user listening count of the tracks available, with an exponential decay applied based on the lapse since the last listening, i.e. the more recent and frequent a track is listened to, the higher the interaction value. The user embedding is represented as a 256-dimensions vector.

However, the user listening histories are proprietary and represent sensitive data. Additionally, the detailed derivation of the embeddings is an internal procedure at Deezer for the recommendation algorithm. Hence, in order to allow the reproducibility of the current work, we directly release the pre-computed embeddings for the anonymized users present in our dataset.

5. RESULTS

We evaluate the two models according to the evaluation protocol proposed in Section 3. First, we evaluate the audio based model with the 3 scenarios: MO-MG, MO-SG, SO-SG. Then, we evaluate the User+Audio model in the SO-SG scenario. Last, we perform the user satisfaction-based evaluation on both models for the SO-SG scenario. In all evaluation protocols, the metrics were macro-averaged.

Figure 3. Architecture of the Audio+User-based model.

	AUC	Recall	Precision	f1-score
car	0.56	0.96	0.47	0.63
gym	0.71	0.87	0.58	0.7
happy	0.58	0.87	0.37	0.52
night	0.59	0.97	0.48	0.64
relax	0.77	0.8	0.61	0.69
running	0.65	0.91	0.56	0.69
sad	0.77	0.72	0.54	0.61
summer	0.6	0.97	0.61	0.75
work	0.53	0.99	0.47	0.64
workout	0.75	0.84	0.52	0.64
average	0.65	0.89	0.52	0.65

Table 2. Results of the audio-based model (multi-label outputs) on the user-agnostic dataset (multiple groundtruth), MO-MG scenario.

5.1 Audio-based Multi-output Multi-groundtruth (MO-MG Scenario)

Table 5.1 shows the results of the audio-based multi-label classification model on our collected dataset without considering the user. The results are consistent with previous studies on context auto-tagging [13]. They show that certain contexts are easier to predict using only the audio input. These are general contexts with similar music style preferences by different users, e.g. 'gym' and 'relax'. By contrast, other contexts are harder to predict from audio only as users listen to more personalized music, e.g. 'work' and 'car'. In consequence, we hypothesis that the variance of the AUC scores across contexts is related to the context dependency on users. Precisely, some contexts could depend more on users than others, making the latter harder to classify without considering the user information.

5.2 Audio-based Multi-output Single-groundtruth (MO-SG Scenario)

Table 5.2 shows the results of the same audio-based multi-label classification model which we now evaluate considering the user. The same audio track will now be presented

	AUC	Recall	Precision	f1-score
car	0.54	0.87	0.09	0.17
gym	0.66	0.6	0.18	0.27
happy	0.57	0.67	0.08	0.14
night	0.57	0.6	0.11	0.19
relax	0.74	0.53	0.25	0.34
running	0.6	0.57	0.15	0.23
sad	0.75	0.52	0.21	0.3
summer	0.58	0.78	0.17	0.29
work	0.52	0.55	0.09	0.15
workout	0.71	0.41	0.17	0.24
average	0.62	0.61	0.15	0.23

Table 3. Results of the audio-based model (multi-label outputs) on the user-based dataset (single ground-truth), MO-SG scenario

	AUC	Recall	Precision	f1-score
car	0.54	0	0.03	0.001
gym	0.66	0.44	0.17	0.24
happy	0.57	0	0	0
night	0.57	0.004	0.14	0.007
relax	0.74	0.6	0.23	0.33
running	0.6	0.05	0.15	0.07
sad	0.75	0.003	0.16	0.006
summer	0.58	0.36	0.18	0.25
work	0.52	0	0.2	0
workout	0.71	0.13	0.18	0.15
average	0.62	0.16	0.14	0.11

Table 4. Results of the audio-based model (forced to single-label output) on the user-based dataset (single ground-truth), SO-SG scenario

several times to the system, i.e. for each user who has annotated this track. While the groundtruth is now single-label and will change for each user, the system will output the same estimated tags independently of the user, i.e. the system does not consider the user as input. We observe a sharp decrease in the precision of the model due to false positive predictions for each user. Indeed, since the output of the system is multi-label, it will output several labels for each track, many of them will not correspond to the current user. The high recall of the model shows that it often predicts the right contextual use for many users. However, it also predicts wrong contexts for many other users. That is due to the limitation of the model which predicts all suitable contexts for all users.

5.3 Audio-based Single-output Single-groundtruth (SO-SG Scenario)

Table 5.3 shows the results of the same audio-based multi-label classification model when restricted to a single prediction per track. While this is not the real-world case of using the audio-based model, it allows a direct comparison to the single-label User+Audio based model. In this case, we see a sharp drop in the recall due to the limitation of a single prediction per track.

	AUC	Recall	Precision	f1-score
car	0.61	0.12	0.13	0.13
gym	0.71	0.16	0.24	0.19
happy	0.64	0.22	0.12	0.16
night	0.61	0.03	0.14	0.05
relax	0.76	0.41	0.29	0.34
running	0.69	0.26	0.22	0.24
sad	0.83	0.5	0.33	0.4
summer	0.65	0.2	0.3	0.24
work	0.58	0.03	0.12	0.04
workout	0.75	0.37	0.2	0.26
average	0.68	0.23	0.21	0.2

Table 5. Results of the audio+user model (single-label output) on the user-based dataset (single ground-truth), SO-SG scenario.

	Accuracy	Recall	Precision	f1-score
Audio	0.21	0.204	0.243	0.216
Audio+User	0.254	0.246	0.295	0.26

Table 6. Comparison of user-based evaluation for the two models

5.4 Audio+User Single-output Single-groundtruth (SO-SG Scenario)

Table 5.4 shows the results for the proposed Audio+User model. Comparing these results with the ones presented in Table 5.3, we observe that the model is performing better than the audio-based model for almost all metrics and labels. The f1-score almost doubles when adding the user information. Additionally, for certain labels as *car, happy, running, sad, summer, work*, the influence of adding the user information is obvious compared to all cases of audio-based evaluation when comparing the AUC values. This is consistent with our hypothesis that for certain labels the influence of user preferences is much stronger than for other labels.

5.5 User Satisfaction-focused Scenario

Finally, we assess the user satisfaction by evaluating the performance of the two models on each user independently. We replace the AUC metric with accuracy because AUC is not defined in the case of certain users where a specific label is positive for all samples. Table 5.5 shows the average performance of each model when computed per user. In this case, we observe how the Audio+User model satisfies the users more on average in terms of all evaluation metrics. By investigating the recall and precision, we noticed that our model results in a larger number of true positives, i.e. predicting the correct context for each user, and a lower number of false positives, i.e. less predictions of the wrong contextual tags for each user. The audio-based model is prone to a higher false positives due to predicting the most probable context for a given track regardless of the user. To sum up, including the user information in the model has successfully proven to improve the estimation of the right contextual usages of tracks.

6. CONCLUSION AND FUTURE WORK

Predicting the contextual use of music tracks is a challenging problem with multiple factors affecting the user preferences. Through our study, we showed that including the user information, represented as user embeddings based on the listening history, improves the model's capability of predicting the suitable context for a given user and track. This is an important result towards building context-aware recommendation systems that are user-personalized, without requiring the exploitation of extensive user private data such as location tracking [3]. However, there is still large room for improvement to successfully build such systems.

Our current model relies on using the audio content, which is suitable for the cold-start problem of recommending new tracks [6, 23]. However, constructing representative user embeddings requires active users in order to properly infer the listening preferences. Future work could investigate the impact of using different types of user information, such as demographics [8], which could be suitable for the user cold-start or less active users too.

Additionally, we focused on the case of a single contextual tag for each user and track pair. In practice, a user could listen to the same track in multiple contexts, i.e. tag prediction would be modelled a multi-label classification problem at the user level. Future studies could further investigate this more complex case of adding the user information in the multi-label settings.

Finally, while we have proven the advantage of our system on a subset of contexts. Extending the study to a larger number of possible contexts still needs to be addressed. In reality, users listen to music in more diverse contexts, adding levels of complexity to the addressed problem.

7. ACKNOWLEDGEMENT

8. REFERENCES

[1] Thierry Bertin-Mahieux, Douglas Eck, Francois Maillet, and Paul Lamere. Autotagger: A model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, 37(2):115–135, 2008.

[2] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. Recsys challenge 2018: Automatic music playlist continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018.

[3] Zhiyong Cheng and Jialie Shen. Just-for-me: An adaptive personalization system for location-aware social music recommendation. In *Proceedings of International Conference on Multimedia Retrieval*, 2014.

[4] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*, 2016.

[5] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017.

[6] Szu-Yu Chou, Yi-Hsuan Yang, Jyh-Shing Roger Jang, and Yu-Ching Lin. Addressing cold start for next-song recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

[7] Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. In *Advances in neural information processing systems*, 2008.

[8] Bruce Ferwerda and Markus Schedl. Investigating the relationship between diversity in music consumption behavior and cultural dimensions: A cross-country analysis. In *UMAP*, 2016.

[9] Michael Gillhofer and Markus Schedl. Iron maiden while jogging, debussy for dinner? In *International Conference on Multimedia Modeling*. Springer, 2015.

[10] Alinka E Greasley and Alexandra Lamont. Exploring engagement with music in everyday life using experience sampling methodology. *Musicae Scientiae*, 15(1):45–71, 2011.

[11] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016.

[12] Mohammad Hossin and MN Sulaiman. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):1, 2015.

[13] Karim Ibrahim, Jimena Royo-Letelier, Elena Epure, Geoffroy Peeters, and Gael Richard. Audio-based auto-tagging with contextual tags for music. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[14] Marius Kaminskas and Francesco Ricci. Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, 6(2-3):89–119, 2012.

[15] Taejun Kim, Jongpil Lee, and Juhan Nam. Sample-level cnn architectures for music auto-tagging using raw waveforms. In *Proceedings of the 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2018.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[17] Paul Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37:101 – 114, 2008.

[18] Adrian C North and David J Hargreaves. Situational influences on reported musical preference. *Psychomusicology: A Journal of Research in Music Cognition*, 15(1-2):30, 1996.

[19] Martin Pichl, E. Zangerle, and G. Specht. Towards a Context-Aware Music Recommendation Approach: What is Hidden in the Playlist Name? In *Procedings of International Conference on Data Mining Workshop (ICDMW)*, 2015.

[20] Jordi Pons Puig, Oriol Nieto, Matthew Prockup, Erik M Schmidt, Andreas F Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, 2018.

[21] Markus Schedl, Arthur Flexer, and Julián Urbano. The neglected user in music information retrieval research. *Journal of Intelligent Information Systems*, 41(3):523–539, 2013.

[22] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2011.

[23] Andreu Vall, Hamid Eghbal-zadeh, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. Music playlist continuation by learning from hand-curated examples and song features: Alleviating the cold-start problem for rare and out-of-set songs. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, 2017.

[24] Xinxi Wang, David Rosenblum, and Ye Wang. Context-aware mobile music recommendation for daily activities. In *Proceedings of the 20th ACM international conference on Multimedia*. ACM, 2012.

[25] Minz Won, Sanghyuk Chun, Oriol Nieto, and Xavier Serrc. Data-driven harmonic filters for audio representation learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[26] Minz Won, Sanghyuk Chun, and Xavier Serra. Toward interpretable music tagging with self-attention. *arXiv preprint arXiv:1906.04972*, 2019.

[27] Karthik Yadati, Cynthia Liem, Martha Larson, and Alan Hanjalic. On the automatic identification of music for common activities. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 2017.

MULTIPLE F0 ESTIMATION IN VOCAL ENSEMBLES USING CONVOLUTIONAL NEURAL NETWORKS

Helena Cuesta[1]   **Brian McFee**[2]   **Emilia Gómez**[1,3]

[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona
[2] Music and Audio Research Lab & Center for Data Science, New York University, USA
[3] European Commission, Joint Research Centre, Seville
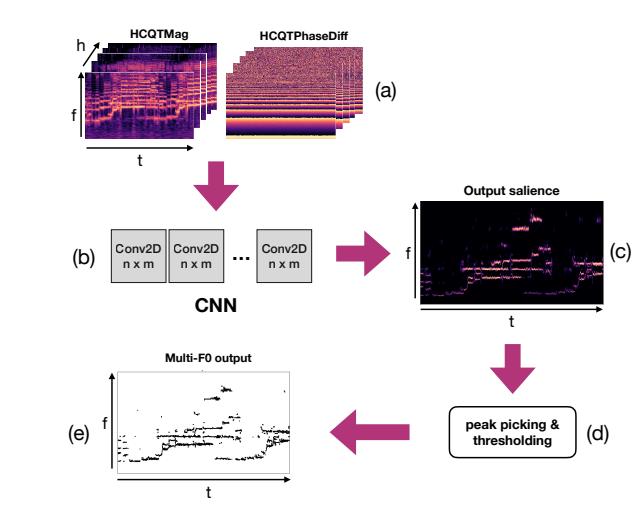
`helena.cuesta@upf.edu`

ABSTRACT

This paper addresses the extraction of multiple F0 values from polyphonic and *a cappella* vocal performances using convolutional neural networks (CNNs). We address the major challenges of ensemble singing, i.e., all melodic sources are vocals and singers sing in harmony. We build upon an existing architecture to produce a pitch salience function of the input signal, where the harmonic constant-Q transform (HCQT) and its associated phase differentials are used as an input representation. The pitch salience function is subsequently thresholded to obtain a multiple F0 estimation output. For training, we build a dataset that comprises several multi-track datasets of vocal quartets with F0 annotations. This work proposes and evaluates a set of CNNs for this task in diverse scenarios and data configurations, including recordings with additional reverb. Our models outperform a state-of-the-art method intended for the same music genre when evaluated with an increased F0 resolution, as well as a general-purpose method for multi-F0 estimation. We conclude with a discussion on future research directions.

1. INTRODUCTION

Ensemble singing is a well-established practice across cultures, found in a great diversity of forms, languages, and levels. However, all variants share the social aspect of collective singing, either as a form of entertainment or expressing emotions. In *The Science of the Singing Voice* [1], Sundberg claims that choral singing is one of the most widespread types of singing. In Western classical music, a *choir* is usually a group of singers divided into four sections: soprano, alto, tenor, bass (SATB); however, there exist many other forms of polyphonic singing, involving a diverse number of singers, parts, and vocal ranges. One example of such variants is a vocal quartet, where four singers—commonly with distinct vocal ranges—sing in

Figure 1. Overview of the proposed method. **(a)** Input features. **(b)** Convolutional architecture diagram. **(c)** Output salience map. **(d)** Peak picking and thresholding step to obtain F0 values from the salience activation in (c). **(e)** Multiple F0 representation, output of the framework.

harmony. Vocal quartets usually follow the SATB configuration; therefore, they are different from a *standard* choir in that there is only one singer per section.

Ensemble singing has not been widely studied in the field of Music Information Retrieval (MIR) in the recent years. We find a few early works focused on the acoustic properties of choral singing [2–4], and also a few more recent studies about some expressive characteristics of polyphonic vocal music, such as singer interaction and intonation [5–10], analysis of unison singing [11, 12], or choir source separation [13]. Most of these studies rely on individual recordings of each voice in the ensemble, which enable the automatic extraction of fundamental frequency (F0) contours from each isolated voice. The general applicability of these approaches is limited by the fact that vocal groups are rarely recorded with individual microphones. Another common strategy is using the polyphonic audio recordings along with their associated synchronized scores. However, the process of synchronizing a choral audio recording to a score is not straightforward, and therefore results are not always entirely trustworthy. Manual an-

notation is also one solution to obtain reliable F0 contours, but the process is highly time-consuming and expensive.

An alternate approach is to analyze a mixed, polyphonic recording to produce multiple F0 estimations simultaneously. This process enables the use of polyphonic singing recordings in the wild, and eliminates the need for separate audio recordings for each singer. However, multi-F0 estimation in polyphonic vocal music has been less often studied, likely due to the complexity and variety of sounds that a singer can produce, the timbre similarity between singers' voices, as well as the scarcity of annotated data of this kind [14].

In this paper, we focus on multiple F0 estimation for vocal ensemble audio mixtures. We build upon previous work on using neural networks to obtain an intermediate salience representation suitable for several tasks, including multi-F0 estimation [15]. In particular, we experiment with a set of convolutional neural networks (CNN), and combine magnitude and phase information, which is commonly neglected in the literature. We experiment with different fusion strategies and analyze the generalization capabilities of deep learning models in the presence of unison and reverbs. Figure 1 shows a diagram of the proposed method.

Following research reproducibility principles, data generation scripts and models are accessible [1] .

2. RELATED WORK

Multiple F0 estimation (also referred to as multi-F0 estimation, or multi-pitch estimation) is a sub-task of automatic music transcription (ATM) that consists of detecting multiple concurrent F0 values in an audio signal that contains several melodic lines at the same time [14, 16, 17]. Benetos et al. [17] summarize the main challenges of ATM as follows: polyphonic mixtures having multiple simultaneous sources with different pitches, loudness, and timbre properties; sources with overlapping harmonics; and the lack of polyphonic music datasets with reliable ground truth annotations, among others. They organize ATM approaches into four categories: frame-level (or multi-F0 estimation), note-level (or note-tracking), stream-level (or multi-F0 streaming), and notation-level. Our work focuses on the first category.

While monophonic F0 estimation is a well-researched topic, with state-of-the-art systems with excellent performances [18–21], multiple F0 estimation is still challenging. Research on this topic is commonly divided into several groups according to the nature of the employed methods. For instance, in [17] they report four categories: traditional signal processing methods, probabilistic methods, non-negative matrix factorization (NMF) methods, and neural networks.

Klapuri [22] proposed a signal processing based method for multi-F0 estimation in polyphonic music. He calculates the salience of F0 candidates by summing the amplitudes of its harmonic partials, and then uses an iterative method

where at every step an F0 is estimated and cancelled from the mixture before moving to the next iteration to estimate the next F0. The same author presented in [23] a similar method that incorporates information about human perception by means of an auditory model before the iterative process.

The system presented by Duan et al. [24] uses maximum-likelihood approach with the power spectrum as input. Spectral peaks are detected and two separate regions are defined accordingly: the peak region and the non-peak region, using a tolerance of half semitone from the detected peaks. In the maximum-likelihood process, both sets are treated independently, and the process of detecting F0 consists of optimizing a joint function that maximizes the probability of having harmonics that explain the observed peaks and minimizing the probability of having harmonics in the non-peak region. The F0 estimates are post-processed using neighbouring frames' estimates to produce more stable F0 contours.

A recent example of a multiple F0 estimation framework that employs neural networks is the system by Bittner et al. [15], DeepSalience (DS): a CNN trained to produce a multi-purpose pitch salience representation of the input signal. It is designed for multi-instrument pop/rock polyphonic music, and it provides an intermediate representation for MIR tasks such as melody extraction and multi-F0 estimation, outperforming state-of-the-art approaches in both cases. Following the premise that a pitch salience function is a suitable representation to extract F0 values, also exploited in [22, 23, 25, 26], this work keeps up with the advancements of deep neural networks to build data-driven salience functions. They use the harmonic constant-Q transform (HCQT) as input feature, which is a 3-dimensional array indexed by harmonic index h, frequency f, and time t: $\mathcal{H}[h, f, t]$. It comprises a set of constant-Q transforms (CQT) stacked together, each of them with its minimum frequency scaled by the harmonic index: $h \cdot f_{min}$.

While the above methods are well-suited for multiple F0 estimation in multi-instrumental music, *a cappella* polyphonic vocal music has several particularities that justify the need for dedicated techniques. One of the most significant challenges of analyzing vocal ensembles is due to harmonies occurring between distinct, overlapping vocal ranges. The timbre similarity, strong harmonic relationships, and overlapping frequency ranges hinder the extraction of concurrent F0 values in such music signals.

McLeod et al. [27] present a system for automatic transcription of polyphonic vocal music, which includes an initial step of estimating multiple F0s, and a second step of voice assignment, where each detected F0 is assigned to one of the SATB voices. They combine an acoustic model based on the factorization of an input log-frequency spectrogram for the multi-F0 estimation with a music language model based on hidden Markov models (HMM) for the voice assignment step. An earlier version of this method was presented in [14]; however, in the latter work the authors include a model integration step where the output of

Dataset	# of songs	Duration (hh:mm:ss)
Choral Singing Dataset [8]	3 songs	00:07:14
Dagstuhl ChoirSet [29]	2 songs	00:55:30
ESMUC Choir Dataset	3 songs	00:21:08
Barbershop Quartets [2]	22 songs	00:42:10
Bach Chorales [3]	26 songs	00:58:20

Table 1: Overview of the datasets used in this paper. The reported durations refer to the original mixtures before re-mixing stems and data augmentation. Dagstuhl ChoirSet and ESMUC Choir Dataset contain several takes per song.
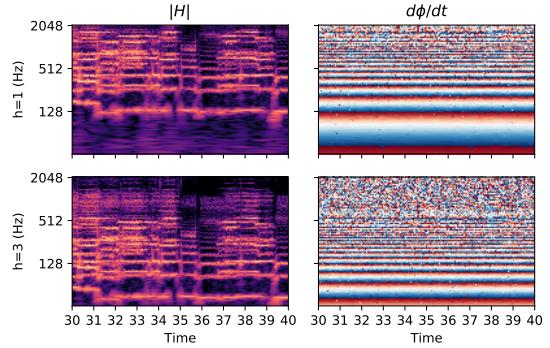
the music language model is further used in the acoustic model to improve the estimation of F0s. Their results show that integrating both parts of the system improves the performance of the voice assignment, and also of the multi-F0 estimation, since it eliminates many false positives.

Su et al. [28] also address some of the aforementioned issues by proposing an unsupervised method for multi-F0 estimation of choir and symphonic music. Their approach uses time-frequency reassignment techniques such as the synchrosqueezing transform (SST), which aims to better discriminate closely-located spectral components, such as unisons. They use an improved technique called ConceFT, which is based on the idea of multi-taper SST, but was proved to estimate instantaneous frequencies in noisy signals more precisely. These methods measure pitch salience and enhance the stability and localization of the F0 features needed for multi-F0 estimation.

3. DATASET

The lack of an appropriate and large enough annotated dataset has been a bottleneck in the use of machine learning techniques for multiple F0 estimation in ensemble singing. We address this difficulty by constructing a dataset that comprises several multi-track datasets of polyphonic singing with F0 annotations. We created a dataset by aggregating several existing multi-track polyphonic singing datasets. Table 1 shows an overview of the characteristics of each dataset individually. In this section, we describe them in more detail, as well as explain the process of data augmentation.

We use five datasets of similar characteristics. First, the Choral Singing Dataset (CSD) [8], a publicly available multi-track dataset of Western choral music. It comprises recordings of three SATB songs performed by a choir of 16 singers, four per section (4S4A4T4B), and it contains separate audio stems for each singer. Besides, it includes F0 annotations for each singer, which are automatically extracted and manually corrected. Similarly, the ESMUC Choir Dataset (ECS) is a proprietary dataset that comprises three songs performed by a choir of 13 singers (5S3A3T2B); it also includes audio stems for each

Figure 2. Input features examples for 10-seconds excerpt of a file from the training set. HCQT magnitude is depicted in the left column, and HCQT phase differentials in the right column, for $h = 1$ (top) and $h = 3$ (bottom).

singer and F0 annotations. The third dataset with comparable characteristics is the Dagstuhl ChoirSet (DCS) [29]: it consists of recordings of two songs performed by a choir of 13 singers (2S2A4T5B), and two different SATB quartets. This dataset also provides the audio stems and automatically extracted F0 annotations. Finally, we also add two commercial datasets: the Bach Chorales (BC) [3] and the Barbershop quartets (BSQ) [2]. They contain 26 and 22 songs, respectively, performed by vocal quartets—SATB in the first case, and tenor, lead, baritone, bass in the second case—as well as automatically extracted F0 annotations.

We exploit the multi-track nature of all datasets to create artificial mixtures of stems. We use PySox [30] to create all the possible combinations of singers, with the constraint of having one singer per part (SATB). In parallel, we also generate the multi-F0 annotations by combining the individual F0 contours of each singer in the mixture.

Besides creating the audio mixtures from individual recordings, we include two additional steps to improve generalization. First, we augment our dataset by means of pitch-shifting individual voices and re-mixing them. Particularly, we use pitch-shifting at a semitone scale: -2 to $+2$ semitones from the original signal. Second, our dataset contains two versions of each audio clip: the original one (obtained by mixing together individual stems), and the same song with reverb. We use the *Great Hall* impulse response (IR) from the Room Impulse Response Dataset in Isophonics [31], and convolve it with the audio mixtures of our dataset. For both tasks, we use MUDA, a software framework for musical data augmentation [32].
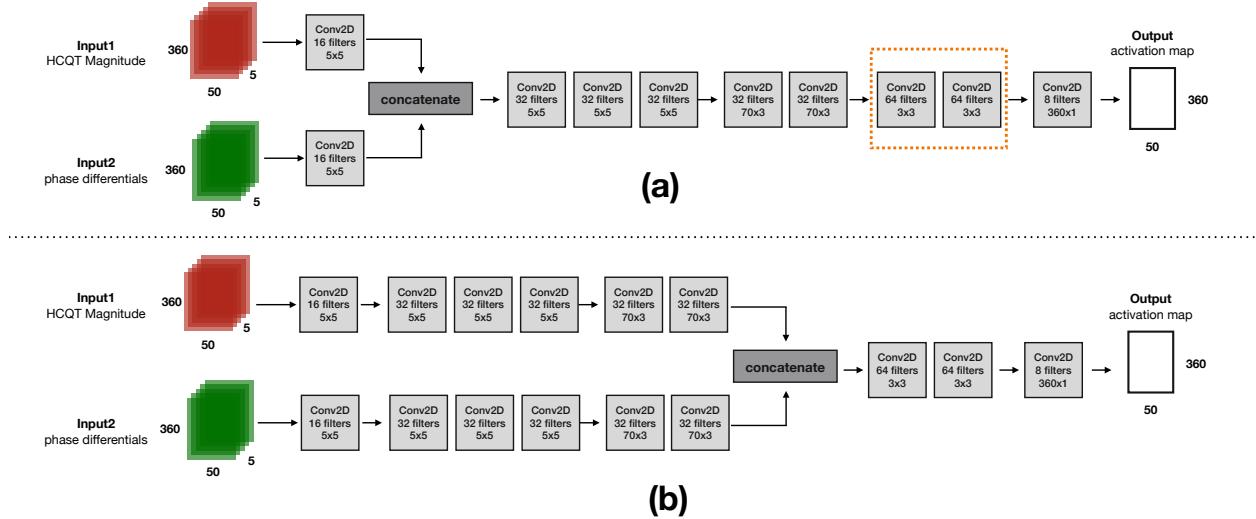
The dataset consists of 22910 audio files of diverse durations, from 10 seconds to 3 minutes. We split it into training (75%, 17184 files), validation (10%, 2291 files), and test (15%, 3435 files) subsets.

4. PROPOSED METHOD

In this section, we describe the input features, the target representations, the convolutional architectures we design, and the experiments we conduct.

Figure 3. Proposed convolutional architectures. **(a)** Early/Shallow and Early/Deep: the two layers inside the orange dotted rectangle are only part of Early/Deep. **(b)** Late/Deep: the concatenation of both inputs' contribution happens later in the network. In all networks, each layer is preceded by a batch-normalization step and the output of each layer is passed through a rectified linear unit activation function, except for the last layer, which uses a sigmoid.

4.1 Input features

Our networks have two separate inputs: the HCQT magnitude and the HCQT phase differentials. The HCQT is a 3-dimensional array $\mathcal{H}[h, t, f]$, indexed by harmonic (h), frequency (f), and time (t). It measures the hth harmonic of frequency f at time t, where $h = 1$ is the fundamental. This representation is based on computing a standard constant-Q transform (CQT) for each harmonic where the minimum frequency (f_{min}) is scaled by the harmonic number, $h \cdot f_{min}$. Detailed descriptions of the HCQT are presented in [15, 33]. For the HCQT calculation we use 60 bins per octave, 20 cents per bin, 6 octaves, and a minimum frequency of 32.7 Hz, which corresponds to a C1. We use five harmonics, so that $h \in \{1, 2, 3, 4, 5\}$ to compute the frequencies of the partials. Phase information is often discarded from neural network inputs, which commonly use magnitude representations such as the magnitude of the short-time Fourier transform (STFT). However, we also use the associated phase differentials. From signal processing theory we know that the phase differential of a signal contributes to a more precise calculation of the instantaneous frequency (ω_{ins}) [34]:

$$\omega_{ins} = \frac{\delta\varphi(t)}{\delta t} \rightarrow f_{ins} = \frac{1}{2\pi}\frac{\delta\varphi(t)}{\delta t} \tag{1}$$

where $\varphi(t)$ is the phase spectrum of the audio signal.

All audio files are resampled to a sampling rate of 22050 Hz, and we use a hop size of 256 samples. An example of the two input features examples is displayed in Figure 2.

4.2 Output representation

The output targets we use to train our networks are time-frequency representations with the same dimensions (2-D) as one of the input channels, i.e., $\mathcal{H}[1]$. We use the

ground truth F0 annotations (see Section 3) and assign each F0 value to the nearest time-frequency bin in the 2-D representation—which has the same time and frequency resolutions as the input—with a magnitude of 1. Non-active bins are set to 0, and we apply Gaussian blur with standard deviation 1 in the frequency direction to account for possible imprecisions in the predictions. We follow the same procedure as in DeepSalience and set the energy decay from 1 to 0 to cover half a semitone in frequency.

4.3 Models

Figure 3 depicts three convolutional architectures we propose: Early/Shallow, Early/Deep, and Late/Deep.

4.3.1 Early/Shallow and Early/Deep models

These models, inspired by DeepSalience, are illustrated in Figure 3a. They both consist of a fully convolutional architecture with two separate inputs: one for the HCQT magnitude and a second one for the HCQT phase differentials. Each of these inputs is first sent to a convolutional layer with 16 (5×5) filters. Then, the outputs of these two layers are concatenated. (5×5) filters cover approximately 1 semitone in frequency and 50 ms in time. After the concatenation, data passes through a set of convolutional layers including two layers with 32 (70×3) filters, which cover 14 semitones in frequency and are suitable for capturing harmonic relations within an octave. In the Early/Deep model we add two 64 (3×3) layers before the last layer with 8 filters that cover all frequency bins.

4.3.2 Late/Deep model

Late/Deep diagram is displayed in Figure 3b, and it follows a similar structure to Early/Shallow and Early/Deep. However, in this case both inputs are handled separately

until the layer with (70×3) filters; then, we concatenate both data streams and add the same layers: two layers with 64 filters (3×3), and the last layer with 8 filters that cover the whole frequency dimension, i.e., 360 bins.

In all models, batch normalization is applied at the input of every layer, and the outputs are passed through rectified linear units (ReLU), except for the output layer, which uses logistic activation (sigmoid) to map the output of each bin to the range [0, 1]. Using sigmoid at the output enables the interpretation of the activation map as a probability function, where the value between 0 and 1 represents the probability that a specific bin belongs to the set of F0s present in the input signal. All models in the experiments described next are trained to minimize binary cross-entropy between the target, $y[t, f]$, and the prediction, $\hat{y}[t, f]$, both of them values in the range [0, 1]:

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \qquad (2)$$

We use the Adam optimizer [35] with a learning rate of 0.001, and train for 100 epochs with a batch size of 16 patches of shape $(360, 50)$. We perform early stopping when the validation error does not decrease for 25 epochs.

4.4 Experimental setup

4.4.1 Evaluation metrics

We evaluate the models using the frame-wise metrics Precision, Recall, F-Score (or F-measure) as they are defined in the MIREX multiple-F0 estimation task [36] using the `mir_eval` library [37].

4.4.2 Experiment 1: fusion strategy

In the first experiment we use the whole dataset split into train-validation-test subsets (see Section 3), to measure the general performance of the three models. With this experiment we study the influence of magnitude and phase information fusion at an early stage of the network, i.e., Early/Shallow-Deep, or later, i.e., Late/Deep. For each model, we use the validation set to optimize the threshold we apply to the peaks extracted from the output salience representation. The optimal threshold is the one that maximizes the average accuracy across the validation set in each case. In addition, we train the Late/Deep model without the phase information, i.e., we remove the branch of the network dedicated to the phase. We intend to verify the hypothesis that including the phase as input to the network leads to more precise results.

4.4.3 Experiment 2: comparative analysis

In this experiment we evaluate the performance of our best-performing model from Experiment 1 on the BSQ [2]. This is one of the datasets used in [27, 38], allowing for a direct comparison between their method—also designed for ensemble singing—and the model we propose. These data are part of our original training dataset, but in this experiment we train the model excluding all the BSQ audio files, and then use them for exclusively for evaluation.

4.4.4 Experiment 3: generalization

In this last experiment, we aim to explore the effect of unison and reverb. Since vocal ensembles are commonly captured using a room microphone, such recordings usually contain reverb or similar effects, caused by the room acoustics. We train our best-performing model excluding all audio files with reverb from the dataset, and then evaluate it with conventional choir recordings from the dataset presented in [28], which is not part of our working dataset. In addition, we evaluate this model on a subset of reverb files from the original test set and compare the performance of this model to the model trained in Experiment 1.

5. RESULTS

5.1 Experiment 1: fusion strategy
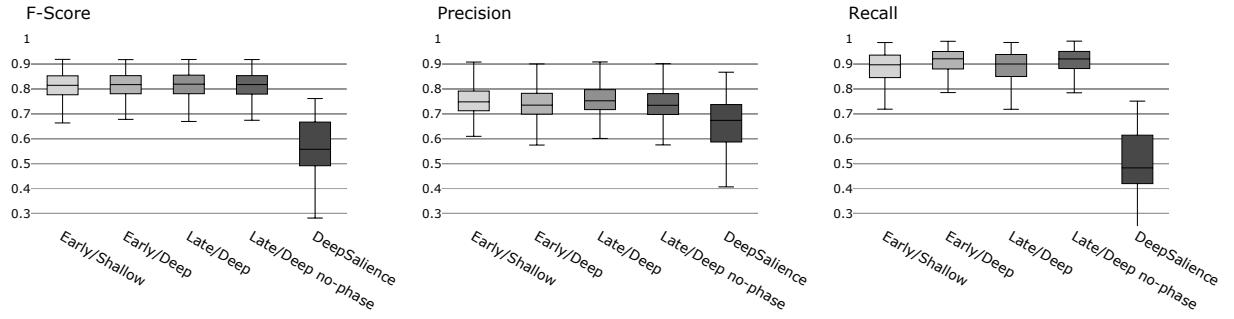
Results for Experiment 1 are depicted in Figure 4. While the three models have similar results, Late/Deep is slightly better in terms of F-Score, suggesting that the late fusion of magnitude and phase information is more robust than the early fusion. Figure 5 shows an excerpt of the multiple F0 output (red) together with their associated ground truth reference (black). We compare these results to DeepSalience using a detection threshold of 0.2 (optimized beforehand on the evaluation material), which has a lower performance, which we attribute to distribution shift from its training set. Additionally, we observe how the Late/Deep without phase information has a similar F-Score but lower precision, showing that including phase differentials as input is helpful for obtaining more accurate results. Since Late/Deep is the model with the best performance, we use it in the subsequent experiments.

5.2 Experiment 2: comparative analysis
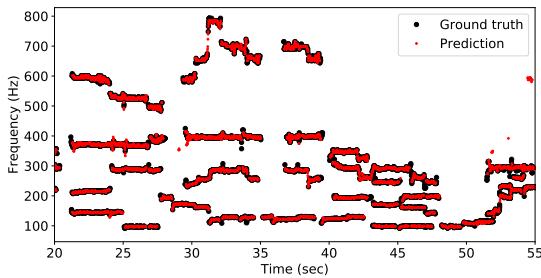
Experiment 2 results are summarized in Table 2 in terms of F-Score, and Precision and Recall (when available). We evaluate the predictions from our model on the BSQ dataset, and compare the results to the ones reported with MSINGERS [14], and VOCAL4-VA, the fully-integrated model from [27]. We use two different pitch tolerances: one semitone (100 cents) and 20 cents. While one semitone resolution is enough for transcription purposes, for analysis such as the ones described in Section 1, i.e., intonation and singer interaction, more pitch resolution is required. We observe that our model outperforms both baseline methods with two different pitch tolerances. In the 20 cents evaluation, the baseline models experience a performance drop (-17% and -26%), whereas the decrease in our model is much smaller, around -2% in the F-Score. This difference presumably resides in the fact that our model uses phase information to refine F0 estimates, therefore extracting a more precise value.

5.3 Experiment 3: generalization

Results from this experiment show that our model outperforms the method in [28] on their choir dataset: when we

Figure 4. Evaluation results on the non-pitch-shifted audio files of the test set. We compare our three models to Deep-Salience as a baseline, as well as to the Late/Deep network trained without the phase differentials (Late/Deep no-phase). Note that outliers are excluded from the plots for an easier visualization.

Figure 5. Example output from Experiment 1, Late/Deep model. Predictions (red) are plotted over the ground truth reference (black).

Method	100 cents			20 cents		
	F	P	R	F	P	R
MSINGERS [14]	0.708 (0.06)	0.685 (0.06)	0.736 (0.07)	0.537 (0.07)	0.620 (0.07)	0.477 (0.08)
VOCAL4-VA [27]	0.757 (0.06)	-	-	0.490	-	-
Late/Deep	**0.846** (0.03)	0.812 (0.03)	0.884 (0.04)	**0.831** (0.03)	0.797 (0.03)	0.868 (0.04)

Table 2: Multi-F0 estimation results (F-Score (F), precision (P), and recall (R)) on the Barbershop quartets, for different pitch tolerances. Values in parentheses refer to the standard deviation. Best scores are highlighted in bold.

calculate the average F-Score across the whole dataset, using the threshold optimized on the validation set, we obtain 0.704, while their best-performing method reaches an average F-Score of 0.653. Note that this dataset contains short excerpts of commercial choir recordings, with several singers per section and a large reverb effect, which differs from our training material. Therefore, these results suggest that our model is robust to recordings in such context. However, a larger experiment with similar data would be necessary, since this dataset is very small. The second part of this experiment is the evaluation of a subset of ten reverb files from the test set. In terms of F-Score, and as expected, the model that includes reverb files in the training set (Experiment 1) improves by slightly more than 10% on

average with respect to the model that excludes these files (Experiment 3). Therefore, we conclude that the presence of both reverb and dry signals in the training set is beneficial for the performance of a wider range of recording conditions such as reverb.

6. CONCLUSIONS

In this paper, we proposed a set of novel convolutional architectures for multiple F0 estimation in *a cappella* ensemble singing, combining magnitude and phase information. For training, we created an annotated dataset of polyphonic singing voice by aggregating several existing datasets, and augmented it by means of pitch-shifting and reverberation.

We conducted several experiments to evaluate different aspects of the detection process. We evaluated the overall performance of three models as compared to a deep learning based multi-purpose multiple F0 estimation system, and found that our models outperform the baseline when applied on ensemble singing. We also verified that using phase information at the input, together with the magnitude, improves the precision of the F0 estimates. We compared our best-performing model to one existing approach for multiple F0 estimation in vocal ensembles, and demonstrated that it outperforms it with two different F0 resolutions (100 and 20 cents). In addition, we compared our model to an approach specifically designed for choir and symphonic music and found that our model is robust in conditions of unison and high reverb. However, further experiments with a larger amount of data are required to verify these findings.

Although our results are a strong contribution to addressing the limitations of deep learning architectures for vocal music, there are some further steps that would potentially improve the performance of our models. Informal experiments showed that post-processing the output F0 contours increases their time continuity; therefore, the overall quality of the output improves. Further steps also include not only estimating the F0 values frame-wise, but also assigning each of them to a singer, which is a challenging task if the number of singers is not known a priori.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] J. Sundberg, *The Science of the Singing Voice*. Northern Illinois University Press, 1987.

[2] T. D. Rossing, J. Sundberg, and S. Ternström, "Acoustic comparison of voice use in solo and choir singing," *The Journal of the Acoustical Society of America*, vol. 79, no. 6, pp. 1975–1981, 1986.

[3] S. Ternström, "Perceptual evaluations of voice scatter in unison choir sounds," *STL-Quarterly Progress and Status Report*, vol. 32, pp. 041–049, 1991.

[4] S. Ternström, "Choir acoustics – an overview of scientific research published to date," *Speech, Music and Hearing Quarterly Progress and Status Report*, vol. 43, no. April, pp. 001–008, 2002.

[5] S. Zadig, G. Folkestad, and V. Lyberg-Ahlander, "Multi-track recordings of choral singers: Development and validation of a method to identify activities and interaction in the choral voice, based on recordings of the individual singers," *Bulletin of Empirical Music Education Research*, vol. 7, no. 1, pp. 1–20, 2016.

[6] J. Devaney, M. I. Mandel, and I. Fujinaga, "A study of intonation in three-part singing using the automatic music performance analysis and comparison toolkit (AMPACT)," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 2012, pp. 511–516.

[7] J. Dai and S. Dixon, "Analysis of interactive intonation in unaccompanied SATB ensembles," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 599–605.

[8] H. Cuesta, E. Gómez, A. Martorell, and F. Loáiciga, "Analysis of intonation in unison choir singing," in *Proceedings of the International Conference of Music Perception and Cognition (ICMPC)*, Graz, Austria, 2018, pp. 125–130.

[9] J. Dai and S. Dixon, "Singing together: Pitch accuracy and interaction in unaccompanied unison and duet singing," *The Journal of the Acoustical Society of America*, vol. 145, no. 2, pp. 663–675, 2019.

[10] C. Weiss, S. J. Schelcht, S. Rosenzweig, and M. Müller, "Towards Measuring Intonation Quality of Choir Recordings: A Case Study on Bruckner's Locus Iste," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 276–283.

[11] H. Cuesta, E. Gómez, and P. Chandna, "A framework for multi-f$_0$ modeling in satb choir recordings," in *Proceedings of the Sound and Music Computing Conference (SMC)*, Málaga, Spain, 2019, pp. 447–453.

[12] P. Chandna, H. Cuesta, and E. Gómez, "A deep learning based analysis-synthesis framework for unison singing," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[13] D. Petermann, P. Chandna, H. Cuesta, J. Bonada, and E. Gómez, "Deep learning based source separation applied to choir ensembles," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[14] R. Schramm and E. Benetos, "Automatic transcription of a cappella recordings from multiple singers," in *Proceedings of the AES Conference on Semantic Audio*, Erlangen, Germany, 2017.

[15] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for F0 tracking in polyphonic music," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 63–70.

[16] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, 2013.

[17] E. Benetos, S. Dixon, and Z. Duan, "Automatic Music Transcription : An Overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2018.

[18] A. D. Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[19] A. Camacho and J. G. Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 124, no. 3, pp. 1638–1652, 2008.

[20] M. Mauch and S. Dixon, "pYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 659–663.

[21] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A Convolutional Representation for Pitch Estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 161–165.

[22] A. P. Klapuri, "Multiple fundamental frequency estimation by summing harmonic amplitudes," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2006, pp. 216–221.

[23] A. Klapuri, "Multipitch analysis of polyphonic music and speech signals using an auditory model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 255–266, Feb 2008.

[24] Z. Duan, B. Pardo, and C. Zhang, "Multiple Fundamental Frequency Estimation by Modeling Spectral Peaks and Non-peak Regions," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 8, pp. 2121–2133, 2010.

[25] M. P. Ryynänen and A. P. Klapuri, "Automatic transcription of melody, bass line, and chords in polyphonic music," *Computer Music Journal*, vol. 32, no. 3, pp. 72–86, 2008.

[26] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 20, pp. 1759–1770, 08 2012.

[27] A. McLeod, R. Schramm, M. Steedman, and E. Benetos, "Automatic transcription of polyphonic vocal music," *Applied Sciences*, vol. 7, no. 12, 2017.

[28] L. Su, T.-Y. Chuang, and Y.-H. Yang, "Exploiting frequency, periodicity and harmonicity using advanced time-frequency concentration techniques for multipitch estimation of choir and symphony." in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, New York City, USA, 2016, pp. 393–399.

[29] S. Rosenzweig, H. Cuesta, C. Weiss, F. Scherbaum, E. Gómez, and M. Müller, "Dagstuhl ChoirSet: A Multitrack Dataset for MIR Research on Choral Singing," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 98–110, 2020.

[30] R. M. Bittner, E. Humphrey, and J. P. Bello, "Pysox: Leveraging the audio signal processing power of sox in python," in *Proceedings of the International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, 2016.

[31] R. Stewart and M. Sandler, "Database of omnidirectional and b-format room impulse responses," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010, pp. 165–168.

[32] B. McFee, E. J. Humphrey, and J. P. Bello, "A software framework for musical data augmentation." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 248–254.

[33] R. M. Bittner, B. McFee, and J. P. Bello, "Multitask learning for fundamental frequency estimation in music," *ArXiv*, vol. abs/1809.00381, 2018.

[34] B. Boashash, "Estimating and Interpreting the Instantaneous Frequency of a Signal. I. Fundamentals," *Proceedings of the IEEE*, vol. 80, no. 4, pp. 520–538, 1992.

[35] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ArXiv*, vol. abs/1412.6980, 2014.

[36] M. Bay, A. F. Ehmann, and S. J. Downie, "Evaluation of multiple-f0 estimation and tracking systems"," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, 2009, pp. 315–320.

[37] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[38] R. Schramm, A. McLeod, M. Steedman, and E. Benetos, "Multi-pitch detection and voice assignment for a cappella recordings of multiple singers," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

THE MULTIPLE VOICES OF MUSICAL EMOTIONS: SOURCE SEPARATION FOR IMPROVING MUSIC EMOTION RECOGNITION MODELS AND THEIR INTERPRETABILITY

Jacopo de Berardinis[1,2] **Angelo Cangelosi**[1] **Eduardo Coutinho**[2]

[1] Machine Learning and Robotics Group, University of Manchester
[2] Applied Music Research Lab, University of Liverpool

`jacopo.deberardinis@manchester.ac.uk`

ABSTRACT

Despite the manifold developments in music emotion recognition and related areas, estimating the emotional impact of music still poses many challenges. These are often associated to the complexity of the acoustic codes to emotion and the lack of large amounts of data with robust golden standards. In this paper, we propose a new computational model (EmoMucs) that considers the role of different musical voices in the prediction of the emotions induced by music. We combine source separation algorithms for breaking up music signals into independent song elements (vocals, bass, drums, other) and end-to-end state-of-the-art machine learning techniques for feature extraction and emotion modelling (valence and arousal regression). Through a series of computational experiments on a benchmark dataset using source-specialised models trained independently and different fusion strategies, we demonstrate that EmoMucs outperforms state-of-the-art approaches with the advantage of providing insights into the relative contribution of different musical elements to the emotions perceived by listeners.

1. INTRODUCTION

The ability of music to express and induce emotions [15,21] and act as a powerful tool for mood regulation [28] are well-known and demonstrable. Indeed, research shows that music listening is a commonly used, efficacious, and adaptable device to achieve regulatory goals [31], including coping with negative experiences by alleviating negative moods and feelings [17].

Crucial to this process is selecting the music that can facilitate the listener to achieve a determined mood regulation target, which often is not an easy task. In order to support listeners in this process, emotion-aware music recommendation systems became popular as they offer the possibility to explore large music libraries using affective cues. Indeed, recommending music based on the emotion of the listener at home [10] or background music personalised for the ones present in a restaurant would provide a more personal and enjoyable user experience [14].

At the core of these systems is music emotion recognition (MER), an active field of research in music information retrieval (MIR) for the past twenty years. The automatic prediction of emotions from music is a challenging task due to the subjectivity of the annotations and the lack of considerable data for effectively training supervised models. Song et al. [29] also argued that MER methods tend to perform well for genres such as classical music and film soundtracks, but not yet for popular music [25]. In addition, it is difficult to interpret emotional predictions in terms of musical content, especially for models based on deep neural networks. Although a few approaches exist for interpretable MER [5], the recognition accuracy of these methods is compromised, with the resulting performance loss often referred to as "cost of explainability".
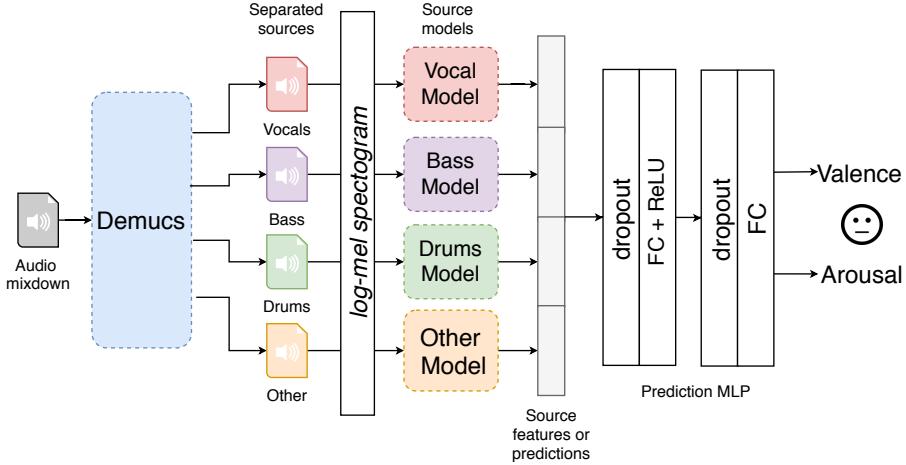
As different voices within a composition can have a distinct emotional impact [13], our work leverages state-of-the-art deep learning methods for music source separation (MSS) to reduce the complexity of MER when limited training data is available. The proposed architecture (EmoMucs) is based on combining source separation methods with a parallel block of source-specialised models trained independently, subsequently aggregated with a fusion strategy. To benchmark our idea, we evaluated EmoMucs on the popular music with emotional annotations (PMEmo) dataset [38], and compared its performance with two reference deep learning models for MER. Experimental results demonstrate that our method achieves better performance for valence recognition, and comparable ones for arousal, while providing increased interpretability.

The main technical contributions are manifold: (i) we provide an in-depth evaluation of two reference models for MER, (ii) we propose a computational model that achieves an improved performance on the current baselines, under similar experimental conditions, and finally (iii) we show that our model provides no cost of interpretability.

The rest of the paper is structured as follows: Section 2 gives a primer on MER and an overview of related work on content-based methods, whilst Section 3 outlines the base-

Figure 1. An overall architecture illustrating our proposed model, EmoMucs.

line architectures and our novel EmoMucs model. Section 4 details the experimental evaluation carried out, including our results on interpretability. Finally, Section 5 draws conclusions and gives direction for future work.

2. BACKGROUND AND RELATED WORK

2.1 A primer on music emotion recognition

Prior to introducing content-based methods for MER, we provide the reader with the fundamentals concepts of the task and refer to [3, 16, 34, 35] for a detailed overview.

Induced vs perceived emotions. Perceived emotions refers to the recognition of emotional meaning in music [29]. Induced (or felt) emotions refer to the feelings experienced by the listener whilst listening to music.

Annotation system. The conceptualisation of emotion with its respective emotion taxonomy remains a longstanding issue in MER [29]. There exist numerous emotion models, from miscellaneous [19] to domain specific [36], categorical [11] and dimensional [24], with the latter two being the prevailing ones. Whilst the categorical model focuses on all the emotions evolving from universal innate emotions like happiness, sadness, fear and anger [11], the dimensional model typically comprises an affective two-dimensional *valence-arousal* space. Valence represents a pleasure-displeasure continuum, whilst arousal outlines the activation-deactivation continuum [24].

Time scale of predictions. Predictions can either be static or dynamic. In the former case, the representative emotion of a song is given by one valence and arousal value [16]. Emotion annotations can also be obtained over time (e.g. second-by-second valence-arousal labels), thus resulting in dynamic predictions [27].

Audio features. Musical compositions consist of a rich array of features such as harmony, tempo, loudness and timbre and these all have an effect on emotion. Previous work in MIR has fuelled around developing informative acoustic features [16]. However, as illustrated in other works [18, 22, 26] and to the best of our knowledge, there exists no dominant single feature for MER.

2.2 Methods for content-based MER

The field of MIR has followed a similar path to other machine learning ones. Prior to the deep learning era, most methods relied on manual audio feature extraction. Huq et al. [12] give an overview on how musical features were traditionally extracted and fed into different architectures such as support vector machines, k-nearest neighbours, random forests, deep belief networks and other regression models. These methods were tested for MER on Russell's well-established arousal and valence emotion model [24].

These were succeeded with deep learning methods. Such techniques, like binarised and bi-directional long short-term memory recurrent neural networks (LSTM-RNN) and deep belief networks, have also been successfully employed for valence and arousal prediction [33]. Other methods again used LSTM-RNNs for dynamic arousal and valence regression, on the acoustic and psychoacoustic features obtained from songs [6]. Most of these works stemmed from entries in the MediaEval emotion challenge [2]. Multimodality has also been an interest for this research community, where [9] looked into MER based on both the audio signal and the lyrics of a musical track. Again, deep learning methods such as LSTM-RNNs are at the core of the architectures proposed.

An important factor in machine learning has been to build interpretable models, to make them applicable to a wider array of applications. To the best of our knowledge, only a few works have attempted to build an interpretable model for MER. In [37], different model classes were built over the extracted and selected features. These vital features were filtered and wrapped, followed by shrinkage methods. In [5], a deep network based on two-dimensional convolutions is trained to jointly predict "*mid-level perceptual features*", related to emotional qualities of music [1], with emotion classes in a categorical annotation space.

Our work focuses on the prediction of induced emotions at a global time scale (static predictions). This is done in a continuous annotation space, as adopting a categorical one would not exhibit the same richness in induced human emotion [35]. The idea of using MSS methods for MER was first investigated in [32]. Our work differs in

the following: (i) we focus on valence-arousal MER and address the former as a regression task; (ii) our methods rely on state-of-the-art deep learning methods with no need of traditional methods for audio feature extraction; (iii) we investigate different fusion strategies and training approaches, and (iv) we provide an insightful analysis for the interpretability of our models.

3. METHODOLOGY

Our approach is based on the observation that different musical sources in a composition can evoke distinct emotional responses from the listeners [13]. Given a music piece, they can contribute differently to the overall induced emotion. For instance, the bass and the vocal lines of a track can be more informative to predict valence, whereas drums might have more impact on arousal. Nevertheless, our aim here is not to provide a general explanation of the emotional influence of musical parts, as this is often an individualistic property belonging to each track.

Instead, we propose a computational model for MER based on a decomposition of the original audio signal to the possible sources (e.g. vocals, drums, bass) that can be detected from it. By doing so, it will be easier for the model to process the audio stream whilst searching for emotion-related patterns in every single source. The aggregation of the resulting source-specific models within a single architecture would also account for the possible inter-source relationships. This approach can thus be considered as a way to provide prior knowledge to a model in order to reduce the complexity of the learning task when limited data is available – a recurring issue in MER.

Considering the technical challenges in MER, the design of a computational model based on music source separation has the potential to (i) improve the performance of the current solutions with the same amount of training data; (ii) provide a modular architecture which can be further adapted and fine-tuned with respect to each source-specific module, and (iii) quantify the contribution of each source to the final prediction for improved interpretability.

Our model, EmoMucs, achieves this through a multiplexed framework for emotion recognition. The architecture of our model is illustrated in Figure 1, with its building blocks explained in the following subsections.

3.1 Music source separation module

In the final step of music production, the tracks corresponding to each individual instrument [1] are mixed together in a single audio file known as mix-down. Music source separation (MSS) aims at reconstructing the individual sources from a mix-down. A reference categorisation of these sources is the *SiSec Mus* evaluation campaign [30], which is based on the following classes: (i) *vocals*, (ii) *drums*, (iii) *bass* and (iv) *other*. Given a mixdown, the goal of a MSS model is to generate a waveform for each of the four original sources.

Most of the approaches for MSS operate on the spectrograms generated by the short-time Fourier transform

_____

[1] We use *voice*, *instrument* and *source* interchangeably.

(STFT). They are trained to produce a mask on the magnitude spectrums for each frame and source [8]. The output audio is then obtained through an inverse STFT on the masked spectrograms, reusing the input mixture phase. However, a technical limitation of these approaches is the information loss resulting from the mix of sources, which cannot be easily recovered through masking.

For this reason we use *Demucs* [8], a recent deep learning model for MSS directly operating on the raw input waveform. Instead of relying on a masking approach, Demucs is inspired by models for music generation in the waveform domain. It implements a U-net architecture with a convolutional encoder-decoder, and a bidirectional LSTM between them to increase the number of channels exponentially with depth [7].

Given an audio track, our system starts by feeding it to Demucs. This results into four different source tracks – one for each SiSec Mus class. To ensure comparability of our architecture with the baseline methods for MER, we compute the log-mel spectogram of each source track.

3.2 Source models and fusion strategies

As illustrated in Figure 1, the log-mel spectrogram of each source track is then passed to the specific model associated to that source (e.g. the vocal's spectrogram is fed to the vocal model). By disentanglement, each sub-model processes a single voice independently, and learns the corresponding source-specific musical features for emotion recognition. This approach thus provides a high degree of flexibility, as it makes it possible to design the architecture of each sub-model specifically for the corresponding source. Nonetheless, to guarantee a fair comparison of our architecture with the current methods for MER, we use one of our baselines as the architecture for all source models.

The baseline models are based on two common deep learning architectures for MER, illustrated in Figure 2. The first is a one-dimensional convolutional neural network (C1D) that was proposed in [9] as an audio model for multimodal MER. The architecture consists of two one-dimensional convolutional layers followed by max-pooling and batch normalisation. Its resulting feature maps are then passed to two fully-connected layers with dropout masks to improve generalisation. The second baseline comprises a VGG-style network, demonstrated to be effective in several MIR tasks [4]. This musically-engineered model, C2D, consists of 5 two-dimensional convolutional blocks, each separated by max pooling and dropout layers. The two-dimensional pooling operators progressively decrease the size of each feature map, while keeping the same number of kernels (32) after each block. After the convolutional blocks, two-dimensional average pooling is applied to ensure that the resulting feature map is of size 32×1. Following dropout, a single fully-connected layer is then employed to predict arousal and valence.

The architecture of our source models can be either C1D or C2D, resulting in two different implementations of EmoMucs – *EmoMucs-C1D* and *EmoMucs-C2D*. To yield a final prediction of valence (V) and arousal (A), the features from the source models are concatenated and passed

Figure 2. The baseline models our system C1D and C2D. These are the building blocks for the source models.

to two fully-connected layers with dropout. Assuming C1D is chosen as the architecture for our source models, there are three possible ways to access the features of a source model. This gives three fusion strategies: *early* (E), *mid* (M) and *late* (L), depicted in Figure 2. The first strategy considers the features obtained after the convolutional layers; *mid* fusion concatenates the features learned after the first fully-connected layers; and the *late* strategy considers the output of each source model, which correspond to the VA predictions. From the definition of C2D, the *mid*-level fusion strategy is not possible with this architecture.

3.3 Our training approach

Considering the different role of each source model, there are three main strategies for training EmoMucs: *full*, *freeze* and *fine-tune*. The first approach consists in training the whole network from scratch and propagating the gradient back to the source models from the last fully-connected layer of EmoMucs. In contrast, the last two strategies are based on pre-training each source model separately as a first step. The full network is then trained until the concatenation level, for the *freeze* mode, or until the first convolutional layer of each source model, for the *fine-tune* mode. This last choice can be considered as a sort of fine-tuning strategy and should be implemented with small learning rates to avoid the source models to catastrophically forget what they have already learned independently.

4. EXPERIMENTS

Our method is validated using the baseline models C1D and C2D trained on the mix-downs as reference models. These are denoted as C1D-M and C2D-M respectively. The performance of the baselines is then compared with each source model trained independently. In particular, we compare CXD-M with CXD-{V | B | D | O}, where V, B, D, O denote the *vocals*, *bass*, *drum*, and *other* sources respectively, and X is a placeholder for 1, 2. This allows to verify how informative each source model is, and whether one of them outperforms the correspondent mix-down baseline.

Secondly, we experiment with the different fusion and training strategies of EmoMucs using all the source models. Similarly to the previous case, the performances of EmoMucs with either C1D or C2D architectures for its source models (denoted as EmoMucs-C1D and EmoMucs-C2D) are compared to C1D-M and C2D-M.

We evaluate the accuracy of the valence-arousal predictions with the root-mean-squared error (RMSE) and the R^2 score. The latter is the coefficient of determination, with the best score being 1 when the variability of the target data is fully captured by the regressor. Conversely, a score equal to 0 corresponds to a model always predicting the expected value of the target. To avoid biasing our evaluations on a single test set, each run employs nested cross-validation with 5 splits for the outer and inner folds.

4.1 Dataset

As mentioned in [25, 29], the current methods for MER perform well for genres such as classical music and film soundtracks, but their performances are still poor for popular music. For this reason, we chose the *popular music with emotional annotations* (PMEmo) dataset [38] for our experiments. This collection contains valence-arousal induced emotions for 794 songs, annotated by 457 subjects, and also provides: song metadata, music chorus clips in MP3 format and pre-computed audio features.
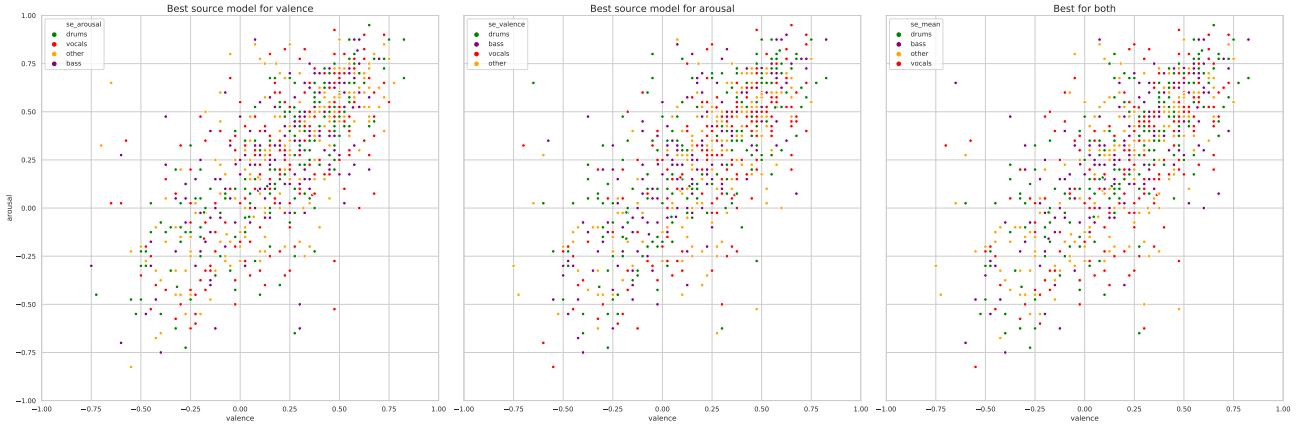
For our experiments, we consider the static valence-arousal annotations. As our model needs raw-audio data to feed Demucs and generate the separated sources from a given mix-down, we use 20-second randomly selected clips from each chorus. For 59 tracks with duration shorter than 20 seconds, we apply zero padding at the end of the clip to ensure fixed-size input. On average, the padding operation is used to compensate for 4.35 seconds. The arousal and valence annotations are scaled to the $[-1, 1]$ interval for improving the stability of the model. We choose not to augment the dataset as such strategy can potentially affect the emotional impact of music on listeners.

4.2 Implementation details

We use Librosa 0.7.2 [20] for computing the log-mel spectograms from the tracks, with a fast Fourier transform (FFT) window size of 512, 256 samples between successive frames and 96 Mel bands. Our models are implemented in PyTorch [23], and the source code to replicate these experiments is available at `github.com/jonnybluesman/emomucs`.

4.3 Experimental results

The results of our experiments are reported in Tables 1 and 2. From Table 1, we notice that the C2D architecture is

Figure 3. Target valence-arousal annotations of PMEmo in the $[-1, 1]$ interval, using colours to distinguish the C2D-based source model that better predicted each single annotation w.r.t. valence and arousal RMSE, together with their mean.

more accurate than C1D in all scenarios. In addition, the former baseline is also a more parsimonious model, due to the less number of parameters. All the baselines trained on the mix-down are considerably better than the source models considered independently. For both the architectures, we find that the drums model is the best between the source models at predicting valence, whereas the bass model is the worst for arousal. The performance disagreement for the source models using different architectures (e.g. C1D-V and C2D-V) suggests that the convolutional architecture plays a crucial role for the type of musical features that can be learned.

Baseline	# params	Input	RMSE		R2	
			V	A	V	A
C1D	86354	**M**	**.2600**	**.2444**	**.3489**	**.5573**
		V	.3048	.3214	.1131	.2426
		B	.2890	.3311	.2029	.1963
		D	.2710	.2961	.3000	.3572
		O	.2723	.2936	.2925	.3679
C2D	37698	**M**	**.2466**	**.2285**	**.4143**	**.6100**
		V	.2701	.2750	.3039	.4455
		B	.2762	.2924	.2718	.3732
		D	.2587	.2855	.3613	.4024
		O	.2633	.2748	.3381	.4462

Table 1. Evaluation of the baseline models trained on the mix-down (C1D-M, C2D-M) together with the corresponding source models. V, B, D, O denote *vocals*, *bass*, *drums* and *other* and they refer to the source models. Bold text highlights the best results for each baseline model. For both cases, the mix-down model achieves the best results.

As can be seen from Table 2, combining all source models in a single network has a crucial impact on the performance of the model. We conjecture that this is achieved by the architecture of EmoMucs, which makes it possible to account for all the possible inter-source relationships. In particular, EmoMucs-C1D with mid-level feature fusion and *freeze* mode training achieves a R^2 score of 0.4332 for valence, which is a considerable increase compared to 0.3489 for C1D. This is also reflected with a decrease of

the RMSE for valence, accounting for 0.2428 instead of 0.26. Considering that the valence-arousal annotations are scaled to the $[-1, 1]$ interval, we divide these values by 2 for a more intuitive interpretation of the error. Hence, 0.2428 and 0.26 can be considered as errors of 12.14% and 13% in the annotation interval. Analogously, EmoMucs C2D with late fusion and *freeze* mode training achieves valence $R^2 = 0.4814$ and RMSE = 0.2320 (11.6%), instead of $R^2 = 0.4143$ and RMSE = 0.2466 (12.33%) for the C2D baseline. On the other hand, the arousal predictions of EmoMucs are comparable to those of the baselines.

4.4 Interpretability

As the architecture of EmoMucs is based on a concatenation of features learned by each source model at a specific layer, it is possible to trace the contribution of each voice as well as those emerging from their interrelated connections. This form of interpretability is architecturally supported by our deep neural network, and it comes at no performance loss. This contrasts the work of Chowdhury et al. [5], who measured the "cost of explainibility" of their model by trading accuracy for interpretability.

A simple way to interpret EmoMucs is to isolate the performance of each model independently, as done in Table 1. It is also compelling to analyse the regression accuracy for each track in the dataset and visualise them together with the target annotations in the valence-arousal space. In Figure 3, this is done separately for valence and arousal by associating each target data point with a colour related to its best source model (the one with lowest valence and arousal RMSE for that target). If source models specialise in certain regions of the annotation space, e.g. drums and high arousal, we would expect them to form clusters in the annotation space. However, this hypothesis is rejected as Figure 3 does not suggest any clear specialisation of the source models in the annotation space. This supports our previous observation that each track has intrinsic features related to its emotional impact. For instance, two distinct tracks with very similar annotations can have a considerably different emotional influence from their sources.

Figure 4 reports the performance (R^2 score for valence

		Early				Mid				Late			
		RMSE		R2		RMSE		R2		RMSE		R2	
Model	Training	V	A	V	A	V	A	V	A	V	A	V	A
EmoMucs-C1D	freeze	.2536	.2580	.3803	.5064	**.2428**	**.2435**	**.4332**	**.5615**	.2453	.2475	.4208	.5470
	finetune	.2562	.2624	.3655	.4878	.2516	.2492	.3875	.5395	na			
	full	.2536	.2628	.3787	.4850	.2625	.2651	.3371	.4794				
EmoMucs-C2D	freeze	.2373	**.2307**	.4584	**.6046**	na				**.2320**	**.2322**	**.4814**	**.6004**
	finetune	.2444	.2442	.4256	.5560					na			
	full	.2541	.2543	.3793	.5212								

Table 2. Comparison of EmoMucs models with different fusion and training strategies.

Figure 4. EmoMucs-C2D trained with different combinations of source models.

and arousal) of EmoMucs-C2D using late fusion and *freeze* training mode for different combinations of source models. The best performance is achieved when using all sources (V-B-D-O). The contribution of source models varies with their combination. When using two sources, the combination of the *bass* and the *other* models gives better performance in the valence space, but for the arousal space an improved result is achieved when combining *vocals* with *other*. When three sources are considered, the combination of *drums*, *bass* and *other* achieves the best R^2 for valence, whereas, for arousal, excluding *bass* gives comparable results to EmoMucs-C2D with all the sources.

5. CONCLUSIONS

The task of computational music emotion recognition (MER) is particularly challenging due to several factors such as subjectivity within annotations, scarcity of labelled data for training supervised models, and inadequate data augmentation strategies. There is common belief that the current models perform well for classical music and film soundtracks, but their performances are still poor for popular music. To the best of our knowledge, improving the interpretability of MER models jeopardises their performance, thus introducing a "cost of explainability".

In this paper we introduced EmoMucs, a deep learning architecture for MER based on music source separation. First, our method separates the audio signal into different sources associated to vocals, drums, bass and other voices of the mix-downs. Different sub-models are then used to process each source independently, with their features being aggregated according to a fusion strategy.

We evaluated EmoMucs on the popular music with emotional annotations (PMEmo) dataset, and compared its performance with two common deep learning models for

MER trained on the mix-downs. Our results demonstrate that EmoMucs outperforms the baseline models for valence, and achieves comparable performance for arousal, while providing increased interpretability.

Our work achieves the following: (i) improved performance of the current solutions with the same amount of training data; (ii) a modular architecture which can be further adapted and fine-tuned with respect to each source-specific module, and (iii) a quantified contribution of each source to the final prediction for more interpretability.

The implementation of EmoMucs considered in our experiment is designed to prioritise the comparability of our approach to other baselines. In our future endeavours, we plan on optimising the architecture and hyper-parameters of each source model in order to specialise their design to the corresponding sources. Additionally, a study based on the analysis of the activations of the fusion layer would provide more detailed insights regarding the contribution of each source-model, thereby increasing the interpretability of our method and its potential applications.

6. REFERENCES

[1] Anna Aljanaki and Mohammad Soleymani. A data-driven approach to mid-level perceptual musical feature modeling. *arXiv preprint arXiv:1806.04903*, 2018.

[2] Anna Aljanaki, Yi-Hsuan Yang, and Mohammad Soleymani. Developing a benchmark for emotional analysis of music. *PLOS ONE*, 12:e0173392, 03 2017.

[3] Mathieu Barthet, György Fazekas, and Mark Sandler. Music emotion recognition: From content- to context-based models. In *CMMR*, 2012.

[4] Keunwoo Choi, George Fazekas, Mark B. Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. *ArXiv*, abs/1703.09179, 2017.

[5] Shreyan Chowdhury, Andreu Vall, Verena Haunschmid, and Gerhard Widmer. Towards explainable music emotion recognition: The route via mid-level features. In *ISMIR*, 2019.

[6] Eduardo Coutinho, George Trigeorgis, Stefanos Zafeiriou, and Björn Schuller. Automatically estimating emotion in music with deep long-short term memory recurrent neural networks. pages 1–3, 01 2015.

[7] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org, 2017.

[8] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.

[9] Rémi Delbouys, Romain Hennequin, Francesco Piccoli, Jimena Royo-Letelier, and Manuel Moussallam. Music mood detection based on audio and lyrics with deep neural net. In *ISMIR*, 2018.

[10] S. Dornbush, K. Fisher, K. McKay, A. Prikhodko, and Z. Segall. Xpod - a human activity and emotion aware mobile music player. In *2005 2nd Asia Pacific Conference on Mobile Technology, Applications and Systems*, pages 1–6, 2005.

[11] Paul Ekman. An argument for basic emotions. 1992.

[12] Arefin Huq, Juan Pablo Bello, and Robert Rowe. Automated music emotion recognition: A systematic evaluation. *Journal of New Music Research*, 39(3):227–244, 2010.

[13] David Huron, Neesha Anderson, and Daniel Shanahan. "you can't play a sad song on the banjo:" acoustic factors in the judgment of instrument capacity to convey sadness. *Empirical Musicology Review*, 9(1):29–41, 2014.

[14] Alejandro Jaimes, Nicu Sebe, and Daniel Gatica-Perez. Human-centered computing: A multimedia perspective. pages 855–864, 01 2006.

[15] Patrik N Juslin and John A. Sloboda. Handbook of music and emotion: Theory, research, applications. 2011.

[16] Youngmoo Kim, Erik Schmidt, Raymond Migneco, Brandon Morton, Jeffrey Scott, Jacquelin Speck, and Douglas Turnbull. Music emotion recognition: A state of the art review. *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, 01 2010.

[17] Emmanuel Kuntsche, Lydie Le Mével, and Irina Berson. Development of the four-dimensional motives for listening to music questionnaire (mlmq) and associations with health and social issues among adolescents. *Psychology of Music*, 44:219–233, 2016.

[18] Karl F MacDorman, Stuart Ough Chin-Chang Ho. Automatic emotion prediction of song excerpts: Index construction, algorithm design, and empirical comparison. *Journal of New Music Research*, 36(4):281–299, 2007.

[19] Stephen Mcadams, Bradley Vines, Sandrine Vieillard, B. Smith, and R. Reynolds. Influences of large-scale form on continuous ratings in response to a contemporary piece in a live concert setting. *Music Perception*, 22:297–350, 12 2004.

[20] Brian McFee, Vincent Lostanlen, Matt McVicar, Alexandros Metsai, Stefan Balke, Carl Thomé, Colin Raffel, Ayoub Malek, Dana Lee, Frank Zalkow, Kyungyun Lee, Oriol Nieto, Jack Mason, Dan Ellis, Ryuichi Yamamoto, Scott Seyfarth, Eric Battenberg, Rachel Bittner, Keunwoo Choi, Josh Moore, Ziyao Wei, Shunsuke Hidaka, nullmightybofo, Pius Friesch, Fabian-Robert Stöter, Darío Hereñú, Taewoon Kim, Matt Vollrath, and Adam Weiss. librosa/librosa: 0.7.2, January 2020.

[21] Leonard B Meyer. *Emotion and meaning in music*. University of chicago Press, 2008.

[22] Luca Mion and Giovanni De Poli. Score-independent audio features for description of music expression. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):458–466, 2008.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[24] Jonathan Posner, James A Russell, and Bradley S Peterson. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and psychopathology*, 17(3):715–734, 2005.

[25] Pasi Saari, Tuomas Eerola, and Olivier Lartillot. Generalizability and simplicity as criteria in feature selection: Application to mood classification in music. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1802–1812, 2010.

[26] Erik M Schmidt and Youngmoo E Kim. Prediction of time-varying musical mood distributions from audio. In *ISMIR*, pages 465–470, 2010.

[27] Erik M Schmidt, Douglas Turnbull, and Youngmoo E Kim. Feature selection for content-based, time-varying musical emotion regression. In *Proceedings of the international conference on Multimedia information retrieval*, pages 267–274, 2010.

[28] Thomas Schäfer, Peter Sedlmeier, Christine Städtler, and David Huron. The psychological functions of music listening. *Frontiers in Psychology*, 4:511, 2013.

[29] Yading Song and D Simon. How well can a music emotion recognition system predict the emotional responses of participants? In *Sound and Music Computing Conference (SMC)*, pages 387–392, 2015.

[30] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. The 2018 signal separation evaluation campaign. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 293–305. Springer, 2018.

[31] Annelies van Goethem and John Sloboda. The functions of music for affect regulation. *Musicae Scientiae*, 15(2):208–228, 2011.

[32] Jieping xu, Xirong Li, Yun Hao, and Gang Yang. Source separation improves music emotion recognition. 04 2014.

[33] Mingxing Xu, Xinxing Li, Haishu Xianyu, Jiashen Tian, Fanhang Meng, and Wenxiao Chen. Multi-scale approaches to the mediaeval 2015 "emotion in music" task. In *MediaEval*, 2015.

[34] Xinyu Yang, Yizhuo Dong, and Juan Li. Review of data features-based music emotion recognition methods. *Multimedia Systems*, 24:365–389, 2017.

[35] Yi-Hsuan Yang and Homer H. Chen. Machine recognition of music emotion: A review. *ACM Trans. Intell. Syst. Technol.*, 3:40:1–40:30, 2012.

[36] Marcel Zentner, Didier Grandjean, and Klaus Scherer. Emotions evoked by the sound of music: Characterization, classification, and measurement. *Emotion (Washington, D.C.)*, 8:494–521, 09 2008.

[37] JiangLong Zhang, Huang Xianglin, Lifang Yang, and Liqiang Nie. Bridge the semantic gap between pop music acoustic feature and emotion: build an interpretable model. *Neurocomputing*, 208, 06 2016.

[38] Kejun Zhang, Hui Zhang, Simeng Li, Changyuan Yang, and Lingyun Sun. The pmemo dataset for music emotion recognition. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ICMR '18, page 135–142, New York, NY, USA, 2018. Association for Computing Machinery.

BISTATE REDUCTION AND COMPARISON OF DRUM PATTERNS

Olivier Lartillot
RITMO Centre for Interdisciplinary Studies in
Rhythm, Time and Motion, University of Oslo
`olivier.lartillot@imv.uio.no`

Fred Bruford
Centre for Digital Music
Queen Mary University, United Kingdom
`fred.bruford@gmail.com`

ABSTRACT

This paper develops the hypothesis that symbolic drum patterns can be represented in a reduced form as a simple oscillation between two states, a Low state (commonly associated with kick drum events) and a High state (often associated with either snare drum or high hat). Both an onset time and an accent time is associated to each state. The systematic inference of the reduced form is formalized. This enables the specification of a rhythmic structural similarity measure on drum patterns, where reduced patterns are compared through alignment. The two-state representation allows a low computational cost alignment, once the complex topological formalization is fully taken into account. A comparison with the Hamming distance, as well as similarity ratings collected from listeners on a drum loop dataset, indicates that the bistate reduction enables to convey subtle aspects that goes beyond surface-level comparison of rhythmic textures.

1. INTRODUCTION

One of the most fundamental areas of both Music Information Retrieval (MIR) and music cognition research concerns the modelling of musical similarity, due to the essential importance of similarity in music perception and the practical utility of similarity models in music retrieval applications such as recommendation systems. Modelling similarity for drum patterns is a particular challenging variant of this research, though one with potential application in a wide range of intelligent music production tools such as drum pattern recommendation systems or automatic drum pattern generation systems.

The challenges of drum pattern similarity modelling lies in the complex nature of polyphonic rhythm perception. The integration of coincident rhythms into a resultant 'multirhythm' has been argued to be an essential feature of polyphonic rhythm perception, especially in the context of drumming music [1]. Rhythmic interactions may also be a significant aspect of polyphonic rhythm perception, with complex rhythmic structures said to emerge from the interaction between rhythmic levels [2]. In the context of drum

patterns specifically, rhythmic interaction and instrumental configuration has been shown to have a significant affect on the perception of syncopation, a fundamental rhythmic property [3]. However, in much existing work on drum pattern similarity modelling, similarity is modelled as a linear combination of monophonic rhythm similarity models calculated on each instrument part, without consideration to rhythmic interactions or integration.

Our aim is to design new models of similarity for drum patterns that take into account both rhythmic interaction and rhythmic integration. The bistate reduction algorithm proposed in this paper attempts to extract the core structure of drum pattern as an interaction between two states, 'Low' and 'High', and to use this reduced representation to compare drum patterns through alignment.

The drum patterns used in this paper are part of a varied set of 160 stylistically accurate symbolic patterns recorded by human drummers on an electronic drum kit, taken from the Groove library of BFD3 [4], a commercially available virtual drum kit plugin. Originally collected by [5], the pattern dataset is drawn from a wide range of genres and subgenres, with 8 genre groups used: Hiphop/Dance, Funk, Blues/Country, Pop, Reggae/Latin, Rock, Metal and Jazz. In addition to multiple genres, they are of varied complexity and function, with some containing fills.
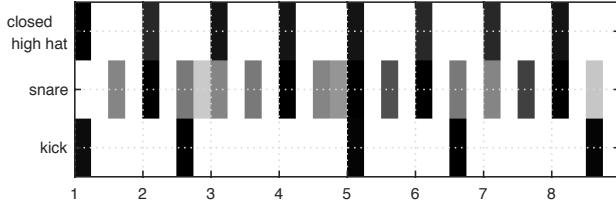
In Section 2, we discuss in greater detail approaches to modelling monophonic rhythm similarity and assess the state-of-the-art in drum pattern similarity modelling and its limitations. In Section 3 and Section 4, we begin discussion of the bistate reduction algorithm and bistate sequence alignment technique respectively as a novel means of representing complex drum patterns and estimating the perceived similarity between them. In Section 5, the sequence alignment algorithm is evaluated in its ability to predict human similarity ratings for pairs on drum patterns in our dataset.

2. RELATED WORK
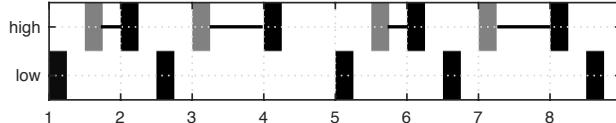
Models of rhythmic similarity are significant in both music perception and music informatics research due to rhythm's fundamental importance in music. For drum patterns in particular, models of rhythmic similarity have various practical applications such as automatic generation of variations on drum patterns [6,7], or in visually mapping drum patterns by similarity to facilitate exploration of drum pattern libraries [8,9].

Figure 1. Drum pattern *Early RnB 33*. Velocity is represented in grey level, from 0 in white to maximum value in black.
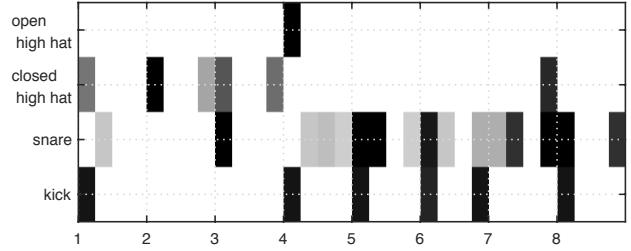
Figure 2. Reduction of *Early RnB 33*. State accents are shown in black and state onsets (if different from accents) in grey. 'Low' corresponds to ↓ and 'high' to ↑.

Figure 3. Drum pattern *Reggae Grooves Fill 3*.

Figure 4. Reduction of *Reggae Grooves Fill 3*.
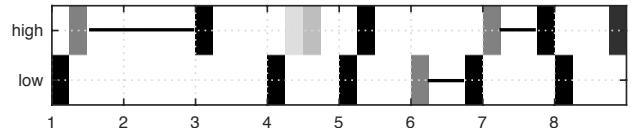
Many models of rhythmic similarity for drum patterns are based on monophonic rhythm similarity models. These methods are based on multiple possible representations of rhythms. The most common method is as a vector consisting of either an onset or silence at each metrical position of the rhythm. The Hamming distance, a simple and popular model, works by counting the number of metrical positions in two rhythm vectors that differ in value (one onset, the other rest) [10]. The swap or directed-swap distance can also be calculated between rhythm vectors by counting the number of swap operations (exchanges between adjacent metrical positions) that need to be performed to turn one rhythm into the other [11]. Other possible rhythm similarity measures use inter-onset intervals (IOIs), vectors of the distance between each successive onset as a multiple of the smallest possible metrical position. An example of this is the chronotonic distance [10]. A thorough overview of numerous approaches to rhythm similarity modelling can be found in [10].

Typically, drum pattern similarity is modelled by stacking these monophonic rhythm similarity models calculated on each part separately. This has been found to be successful in a few cases; [7] found that the Hamming distance and directed-swap distance correlated strongly with human ratings of similarity for drum patterns, with the Hamming distance outperforming the directed-swap. In [8], the authors found a 2D projection of the Hamming distance could partially cluster drum patterns in a way that matched their genre tags for a small dataset of patterns. In a pilot study, a model similar to the Hamming distance was also found to correlate to listeners' similarity ratings for drum patterns [12]. This model counts whether the number of metrical positions where rhythms differ, but adds a weighting metric based on metrical awareness.

The limitation of these models is that by stacking monophonic rhythm similarity measures calculated on each part separately, they fail to account for effects of rhythmic interaction between instrument parts, or the perception of re-

sultant 'multirhythms' due to the perceptual integration of simultaneous rhythmic streams. Additionally, while fixed weighting schemes have been used to weight more perceptually important kit instruments [6], these may not be flexible enough to account for the possibly changing contextual importance of different instruments in a drum pattern.

3. BISTATE REDUCTION

Basic drum patterns are usually defined by the alternation of, typically, bass (or "kick") drum and snare drum strokes, with a further subdivision on the ride cymbal or hi-hat [1]. The ordering in this definition implicitly indicates that the bass and snare drums alternation is of higher level of salience, while the cymbal or hi-hat subdivision is of lesser importance.

3.1 Dominant Drum Selection

The first hypothesis guiding the approach presented in this paper is that drum patterns can be reduced by only focusing on these two most dominant classes of drum strokes. In the definition of basic drum patterns above, the two dominant drums are said to be bass and snare drums. But sometimes the snare drums can be replaced by, for instance, closed hi-hat.

For a given drum pattern, we propose a simple method for selecting the two dominant drums. It consists in ordering the drums in a decreasing hierarchical order, and selecting the two first drums, in this ordered list, that are active in the given drum pattern. The first selected drum will be called the *Low* drum, as it often relates to the bass drum and to the use of lower frequencies, while the second selected drum will be called the *High* drum.

For the drum patterns discussed throughout the paper, the reduction was performed by using the following hierarchical ordering: 1. kick drum, 2. snare drum, 3. closed hi-hat. It was not necessary to consider other drums, since at least two of these three drums were active in each drum

---

[1] Cf. for instance `https://en.wikipedia.org/wiki/Drum_beat`

pattern. In other words, a drum pattern featuring kick drum and closed hi-hat, but no snare drum, will use kick drum as Low and closed hi-hat as High. A drum pattern featuring snare drum and closed hi-hat but no kick drum will use snare drum as Low and closed hi-hat as High.

For the example shown in Figures 1 and 3, the closed hi-hat sequence is considered as of less importance and therefore ignored.

3.2 Two-State Alternation

The second hypothesis developed in the proposed approach is that drum patterns can be reduced as an *alternation* between the Low and High drums. This can be formalised as an alternation between two states, respectively designated ↓ and ↑. Figures 2 and 4 shows the reduction corresponding to the sequences in Figures 1 and 3.

3.2.1 Handling of Simultaneous Strokes

At first sight, to detect the ↓ and ↑ *states*, we would simply need to look for the location of alternation between the Low and High drums. The difficulty comes from the fact that the two drums are often played together. In such occasions, inference of ↓ or ↑ state relies on the context defined by the recent past.

For instance in Figure 3, kick and snare drums are played together at the beginning of beats 5, 6 and 8. Because the kick drum plays a clear ↓ at the beginning of beat 4 (i.e., 4.1), which continues with snare drum playing gently (from 4.2 to 4.4). We haven't heard any clear ↑ state. Therefore, when snare drum and kick drums are superposed on the next beat (5.1), this is perceived as a ↑. But because the snare drum is played loudly on the next tick (5.2), the next beat (6.1) is rather perceived as a ↓. Same for 8.1.
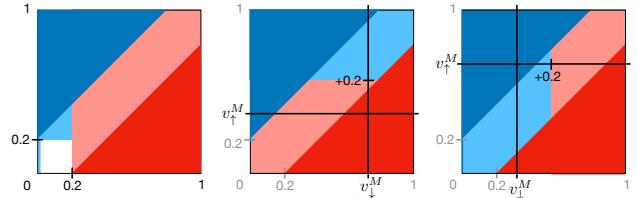
We propose to explain this phenomenon by hypothesising that the categorisation ↓ vs. ↑ is based on comparing each new drum stroke (or superposed strokes) to mental models of ↓ and ↑ states. And in our modelling, this can be simplified by comparing to only the mental model related to one of the states ↓ and ↑, namely the state considered as currently active.

More precisely, the mental model is represented by a state $s \in \{\downarrow, \uparrow, 0\}$ (where state 0 is used solely when starting the analysis, before the first actual state ↓ or ↑ has been detected) altogether with a series of two velocities $(v_\downarrow^M, v_\uparrow^M) \in [0,1]^2$, called *profile* of the mental model, and first initiated to $(0, 0)$. The profile is updated with the current configuration

$$(v_\downarrow^M, v_\uparrow^M) = (v_\downarrow, v_\uparrow) \qquad (1)$$

every time there is a state transition.

At any given successive point in the drum pattern, featuring at that instant a Low drum stroke of velocity $v_\downarrow \in [0,1]$ (with 0 indicating no stroke) and a High drum stroke of velocity $v_\uparrow \in [0,1]$, we consider the following alternatives, which are tested in the indicated order until a suitable condition is found:

Figure 5. State transition diagrams, where the current state is respectively 0 (left diagram), ↓ (middle) and ↑ (right). For given values for $v_\downarrow$ (X axis) and $v_\uparrow$ (Y axis) is indicated the next state: ↓ (red) ↑ (blue), 0 (white). See the text for more explanation.

1. If $v_\downarrow > v_\uparrow + .2$, the next state is ↓. This corresponds to the region in dark red in each diagram in Figure 5. The profile is updated. In other words, when the Low stroke is significantly stronger than the High stroke, the new state is clearly ↓.

2. Similarly, if $v_\uparrow > v_\downarrow + .2$, the next state is ↑ (dark blue region in Figure 5) and its profile updated.

3. If the current state is 0 (i.e, undefined), the next state is ↓ if $v_\downarrow > .2$ (light red region in the left diagram of Figure 5). In other words, at the very beginning of a drum pattern, an ambiguous mixture of Low and High drums is associated with ↓, because it is considered as the default state (while ↑ is a departure from the default state). If on the contrary $v_\downarrow \leq .2$ the next state is ↑ if $v_\uparrow > .2$ or if $v_\downarrow = 0$ (light blue region).

4. If $v_\downarrow = 0$ and if the current state is ↓, the next state is ↑ if $v_\uparrow > \frac{v_\uparrow^M}{2}$. If on the contrary the current state is already ↑, the profile is updated only if $v_\uparrow > v_\uparrow^M$.

5. If the current state is ↓ and $v_\uparrow > v_\uparrow^M + .2$, the next state is ↑ (light blue region in the middle diagram of Figure 5). In other words, a significant increase in velocity of the High strike (even if the velocity remains lower than the Low strike's one) triggers a transition to the ↑ state.

6. Similarly, if the current state is ↑ and $v_\downarrow > v_\downarrow^M + .2$, the next state is ↓ (light red region in the right diagram of Figure 5).

3.2.2 State Onset and Accent Times

The approach presented in the previous section leads to the generation of a sequence of states ↑ and ↓. Successive repetitions of a same state are reduced into one single state. This practically means in particular that successive repetitions of a given drum (while the other drum remains silent) are collapsed into one single state.

However, if the most accented stroke is not the first one, this needs to be specifically indicated in the pattern description, as it plays an important role in the pattern characteristics. For that reason, with each state (excepted the initial 0 state) are associated three attributes:

	↓!	(↓)	↑!	(↑)
↓!	$S\downarrow$	$S\downarrow$	O	$A\downarrow$
(↓)	$S\downarrow$		$B\uparrow$	
↑!	O	$A\uparrow$	$S\uparrow$	$S\uparrow$
(↑)	$B\downarrow$		$S\uparrow$	

Table 1. Alignment configurations, triggered by state transitions, in the first sequence (left column) or/and in the second sequence (top row). In the absence of transition in any of the two sequences, no alignment configuration needs to be defined.

- whether it is $\downarrow$ or $\uparrow$

- the *onset time*, i.e., the temporal position of the first stroke in that successive repetition of strokes

- the *accent time*, i.e., the temporal position of the most accented stroke.

4. BISTATE SEQUENCE ALIGNMENT

The third hypothesis developed in this paper is that the perceived distance between two drum patterns can be approximated by aligning the two corresponding bistate sequences and estimating the alignment mismatches.

4.1 Alignment principle

To allow the formalisation of this alignment, the set of two states $\{\downarrow, \uparrow\}$ is further decomposed into four states $\{\downarrow!, \uparrow!, (\downarrow), (\uparrow)\}$ in order to take into account all the temporal relationships:
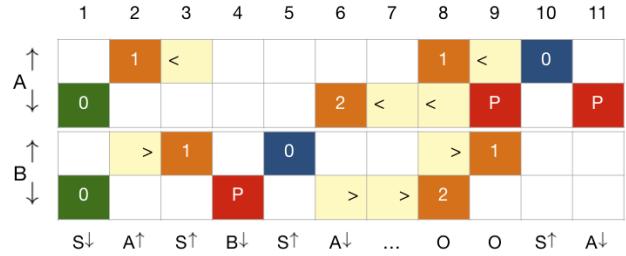
- $\downarrow!$ and $\uparrow!$ corresponds to a new transition to, respectively, $\downarrow$ and $\uparrow$.

- $(\downarrow)$ and $(\uparrow)$ indicates a continuation of the corresponding states $\downarrow$ and $\uparrow$.

This leads to a 4×4 matrix of possible alignment configurations, shown in Table 1. We can distinguish three types of alignment configurations:

- The two sequences are in same state ("S"), both low ("$S\downarrow$") or both high ("$S\uparrow$").

- One sequence introduces the opposite state ahead of the other sequence: "A" if it the first sequence ahead, "B" if the second sequence.

- The two sequences are in opposite states ("O").

The possible transitions between configurations are the following, as described in Table 1:

- Starting from the same state (for instance $S\downarrow$), then both sequences transition at the same time (for instance $S\uparrow$).

- Starting from the same state (for instance $S\downarrow$), then one sequence transitions first (for instance $A\uparrow$).

Figure 6. Alignment between two reduced drum patterns A and B. Penalties are given for each individual state, indicated with a number or 'P'. See the text for more explanation.
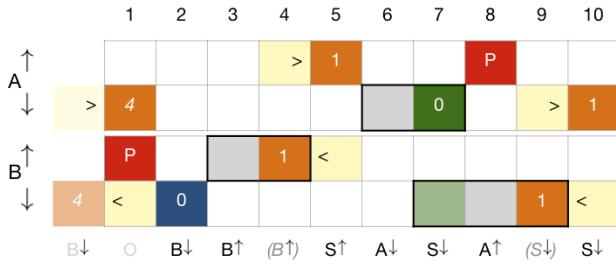
- From configuration A or B, if only one sequence transitions, this leads to S, while if both sequences transition at the same time, this leads to opposite states O.

- From O, another double-transition leads to another O, while a single transition leads to either A or B.

4.2 Numerical distance estimation

The proposed distance model is based on an evaluation of how well each of the two drum patterns aligns with the other. For each successive state of each drum pattern, a penalty is given if that state is not found in the other sequence around the same temporal position. Offset in the temporal positions of the state on the two sequences also contributes to the penalty.

The basic mechanisms are illustrated below using the toy example shown in Figure 6. In this first example, we assume that both sequences start at the same state ($\downarrow$).

- The first state starts at the same time ($t = 1$) on both sequences, leading to a $S\downarrow$ configuration with 0 penalty (shown in green in the Figure).

- The next event appears first in sequence A ($A\uparrow$ at $t = 2$). The same transition appears next in sequence B ($S\uparrow$ at $t = 3$). It is thus a quasi-synchronised transition, with 1 tick delay between the two sequences. A penalty of 1 is therefore given to the new state in each sequence. (The onset of each state is shown in orange while the position of the opposite state is shown in yellow.)

- A transition $B\downarrow$ at $t = 4$ is followed by another transition back to $S\uparrow$. This state $\downarrow$ is deleted by giving the maximum penalty P (in red) and the new state $\uparrow$ at $t = 5$ (in blue) is ignored.

- A transition $A\downarrow$ at $t = 6$ is followed by a double transition O at $t = 8$, which is considered as both a 2-tick-delayed transition of B to state $\downarrow$ and as the start of a new transition $A\uparrow$. Same for the subsequent double transition O at $t = 9$. The new transition $A\downarrow$ being followed by a transition $S\uparrow$, that state $\downarrow$ is removed.

321

Figure 7. Alignment between two reduced drum patterns A and B. Boxed series of cells show states containing repeated strokes, where the accent is the rightmost cell.

- Sequence A continuing after the end of sequence B, any transition to the opposite state $A \downarrow$ (here only at $t = 11$) is deleted.

In Figure 7, the sequences starting in opposite states, an additional $\downarrow$ state is added before the start of sequence B, to which is given a penalty of 4. In this example, since the next event is a $\downarrow$ in sequence B, the previous $\uparrow$ is removed.

The rest of Figure 7 illustrates the handling of states containing repeated strokes with the accentuated stroke appearing after the initial onset. Concerning the transition to $\uparrow$ first by B at $t = 3$ and then by A at $t = 5$, the penalty is only 1 because it is computed between the accent in sequence B (at $t = 4$) and the onset of sequence A. For the same reason, the next transition to $\downarrow$ at $t = 7$ is considered without penalty. While B remains in $\downarrow$, with its attack at $t = 9$, A transitions to $\uparrow$ with an attack at $t = 8$, which is therefore deleted.

Each penalty is weighted with respect to the corresponding accent velocity. For instance in the case of a drum stroke of very low intensity (such as a ghost note), the contribution of this penalty to the overall distance will be low. Finally the distance between the two sequences is defined as the maximum between the two weighted sums.

5. EVALUATION

To investigate the potential use of the bistate sequence alignment algorithm for drum pattern similarity modelling, we examined its calculated distances between 80 pairs of drum patterns in our dataset in relation to human-provided perceptual similarity ratings.

5.1 Methodology

We collected similarity ratings for the drum patterns via an online listening test. The symbolic patterns were first synthesized into audio via a generic sampled drum kit. 21 listeners rated similarity for the 80 pairs of patterns using a continuous scale. Median internal consistency for all participants calculated using the ICC (2,1) for the repeated pairs was 0.85, equaling good agreement. The inter-rater agreement for the 21 listeners calculated using the same ICC (2,1) was 0.73, equally moderate-to-good agreement. The spreads of ratings for each comparison were normally

Similarity Model	r	p
Hamming Distance	0.604	2.97e-9
Hamming Distance (2 channels)	0.539	2.58e-7
Bistate Sequence Alignment	0.556	8.49e-8
min(Hamming (2 chan), Alignment)	0.606	2.65e-9
min(Hamming, Alignment)	0.692	1.21e-12

Table 2. Pearson correlation coefficient r and *p-value* between mean similarity ratings and distance models.

distributed. More information on the collection of this dataset may be found in [5].

To test the overall extent to which the bistate sequence alignment algorithm corresponds to perceived similarity, we calculated the Pearson correlation between the distance calculated by the bistate sequence alignment algorithm and the mean of human-supplied similarity ratings. We used the established Hamming distance as a reference for evaluation, since, as indicated in Section 2, it gave the best results in previous works. The first step of our approach, presented in section 3.1, can be studied separately by also computing the Hamming distance on the two main channels of the drum patterns.
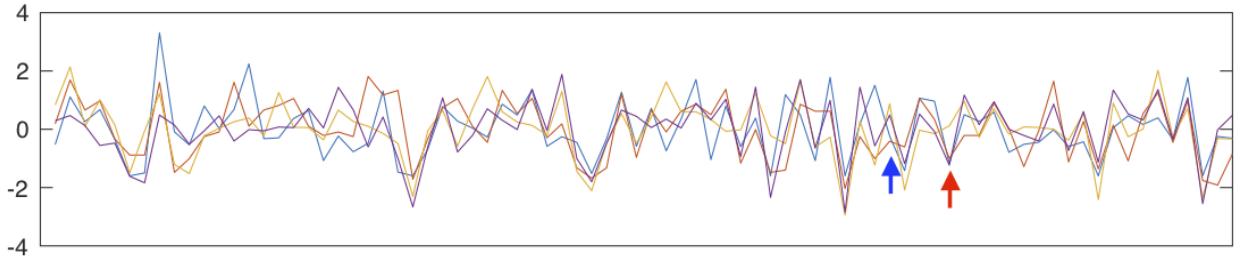
5.2 Results and Discussion

The results can be seen in Table 2. Various values for the parameter P were tried and the best results were obtained with $P = 8$. The Hamming distance's correlation on the complete drum patterns is slightly stronger than the bistate sequence alignment, which is itself very slightly better than the Hamming distance computed on the two main drum channels.
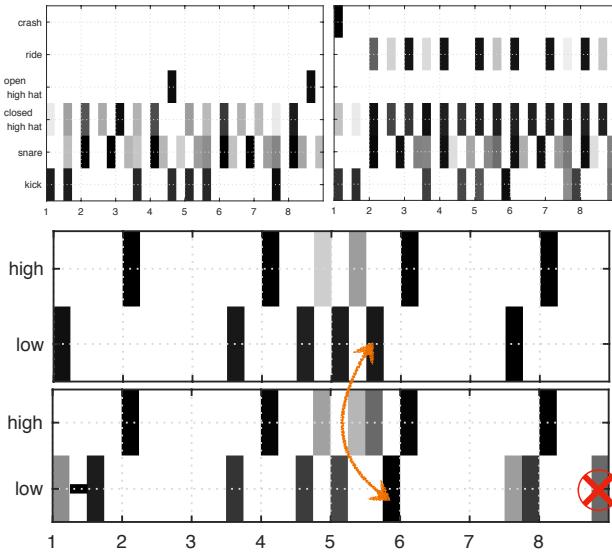
When combining the Hamming distance (computed on the whole drum loops) with the alignment by taking the minimum of them both, the correlation is better than the Hamming distance alone, with a correlation of 0.692 vs 0.604. This increase in correlation was found to be statistically significant (t=1.748, p=0.04). This could indicate that both these two algorithms capture fundamentally different aspects of similarity, with the Hamming distance capturing low-level similarities between rhythms, and the bistate sequence alignment capturing qualities relating to rhythmic interaction and structure.

The difference between the distances can be seen in Figure 8 where the similarity ratings of all 80 pairs are plotted alongside distances calculated by both the Hamming distance and the bistate sequence alignment algorithm.

The differences between the bistate sequence alignment algorithm and Hamming distance can be further demonstrated through viewing of some particular examples. Figure 9 shows a pair of *Soul Grooves* patterns that share very similar kick and snare drum patterns while differing on other drum tracks. Since the similarity was judged by the listeners as high, this demonstrates the interest, in particular cases such as this one, in focusing the comparison of drum patterns on the main kick and snare drums. And indeed, computing the Hamming distance on those two main channels leads to a similarly low distance value (cf. red arrow in Figure 8).

Figure 8. Comparison of the Z-score of the new proposed distance (in blue), the Hamming distance (in yellow), the Hamming distance on the two main channels (in purple) and the listeners' ratings (in red), for each of the 80 pairs of drum patterns. The red and blue arrows indicate the examples shown in, respectively, Figures 9 and 10.
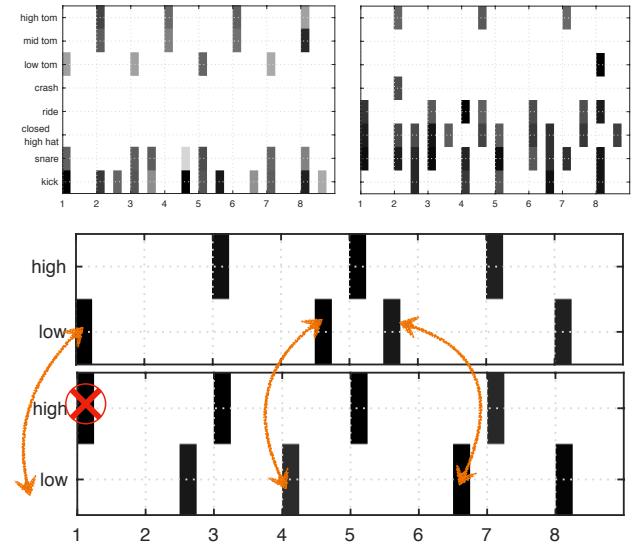
Figure 9. Alignment between drum patterns *Soul Grooves 2* (top left and center) and *Soul Grooves 31* (top right and bottom). The only penalties contributing to the distance is a 1-tick offset (in orange) and a deleted state (in red).

Figure 10. Alignment between drum patterns *Batucara 3* and *Cascara 5* using the same conventions as in Figure 9.

Figure 10 illustrates the interest of the method beyond the simple selection of two main channels. In this example, the two sequences differ significantly at the surface level, even if we restrict the scope on the kick and snare drums. On the other hand, the corresponding reduced bistate sequences are similar and show a clear alignment, leading to a relatively low distance on par with the listeners' jugement.

6. CONCLUSIONS

This article proposed an attempt to reduce drum patterns into an underlying core structure with the aim to compare drum patterns by aligning their reduced representation one with each other. Clearly the problem of rhythmic reduction is of high complexity, and is addressed here solely as an intermediary step for the establishment of a distance measure between drum patterns. These mechanisms of selection of dominant drums and of inference of Low and High states could be studied further in order to provide more advanced description of drum patterns.

As it could have been expected, the comparison between the proposed distance and a simple surface-level Hamming distance concerning their abilities to mimic listeners similarity judgments shows that for the most part, listeners rely on surface characteristics of the overall rhythmic texture to compare drum patterns. However, in particular cases the underlying core structure can be of importance as well, and this is where the proposed distance can be of interest. Combining surface-level and deeper-level representations seems to improve the overall similarity modelling. We may hypothesise that further progress in this endeavour could be made possible through the integration of more refined cognitive models.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] W. Anku, "Principles of rhythm integration in african drumming," *Black Music Research Journal*, pp. 211–238, 1997.

[2] S. Handel and G. R. Lawson, "The contextual nature of rhythmic interpretation," *Perception & Psychophysics*, vol. 34, no. 2, pp. 103–120, 1983.

[3] M. A. Witek, E. F. Clarke, M. L. Kringelbach, and P. Vuust, "Effects of polyphonic context, instrumentation, and metrical location on syncopation in music," *Music Perception: An Interdisciplinary Journal*, vol. 32, no. 2, pp. 201–217, 2014.

[4] FXpansion, "BFD3 https://www.fxpansion.com/products/bfd3/," Accessed May 2020.

[5] F. Bruford, M. Barthet, S. McDonald, and M. Sandler, "Modelling musical similarity for drum patterns: A perceptual evaluation," in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, 2019, pp. 131–138.

[6] R. Vogl, M. Leimeister, C. Ó. Nuanáin, M. Hlatky, S. Jordà Puig, and P. Knees, "An intelligent interface for drum pattern variation and comparative evaluation of algorithms," *Journal of the audio engineering Society. 2016; 65 (7/8): 503-13.*, 2016.

[7] C. O. Nuanáin, P. Herrera, and S. Jorda, "Target-based rhythmic pattern generation and variation with genetic algorithms," in *Sound and Music Computing Conference*, 2015.

[8] F. Bruford, M. Barthet, S. McDonald, and M. Sandler, "Groove Explorer: An Intelligent Visual Interface for Drum Loop Library Navigation," in *IUI Workshops*, 2019.

[9] D. Gómez-Marín, S. Jorda, and P. Hererra, "Drum rhythm spaces: from global models to style-specific maps," in *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2017.

[10] G. T. Toussaint, "A comparison of rhythmic similarity measures." in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2004.

[11] C. Guastavino, F. Gómez, G. Toussaint, F. Marandola, and E. Gómez, "Measuring similarity between flamenco rhythmic patterns," *Journal of New Music Research*, vol. 38, no. 2, pp. 129–138, 2009.

[12] D. Gómez-Marín, S. Jordà, and P. Herrera, "Pad and Sad: Two awareness-Weighted rhythmic similarity distances," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

Paper Session 3

MULTI-INSTRUMENT MUSIC TRANSCRIPTION BASED ON DEEP SPHERICAL CLUSTERING OF SPECTROGRAMS AND PITCHGRAMS

Keitaro Tanaka[1,*]   **Takayuki Nakatsuka**[1]   **Ryo Nishikimi**[2]
Kazuyoshi Yoshii[2]   **Shigeo Morishima**[3]

[1] Waseda University, Japan [2] Kyoto University, Japan

[3] Waseda Research Institute for Science and Engineering, Japan
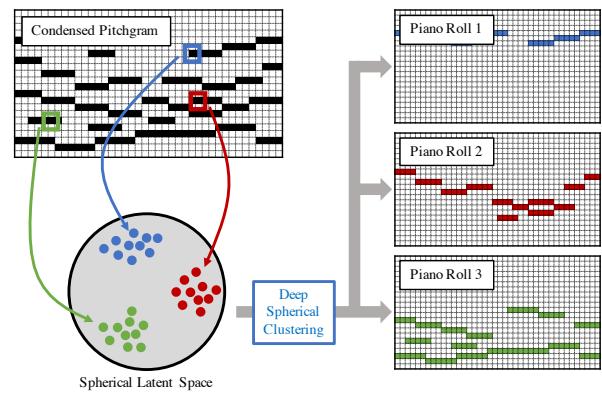
[*]phys.keitaro1227@ruri.waseda.jp

ABSTRACT

This paper describes a clustering-based music transcription method that estimates the piano rolls of arbitrary musical instrument parts from multi-instrument polyphonic music signals. If target musical pieces are always played by particular kinds of musical instruments, a way to obtain piano rolls is to compute the pitchgram (pitch saliency spectrogram) of each musical instrument by using a deep neural network (DNN). However, this approach has a critical limitation that it has no way to deal with musical pieces including undefined musical instruments. To overcome this limitation, we estimate a condensed pitchgram with an existing instrument-independent neural multi-pitch estimator and then separate the pitchgram into a specified number of musical instrument parts with a deep spherical clustering technique. To improve the performance of transcription, we propose a joint spectrogram and pitchgram clustering method based on the timbral and pitch characteristics of musical instruments. The experimental results show that the proposed method can transcribe musical pieces including unknown musical instruments as well as those containing only predefined instruments, at the state-of-the-art transcription accuracy.

1. INTRODUCTION

The problem of estimating the fundamental frequencies of multiple periodic signals, which is called multi-pitch estimation (MPE) [1], is an important task of music information retrieval (MIR) since it plays a basic role in automatic music transcription (AMT), which is a task of converting music signals into a symbolic form of music notation [2]. The conventional approaches to MPE primarily focused on transcribing single-instrument music signals. The accuracy of this single-instrument MPE (SI-MPE) has been greatly improved by deep learning. Recently, some studies have extended SI-MPE and have tackled the problem

Figure 1. Each bin of a condensed pitchgram is embedded on a spherical latent space taking into account the timbral characteristics. Piano rolls of each instrument part is obtained by deep spherical clustering on the space.

of multi-instrument MPE (MI-MPE) for further generalization. An MI-MPE is a task which estimates the pitchgrams (pitch saliency spectrograms) of every musical instrument from a music signal consisting of multiple instruments. The difficulty of MI-MPE in addition to SI-MPE is the necessity of estimating the corresponding instrument part which the pitchgram belongs to. To alleviate this difficulty, previous studies [3, 4] for MI-MPE limited their target musical instruments to a small number of predefined instruments. One of the solutions to this problem is applying a classification technique to MI-MPE and separate the music signal into each pitchgram.

These classification-based methods have been successful [3, 4] in the framework of supervised learning, especially for classical music where the constituent instruments are mostly fixed. However, in modern music (*e.g.*Pops and EDMs) where a larger number of instruments often appear, it would be ideal to have no limit on target instruments in order to achieve better AMT.

In the field of speech separation, several studies [5–7] have attempted a similar task of separating arbitrary speakers. When handling arbitrariness of the target sources in DNNs, technical problems related to permutations arise. Specifically, DNNs deterministically map inputs to a defined set of sources in each dimension, and thus does not allow permutation between different targets. To solve this permutation problem, a method called deep clustering

has been proposed that treats speech separation for arbitrary speakers as a clustering problem, rather than a classification problem [5]. This approach avoids the above-mentioned problem and achieves optimal clustering at the same time by constructing an affinity matrix.

In this paper, we propose a new method to estimate the piano rolls of arbitrary musical instrument parts from multi-instrument polyphonic music signals based on deep clustering (Figure 1). We estimate a condensed pitchgram which shows all played pitches, with an existing instrument-independent neural multi-pitch estimator, and then separate the pitchgram into a specified number of musical instrument parts with a deep spherical clustering technique. Also, by considering the spectrogram in addition to the pitchgram in clustering phase, the optimal part estimation can be performed based on both the timbral and pitch characteristics of the instruments contained in the music signal. Furthermore, since there is a complementary relationship between MPE and sound source separation [8–10], we propose a joint spectrogram and pitchgram clustering method which can improve the transcription accuracy.

To verify that our method can transcribe arbitrary musical instruments, we conducted experiments of MI-MPE for various musical instruments. Experimental results show that the method can successfully handle a wide variety of instruments including those unseen during training. Although our method does not set any limitation on applicable instruments, the results suggest that it performs comparably to the state-of-the-art classification method [3].

Our main contribution of this study is the proposal of a new clustering-based method to transcribe arbitrary musical instrument parts from a music signal. To our knowledge, this is the first attempt of MI-MPE at frame-level without any restriction on used instruments. Furthermore, we show that the deep clustering method can be applied to tasks other than speech separation, and describe its potential in several sound related tasks.

2. RELATED WORK

In this section, we limit our scope to studies related to MI-MPE and methods dealing with arbitrariness of DNNs. Brief explanations of each study will be provided in the following subsections.

2.1 Multi-instrument Multi-pitch Estimation

Although AMT has been well studied, it still remains a challenging task [11]. Among the various tasks associated to AMT, MI-MPE is particularly difficult because it requires to simultaneously perform SI-MPE and instrument part estimation for each estimated note [2].

MI-MPE has commonly been tackled as a problem of stream-level transcription: grouping estimated notes and making continuous pitch contours for each part. Duan *et al.* [12] proposed a constrained clustering approach against the result of MPE. The clustering is performed under the constraint of consistency in each part of uniform
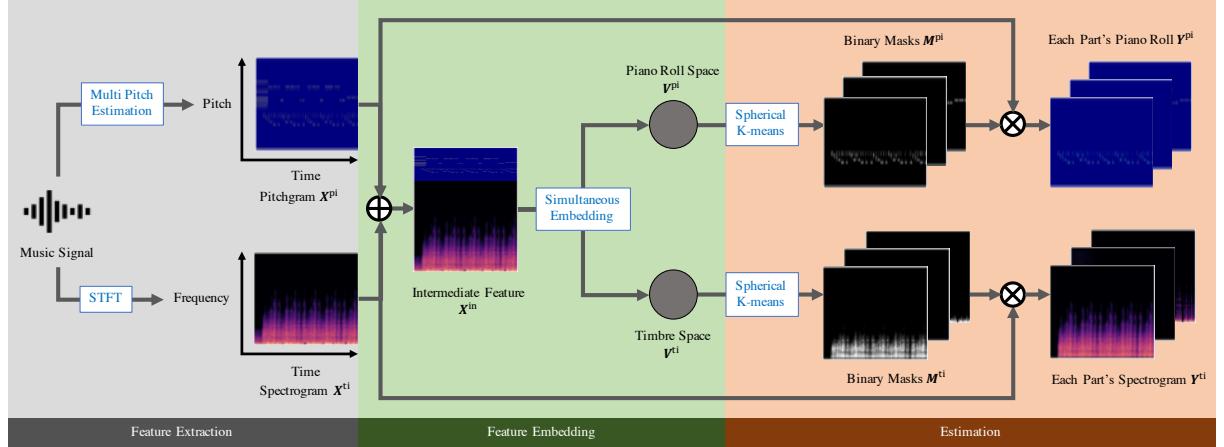
discrete cepstrum. Their method can be used in complement with various MPE algorithms [13–15], and does not require any source model trained with isolated recordings of the underlying instruments. Following this study, Arora *et al.* [16] took a similar approach. They used probabilistic latent component analysis for MPE and source-specific feature extraction, and hidden Markov random fields for clustering into each instrument part. These two methods can deal with a variety of instruments, but due to their algorithms, each instrument must be a monophonic instrument which plays only one note at a time. Unlike these methods, Benetos *et al.* [17] focused on the differences in the sounds played by each instrument. They used spectral templates that correspond to sound states, supported by the shift-invariant probabilistic latent component analysis method. To conduct MPE for each instrument, they controlled the order of these templates by using hidden Markov model-based temporal constraints.

In recent years, some studies have tackled MI-MPE as a frame-level simultaneous MPE and musical instrument recognition problem. Wu *et al.* [3] proposed a DNN model based on the DeepLabV3+ [18] and U-Net structure [19]. They considered MI-MPE as a semantic segmentation problem on the time-frequency bins generated from music signals, where each object class represents a certain musical instrument. Most recently, Cerberus Network was proposed by Manilow *et al.* [4]. This model was built upon the preceding Chimera Network [20] which was developed for speech separation, adding a module that produces separated piano rolls for each instrument. The drawback of these methods is that only musical instruments included in the predefined set can be transcribed. In order to apply classification-based methods to source separation, output classes and object instances must be represented explicitly. Therefore, it is difficult to use these methods in the general case.

2.2 Arbitrariness with DNNs

In order to allow extraction of a piano roll of arbitrary instruments from an audio signal, the prediction itself must take place in a process where the instruments are unidentified, *i.e.*, individual piano rolls are referred to as instrument one, two, three instead of their specific identity such as piano, guitar, violin, etc. However, when doing so by using a DNN based approach, a problem of permutation arises as previously mentioned. Specifically, when the piano rolls of individual instruments are extracted but the instrument type is unknown, the loss between predicted piano roll and ground truth cannot be calculated straightforwardly since the correspondence of instrument type between these two remains unknown.

A similar problem has been addressed in the studies for speaker-independent speech separation [5–7], whose goal is to separate a piece of audio consisting of multiple people speaking simultaneously into individual speakers audio. Unless given an image or video of the target speaker, correspondence between separated audio and ground truth audio cannot be established, and hence the task poses the

Figure 2. Overview of the proposed method. We extract two audio features $\boldsymbol{X}^{\mathrm{pi}}$ and $\boldsymbol{X}^{\mathrm{ti}}$ from an input music signal and concatenate them as an intermediate feature $\boldsymbol{X}^{\mathrm{in}}$. The feature $\boldsymbol{X}^{\mathrm{in}}$ is mapped into two latent space, piano roll space $\boldsymbol{V}^{\mathrm{pi}}$ and timbre space $\boldsymbol{V}^{\mathrm{ti}}$. We generate binary masks $\boldsymbol{M}^{\mathrm{pi}}$ and $\boldsymbol{M}^{\mathrm{ti}}$ from each space by clustering. These masks are applied to corresponding features, and we obtain piano rolls for each musical instrument part $\boldsymbol{Y}^{\mathrm{pi}}$ and separated spectrograms $\boldsymbol{Y}^{\mathrm{ti}}$.

same problem of permutation. To address this problem, methods such as permutation invariant training (PIT) [6], and deep clustering [5] have been proposed recently.

PIT tackled the permutation problem by calculating a loss function for all possible pairs of predicted values and ground truth, while optimization is only conducted for the pair with minimum loss. Although its implementation is simple and it can be combined with other learning techniques, its computational complexity remains considerably high. In details, when N sources are included in the target mixture, $N!$ possible permutations must be calculated in their algorithm.

On the other hand, deep clustering avoids the permutation problem by optimizing an embedded representation of the desired output, so that the class separation can be conducted via clustering in the embedded space at inference time. Given a $X \times D$ matrix $\boldsymbol{A}$ as the embedded representation, where X is the time-frequency index and D is the embedding dimension, the affinity matrix $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ is calculated. In the same manner, the affinity matrix $\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}$ is obtained for the ground truth data $\boldsymbol{B}$ which is a $X \times N$ matrix, where N represents the number of speakers. The optimization is conducted to minimize the distance between the two affinity matrices $||\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} - \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}||_F^2$. Here, deep clustering succeeds in circumventing the permutation problem as $(\boldsymbol{A}\boldsymbol{P})(\boldsymbol{A}\boldsymbol{P})^{\mathrm{T}} = \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ for any $D \times D$ permutation matrix $\boldsymbol{P}$. Furthermore, since optimization is conducted on the transformed $X \times X$ matrix, the target data may include any number of sources. For these advantages, we adopt the deep clustering method in our framework as described in Section 3.

3. PROPOSED METHOD

This section describes our proposed clustering-based method for the transcription of arbitrary musical instrument parts (Figure 2). Our framework consists of three parts: a feature extraction part, a feature embedding part to obtain piano roll space and timbre space, and an esti-

mation part based on deep spherical clustering. We first pretrain the feature extraction part and the feature embedding part individually for the stabilization of early learning stages, then optimize both parts in conjunction through the entire learning.

3.1 Problem Configuration

Let $\boldsymbol{S} = \{\boldsymbol{s}_k \in \mathbb{R}^l\}_{k=1}^K$ be a set of mixture audio signals, where $l = 44.1$ [kHz] $\times$ 10 [sec] is a length of the signal, and K is the number of mixture audio signals. We assume that each $\boldsymbol{s}_n$ consists of three instrument parts. Let $\boldsymbol{Y}^{\mathrm{pi}} = \{\boldsymbol{y}_n^{\mathrm{pi}} \in [0,1]^{T \times C}\}_{n=1}^{N+1}$ be a set of pitchgrams of piano rolls, where T is the number of time frames, C is the number of constant-Q transform (CQT) frequency bins and N is the number of musical instrument parts. Our goal is to train a DNN f that maps $\boldsymbol{S}$ to $\boldsymbol{Y}^{\mathrm{pi}}$. Here, we incorporated two key ideas into f for the performance improvement and the stable training. Let $\boldsymbol{Y}^{\mathrm{ti}} = \{\boldsymbol{y}_n^{\mathrm{ti}} \in \mathbb{R}^{T \times F}\}_{n=1}^N$ be a set of corresponding spectrograms of the piano rolls, where F is the number of short-time Fourier transform (STFT) frequency bins. We train f that maps $\boldsymbol{S}$ to not only $\boldsymbol{Y}^{\mathrm{pi}}$, but also $\boldsymbol{Y}^{\mathrm{ti}}$ for improving the performance of a transcription. To achieve this with the stable training, we introduce an intermediate supervision, which consists of two semantic features. Let $\boldsymbol{X}^{\mathrm{pi}} \in [0,1]^{T \times C}$ and $\boldsymbol{X}^{\mathrm{ti}} \in \mathbb{R}^{T \times F}$ be a set of pitch characteristics and a set of timbral characteristics, respectively. We divided the f into two networks: feature extraction network g that maps $\boldsymbol{S}$ to $\boldsymbol{X}^{\mathrm{pi}}$, and feature embedding network h that maps the concatenation of $\boldsymbol{X}^{\mathrm{pi}}$ and $\boldsymbol{X}^{\mathrm{ti}}$ to $\boldsymbol{Y}^{\mathrm{pi}}$ and $\boldsymbol{Y}^{\mathrm{ti}}$. Firstly, the network g and h are trained individually for the stable training, and then our full network $f\ (= h \circ g)$ are jointly trained for the overall optimization.

3.2 Feature Extraction

In the feature extraction stage, a pitchgram and spectrogram are obtained from the input music signal, as pitch and timbral characteristics of each instrument are impor-

tant for estimating piano rolls of each musical instrument part. For pitch, we computed a condensed pitchgram of the given music signal, in a form similar to a logarithmic frequency spectrogram using an instrument-independent neural multi-pitch estimator [21]. This network receives harmonic constant-Q transform (HCQT) as its input and outputs a condensed pitchgram denoted by $\boldsymbol{X}^{\mathrm{pi}}$. A value of each pitchgram bin is proportional to its salience.

We computed an STFT spectrogram $\boldsymbol{X}^{\mathrm{ti}}$ of the given music signal. Although there are other possible representations for timbral characteristics, we use STFT following the original deep clustering [5]. To reduce variations in total volume of input signals, the STFT spectrogram is normalized so that each time-frequency bin has a mean of zero and a standard deviation of one. Details are in Section 4.1.

3.3 Feature Embedding

We adopt joint learning of piano roll transcription and sound source separation. They are known to have a complementary relationship and have been reported to improve performance when they are learned simultaneously [8–10]. Following this knowledge, we propose a network based on deep spherical clustering that allows joint learning of transcription and separation. To learn the obtained pitch and timbral feature at the same time, we concatenate them along each frequency axis. This input feature $\boldsymbol{X}^{\mathrm{in}} \in \mathbb{R}^{T \times (C+F)}$ is used as the input to our network. The network maps the input feature $\boldsymbol{X}^{\mathrm{in}}$ to two separate latent spaces: piano roll space $\boldsymbol{V}^{\mathrm{pi}}$ and timbre space $\boldsymbol{V}^{\mathrm{ti}}$. The structure of our network is shown in Figure 3, where D and D' are the embedded dimensions of piano roll and timbre space. It consists of a three layer Bidirectional Long short-term memory (BLSTM), a fully connected (FC) layer for each space with tanh activation, and finally L^2 normalization. L^2 Normalization is conducted so that the piano roll space and timbre space respectively form a D and D' dimensional hypersphere.

The binary masks are made from the two latent spaces and applied to the pitchgram and the spectrogram later. In order to generate masks by clustering, all time-frequency bins have to be located ideally on the spherical latent spaces, i.e., bins of the same source are close and bins of different sources are far apart. This can be achieved by constructing the affinity matrix of each space, $\boldsymbol{V}^{\mathrm{pi,ti}}\boldsymbol{V}^{\mathrm{pi,ti}\mathrm{T}}$. Since $\boldsymbol{V}^{\mathrm{pi,ti}}$ is L^2 normalized, $TC \times TC$ or $TF \times TF$ matrix $\boldsymbol{V}^{\mathrm{pi,ti}}\boldsymbol{V}^{\mathrm{pi,ti}\mathrm{T}}$ show cosine similarity of all time-frequency bins. Let $TC \times (N+1)$ matrix $\hat{\boldsymbol{M}}^{\mathrm{pi}}$ and $TF \times N$ matrix $\hat{\boldsymbol{M}}^{\mathrm{ti}}$ represent correct masks, where N is the number of musical instrument parts. We assume that each time-frequency bin is attributed to only one source. If more than one source share the same bin, the bin is assigned to the dominant source which has the largest volume (MIDI velocity) or the largest power spectrogram. $\hat{\boldsymbol{M}}^{\mathrm{pi,ti}}$ thus take binary value, one for assigned bin and zero for the opposite, and affinity matrix $\hat{\boldsymbol{M}}^{\mathrm{pi,ti}}\hat{\boldsymbol{M}}^{\mathrm{pi,ti}\mathrm{T}}$ also have binary value. We can train this network using $\hat{\boldsymbol{M}}^{\mathrm{pi,ti}}\hat{\boldsymbol{M}}^{\mathrm{pi,ti}\mathrm{T}}$ as target affinity matrix of $\boldsymbol{V}^{\mathrm{pi,ti}}\boldsymbol{V}^{\mathrm{pi,ti}\mathrm{T}}$.

Note that we prepare an extra dimension for $\hat{\boldsymbol{M}}^{\mathrm{pi}}$. Because the condensed pitchgram $\boldsymbol{X}^{\mathrm{pi}}$ is a prediction, $\boldsymbol{X}^{\mathrm{pi}}$ may include misestimations, i.e., false negatives and false positives. Among them, false positives should be treated as exceptions because they have no true instrumental attribution. We therefore prepare an additional dimension for bins which are silent in the ground truth, thus true negatives and false positives are put in this dimension. We also retain $\boldsymbol{X}^{\mathrm{ti}}$ bins whose magnitude is greater than the original maximum magnitude minus 40 dB. This prevents the network from considering about small power bins too much.

3.4 Training Strategy

Training of the multi-pitch estimator is conducted by minimizing the cross entropy loss shown in Eqn (1),

$$\mathcal{L}_{DS} = -\hat{\boldsymbol{X}}^{\mathrm{pi}} \log(\boldsymbol{X}^{\mathrm{pi}}) - (1 - \hat{\boldsymbol{X}}^{\mathrm{pi}}) \log(1 - \boldsymbol{X}^{\mathrm{pi}}) \quad (1)$$

where $\hat{\boldsymbol{X}}^{\mathrm{pi}}$ and $\boldsymbol{X}^{\mathrm{pi}}$ represent the ground truth condensed pitchgram and the estimated condensed pitchgram. Both have values ranging from zero to one. Training of simultaneous embedding part is conducted to minimize Eqn (2).

$$\mathcal{L}_{DC}{}^{\mathrm{pi,ti}} = ||\boldsymbol{V}^{\mathrm{pi,ti}}\boldsymbol{V}^{\mathrm{pi,ti}\mathrm{T}} - \hat{\boldsymbol{M}}^{\mathrm{pi,ti}}\hat{\boldsymbol{M}}^{\mathrm{pi,ti}\mathrm{T}}||_F^2 \quad (2)$$

To reduce computational costs, we used a variation of Eqn (2) in practice.

$$\mathcal{L}_{DC}{}^{\mathrm{pi,ti}} = ||\boldsymbol{V}^{\mathrm{pi,ti}\mathrm{T}}\boldsymbol{V}^{\mathrm{pi,ti}}||_F^2 - 2||\boldsymbol{V}^{\mathrm{pi,ti}\mathrm{T}}\hat{\boldsymbol{M}}^{\mathrm{pi,ti}}||_F^2$$
$$+ ||\hat{\boldsymbol{M}}^{\mathrm{pi,ti}\mathrm{T}}\hat{\boldsymbol{M}}^{\mathrm{pi,ti}}||_F^2 \quad (3)$$

Direct construction of the original affinity matrix is avoided in Eqn (3) because TC and TF are much greater than D and D' [5]. Using these two kinds of losses, the total loss function is described as Eqn (4).
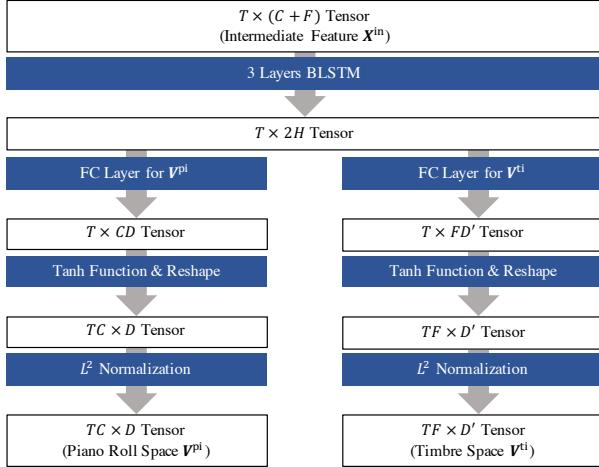
$$\mathcal{L}_{total} = \mathcal{L}_{DS} + \alpha\mathcal{L}_{DC}{}^{\mathrm{pi}} + \beta\mathcal{L}_{DC}{}^{\mathrm{ti}} \quad (4)$$

α and β in Eqn (4) are parameters to decide weights of each loss. We set them both at 0.000001 in our experiment.

For the stabilization of early learning stage, we first pretrained multi-pitch estimator and simultaneous embedding network respectively using the loss in Eqn (1) and Eqn (3). After pretraining, global optimization was conducted through end-to-end training by Eqn (4). We used Adam optimizer [22] for every training.

3.5 Estimation

At inference time, we generate two binary masks $\{\boldsymbol{M}_i^{\mathrm{pi}}\}_{i=1,...,N+1}$ and $\{\boldsymbol{M}_j^{\mathrm{ti}}\}_{j=1,...,N}$ for $\boldsymbol{X}^{\mathrm{pi}}$ and $\boldsymbol{X}^{ti}$ respectively from learned latent spaces $\boldsymbol{V}^{\mathrm{pi}}$ and $\boldsymbol{V}^{\mathrm{ti}}$. Mask generation is conducted by clustering the embedded features. Here, since the two spaces are hyperspherical shaped, we execute clustering by means of spherical k-means [23] though original deep clustering simply uses k-means. Because spherical k-means groups features based

Figure 3. Details of simultaneous embedding part in Figure 2. $2H$ is the number of hidden nodes in BLSTM.

on their distance on a hypersphere, i.e. cosine distance, this should be applied for our purpose rather than k-means. Piano rolls of each musical instrument part and silent part $\{Y_i^{\mathrm{pi}}\}_{i=1,\ldots,N+1}$ are calculated by Eqn (5).

$$Y_i^{\mathrm{pi}} = X^{\mathrm{pi}} \otimes M_i^{\mathrm{pi}} \qquad (5)$$

Additionally, spectrograms of each part $\{Y_j^{\mathrm{ti}}\}_{j=1,\ldots,N}$ are obtained by Eqn (6), which can be converted to separated sounds of each instrument via inverse STFT.

$$Y_j^{\mathrm{ti}} = X^{\mathrm{ti}} \otimes M_j^{\mathrm{ti}} \qquad (6)$$

In Eqn (5) and Eqn (6), element wise product is described as $\otimes$. Since M_j^{ti} is only for retained bins of X^{ti}, other bins are shared with all sources.

4. EVALUATION

4.1 Data

We used the Slakh2100-orig dataset [24] for our evaluation. The dataset contains 1500 training tracks, 375 validation tracks, and 225 test tracks. Each track is composed of multiple instruments, and the dataset consists of both mixed and separated sound sources with their MIDI data. It contains twelve kinds of instruments: piano, bass, guitar, drums, strings, synth pad, reed, brass, organ, pipe, synth lead, and chromatic percussion. We eliminated drums and chromatic percussion from the data to focus on instruments where pitch is important, *i.e.*, we used the other ten instruments for the experiment. To demonstrate the capability of estimating the piano rolls of arbitrary musical instrument part, we only used seven instruments (piano, bass, guitar, strings, synth pad, reed, and brass) for the training and validation data. We evaluated the performance using test data; above seven for the closed condition (seen instruments), and ten for the open condition (unseen instruments).

Training samples are constructed by cutting the tracks into ten seconds segments. Ground truth for condensed pitchgram is prepared by overlaying the MIDI data for the constituent sound sources. To make the mixture signal and the ground truth of condensed pitchgram, we overlaid both cut sound sources and MIDI data. Here, segments that do not have instrument sound for more than five seconds are omitted. The mixture MIDI data are binarized and gaussian blurred according to [21]. The musical recordings are mono-channel and their sampling rates are 44.1kHz. We computed STFT using Hann window with a size of 2048 time frames $\approx$ 50ms. The hop size is 512 frames $\approx$ 11ms for both STFT and HCQT. HCQT is computed for harmonics of {0.5, 1, 2, 3, 4, 5} with the minimum frequency 32.7Hz (C1) over six octaves. Our implementation uses the librosa library [25]. In total, 11 hours of training data and 3 hours of validation data were generated. For test data, 6 hours of data were generated for each condition.

4.2 Experimental Conditions

We evaluated the frame-level accuracy of transcriptions for each instrument part in the mixture. For the experiment, we fixed the number of mixed instruments to three, *i.e.*, $N = 3$. The transcription accuracy is evaluated by precision, recall, and F-measures. We count the pitchgram bin of a certain instrument as correct when binary values of estimation result and ground truth match with a correct part attribution. These metrics are calculated with Eqn (7),

$$P = \frac{TP}{TP + FP}, \ R = \frac{TP}{TP + FN}, \ F = \frac{2PR}{P + R} \quad (7)$$

where TP, FP, and FN are the number of true positive, false positive, and false negative, respectively. These values are calculated by the mir_eval [26] library.

To compare with the existing state-of-the-art classification-based method, we reimplemented [3] with eight output classes: the seven known instruments above and a non-instrument class. For a fair comparison between our clustering approach and the existing classification approach while considering the correctness of part attribution, we set evaluation conditions as follows:
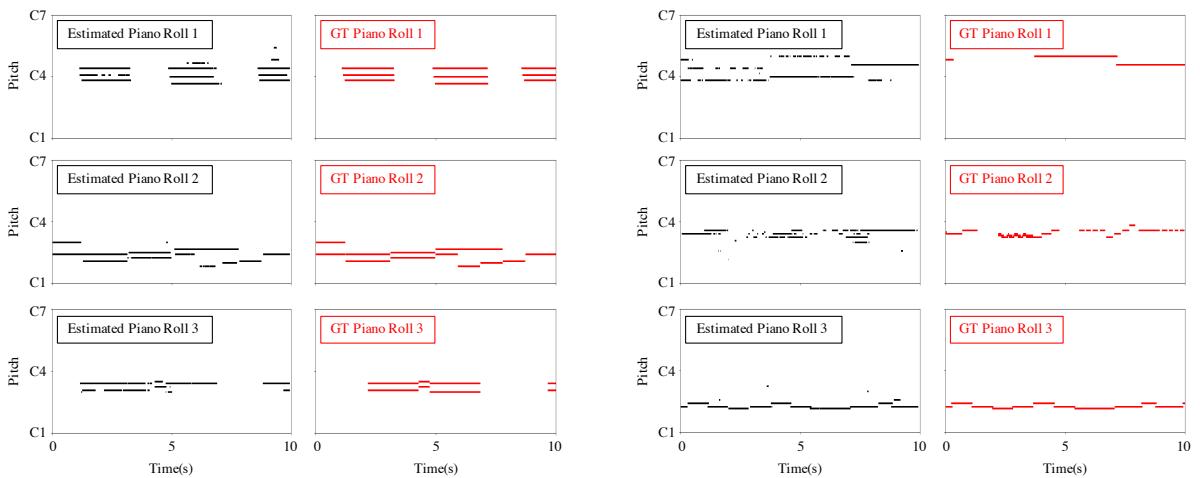
1. In the clustering approach, part attribution is not conducted explicitly. Thus, clusters are assigned to each instrument source by optimizing the F-measure.

2. In the classification approach under the closed condition, estimated parts are directly used as part assignments.

3. In the classification approach under the open condition, by design, part attribution cannot be conducted for unknown instrument sources. Thus, estimated parts are reassigned to each instrument source included in the audio by optimizing the F-measure.

4.3 Experimental Results

The experimental results are shown in Table 1. Our proposed method outperformed the state-of-the-art classification-based method [3] in the transcription of unknown instruments under the open condition. Furthermore, the F-measure score of unknown instruments

| Instrument | Closed condition | | | | | | Open condition | | | | | |
| | [3] | | | Our method | | | [3] | | | Our method | | |
	P	R	F	P	R	F	P	R	F	P	R	F
Piano	51.28	46.50	**45.87**	62.02	39.61	44.07	52.51	48.04	**47.37**	61.87	38.90	43.64
Bass	73.75	58.79	**64.04**	39.72	50.78	42.24	74.27	59.66	**64.67**	40.59	51.88	43.23
Guitar	46.64	36.72	37.69	52.91	35.45	**39.46**	44.59	37.12	37.25	53.45	36.50	**40.32**
Strings	55.27	56.79	**52.74**	66.35	48.74	52.40	53.21	56.97	**52.05**	65.31	48.40	52.04
Synth pad	43.72	44.80	**42.07**	49.65	35.12	38.70	44.42	46.89	**43.91**	51.99	36.58	40.81
Reed	28.53	33.90	29.27	29.87	37.37	**31.53**	26.92	31.72	27.53	28.87	35.46	**30.04**
Brass	35.24	25.12	24.50	37.10	30.23	**29.53**	37.66	25.67	25.89	36.78	30.64	**30.26**
Organ	—	—	—	—	—	—	20.14	19.01	16.89	36.62	28.57	**29.11**
Pipe	—	—	—	—	—	—	22.62	27.13	23.02	38.37	39.49	**35.22**
Synth lead	—	—	—	—	—	—	20.58	17.44	17.59	29.41	25.11	**24.98**

Table 1. Comparative results of MI-MPE on the Slakh2100-orig dataset [24] with classification-based method by [3] and our method. P, R, and F are precision, recall and F-measure, respectively, defined in Eqn (7).

Figure 4. Transcribed piano rolls of each instrument part from the mixture signals. The left pairs are successful cases (Track01879) and the right pairs are failure cases (Track01878). The left column shows the estimated piano rolls (black) and the right column shows the ground truths (red). Each row shows the corresponding part, respectively.

was comparable to that of known instruments in our method, while the score significantly decreased in the classification-based method. Our method also succeeded in transcribing known instruments under both conditions at an accuracy equivalent to the classification-based method.

Examples of estimated piano rolls using our method are illustrated in Figure 4. In the successful cases, although some errors are present, it can be seen that our proposed method well-conducted pitch estimation and instrument assignment. In the failure cases, some notes which have to appear in piano roll two are transcribed in piano roll one around three seconds, in addition to many misestimations.

4.4 Discussion

Our method can obtain separated sounds of each instrument part in addition to their piano rolls; however, matching the estimated piano rolls and the instrument part labels still have to be done manually. One of the most interesting directions of this research is the automation of this process.

Also, we assume that each time-frequency bin is attributed to only one source as mentioned in Section 3.3 though different instruments may share the same bin in practice. To deal with this case, another direction is to introduce the von Mises-Fisher (vMF) distribution [27, 28] into the hyperspherical latent space and perform soft clustering based on this distribution.

5. CONCLUSION

This paper presented a method for transcription of arbitrary musical instrument parts based on deep spherical clustering. Timbral and pitch characteristics of the music signal are simultaneously considered in the transcription, through joint clustering of a pitchgram and a spectrogram. The experimental results showed that the proposed method is capable of transcribing musical pieces including musical instruments not in training data. We plan to automate the matching process and introduce the vMF distribution into the hyperspherical latent space for future work.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] M. G. Christensen, P. Stoica, A. Jakobsson, and S. H. Jensen, "Multi-pitch estimation," *Signal Processing*, vol. 88, no. 4, pp. 972–983, 2008.

[2] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine (SPM)*, vol. 36, no. 1, pp. 20–30, 2019.

[3] Y. Wu, B. Chen, and L. Su, "Polyphonic music transcription with semantic segmentation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 166–170.

[4] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 771–775.

[5] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 31–35.

[6] D. Yu, M. Kolbæk, Z. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 241–245.

[7] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 246–250.

[8] S. Ewert, B. Pardo, M. Muller, and M. D. Plumbley, "Score-informed source separation for musical audio recordings: An overview," *IEEE Signal Processing Magazine (SPM)*, vol. 31, no. 3, pp. 116–124, 2014.

[9] Z. Duan and B. Pardo, "Soundprism: An online system for score-informed source separation of music audio," *IEEE Journal of Selected Topics in Signal Processing (JSTSP)*, vol. 5, no. 6, pp. 1205–1215, 2011.

[10] T. Nakano, K. Yoshii, Y. Wu, R. Nishikimi, K. W. Edward Lin, and M. Goto, "Joint singing pitch estimation and voice separation based on a neural harmonic structure renderer," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019, pp. 160–164.

[11] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: Challenges and future directions," *Journal of Intelligent Information Systems (JIIS)*, vol. 41, 2013.

[12] Z. Duan, J. Han, and B. Pardo, "Multi-pitch streaming of harmonic sound mixtures," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 22, no. 1, pp. 138–150, 2014.

[13] Z. Duan, B. Pardo, and C.Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 18, no. 8, pp. 2121–2133, 2010.

[14] M. Wu, D. Wang, and G. J. Brown, "A multipitch tracking algorithm for noisy speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 11, no. 3, pp. 229–241, 2003.

[15] Z. Jin and D. Wang, "Hmm-based multipitch tracking for noisy and reverberant speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 19, no. 5, pp. 1091–1102, 2011.

[16] V. Arora and L. Behera, "Multiple f0 estimation and source clustering of polyphonic music audio using plca and hmrfs," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 23, no. 2, pp. 278–287, 2015.

[17] E. Benetos and S. Dixon, "Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model," *The Journal of the Acoustical Society of America (JASA)*, vol. 133, pp. 1727–1741, 2013.

[18] L. C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," in *eprint arXiv:1706.05587*, 2017.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 9351, pp. 234–241, 2015.

[20] Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, and N. Mesgarani, "Deep clustering and conventional networks for music separation: Stronger together," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 61–65.

[21] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for f_0 estimation in polyphonic music," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 63–70.

[22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2014.

[23] I. S. Dhillon, J. Fan, and Y. Guan, "Efficient clustering of very large document collections," *Data Mining for Scientific and Engineering Applications*, vol. 2, 2001.

[24] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019.

[25] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Python in Science Conference (SciPy)*, 2015, pp. 18–24.

[26] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mir_eval: A transparent implementation of common mir metrics," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[27] S. Gopal and Y. Yang, "Von mises-fisher clustering models," *International Conference on Machine Learning (ICML)*, vol. 1, pp. 269–302, 2014.

[28] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra, "Clustering on the unit hypersphere using von mises-fisher distributions," *Journal of Machine Learning Research (JMLR)*, vol. 6, pp. 1345–1382, 2005.

SuPP & MaPP: ADAPTABLE STRUCTURE-BASED REPRESENTATIONS FOR MIR TASKS

Claire Savard[1] **Erin H. Bugbee**[2] **Melissa R. McGuirl**[3]

Katherine M. Kinnaird[4]

[1] Department of Physics, University of Colorado-Boulder, USA

[2] Department of Biostatistics, Brown University, USA

[3] Division of Applied Mathematics, Brown University, USA

[4] Department of Computer Science and Program in Statistical & Data Sciences, Smith College, USA

`claire.savard@colorado.edu`

ABSTRACT

Accurate and flexible representations of music data are paramount to addressing MIR tasks, yet many of the existing approaches are difficult to interpret or rigid in nature. This work introduces two new song representations for structure-based retrieval methods: *Surface Pattern Preservation (SuPP)*, a continuous song representation, and *Matrix Pattern Preservation (MaPP)*, SuPP's discrete counterpart. These representations come equipped with several user-defined parameters so that they are adaptable for a range of MIR tasks. Experimental results show MaPP as successful in addressing the cover song task on a set of Mazurka scores, with a mean precision of 0.965 and recall of 0.776. SuPP and MaPP also show promise in other MIR applications, such as novel-segment detection and genre classification, the latter of which demonstrates their suitability as inputs for machine learning problems.

1. INTRODUCTION

This paper builds on the traditions of matrix representations of songs started by Foote [1]. Specifically, we are principally interested in the cover song identification task. Recent content-based approaches to this task include building objects that compare one recording against a second one [2, 3], comparing slices of recordings to each other [4], creating a graph of songs on which to perform clustering [5], and using deep learning [6].

Structure-based approaches to the cover song task begin by creating representations encoding songs' structural information. These structural representations do not always explicitly encode where repetitions occur (for example, [7]). In contrast, the aligned hierarchies encode every possible repeated structure's placement within a song [8]. Between

these approaches is recent work by McGuirl et al. [9] which develops start-end (SE) and start(normalized)-length (S_NL) diagrams. These structure-based musical representations seek to balance the amount of structural information provided by leveraging ideas from Topological Data Analysis (TDA), a field of applied mathematics.

We address issues of SE and S_NL diagrams, discussed in Section 2.2, by transforming S_NL diagrams into surface and matrix representations, called Surface Pattern Preservation (SuPP) and Matrix Pattern Preservation (MaPP). SuPP and MaPP are analogous to TDA persistence surfaces and persistence images [10], respectively. These novel representations can be thought of as two versions of the same concept, with SuPP as the complete representation containing all possible structural information, and MaPP as its down-sampled computationally friendly extension. MaPP can be embedded into Euclidean space [1] making calculations straightforward using distance functions. Thus, MaPPs are more usable for machine learning algorithms than their predecessors.

Additionally, SuPP and MaPP are both adaptable to varying MIR tasks, such as the cover song task, novel-segment detection, and genre classification, whereas the aligned hierarchies, SE diagrams, and S_NL diagrams were created specifically for the cover song task. We present ways in which SuPP and MaPP may be adapted for these different tasks with a larger focus on the cover song task and how these new methods compare with results from S_NL diagrams and another extension of the aligned hierarchies.

2. BACKGROUND

Continuing the work begun with the aligned hierarchies [8] and extended in S_NL diagrams [9], SuPP and MaPP are consecutive representations that are smoothings of S_NL diagrams. Additionally, MaPP combines the strengths of the aligned hierarchies and S_NL diagrams to build a representation that can be used in standard machine learning algorithms such as k-means clustering or support vector machines. In this section, we briefly review aligned hierarchies and S_NL diagrams while highlighting their limitations to motivate the novel representations proposed in this work.

---

[1] In fact, MaPP can be embedded into any inner product space.

2.1 Aligned Hierarchies

The aligned hierarchies representation encodes all possible hierarchical structure decompositions of a song on a single common time axis [8]. While this representation clearly shows the length of each repeated section, it does not visually emphasize the differences between the lengths of repeats using white space. Also, the distance measure for the aligned hierarchies is inefficient to compute and is quite coarse [8]. First, the distance measure notes only when two repeats of the same size line up exactly in terms of placement within the song. Second, comparisons between songs under this distance measure require both songs to be the same number of beats. This last limitation makes using the aligned hierarchies impractical for cover song detection.

2.2 SE and S_NL Diagrams

Motivated by TDA theory, SE and S_NL diagrams extend the aligned hierarchies and overcome the rigidity in comparing songs with aligned hierarchies. While SE and S_NL diagrams are more flexible and computationally efficient than their predecessor, they were built with the cover song task in mind and are difficult to adapt for other MIR tasks. Furthermore, they are not suitable to use with machine learning algorithms.

2.2.1 Topological Data Analysis Inspiration

Broadly, TDA is concerned with extracting quantifiable shape features from large, complex datasets [11–14]. Though not extracting topological information typically sought by TDA methods, SE and S_NL diagrams draw their inspiration from persistence diagrams [11, 12] which track how topological features persist across an increasing sequence of spatial scales. Just as persistence diagrams are a collection of 2-D points whose x- and y-coordinates represent the length scales at which the topological features appear and disappear, SE and S_NL diagrams are collections of 2-D points whose x- and y-coordinates represent the start and end (or length) times, respectively, of repetitive sections in a song. An advantage of the correspondence between persistence diagrams and SE and S_NL diagrams is that rich mathematical theory from TDA can be extended to the musical representations for computational tasks.

2.2.2 Structure of SE and S_NL

SE and S_NL diagrams extend the aligned hierarchies by transforming them into a representation consisting of a finite collection of points [9]. Specifically, the SE diagram for a song is defined as $\{(s_i, e_i)\}_{i=1}^{N} \subset \mathbb{R}_+^2$, where s_i and e_i are the start and end times, respectively, of the ith repeated structure. Similarly, the S_NL diagram for a song is defined as $\{(\alpha(s_i/M), e_i - s_i)\}_{i=1}^{N} = \{(\bar{s}_i, \ell_i)\}_{i=1}^{N}$, where $\alpha > 0$ is a scaling factor and M is a normalization term related to the length of the song, with s_i and e_i as above. Figure 1 shows the transition from the aligned hierarchies to the S_NL diagram for Chopin's Mazurka Op. 6, No. 1. Both the SE and S_NL diagrams add visual emphasis for the differences between the lengths of the repetitions, but lose the visual width of each repetition.

2.2.3 Shortcomings of Previous Work

Similar to persistence diagrams, the complex structure of SE and S_NL diagrams makes them unsuitable inputs for most statistical analyses and machine learning tasks. For example, these diagrams do not live in an inner product space and simply computing averages in the space of persistence diagrams, and therefore in the spaces of SE and S_NL diagrams, remains a challenge. The notion of an average persistence diagram is defined through the Fréchet mean, which views the space of persistence diagrams as a probability space [15]. The Fréchet mean is computed as the solution of a minimization problem and is not guaranteed to be unique. Moreover, it is non-trivial to compute.

Without a unique and easy-to-compute mean or an inner product structure, the utility of SE and S_NL diagrams for MIR tasks is limited. In particular, SE and S_NL diagrams cannot be used as inputs for most classification and regression models, such as support vector machines, which would otherwise be useful for tasks like genre classification.

2.3 SuPP and MaPP Inspiration

This work continues to leverage TDA theory in the creation of two new structural representations, SuPP and MaPP. Just as persistence diagrams are transformed into persistence surfaces and persistence images [10] through a weighted sum of Gaussian functions centered at each point in a given persistence diagram, here we transform SE and S_NL into SuPP and MaPP. The mathematical details of this transformation are provided in Section 3. Like persistence images, MaPPs can be embedded into an inner product space, such as Euclidean space, and can therefore be used as inputs for machine learning algorithms.

3. METHODS

In this section, we define Surface Pattern Preservation (SuPP) and Matrix Pattern Preservation (MaPP). SuPP is a surface representation of S_NL diagrams [9] that allows similar repeated sections to be viewed as a single structure. Since comparing surfaces computationally is difficult, we introduce MaPP, the discrete matrix version of SuPP. As a matrix, MaPP allows for pairwise comparisons using common distances such as the Euclidean and Frobenius metrics. The procedure [2] for creating SuPP and MaPP from S_NL diagrams is outlined below and summarized in Algorithm 1.
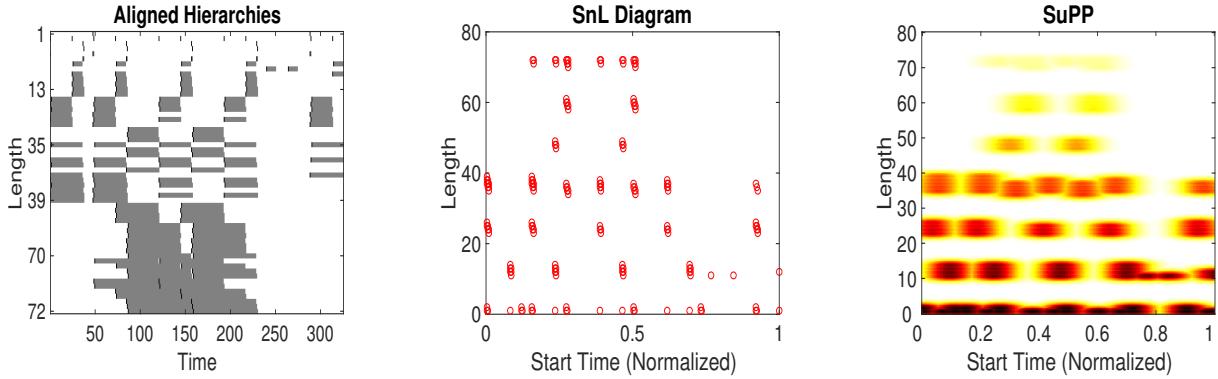
3.1 Surface Pattern Preservation (SuPP)

SuPP is a smoothing of a song's S_NL diagram that maintains structural information of the song. This smoothing is achieved by placing a 2-D Gaussian at each point in a song's S_NL diagram and aggregating overlapping Gaussians.

3.1.1 Defining Repeats as Gaussians

Defining the Gaussians that represent the repeated structures of the piece is the heart of the transformation from

---

[2] Code is available on GitHub: `https://github.com/cgsavard/ICERM_compare_songs`

Figure 1: From left to right, the aligned hierarchies, S_NL diagram, and SuPP corresponding to the score of Mazurka Op. 6, No. 1 by Chopin. The weights applied to SuPP are the weights used for the cover song task found in Section 4.2.1.

the S_NL diagram to SuPP. In general, given a S_NL diagram $\{(\bar{s}_i, \ell_i)\}_{i=1}^N$, we first place a Gaussian (g) over each repeated structure so that $(\bar{s}_i, \ell_i) \rightarrow g_i(\bar{s}, \ell)$ where

$$g_i(\bar{s}, \ell) = \frac{1}{2\pi\sigma_s\sigma_\ell} e^{-\left(\frac{(\bar{s}-T_s(\bar{s}_i))^2}{2\sigma_s^2} + \frac{(\ell-T_\ell(\ell_i))^2}{2\sigma_\ell^2}\right)}, \quad (1)$$

$T = (T_s, T_\ell) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defines the center of each of the Gaussians, σ_s is the standard deviation in the start direction, and σ_ℓ is the standard deviation in the length direction.

The Gaussians can either be centered at the beginning of a repeated structure, so that $T(\bar{s}_i, \ell_i) = (T_s(\bar{s}_i), T_\ell(\ell_i)) = (\bar{s}_i, \ell_i)$ is the identity, or in the middle of a repeated structure, so that $T(\bar{s}_i, \ell_i) = (T_s(\bar{s}_i), T_\ell(\ell_i)) = (\bar{s}_i + \frac{\ell_i}{2}, \ell_i)$. This choice is based on the task at hand as well as user preference. Midpoints correspond to the central beat in each repeated section, and are therefore a good indicator for where these structures are occurring, on average, in the song. For our cover song task experiment, we adjust the S_NL diagrams by choosing to center each Gaussian about the repeated structures' midpoints rather than their start times (i.e., $T(\bar{s}_i, \ell_i) = (\bar{s}_i + \frac{\ell_i}{2}, \ell_i)$). Figure 2 illuminates how using the start or midpoint of the repeated section for the Gaussians compares visually for the repetitions in aligned hierarchies. Other distributions aside from a Gaussian can also be used and may be preferable for certain MIR tasks.
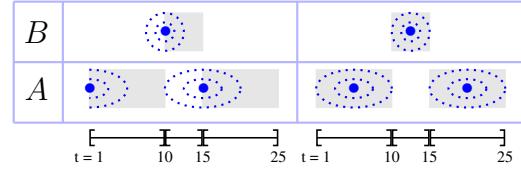
After determining the appropriate placement of the Gaussians, the next step is to set the two standard deviation parameters that govern the shape of these Gaussians. The standard deviation σ_s determines the width of the Gaussians with respect to the start (or time) axis; that is, the axis representing where each repeat in the song - now represented as a Gaussian - exists. The standard deviation σ_ℓ determines the width of the Gaussians with respect to the length axis. These standard deviations control the extent to which nearby Gaussians will overlap.

Figure 2: How 2-D Gaussians relate to repeated structures of aligned hierarchies when using the S_NL diagram (left) or the altered S_NL diagram with midpoints (right).

3.1.2 Creating the Surface

The next step in the SuPP creation is to aggregate the collection of Gaussians to create a surface. When two or more Gaussians intersect, which occurs when points in the S_NL diagram are close together, the SuPP value at the intersection is set to the maximum of the Gaussians rather than the sum of the Gaussians as is done for persistence surfaces [10]. That is, for any S_NL diagram $\{(\bar{s}_i, \ell_i)\}_{i=1}^N$, we define SuPP as the surface SuPP : $\mathbb{R}^2 \rightarrow \mathbb{R}$, where

$$\text{SuPP}(\bar{s}, \ell) = F(\bar{s}, \ell) * \max_{i \in [1, N]} g_i(\bar{s}, \ell), \quad (2)$$

for some weighting function $F(\bar{s}, \ell) : \mathbb{R}^2 \rightarrow \mathbb{R}$ and $g_i(\bar{s}, \ell)$ defined in Eqn. 1. Combining the Gaussians over all repeats allows repetitive structures that are similar in length and start time to be perceived as the same repeated section. We use the maximum instead of other aggregators because we want these similar structures to be treated as one section. We do not want sections to be more highly weighted if there are many similar structures in that section. The choices of σ_s and σ_ℓ determine how close points in S_NL need to be for their Gaussians to substantially blur together.

Algorithm 1 Algorithm for constructing SuPP and MaPP

Input: Song's S_NL diagram $\{(\bar{s}_i, \ell_i)\}_{i \in I}$
for $i \in I$ **do**
    • Replace $(\bar{s}_i, \ell_i)$ with 2-D Gaussian defined by
      $\mu = (\bar{s}_i, \ell_i)$ and $\sigma = (\sigma_s, \sigma_\ell)$
end for
    • Aggregate all Gaussians to create a surface by using
      the maximum function
    • Apply weighting function to surface to create SuPP
    • Discretize SuPP to create gridded surface
    • Integrate the area under each gridded unit and store
      resulting values in a matrix to create MaPP
Outputs: SuPP, MaPP

The weighting function $F(\bar{s}, \ell)$ governs which type of repeated structures are emphasized in the resulting surface. This weight controls the heights of the Gaussians in SuPP, lifting important sections of the song and suppressing other sections. The surface weight can be varied by the user based on the MIR task at hand. In Section 4, we provide an example of how one may want to set the weighting function.

3.2 Matrix Pattern Preservation (MaPP)

SuPP is a continuous representation carrying all information found in S_NL diagrams. Beyond this, user-specified parameters can be set to emphasize various parts of a song. However, using a continuous surface representation is computationally complex and thus not feasible for many computing tasks. To address this challenge, a transformation of SuPP into a discrete representation with a natural embedding and metric for comparison, such as MaPP, is necessary.

To create MaPP, SuPP is first sectioned by placing a grid over the $\mathbb{R}^2$ plane over which SuPP is defined. This discretization is based on a user-defined resolution. Then, the volume beneath each grid unit of SuPP is computed using numerical integration. These volumes are recorded as entries in a matrix with the same dimensions as the gridded SuPP, resulting in MaPP.

Namely, for a S_NL diagram $\{(\bar{s}_i, \ell_i)\}_{i=1}^{N}$, the corresponding MaPP is a $P \times P$ matrix such that:

$$\text{MaPP}(\text{SuPP})_{jk} = \int_{\beta_k}^{\beta_{k+1}} \int_{\alpha_j}^{\alpha_{j+1}} \text{SuPP}(\bar{s}, \ell) d\bar{s} d\ell \quad (3)$$

where $\alpha_j = j\frac{R_{\bar{s}}}{P}$ and $\beta_k = k\frac{R_\ell}{P}$ are the individual grid widths and heights given by the user-defined resolution P and the ranges $R_{\bar{s}}$ and R_ℓ of the respective axes in the SuPP.

3.3 Embedding MaPP Representations

Since the MaPP representations are matrices, they can be embedded into various metric spaces. There are many pre-existing metrics with strong mathematical theory that can be applied to compute distances between matrices. We use the Frobenius norm. The Frobenius distance between two MaPPs A and B is defined as:

$$d_F(A, B) = \sqrt{\text{Tr}((A - B) * (A - B)^T)}, \quad (4)$$

where Tr indicates the trace [16]. The Frobenius norm measures the "average" value within the difference matrix $A - B$.

With any suitable embedding, we inherently define a classification space for MaPP representations and the songs that they represent. Thus, we can employ various computational techniques to compare songs. We also note that not all MIR tasks rely on song comparisons, and MaPP can also be used for exploration within a song.

4. APPLICATION TO COVER SONG TASK

SuPP and MaPP can be used in structure-based retrieval methods for a variety of MIR tasks. In this section, we show how these representations can address the cover song task, and compare our methods with some previous methods.

4.1 Dataset

To test the efficacy of SuPP and MaPP, we use a collection of Chopin's Mazurka scores as **kern files from the KernScore online database [3] [17]. There are 52 scores in the Mazurka dataset, and we extract two versions for each score. The first version of each piece, referred to as the "expanded" score, plays each repeated section as marked in the score. The second version, referred to as the "non-expanded" score, does not respect these repetition markers and plays marked repeated sections once. As a result, the data consists of 104 songs with pairs of expanded and non-expanded versions for each Mazurka piece.

Each score is initially represented as a thresholded self-dissimilarity matrix (SDM). Following the procedure from [8, 18], we first extract a chroma vector for each beat in the score using the Python library music21. We then build audio shingles [4] for each beat [19–21] by concatenating S number of consecutive chroma vectors, encoding local information for each beat. We set the shingle width to $S = \{6, 12\}$ and then compute the cosine-dissimilarity measure between each pair of audio shingles. We finally create the SDM by thresholding the matrices using $\mathcal{T} = \{0.01, 0.02, 0.03, 0.04, 0.05\}$. This SDM is converted to the aligned hierarchies [8], and then to S_NL diagrams [9]. From the S_NL diagram, we create a SuPP which is then discretized to become a MaPP, as described in Section 3.2.

4.2 Experiment

The cover song identification task, or version detection task, aims to identify recordings that are performances of the same piece of music. We address this task by creating SuPP and MaPP representations for each song and calculating pairwise Frobenius distances between MaPPs. The distance between two MaPPs is a measurement of structure dissimilarity, which is used alongside a clustering technique to deem whether songs are covers of the same work.

4.2.1 Setting SuPP Parameters
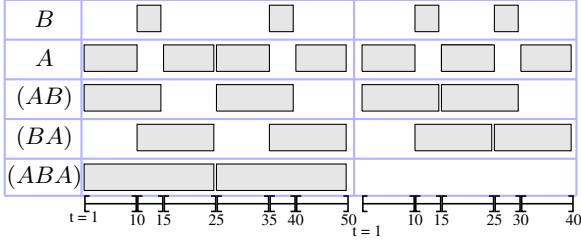
As discussed in Section 3.1, to create the SuPP representation we start by defining where the Gaussians representing each repetition are centered. For addressing the cover song task, the S_NL diagram is adjusted so the horizontal axis reflects the midpoints of the repeated structures instead of their start times. [5] We next define σ_s, σ_ℓ, and the weighting function to determine the size, shape, and height of the 2-D Gaussians placed over each point in the S_NL diagram.

For these experiments, we use a normalized constant standard deviation for σ_s, which determines the width of the Gaussians on the time axis. Recall that the time axis is normalized in the S_NL diagrams so that all songs are placed within the same range. We set $\sigma_s = \frac{1}{M}$ beats for each song, where M is the normalization constant, or the total

---

Figure 3: These aligned hierarchies represent songs of structure ABAABA (left) and ABABA (right). By omitting a middle "A" section in ABAABA, the longer repetitive sections break down while the shorter ones are preserved.

song length, inherited from the $S_N L$ diagram for a given song. This means that repeats of the same length that start within two beats ($2\sigma_s$) of each other will overlap, and thus combine to form a single structure in the SuPP.

The second parameter is σ_ℓ, which determines the width of the Gaussians in the length direction. We use a constant value for this parameter, setting $\sigma_\ell = 1$. Since the songs are not normalized along this dimension in the $S_N L$ diagrams, no normalization term is needed here. This means that there will be overlap between repeats centered at the same beat and those whose lengths differ by up to two beats.

Lastly, we choose a weighting function to apply to our surface that will give more importance to shorter structures than longer structures, due to longer repetitive structures having more variance between cover songs than shorter ones. An example of this phenomenon can be seen in Figure 3; a slight change in the overall song pattern, such as repeating the chorus one fewer time, breaks down the largest structures while maintaining the shorter repeats. To encode this importance on smaller-sized structures, we choose the following piece-wise linearly decreasing function:

$$F(\bar{s}, \ell) = \begin{cases} 1 & \ell \leq \ell_{min} \\ 1 - \frac{\ell - \ell_{min}}{\ell_{max} - \ell_{min}} & \ell_{min} \leq \ell \leq \ell_{max} \\ 0 & \ell \geq \ell_{max} \end{cases} \quad (5)$$

where ℓ_{min} and ℓ_{max} are user-specified bounds on the lengths of repetitive structures included in SuPP. We set $\ell_{min} = 0$ beats and $\ell_{max} = 80$ beats in order to include 98% of structures seen in our data. As the length increases from ℓ_{min} to ℓ_{max}, the weight decreases from one to zero.

4.2.2 Comparing MaPPs

For the cover song task, we set the MaPP resolution to $P = 200$, yielding a 200 x 200 matrix. This choice of resolution follows the work in [10, 22], which shows that this parameter value is robust, though other resolutions may be applied. This process is analogous to the resampling in [7] but is an aggregation within SuPP instead of a sampling.

After creating a MaPP for each song, we compute pairwise Frobenius distances and apply a clustering algorithm. Noting that MaPP encodes a notion of musical structure, the Frobenius distance offers a measure of the dissimilarity between the musical structures of two different pieces. For cover songs, we expect this distance to be low because

songs that cover the same piece of music will likely have similar repeated musical structures.

For the Mazurka scores dataset, each song has exactly one match, namely the expanded version with repeat markers honored and the non-expanded version where the repetition markers are ignored. Therefore, we use mutual k-nearest neighbors with $k = 1$ to pair the songs. Under this technique, two songs are only labeled covers of the same piece if they both claim each other to be their closest "neighbor," corresponding to the smallest distance computed. If there is no mutual nearest neighbor, then that song is not matched to any other in the dataset.

4.2.3 Results
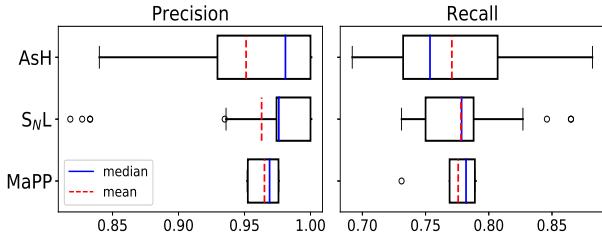
Ten experiments were performed with varying thresholds and shingle numbers applied to the SDMs. Overall, precision and recall values across these experiments were consistent (see Table 1) with mean precision of 0.965 and mean recall of 0.776. Figure 4 shows these results to be comparable to similar experiments on average using $S_N L$ diagrams [9] and aligned sub-hierarchies [6] (AsH) [18]. However, MaPP results have less variability among the ten experiments and thus show more stability. Additionally, MaPP is more flexible in its creation, allowing for more user creativity, and it is more computationally efficient to compare MaPPs than $S_N L$ or AsH representations, as the latter two include optimal matching steps in their comparisons.

We found that songs with few repetitive structures (and thus a scarce $S_N L$ diagram) make up the majority of the songs left without a match or improperly matched. Therefore, our method works best when analyzing songs with ample repetitive structures. An example of expanded and non-expanded versions of a score that were not matched together is shown in Figure 5, visibly due to scarce amounts of repeated structures in the non-expanded $S_N L$ diagram.

---

[6] AsH are extensions of aligned hierarchies that make aggregate comparisons using sections of songs instead of one cohesive structure representation as with aligned hierarchies.

Threshold ($\mathcal{T}$)	Shingle (S)	Precision	Recall
0.01	6	0.952	0.769
	12	0.975	0.778
0.02	6	0.974	0.731
	12	0.964	0.786
0.03	6	0.952	0.769
	12	0.976	0.789
0.04	6	0.952	0.769
	12	0.976	0.789
0.05	6	0.976	0.789
	12	0.954	0.789

Table 1: Experimental results for the cover song task using midpoint versions of $S_N L$ diagrams, normalized constant σ_s, constant σ_ℓ, and linearly decreasing weight along the length axis from Eqn. 5.

Figure 4: A comparison of precision and recall values for varying threshold and shingles for three different methods: MaPP, S_NL diagrams, and AsH. The AsH results do not include the songs which were left unmatched due to empty AsH representations (which varied between 14% and 65% of the dataset) as error, thus inflating the results compared to the other two methods.

Figure 5: Mazurka Op. 68 No. 4 expanded (left) and non-expanded (right) do not match using k-mutual nearest neighbors with $k = 1$ on their corresponding MaPPs due to scarce repeated structures in the non-expanded S_NL diagram.

5. OTHER APPLICATIONS

Following are two additional examples of how SuPP and MaPP can be used to address other MIR tasks: novel-segment detection and genre classification. Preliminary experimental results show promise with both tasks, and future work will include a full analysis of these experiments.

5.1 Novel-Segment Detection
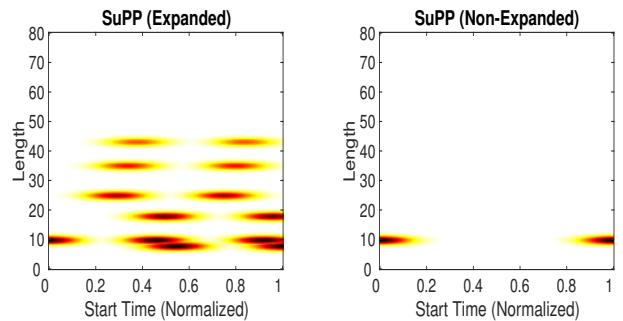
We define novel-segment detection to be finding the boundaries between repeated segments and novelty sections. This is a combination of both the novelty detection and segmentation tasks in MIR [23–25]. These boundaries often distinguish between typical segments within a musical piece, like a verse or a coda, making the segmentation task a natural application of such detection [23, 24].

For this task, we extend the analysis of MaPP from Section 4 by transforming the matrix into a vector; that is, given a MaPP, we create a 1-D vector by taking the sum of each column. This projection yields a time-dependent vector whose entry at a given time step corresponds to the sum of the structure activity measured by MaPP at that time. Global and local minima (as specified by user-specific constraints) of this projection correspond to regions between large amounts of structure. Outliers within the collection of minima correspond to novelty sections, where no repetitive structure is present for a long period of time. Preliminary work using this methodology of locating the minima of the summation projection of MaPP shows promising results, highlighting the flexibility of MaPP in other MIR tasks.

5.2 Genre Classification

In genre classification, we seek to classify songs by the genres assigned to them by their recording company. We use the collection of 104 Chopin's Mazurkas along with a selection of 676 Jazz lead sheets [26] from the *iRb Corpus* in the `**jazz` format to have two genres.

Given that MaPP representations embed into inner product spaces, we can use machine learning algorithms for MIR tasks. MaPP matrix elements become the features for each song with the number of features set by the resolution parameter. Various machine learning algorithms, implemented in sklearn,[7] are applied to our set of MaPPs as constructed in Section 4, representing 104 Mazurka scores and 676 jazz lead sheets. Logistic regression, a Gaussian kernel support vector machine (SVM), and a polynomial kernel SVM distinguish between the Mazurka and jazz pieces with high accuracy of above 94% for each classifier.

6. CONCLUSION

In this paper, we introduce SuPP and MaPP, two musical representations influenced by TDA theory. We describe how SuPP and MaPP are built and give intuition into how parameters may be chosen when applying these representations to the cover song task. Our accuracies using MaPP for this task are comparable to previous studies on average but indicate greater stability among various SDM thresholds and shingles. Preliminary experiments applying SuPP and MaPP to novel-segment detection and genre classification are plausible, demonstrating the adaptability of these representations for distinct MIR tasks.

We discuss how SuPP and MaPP overcome limitations of the aligned hierarchies [8] and S_NL diagrams [9], and how they are adaptable with user-specified parameters allowing for task-specific representations. Unlike its predecessors, MaPP is well suited for machine learning. Future work will demonstrate this through various MIR tasks, such as genre classification, and by applying similar methods to additional datasets including audio data, such as the Da-TACOS dataset [27]. A drawback of SuPP and MaPP is the necessary manual selection of parameters, requiring a deep understanding of the task at hand.

SuPP and MaPP, alongside SE and S_NL diagrams [9], offer inspiring insights and open up a realm of opportunities in the intersection of TDA and MIR. These methods are both grounded in mathematical theory and have practical applications to the field of MIR, as seen by the effective use of MaPP for the cover song task. The experiments in this paper further highlight the utility of TDA-based methods and explore new opportunities for future experimentation in the intersection of TDA and MIR.

---

[7] https://scikit-learn.org/stable/

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] J. Foote, "Visualizing music and audio using self-similarity," *Proc. of ACM Multimedia 1999*, pp. 77–80, 1999.

[2] J. Serrà, X. Serra, and R. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, no. 093017, 2009.

[3] C. J. Tralie, "Early MFCC and HPCP Fusion for Robust Cover Song Identification." *Proc. of 18th ISMIR Conference*, pp. 294–301, 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1417331

[4] D. F. Silva, F. V. Falcão, and N. Andrade, "Summarizing and comparing music data and its application on cover song identification," *Proc. of 19th ISMIR Conference*, pp. 732–739, 2018.

[5] M. Sarfati, A. Hu, and J. Donier, "Community-based cover song detection," *Proc. of 20th ISMIR Conference*, pp. 244–250, 2019. [Online]. Available: http://archives.ismir.net/ismir2019/paper/000028.pdf

[6] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," *Proc. of 20th ISMIR Conference*, pp. 107–114, 2019.

[7] P. Grosche, J. Serrà, M. Müller, and J. Arcos, "Structure-based audio fingerprinting for music retrieval," *Proc. of 13th ISMIR Conference*, pp. 55–60, 2012.

[8] K. M. Kinnaird, "Aligned hierarchies: A multi-scale structure-based representation for music-based data streams," *Proc. of 17th ISMIR Conference*, pp. 337–343, 2016.

[9] M. R. McGuirl, K. M. Kinnaird, C. Savard, and E. H. Bugbee, "SE and SNL diagrams: Flexible data structures for MIR," *Proc. of 19th ISMIR Conference*, pp. 341–347, 2018.

[10] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier, "Persistence images: A stable vector representation of persistent homology," *Journal of Machine Learning Research*, vol. 18, no. 8, pp. 1–35, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-337.html

[11] G. Carlsson, A. Zomorodian, A. Collins, and L. J. Guibas, "Persistence barcodes for shapes," *International Journal of Shape Modeling*, vol. 11, no. 02, pp. 149–187, 2005.

[12] H. Edelsbrunner and J. L. Harer, *Computational Topology: An Introduction*. American Mathematical Society, 2010.

[13] A. Zomorodian, *Topology for Computing*. Cambridge University Press, 2009.

[14] F. Chazal, V. de Silva, M. Glisse, and S. Oudot, *The Structure and Stability of Persistence Modules*, 1st ed. Springer International Publishing, 2016.

[15] Y. Mileyko, S. Mukherjee, and J. Harer, "Probability measures on the space of persistence diagrams," *Inverse Problems*, vol. 27, no. 12, p. 124007, November 2011.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[17] C. Sapp, "Online database of scores in the humdrum file format," *Proc. of 6th ISMIR Conference*, pp. 664–665, 2005.

[18] K. M. Kinnaird, "Aligned sub-hierarchies: A structure-based approach to the cover song task," *Proc. of 19th ISMIR Conference*, pp. 585–591, 2018.

[19] M. Casey, C. Rhodes, and M. Slaney, "Analysis of minimum distances in high-dimensional musical spaces," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 1015–1028, 2008.

[20] M. Casey and M. Slaney, "Song intersection by approximate nearest neighbor search," *Proc. of 7th ISMIR Conference*, pp. 144–149, 2006.

[21] ——, "Fast recognition of remixed audio," *Proc. of 2007 IEEE International Conference on Audio, Speech and Signal Processing*, pp. IV–1425 – IV–1428, 2007.

[22] M. Zeppelzauer, B. Zielinski, M. Juda, and M. Seidl, "Topological descriptors for 3D surface analysis," in *CTIC*, 2016.

[23] M. Hartmann, O. Lartillot, and P. Toiviainen, "Musical feature and novelty curve characterizations as predictors of segmentation accuracy," *Proc. of the Sound and Music Computing Conference*, pp. 365–372, 2017.

[24] J. Foote, "Automatic audio segmentation using a measure of audio novelty," *Proc. of IEEE International Conference on Multi-Media and Expo*, pp. 452–455 vol.1, 2000.

[25] M. Müller, T. Prätzlich, and J. Driedger, "A cross-version approach for stabilizing tempo-based novelty detection," *Proc. of* 13th *ISMIR Conference*, pp. 427–432, 2012.

[26] Y. Broze and D. Shanahan, "The iRb Corpus in `**jazz` format," http://musiccog.ohio-state.edu/home/index.php/iRb_Jazz_Corpus, 2012, [Online; accessed 28-September-2016].

[27] F. Yesiler, C. Tralie, A. A. Correya, D. F. Silva, P. Tovstogan, E. Gómez, and X. Serra, "Da-TACOS: A dataset for cover song identification and understanding," *Proc. of* 20th *ISMIR Conference*, pp. 327–334, 2019. [Online]. Available: http://archives.ismir.net/ismir2019/paper/000038.pdf

A METHOD FOR ANALYSIS OF SHARED STRUCTURE IN LARGE MUSIC COLLECTIONS USING TECHNIQUES FROM GENETIC SEQUENCING AND GRAPH THEORY

Florian Thalmann[1]   **Kazuyoshi Yoshii**[1]   **Thomas Wilmering**[2]
Geraint A.Wiggins[3]   **Mark B. Sandler**[2]

[1] Speech and Audio Processing Laboratory, Kyoto University
[2] Centre for Digital Music, Queen Mary University of London
[3] Artificial Intelligence Lab, Vrije Universiteit Brussel

`thalmann.florian.2x@kyoto-u.ac.jp`

ABSTRACT

While common approaches to automatic structural analysis of music typically focus on individual audio files, our approach collates audio features of large sets of related files in order to find a shared musical temporal structure. The content of each individual file and the differences between them can then be described in relation to this shared structure. We first construct a large similarity graph of temporal segments, such as beats or bars, based on self-alignments and selected pair-wise alignments between the given input files. Part of this graph is then partitioned into groups of corresponding segments using multiple sequence alignment. This partitioned graph is searched for recurring sections which can be organized hierarchically based on their co-occurrence. We apply our approach to discover shared harmonic structure in a dataset containing a large number of different live performances of a number of songs. Our evaluation shows that using the joint information from a number of files has the advantage of evening out the noisiness or inaccuracy of the underlying feature data and leads to a robust estimate of shared musical material.

1. INTRODUCTION

Automatic analysis of musical structure from audio is one of the more challenging tasks in music information retrieval (MIR) [1–3]. Reasons for this are the relatively high-level nature of the problem and its dependence on lower-level audio descriptors, which have a tendency to be noisy, as well as the restricted availability of annotated collections in a limited number of musical genres, which are necessary for tackling the problem with solutions based on machine learning. However, the growing number of large

public and online music collections, which are usually annotated with user-curated metadata (song titles, artists, recording information, or dates), can potentially be used for unsupervised structural analysis which may be of considerable musicological value.

This paper introduces an approach for the detection of temporal structure in large collections of musical audio recordings where information from a number of related recordings is combined to improve the quality of results. From a given set of input audio recordings, e.g. different performances of the same song, our method identifies the most commonly occurring sequential structures, relates them to each other and organizes them hierarchically. The individual files can then be described, compared and aligned with each other by referencing this shared structure. Inspired by techniques used in genetic sequencing, we combine the use of different alignment methods, including dynamic programming (DP) and multiple sequence alignment (MSA) with graph representations and search methods. We evaluate our method on a subset of the Live Music Archive (LMA) of the Internet Archive and analyze the harmonic content of a large number of performances of a selection of songs. We compare the harmonic essence thus obtained with existing lead sheets and illustrate the differences between individual performances in a few qualitative comparisons. Although the examples in this paper focus on formal structure determined by harmonic progressions, the method can easily be used for structure determined by other musical aspects.

2. RELATED WORK

With the omnipresence of large digital audio collections of music, the automatic analysis of large corpora is becoming an increasingly central task in music information retrieval. Recent research in this area has mainly focused on large-scale statistical analysis of audio features [4–6], as well as making these accessible using interactive browsing tools [7–9]. While the analysis of temporal structure, including the identification of musical patterns, motifs, harmonic progressions, or form across corpora, are relatively well established for collections of symbolic music [10–12], only a

few have attempted to use similar analysis methods for audio collections [13–15]. Potential reasons for this are often discussed and may be due to common problems with audio collections, including mislabeling, duplication, differing recording quality, or noisy audio features [4, 6, 16, 17].

Many of the methods used jointly in this paper have previously been applied in different musical contexts. Dynamic programming is often used to align audio recordings of different performances of the same material, either audio-to-score or audio-to-audio [18–20], sometimes considering musical variations and structural differences [21]. While immediate uses include score following, automatic accompaniment, and computer-assisted production, some have used alignment methods for classification tasks such as cover song identification [22] and plagiarism [23].
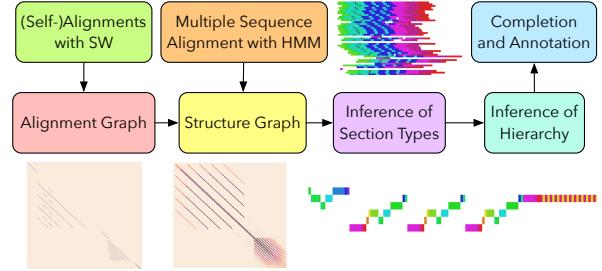
While the alignments methods most commonly used in MIR are pairwise, i.e. applied to align pairs of sequences, such as the Smith-Waterman or the Needleman-Wunsch algorithm [24], there are many methods for directly aligning multiple sequences commonly used in bioinformatics, but only a few have so far been applied to musical data. [25] used *multiple sequence alignment (MSA)* to eliminate conflicts and typos in song lyrics retrieved from the Web. [20] used their own progressive MSA method as well as *Profile HMMs* (Hidden Markov Models) to align different recordings of performances of classical music and found the two methods to lead to comparable results. In [26, 27], which comes closest to the present work, different MSA libraries were used along with pattern mining to detect harmonic patterns in symbolic transcriptions of a set of 138 songs. The authors were able to identify cover songs as well as genre clusters with their most characteristic progressions.

There are generally three common approaches to automatically discovering musical structure in sequences of feature vectors from individual recordings, via repetition, novelty, or homogeneity [1]. The first identifies repeating subsequences whereas the other two identify abrupt changes or comparatively stable areas. Sections often reappear in slightly varied forms, which has been addressed in [28]. Recent methods often use combined approaches using both harmonic and timbre features in order to improve results, e.g. [29]. While music is inarguably organized hierarchically [2], few approaches enable the detection of hierarchical structure [30].

Our approach is purely repetition-based, however, not only in individual recordings, but in the entire collection of interest. This allows the identification of sections occurring only once in a given piece and leads to more robust descriptions of the found sections. Similar to our approach, [31] used automatically detected musical structure to improve chord label quality for individual pieces.

3. METHOD

Our method consists of a five-step process. Given a set of recordings, we create an alignment graph based on a number of self-alignments and pairwise alignments. Then, we use an MSA to partition the alignment graph and create a structure graph. We then search the structure graph for

Figure 1. Overview of the five-step process.

commonly occurring sections and classify them into section types. These section types are then grouped into a hierarchical structure, based on their co-occurrence. Finally, we complete the structure graph with material from the individual recordings that was left out by the MSA and annotate the recordings with section types. Figure 1 shows an overview of the process.

3.1 Alignment Graph

Given a collection A of K related audio recordings we obtain a *feature sequence* $A^k = (a_1^k, ..., a_{N_k}^k)$ of length N_k for each recording $k = 1, ..., K$. Each element a_i^k represents a time segment, such as a beat, bar, or onset, with corresponding feature information, such as chroma vectors or chord labels.

A *local alignment* between two such sequences A^s, A^t is commonly defined as a sequence of pairs $p = (p_1, ..., p_L)$ with $p_l = (s_l, t_l) \in [1, ..., N_s] \times [1, ..., N_t]$ with a monotonicity constraint $1 \leq s_1 \leq ... \leq s_{N_s} \leq N_s$ and $1 \leq t_1 \leq ... \leq t_{N_t} \leq N_t$ and a step condition $p_{l+1} - p_l \in \{(0,1),(1,0),(1,1)\}$. A local *self-alignment* can be defined accordingly with $A^s = A^t$.

In our situation it is advantageous to only consider *diagonal local alignments* in order to reduce ambiguity and noise in the alignment graph. We can achieve this with a more strict step condition $p_{l+1} - p_l = (1,1), \forall l = 1, ..., L\text{-}1$.

Due to repetition and variation in the given musical material, many sensible local alignments may exist between each pair of recordings. For example, if in one recording the first of a pair of musical sections is expanded or another is inserted between the two sections, two independent local alignments are still able to capture the commonality between the two recordings. The same is true for self-alignments, which are able to characterize repetition at different temporal intervals within recordings.

From a large number of such alignments and self-alignments we can then create an *alignment graph* $G_A = (N_A, E_A)$ for collection A with a node for each segment a_i^k and an edge between each aligned segment pair p_l. Due to the alignments being local, not every node in the graph is necessarily connected, and some nodes may have many incident edges.

Due to the large size of many audio collections of interest and the time complexity of alignment algorithms, it may not be feasible to calculate the alignments between ev-

ery possible pair of recordings. However, for our method it has proven to be sufficient to select a small subset of all possible pairings, e.g. n random pairings per recording ($n * K$ alignments) plus all K self-alignments.

3.1.1 Alignment and Self-Alignment Methods Used

Common approaches to alignment are usually designed to find a single global or local alignment for a pair of given sequences. For the reasons outlined above we are more interested in finding multiple local diagonal alignments of reasonable length. Many common approaches can be modified for this purpose. Here we discuss the *Smith-Waterman* algorithm, which we use in our experiments.

Smith-Waterman is a dynamic programming algorithm developed in the context of genetic sequence alignment [32]. For two given input sequences A^s, A^t, the simplest variant with a linear gap penalty generates a scoring matrix $H_{i,j}$ with dimension $N_s + 1 \times N_t + 1$ and with $H_{i,0} = H_{0,j} = 0$. Each $H_{i,j}$ with $i, j > 0$ is then determined as follows:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + sim(a_j^s, a_j^t), \\ H_{i-1,j} - P_G, \\ H_{i,j-1} - P_G, \\ 0 \end{cases} \quad (1)$$

where sim is a similarity function between feature vectors and P_G is a gap penalty. [1] Depending on the nature of the feature vectors, one may choose sim to be a simple cosine similarity, or a function that returns a match score for identical vectors and else a lower mismatch score.

Starting from the highest score in the matrix, the algorithm then finds the most likely alignment path by tracing back the origins of the score and ending at a position with a score of 0. Our modification of the algorithm finds diagonal paths by limiting trace-back to matching pairs containing a maximum number of $\gamma \geq 0$ subsequent diagonal gaps (mismatches). With one iteration of Smith-Waterman, several of these paths may be extracted by gradually removing found paths from the matrix and setting their values to 0.
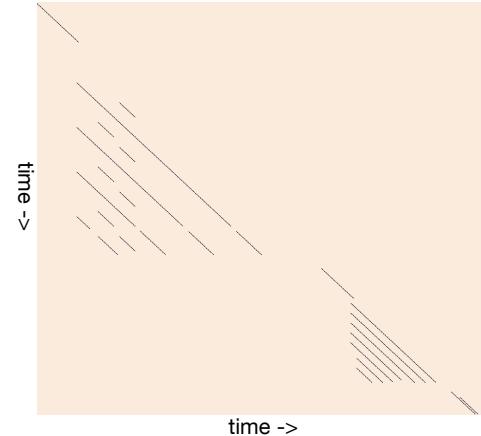
Furthermore, due to the fact that some potential paths can be covered up by higher-rated paths nearby, we propose an *iterative variant* of Smith-Waterman where every element in the neighborhood of a previously found alignment path p is set to zero, i.e. $\forall p_l \in p$ we set $H_{ij} = 0$ for $s_l - \delta \leq i \leq s_l + \delta$ and $t_l - \delta \leq j \leq t_l + \delta$. The parameter $\delta \geq 0$ controls the minimum distance between alignments, which can be used for limiting the number of results. [2]

Additional ways of improving the performance of the algorithm and the quality of the resulting alignments that have proven useful are limiting the number of iterations, setting a score threshold below which alignments are no longer considered, or setting a minimum segment length.

Many of the existing methods can also be modified for *self-alignment*. With Smith-Waterman, we simply treat the

Figure 2. The matrix resulting from a diagonal self-alignment of a recording of *China Doll* from the dataset used in Section 4 (10 longest segments).

trivial diagonal alignment as a previously found path p. Figure 2 shows an example diagonal self-alignment.

3.2 Structure Graph

The next step is to construct a *structure graph* that encapsulates the most common structural characteristics found in the given collection. The goal is to identify a large subgraph G'_A of G_A that can be partitioned into a sequence of partitions $P_1, \ldots, P_M$ of corresponding nodes in G'_A, i.e. $P_m \subset N_A$. Each partition contains at most one segment of each recording, i.e. $k \neq l, \forall a_i^k, a_j^l \in P_m$, and the partitions are strictly ordered temporally, i.e. $i < j, \forall a_i^k \in P_m, a_j^l \in P_{m+1}$. For example, if the nodes of G_A represent bars in A, each partition contains bars of different recordings that can be considered equivalent. Subsequent partitions may be thought of as recurring sequences, although there may be gaps in individual or all recordings between to adjacent partitions. Note that $\cup P_m$ may likely not include all nodes of G_A due to significant structural differences between the individual recordings, hence the definition of G'_A. Figure 3 shows the connection matrix of an example partition where the z-axis shows the number of connections between partition pairs.

We can infer such a partition directly from the connections in G_A in an iterative manner. We experimented with various graph search approaches, e.g. searching for the most densely connected components with at most one node per recording and aligning these components temporally, or with beam search with various partition improvement and modification methods based on functions that rate the clarity of the connection matrix of the partition.

However, while these methods work well for relatively similar input sequences, we obtained better and faster results for more diverse input material with multiple sequence alignments using *Profile HMMs* [24], a method common in genetic sequencing where a reasonable simultaneous alignment is found for all sequences. A Profile HMM is a particular type of Hidden Markov Model with three types of states and a given length L. *Match states* M_j

---

[1] In genetic sequence alignment it is generally more appropriate to replace P_G with a gap penalty function that distinguishes between opening and closing a gap and decreases for longer gaps.

[2] A similar iterative ('recursive') variant was introduced in [19]. However, there each sequence element is involved in at most one pair.

Figure 3. Connection matrix of the partitioned alignment graph of *China Doll* (65 recordings). The z-axis shows the number of connections between the nodes of each partition pair. The beginning of the song features a regularly repeating section (verses and solos), followed by a section that occurs only once but in most versions (interlude and chorus), followed by a short repeated ostinato (outro).
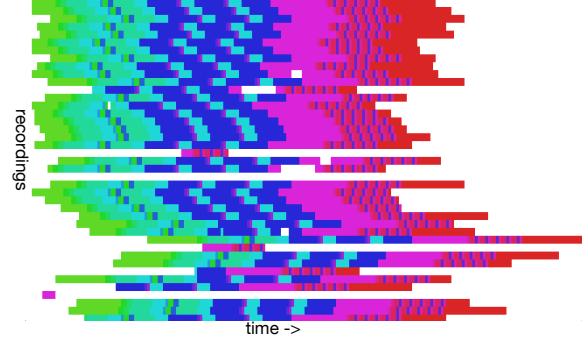
Figure 4. Transition structure of a Profile HMM [24].

represent segments shared between sequences, *insert states* I_j represent possibly multiple consecutive segments particular to an individual sequence, and *delete states* D_j represent segments missing in a particular sequence (the segment represented by the corresponding match state), where $j = 1, \ldots, L$ (see Figure 4).

There are L match and L delete states, as well as $L + 1$ insert states. The emission distributions for the match and insert states are chosen depending on the input feature sequences, e.g. 12-dimensional multivariate gaussian for chroma vectors, or multinomial discrete distributions for chord labels. The model is trained with a number of related sequences, in our case the A^k, using the expectation maximization variant of Baum-Welch. L is usually chosen based on the lengths of the input sequences, e.g. their maximum, median, mean, or minimum length. Individual state sequences for each input sequence can be decoded using the Viterbi algorithm, i.e. we obtain a state label for each a_i^k. We can then infer the M partitions from the segments associated with the L match states ($M \le L$), for example by only keeping the match states that appear in a proportion of at least $0 \le \lambda \le 1$ of all the sequences.

We now define a *structure graph* $G''_A = (N''_A, E''_A)$ whose nodes correspond to the partitions P_m and whose edges are determined by the most common mutual connections in G'_A between the elements of different partitions. More precisely, we add an edge for each node pair $P_m, P_n \in N''_A$ where $P_m \in con(P_n, \mu)$ and $P_n \in$

Figure 5. Juxtaposition of different recordings of *China Doll* where colors represent segment types. The different horizontal offsets illustrate varying lengths of introductory tuning or announcements. The empty lines are mislabeled recordings of different songs.

$con(P_m, \mu)$. The function $con(x, \mu)$ returns the set of $\mu > 0$ nodes in N''_A most strongly connected to x:
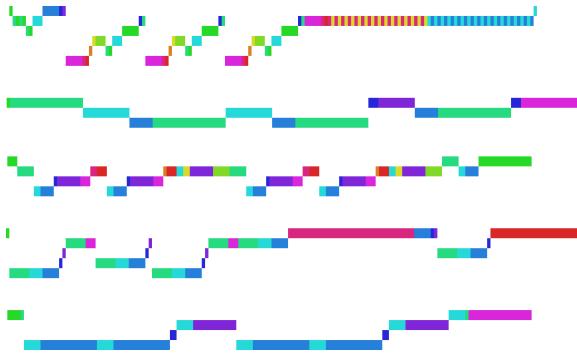
$$con(x, \mu) = \underset{y \in N'_A \setminus x}{\text{argmax}} \; |\{e \in E_A \mid a, b \in e, a \in x, b \in y\}| \\ \mu$$

(2)

where argmax_μ returns the set of μ arguments for which the function is maximized. The parameter μ should be relatively small for best results, e.g. $0 < \mu \le 5$. For illustration, the resulting graph is a pruned simple-graph version of the multigraph of which a connection matrix is shown in Figure 3. The pruned graph contains only nodes representing significantly large partitions are kept and simple edges are established wherever the multigraph has many edges.

3.3 Inference of Section Types and Hierarchies

The structure graph can then automatically be decomposed in order to find *types of sections* recurring in the collection and within single recordings. The connected components C_j in G''_A represent sets of equivalent partitions of segments recurring at different points in time. We sort these components by their lowest partition index $\min_m(P_m \in C_j)$ and group temporally adjacent ones where $P_m \in C_j \iff P_{m+1} \in C_{j+1}$ into sequences. For each of these sequences we can retrieve the corresponding recurrent sections by simply transposing the two-dimensional arrays of indexes, i.e. $(C_{j1}, \ldots, C_{jJ})^T$ with $j1, \ldots, jJ$ being the sorted indexes of the components in a given group. Finally, we merge temporally adjacent groups of sections G, H if for each section in G there is a directly temporally adjacent section in H and vice versa. Figure 5 shows a visualization of the section types thus obtained for the partitioned graph shown in Figure 3.

For music with no recurring sections the above procedure may result in only a few or no section types. We therefore suggest an additional step of inferring boundaries between adjacent connected components C_j, C_{j+1}. Let $\Delta_{j,j+1} = avg_k(n_k - m_k)$ be the average difference in index over all recordings k appearing C_j, C_{j+1}, where m_k, n_k are the indexes of the segments of k, i.e. $a_{m_k}^k \in C_j, a_{n_k}^k \in C_{j+1}$. If $\Delta_{j,j+1} \ge \tau$ for a given thresh-

Figure 6. Visualization of the structural hierarchies of five different songs from the dataset from Section 4.1. The top-most is the unflattened hierarchy of *China Doll* (Figures 3 and 5), all others are flattened. The colors indicate section types, the vertical axis nesting level.

old $\tau > 1$, we introduce a section boundary between C_j and C_{j+1}. For example, for $\tau = 5$ we add a section boundary between any two subsequent components where there are on average 4 segments missing per recording.

Finally, we can simplify the structure by inferring a *hierarchy* from the obtained section types. This can be done using a simple recursive search method that identifies the most frequently recurring adjacent section types and either combines them into a new type or concatenates them if they always co-occur. This hierarchy can then be simplified by further merging adjacent types that always appear together, and finally flattening nested types if their parts only occur within them. Figure 6 shows a few visualizations of example hierarchical structures.

3.4 Annotation of Individual Structures

In a final step of the process, each individual recording in the collection can be annotated using the found shared structure. First, we label each segment in the structure graph with a corresponding section type identified as described in the previous section. Then, using the alignment graph G_A we can infer section types for segments that are not in the structure graph, which may for example be the case if some recordings contain additional repetitions of sections. We consider for each segment a_i^k in $G_A \setminus G'_A$ and check which partition P_m in G''_A it is most strongly connected to, i.e. which partition contains the most segments connected to a_i^k. Note that some segments, sections, or entire recordings may remain unlabeled, if they were not aligned or self-aligned in the first step of the process (Section 3.1), due to being entirely unrelated or their features being too noisy (see Figure 5 for some examples).

Finally, we can annotate each segment that received a section type with a *feature value* derived from the shared structure. We determine a value for each position of every section type by summarizing the original features of all segments associated with that position. For example, we may label the first beat of a section type with the chord label most frequently occurring among all associated beats.

All corresponding segments in individual recordings can then be annotated with that label.

4. EXPERIMENTS

We tested our approach on material from the Grateful Dead collection of the Live Music Archive,[3] which holds more than 13,000 recordings of over 2,000 shows spanning the years 1965 to 1995. The large number of recordings of individual songs and the improvised nature of Grateful Dead's performances make this collection particularly interesting for our work.

4.1 Dataset and Preparations

We created a *dataset*[4] with all performed versions of 15 songs from this collection, selected based on the criteria that a large number of versions exist and that a corresponding studio recording by the Grateful Dead is available that could potentially be used as a reference in the future. The fact that these recordings are live recordings poses additional challenges to the ones outlined in Section 2. Many of them contain a considerable amount of crowd noise which may lead to noisy audio features, and most of these recordings were made by amateurs using their analog tape equipment, which means that many of them are out of tune due to varying tape speed. We addressed the second of these problems and *resampled* the audio files after comparing their rotated chroma features with the ones of the respective studio version. For a ground truth, we *transcribed* the chord progressions beat-by-beat and grouped them into bars and sections for each of the songs with the help of existing lead sheets.[5] This level of granularity is particularly important due to the fact that many of these songs are based on odd meters (e.g. 7/4 in *Estimated Prophet*) or contain metrical changes (e.g. abbreviated 2/4 bars in *China Cat Sunflower*). Tuning ratios, a script for downloading and resampling, and transcriptions are published with the dataset.

The experiments described here[6] are based on a *subset* of the dataset with at most 100 versions of every song. We extracted triadic chord features using [33] (root notes and one of the four qualities major, minor, diminished, and augmented) and summarized them to beats extracted using Madmom.[7] The summarization process is based on the statistical mode of the chords in each temporal segment, i.e. for each segment the chord that was played for the longest. In order to be comparable with the features, we simplified the transcribed chords to triads as well.

We used our own implementation of a Profile HMM and initialized it with L = median input length and with uniform distributions and transition probabilities, except match-match 0.999 and delete-insert 0.01. Our Smith Waterman implementation led to the quickest and best results

---

[3] https://archive.org/details/GratefulDead
[4] https://github.com/grateful-dead-live/fifteen-songs-dataset
[5] e.g. at http://jdarks.com/GDTab.html
[6] Code available at https://github.com/florianthalmann/ismir2020-shared-structure
[7] https://madmom.readthedocs.io
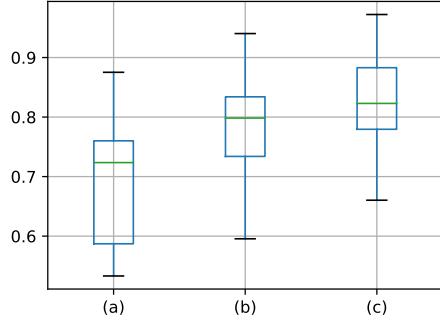
	(a) baseline	(b) annotated	(c) shared
p_G	0.691	0.779	0.825
p_O	0.411	0.461	0.482

Table 1. Proportion of matched chords in groundtruth and output for baseline (extracted chords), annotated recordings, and shared harmonic structure (averaged over all recordings in the case of baseline and annotated).

Figure 7. Distributions of average proportions of matches in ground truth p_G per song. (a) baseline, (b) annotated versions, (c) shared structure.

with the following parameter settings: a single iteration, 10 longest alignment segments, minimum segment length 16, $\gamma = 4$, $\delta = 4$, $\lambda = .1$, $\mu = 1$, $\tau = 2$.
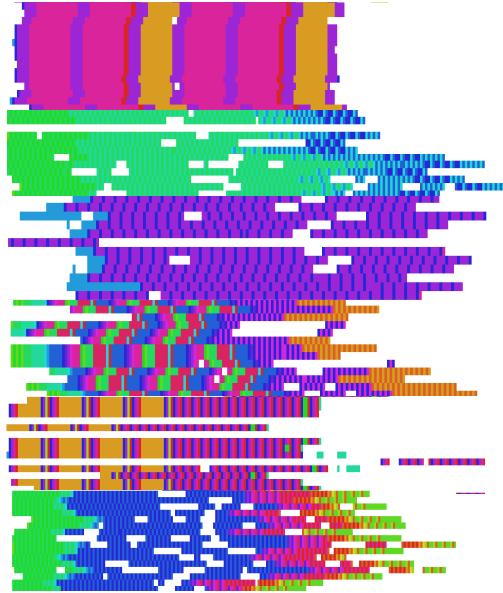
4.2 Results

Due to the fact that there is no previous work with which to compare our method, we chose to perform an evaluation similar to [31] where structural information is used to improve chord prediction accuracy. [8] We used Smith Waterman to align the following sets of sequences with the ground truth transcriptions: (a) the original extracted chord sequences as a baseline (b) the sequences annotated by our method according to Section 3.4 and (c) the shared harmonic structure identified by our method according to Section 3.3. We then calculated two measures for each of the sets of sequences: the proportion of correctly matched ground truth segments p_G as well as the proportion of correctly matched segments in the output p_O, i.e.

$$p_G = \frac{\text{matches in alignment}}{\text{length of groundtruth}}, \ p_O = \frac{\text{matches in alignment}}{\text{length of output}} \tag{3}$$

Table 1 shows the overall values and Figure 7 shows distributions of p_G per song. p_O is lower than p_G due to for example additional repetitions of sections in performances or the high degree of variation and improvisation in many of the songs, i.e. there are deviations from the 'lead sheet' in individual recordings. However, the fact that on average the shared harmonic structure matched with the lead sheet content with an average probability of 82.5% is promising.

---

[8] Note that instead of evaluating the hierarchical structures, which necessitates a non-trivial generalization of the method suggested in [34] and will be done in future work, we evaluate the flattened annotations, which nevertheless result from the process described in sections 3.1 through 3.4.

Figure 8. Visualisation of the segment types for 6 songs (*Box of Rain*, *Eyes of the World*, *Franklin's Tower*, *Sugar Magnolia*, *Casey Jones*, and *China Cat Sunflower*), around ten recordings each.

4.3 Application

As a more qualitative investigation of the potential of our approach we created a simple Web application for the interactive exploration of annotations and alignments along with the underlying recordings. Users can hear the corresponding segments of the recordings by clicking on the colored blocks. Figure 8 compiles six screenshots for different songs that illustrate different shared structures and individual deviations from them. Whereas *Box of Rain* has a very even and simple $AABAAB$ structure with tiny insertions, other songs, such as *Eyes of the World* or *Franklin's Tower*, feature longer more open-ended yet highly repetitive sections. *Casey Jones* is a combination of both with four verse/chorus repetitions followed by an extended jam over part of the chorus.

5. CONCLUSION

We have presented a new method for the extraction of shared temporal structure from a number of related audio recordings and shown with both quantitative and qualitative results how such a method could be useful. For example, it could provide musicologists a way to systematically study and explore larger archives of related recordings, or yield more reliable estimates of audio features for noisy live music recordings. Besides a more extensive evaluation and application, future work could include an expansion of the method for joint use of different kinds of feature vectors, either to improve the inference of sections and hierarchies for less repetition-based music (analogous to other approaches to structure inference) or for a more multidimensional analysis of the musical material.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] J. Paulus, M. Müller, and A. Klapuri, "State of the art report: Audio-based music structure analysis." in *IS-MIR*. Utrecht, 2010, pp. 625–636.

[2] M. Müller, "Music structure analysis," in *Fundamentals of Music Processing*. Springer, 2015, pp. 167–236.

[3] M. Giraud, R. Groult, and F. Levé, "Computational analysis of musical form," in *Computational Music Analysis*. Springer, 2016, pp. 113–136.

[4] N. Collins, "The ubuweb electronic music corpus: an mir investigation of a historical database," *Organised Sound*, vol. 20, no. 1, pp. 122–134, 2015.

[5] M. Mauch, R. M. MacCallum, M. Levy, and A. M. Leroi, "The evolution of popular music: Usa 1960–2010," *Royal Society open science*, vol. 2, no. 5, p. 150081, 2015.

[6] K. R. Page, S. Bechhofer, G. Fazekas, D. M. Weigl, and T. Wilmering, "Realising a layered digital library: exploration and analysis of the live music archive through linked data," in *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries*. IEEE Press, 2017, pp. 89–98.

[7] A. Porter, M. Sordo, and X. Serra, "Dunya: A system for browsing audio music collections exploiting cultural context," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013.

[8] S. Abdallah, E. Benetos, N. Gold, S. Hargreaves, T. Weyde, and D. Wolff, "The digital music lab: A big data infrastructure for digital musicology," *Journal on Computing and Cultural Heritage (JOCCH) - Special Issue on Digital Infrastructure for Cultural Heritage, Part 1*, vol. 10, no. 1, 2017.

[9] A. Allik, F. Thalmann, and M. Sandler, "Musiclynx: Exploring music through artist similarity graphs," *WWW '18 Companion Proceedings of the The Web Conference 2018, Geneva, Switzerland*, 2018.

[10] O. Lartillot, "Efficient extraction of closed motivic patterns in multi-dimensional symbolic representations of music," in *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*. IEEE, 2005, pp. 229–235.

[11] D. Temperley and T. d. Clercq, "Statistical analysis of harmony and melody in rock music," *Journal of New Music Research*, vol. 42, no. 3, pp. 187–204, 2013.

[12] D. Meredith, "Analysing music with point-set compression algorithms," in *Computational Music Analysis*. Springer, 2016, pp. 335–366.

[13] T. Collins, S. Böck, F. Krebs, and G. Widmer, "Bridging the audio-symbolic gap: The discovery of repeated note content directly from polyphonic music audio," in *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.

[14] M. Barthet, M. D. Plumbley, A. Kachkaev, J. Dykes, D. Wolff, and T. Weyde, "Big chord data extraction and mining," in *Proceedings of the 9th Conference on Interdisciplinary Musicology – CIM14*, 2014.

[15] J. Van Balen, J. A. Burgoyne, D. Bountouridis, D. Müllensiefen, and R. C. Veltkamp, "Corpus analysis tools for computational hook discovery," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

[16] B. Sturm, "An analysis of the GTZAN music genre dataset." *Proceedings of the 2nd international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, 2012.

[17] T. Wilmering, G. Fazekas, S. Dixon, K. Page, and S. Bechhofer, "Towards high level feature extraction from large live music recording archives," *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML), 6-11 July, Lille, France*, 2015.

[18] S. Dixon and G. Widmer, "Match: A music alignment tool chest." in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 492–497.

[19] S. Ewert, M. Muller, V. Konz, D. Mullensiefen, and G. A. Wiggins, "Towards cross-version harmonic analysis of music," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 770–782, 2012.

[20] S. Wang, S. Ewert, and S. Dixon, "Robust and efficient joint alignment of multiple musical performances," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 2132–2145, 2016.

[21] M. Grachten, M. Gasser, A. Arzt, and G. Widmer, "Automatic alignment of music performances with structural differences," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013.

[22] J. Serra, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 6, pp. 1138–1151, 2008.

[23] C. Dittmar, K. F. Hildebrand, D. Gärtner, M. Winges, F. Müller, and P. Aichroth, "Audio forensics meets music information retrieval—a toolbox for inspection of music plagiarism," in *2012 Proceedings of the 20th European signal processing conference (EUSIPCO)*. IEEE, 2012, pp. 1249–1253.

[24] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[25] P. Knees, M. Schedl, and G. Widmer, "Multiple lyrics alignment: Automatic retrieval of song lyrics." in *IS-MIR*. Citeseer, 2005, pp. 564–569.

[26] P. Esling and M. G. Bergomi, "Molecular clock synthesis," IRCAM, http://repmus.ircam.fr/esling/projet-atiam-2014.html, Tech. Rep., 2015.

[27] M. G. Bergomi, "Dynamical and topological tools for (modern) music analysis," Ph.D. dissertation, Universià degli Studi di Milano; Universié Pierre et Marie Curie, 2015.

[28] M. Müller and F. Kurth, "Towards structural analysis of audio recordings in the presence of musical variations," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, p. 089686, 2006.

[29] G. Shibata, R. Nishikimi, E. Nakamura, and K. Yoshii, "Statistical music structure analysis based on a homogeneity-, repetitiveness-, and regularity-aware hierarchical hidden semi-markov model," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[30] B. McFee and D. Ellis, "Analyzing song structure with spectral clustering." in *ISMIR*, 2014, pp. 405–410.

[31] M. Mauch, K. C. Noland, and S. Dixon, "Using musical structure to enhance automatic chord transcription." in *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, 2009, pp. 231–236.

[32] T. F. Smith, M. S. Waterman *et al.*, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[33] Y. Wu, T. Carsault, and K. Yoshii, "Automatic chord estimation based on a frame-wise convolutional recurrent neural network with non-aligned annotations," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[34] B. McFee, O. Nieto, M. M. Farbood, and J. P. Bello, "Evaluating hierarchical structure in music annotations," *Frontiers in psychology*, vol. 8, p. 1337, 2017.

A CHORUS-SECTION DETECTION METHOD FOR LYRICS TEXT

Kento Watanabe   **Masataka Goto**

National Institute of Advanced Industrial Science and Technology (AIST), Japan

{kento.watanabe, m.goto}@aist.go.jp

ABSTRACT

This paper addresses the novel task of detecting chorus sections in English and Japanese lyrics text. Although chorus-section detection using audio signals has been studied, whether chorus sections can be detected from text-only lyrics is an open issue. Another open issue is whether patterns of repeating lyric lines such as those appearing in chorus sections depend on language. To investigate these issues, we propose a neural network-based model for sequence labeling. It can learn phrase repetition and linguistic features to detect chorus sections in lyrics text. It is, however, difficult to train this model since there was no dataset of lyrics with chorus-section annotations as there was no prior work on this task. We therefore generate a large amount of training data with such annotations by leveraging pairs of musical audio signals and their corresponding manually time-aligned lyrics; we first automatically detect chorus sections from the audio signals and then use their temporal positions to transfer them to the line-level chorus-section annotations for the lyrics. Experimental results show that the proposed model with the generated data contributes to detecting the chorus sections, that the model trained on Japanese lyrics can detect chorus sections surprisingly well in English lyrics and that patterns of repeating lyric lines are language-independent.

1. INTRODUCTION

The digitization of lyrics collections has opened various areas of lyrics-based research in the Music Information Retrieval (MIR) community, such as research on lyrics browsing [1–3], lyrics genre classification [4–6] and lyrics-to-audio synchronization [7–17]. Lyrics are usually plain text without any annotations, and some researchers have analyzed their structure, such as paragraph structure and topic transitions between paragraphs [18–22]. For example, Fell et al. [18] and Watanabe et al. [19] estimated section boundaries in lyrics text without empty lines but were not able to assign a section label such as verse or chorus to each estimated section. Chorus sections were not detected in lyrics text.

The goal of this paper is to achieve automatic chorus-section detection for lyrics text. This task has not been studied, though chorus-section detection, as well as music structure analysis, for audio signals has been a popular topic of research in the MIR community [23–41]. Since whether chorus sections can be detected from text-only lyrics is an open issue, it is worth investigating this issue from academic viewpoints. Moreover, a chorus-section detection method for lyrics text has potential applications. For example, when listeners want to find lyrics with a chorus section having a particular phrase such as "I love you" for the purpose of singing, reusing its chorus section in a short video clip, etc., it is necessary for a lyrics search system to automatically detect which lines of the lyrics are included in chorus sections. The detected lyric lines of chorus sections could be used in a lyrics viewing function of music services displaying lyrics with those lines highlighted by a different color or typeface. Automatic lyric video generation technologies could give those lines more vivid animations.
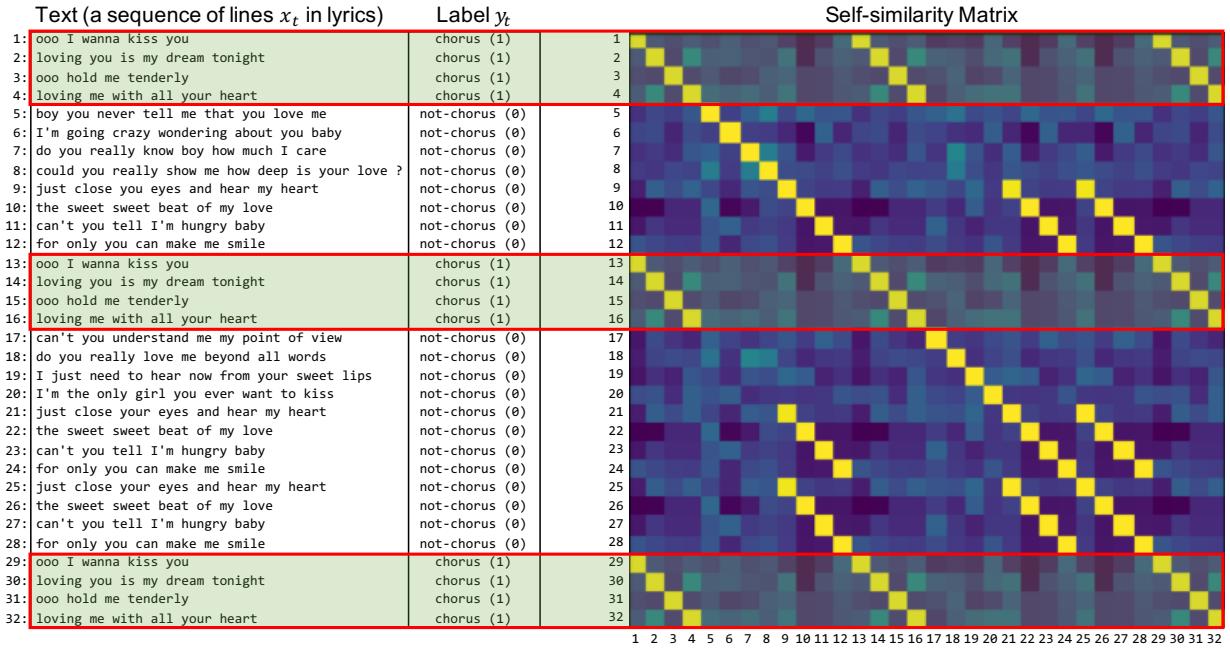
Chorus sections are the most repeated and memorable portions of a song [39]. Since it is not easy to explore heuristic rules to find such sections, most existing chorus-section detection methods for audio signals have leveraged repetitive patterns of those sections within a song. In this paper, we propose a supervised model that can detect chorus sections in English and Japanese lyrics. Our model uses both structural features that represent patterns of repeating lyric lines and linguistic features that are calculated from word2vec [42] and context2vec [43]. To detect chorus sections using only plain text without any labels or even empty lines (i.e., section boundaries), we investigate a model and features effective for chorus-section detection. Experimental results show that our proposed model outperforms alternative baseline models and that combining structural and linguistic features contributes to better performance.

Although such a supervised model needs a large dataset of lyrics with line-level chorus-section annotations for its training, there was no such dataset as there was no prior work on chorus sections in lyrics text. To address this issue of lacking training data, we generated a dataset consisting of 9,313 English and 91,459 Japanese lyrics with chorus-section annotations by utilizing pairs of musical audio signals and their corresponding manually time-aligned lyrics. We first automatically detected chorus sections in audio signals of a song [39]. Then, since each lyric line had the corresponding start time within the song, we could find lyric lines that temporally correspond to the duration of each detected chorus section. We thus obtained the annotated dataset by assigning a `chorus` label to those lyric

Text (a sequence of lines x_t in lyrics)	Label y_t		Self-similarity Matrix
1: ooo I wanna kiss you	chorus (1)	1	
2: loving you is my dream tonight	chorus (1)	2	
3: ooo hold me tenderly	chorus (1)	3	
4: loving me with all your heart	chorus (1)	4	
5: boy you never tell me that you love me	not-chorus (0)	5	
6: I'm going crazy wondering about you baby	not-chorus (0)	6	
7: do you really know boy how much I care	not-chorus (0)	7	
8: could you really show me how deep is your love ?	not-chorus (0)	8	
9: just close you eyes and hear my heart	not-chorus (0)	9	
10: the sweet sweet beat of my love	not-chorus (0)	10	
11: can't you tell I'm hungry baby	not-chorus (0)	11	
12: for only you can make me smile	not-chorus (0)	12	
13: ooo I wanna kiss you	chorus (1)	13	
14: loving you is my dream tonight	chorus (1)	14	
15: ooo hold me tenderly	chorus (1)	15	
16: loving me with all your heart	chorus (1)	16	
17: can't you understand me my point of view	not-chorus (0)	17	
18: do you really love me beyond all words	not-chorus (0)	18	
19: I just need to hear now from your sweet lips	not-chorus (0)	19	
20: I'm the only girl you ever want to kiss	not-chorus (0)	20	
21: just close your eyes and hear my heart	not-chorus (0)	21	
22: the sweet sweet beat of my love	not-chorus (0)	22	
23: can't you tell I'm hungry baby	not-chorus (0)	23	
24: for only you can make me smile	not-chorus (0)	24	
25: just close your eyes and hear my heart	not-chorus (0)	25	
26: the sweet sweet beat of my love	not-chorus (0)	26	
27: can't you tell I'm hungry baby	not-chorus (0)	27	
28: for only you can make me smile	not-chorus (0)	28	
29: ooo I wanna kiss you	chorus (1)	29	
30: loving you is my dream tonight	chorus (1)	30	
31: ooo hold me tenderly	chorus (1)	31	
32: loving me with all your heart	chorus (1)	32	

Figure 1. Example of lyrics with chorus-section annotations and corresponding self-similarity matrix in which each cell represents the similarity between two lyric lines. These lyrics are from "How Deep Is Your Love?" (RWC-MDB-P-2001 No. 81 in the RWC Music Database [44]).

lines and a `not-chorus` label to the other lines. Experimental results show that the model trained with this large automatically generated dataset performs better than the model trained with a smaller manually annotated dataset and that the model trained on Japanese lyrics can detect chorus sections surprisingly well in English lyrics.

2. LYRICS CHORUS-SECTION DETECTION TASK

The left side of Figure 1 shows an example of lyrics with chorus-section annotations (labels). The lyrics of a song are a sequence of lyric lines, each line having a sentence or phrase. In this example there are three highlighted chorus sections that have exactly the same four lines, though in other songs, lyrics of chorus sections are repeated with some modifications. To maximize the applicability, as shown in this example, we assume that the input text of lyrics does not have any section boundaries. Even though some lyrics contain empty lines at those boundaries, those lines are deleted in advance. We also assume that the input text does not have explicit chorus labels such as "(chorus)" at the beginnings of chorus sections. Even though some lyrics contain those labels, they are deleted as well. When lyrics contain a repetition label such as "(* repeat)", it is manually replaced with the corresponding lyric lines.

We formulate this chorus-section detection task as a sequence labeling problem: predicting the `chorus` or `not-chorus` status for each lyric line. Let X_s be the lyrics of a song s composed of T lines of text: $X_s = \{x_1, ..., x_t, ..., x_T\}$. Each lyric line x_t has a binary label y_t. If $y_t = 1$, x_t is in a chorus section. If $y_t = 0$, x_t is not in a chorus section. Y_s denotes a sequence of labels correspond-

ing to X_s: $Y_s = \{y_1, ..., y_t, ..., y_T\}$. In the training step, the model learns the conditional probability $P(Y_s|X_s)$. In the validation/testing step, the trained model has to predict labels Y_s for given lyric lines X_s.

Chorus sections cannot be detected by simply extracting repeated lines since those lines often correspond to non-chorus sections. For example, lyric lines 9–12 and 21–24 in Figure 1 are exactly repeated, but those lines are not in chorus sections. It is also difficult to manually define a set of rules to find various chorus sections. We therefore prepare various features that could be useful for machine learning to deal with various types of chorus sections.

3. COMPUTATIONAL MODELING OF CHORUS SECTIONS IN LYRICS

We propose a neural network-based model for sequence labeling by using structural features that are self-similarity matrix (SSM) representations. SSM representations are widely used in computational music structure analysis, but we use different representations for lyrics. In addition to structural features, our model utilizes linguistic features such as word vectors and sentence vectors calculated from word2vec [42] and context2vec [43], which are widely used in natural language processing.

In the following sections, we first describe nine SSMs for capturing patterns of repeating lyric lines and explain how to encode the SSMs for neural networks (Section 3.1). We then describe the linguistic features obtained by vectorizing the semantic/syntactic information of lines using word2vec and context2vec (Section 3.2). Finally, we describe a neural-network-based sequence labeling model with these structural and linguistic features (Section 3.3).

3.1 Structural Features

Most previous work on music structure analysis for audio signals [23–41] identifies repeated musical sections by using a SSM like that shown in Figure 1. Repeated sections lead to high values in diagonals of the matrix, and those patterns are used to identify the structure. To capture repeated lyric lines that often appear in chorus sections, we also compute the SSM from lyrics text, but the design of the similarity measure to compute each cell of the SSM is important. We propose to use the following nine variations of similarity measures sim_m, where m denotes the variation. Some of the similarities are based on previous studies [18, 19].

String similarity (sim_{str}): a normalized Levenshtein edit distance [45] between the characters of two lyric lines.

Head similarity (sim_{head}): a normalized Levenshtein edit distance between the characters of the first two words of two lyric lines.

Tail similarity (sim_{tail}): a normalized Levenshtein edit distance between the characters of the last two words of two lyric lines.
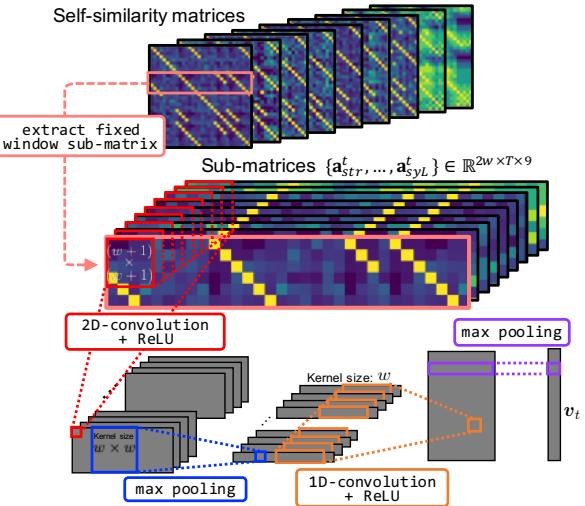
Phonetic similarity (sim_{phone}): To capture rhymes in the lyrics, we calculate a normalized Levenshtein edit distance between the phonetic transcriptions of two lyric lines. We use the CMU pronunciation dictionary [1] to extract the phonetic transcription. For example, the phonetic transcription of "I love you" is [AY1, L, AH1, V, Y, UW1].

Part-of-speech similarity (sim_{pos}): To capture similarities in grammatical structure, we calculate a normalized Levenshtein edit distance between the part-of-speech (POS) sequences of two lines. We use the default POS tagger in the NLTK package [46].

Word vector similarity (sim_{w2v}): To capture the semantic similarity between two lyric lines, we simply average vectors of the words of each lyric line by using pre-trained word2vec [42] and compute their cosine similarity. This "bag of words" representation does not differentiate "dog bites person" from "person bites dog".

Context vector similarity (sim_{c2v}): To consider the word order, we vectorize the lyric lines using pre-trained context2vec [43], an extension of word2vec, which encodes a sequence of words by using Long Short-Term Memory (LSTM) networks [47]. We then compute their cosine similarity to obtain sim_{c2v}.

Word syllable count similarity (sim_{syW}): Since repeated phrases sometimes have the same number of syllables even if their words are different, we use a sequence of word syllable counts on each lyric line. For example, the word syllable counts of the two lyric lines *"Sometimes you lost yourself away"* and *"Everytime you just close your eyes"* [2] are {2, 1, 1, 2, 1} and {2, 1, 1, 1, 1, 1}, respectively. When successive lyric lines have similar syllable count sequences, they are likely to correspond to the repetition of sections. We use dynamic time warping (DTW) [48] to calculate the similarity between syllable count sequences.

Figure 2. Convolutional neural network for SSMs.

Lyric Line syllable count similarity (sim_{syL}): We can also use the total syllable count of all words in each lyric line. For example, in all the chorus sections shown in Figure 1, the total syllable count of the first lyric line is 6 and that of the second line is 8. We calculate the similarity of such total syllable counts of each pair of lyric lines by using the following procedure. (1) We extract a window of four lyric lines $L_t = \{x_t, x_{t+1}, x_{t+2}, x_{t+3}\}$ and shift it over the entire lyrics of a song. (2) The similarity between the lyric lines x_t and $x_{t'}$ is calculated by DTW of L_t and $L_{t'}$.

We thus calculated nine SSMs $\mathbf{A}_m \in \mathbb{R}^{T \times T}$, where each cell is a sim_m explained above. Then, to calculate feature vectors from the above nine SSMs, we exploit a convolutional neural network (CNN) architecture to detect textual macro structures from various patterns in SSMs regardless of their locations and relative sizes in SSMs. Except for network parameters, this CNN architecture is the same as that of Fell et al. [18], as we share the same motivation: to extract translation, scaling and rotation invariant features from the input image (in our case, nine SSMs).

Figure 2 illustrates the CNN structure. After calculating the nine SSMs, we extract fixed-size elongated-rectangle sub-matrices centered on the target lyric line: $\mathbf{a}_m^t = \mathbf{A}_m[t - w + 1, ..., t + w; 1, ..., T] \in \mathbb{R}^{2w \times T}$, where w is a fixed window size. The input of the CNN is nine sub-matrices $\{\mathbf{a}_{str}^t, ..., \mathbf{a}_{syL}^t\} \in \mathbb{R}^{2w \times T \times 9}$, where the number of channels corresponds to the number of SSMs. The kernel size of the first 2D-convolutional layer is $(w + 1) \times (w + 1)$ so that each feature can capture a prospective chorus section. Each resulting tensor is downsampled by max-pooling with $w \times w$ kernel size. We then apply the 1D-convolutional layer with a kernel size of w and the last max-pooling layer downsamples the resulting vector to a scalar. In this network, all convolutional layers employ the ReLU function. We can perform the above procedure independently for each lyric line x_t and obtain the CNN-based feature vector $\boldsymbol{v}_t$.

3.2 Linguistic Features

Some expressions tend to appear in chorus sections. To quantify this tendency, we calculate the difference between

Tri-gram	$P_c - P_n$	Tri-gram	$P_n - P_c$
I'm	0.12%	there's	0.04%
don't	0.11%	I've	0.03%
oh oh oh	0.05%	's a	0.03%
I'll	0.05%	I'd	0.02%
we're	0.04%	but I'	0.02%
you're	0.04%	's not	0.01%
'll be	0.04%	what's	0.01%
I don'	0.04%	na na na	0.01%
Let's	0.03%	yeah yeah yeah	0.01%
you got ta	0.03%	've been	0.01%
I can'	0.03%	't take	0.01%
can't	0.03%	didn't	0.01%

Table 1. Frequent word tri-grams in chorus and non-chorus sections. An apostrophe is regarded as a word.

word tri-gram probabilities in the chorus and non-chorus sections. Table 1 shows the word tri-grams that frequently appear in both of the sections. Here, P_c and P_n denote word tri-gram probabilities in the chorus and non-chorus sections, respectively. As shown in this table, we found that phrases about the future (e.g., *"I'll"* and *"Let's"*) tend to appear in chorus sections more often than do phrases about the past (e.g., *"have been"* and *"didn't"*). To exploit such tendencies, we propose two linguistic features.

Average of word vectors ($ling_{ave}$): We use the average of word vectors of a given lyric line as a feature. The word vectors are obtained using pre-trained word2vec [42], skipping out-of-vocabulary words.

Vector representations for word sequences ($ling_{seq}$): To consider the word order that cannot be modeled by $ling_{ave}$, we use pre-trained context2vec [43] that enables vectorization of a lyric line by putting a sequence of word vectors into the LSTM.

We calculate the above linguistic feature vectors for each lyric line x_t and obtain their concatenated vector $\boldsymbol{u}_t$.

3.3 Neural Network-based Sequence Labeling Model

To solve the sequence labeling problem, we use the standard Bidirectional Long Short-Term Memory (Bi-LSTM) networks [49] to compute the conditional probability $P(Y_s|X_s)$. The neural network structure is illustrated in Figure 3.
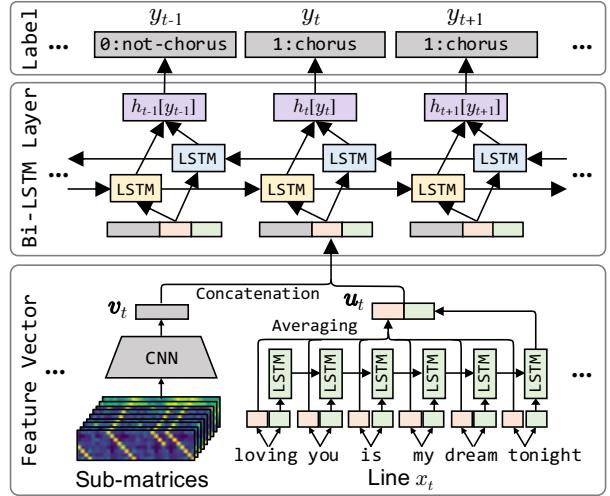
The input to the Bi-LSTM layer at each time step t (lyric line) is a concatenation of two different types of feature vectors: (1) structural feature vectors $\boldsymbol{v}_t$ encoded from nine variations of SSMs in Section 3.1 and (2) linguistic feature vectors $\boldsymbol{v}_t$ encoded in Section 3.2. Formally, the conditional probability $P(Y_s|X_s)$ is calculated by using a softmax function:

$$P(Y_s|X_s) = \frac{\exp(\text{Score}(X_s, Y_s))}{\sum_{Y_s'} \exp(\text{Score}(X_s, Y_s'))}. \quad (1)$$

The Score() is defined as

$$\text{Score}(X_s, Y_s) = \sum_t \text{BN}(h_t[y_t]), \quad (2)$$

where $h_t[y_t]$ is the output of the Bi-LSTM for each time step t and BN() denotes batch normalization [50]. In the model training step, we use a binary cross-entropy loss.

Figure 3. Neural-network-based sequence labeling model for chorus-section detection.

4. EXPERIMENT

Inspired by audio-based chorus-section detection [39], we evaluated the proposed method by using the F-measure (F) that is a harmonic mean of precision (P) and recall (R), $F = (2 \cdot R \cdot P)/(R + P)$, where

$$P = \frac{\text{\# of lyric lines in correctly detected chorus sections}}{\text{\# of lyric lines in detected chorus sections}}.$$

$$R = \frac{\text{\# of lyric lines in correctly detected chorus sections}}{\text{\# of lyric lines in correct (annotated) chorus sections}}.$$

We also used the pair-wise F-measure (p-F), normalized conditional entropy F-measure (n-F) and V-measure (V) that are provided by the Python module `mir_eval` and commonly used to evaluate computational music structure analysis [51].

4.1 Methods Compared

To confirm the effectiveness of our Bi-LSTM method based on the Bi-LSTM model that can learn dependencies between adjacent lyric lines, we compared its performance with that of with two baseline methods:

(1) Heuristic: We implemented the heuristic that "if lines at the end of the lyrics are repeated with small modifications, all those repeated lines are chorus sections" by the following procedure: (i) From the SSM that is the average of the nine SSMs, we extracted diagonals whose cells had values higher than a threshold λ, which was tuned on a development set to be $\lambda = 0.62$. (ii) From the extracted diagonals, we selected the shortest diagonal among diagonals placed at the bottom of the SSM (e.g., the diagonal starting at the cell $\text{SSM}[29; 1]$ in Figure 1). (iii) Successive lines corresponding to the rows where the selected diagonal was located (e.g., lyric lines 29–32 in Figure 1) were assigned the label `chorus`. (iv) Other successive lines that were similar to the `chorus` lines (e.g., lyric lines 1–4 and 13–16 in Figure 1) were also assigned `chorus` labels.

(2) Multi-Layer Perceptron (MLP): Like the Bi-LSTM method, but with the Bi-LSTM model replaced by a standard MLP model. This method ignores transitions between adjacent lyric lines and predicts y_t from x_t only.

We chose the number of kernels for the first and second CNNs to be 200 and 400, respectively. We used $w = 3$ for the window size. In the MLP and Bi-LSTM methods, we chose the dimension of the hidden state to be 600. The word2vec [42] and context2vec [43] were pre-trained on lyrics and were not updated in the model training step of our method. The dimension of their output vectors was 300. We used AdamW for parameter optimization [52]. The initial learning rate was 0.001 with an exponential decay. We used a mini-batch size of 64. Training was run for 100 epochs, and the model used for testing was the one that achieved the best F-measure on the development set.

4.2 Dataset

To train our computational model that predicts whether the label of each lyric line is `chorus` or `not-chorus`, we needed a large amount of lyrics data with line-level chorus-section annotations like those illustrated in Figure 1. Since there was no dataset for this, we generated a large amount of such lyrics data by the following procedure:

(1) We prepared 100,772 pairs of musical audio signals and their corresponding manually time-aligned (temporally synchronized) lyrics [3] . To avoid unreliable lyrics, we confirmed that all lyrics had more than eight lines and less than 120 lines.

(2) We detected chorus sections of every song automatically by using its audio signals. In our experiments, we used the RefraiD method [39] to obtain the start and end times of each chorus section, but other methods could also be used.

(3) If the start time of a lyric line was within any chorus section detected in audio signals, that line was labeled `chorus`; otherwise, it was labeled `not-chorus`.

Of course, not all generated annotations were correct, but by using over 100,000 training data, the model could be robustly trained without being influenced by errors or outliers. The generated training data consisted of 9,313 English and 91,459 Japanese songs, and we called them EN_auto and JA_auto, respectively. [4]

Furthermore, we manually annotated three sets of lyrics data with more reliable line-level chorus-section annotations for three different purposes:

(a) For training comparison: We annotated 1,103 Japanese lyrics and called them JA_man [5] . By comparing the performance of the model trained on JA_auto with that of the model trained on JA_man, we could confirm that our generated data is reliable enough for training purposes.

(b) For tuning model parameters: We annotated the lyrics of 21 English and 79 Japanese songs from RWC-MDB-P-2001 and called them EN_RWC and JA_RWC,

| Method | Training data / Testing data | | | | | | | |
| | EN_auto / EN_test | | | | JA_auto / JA_test | | | |
	F	p-F	n-F	V	F	p-F	n-F	V
Heuristic	57.8	73.8	43.0	35.8	57.1	73.2	43.6	36.3
MLP	74.2	76.8	47.7	43.0	80.6	82.8	62.6	59.1
Bi-LSTM	**78.1**	**77.7**	**50.8**	**47.3**	**83.4**	**83.5**	**64.9**	**61.4**

Table 2. Experimental result: Comparison of different methods (the unit is %).

| Feature | Training data / Testing data | | | | | | | |
| | EN_auto / EN_test | | | | JA_auto / JA_test | | | |
	F	p-F	n-F	V	F	p-F	n-F	V
sim_{all}	77.9	76.1	48.6	45.5	81.2	82.7	63.6	59.6
$ling_{all}$	57.4	59.9	16.5	6.9	55.2	61.8	22.1	16.7
both	**78.1**	**77.7**	**50.8**	**47.3**	**83.4**	**83.5**	**64.9**	**61.4**

Table 3. Experimental result: Importance of using both structural and linguistic features.

respectively. These were used to tune model parameters. **(c) For testing**: We annotated the lyrics of 118 other English songs and 128 other Japanese songs and called them EN_test and JA_test, respectively [6] . These were used to test the chorus-section detection methods.

4.3 Comparison of Different Methods

Table 2 summarizes the evaluated performances of Heuristic, MLP and the proposed Bi-LSTM. We found that MLP and Bi-LSTM performed better than Heuristic. This indicates that methods based on supervised learning are better than a rule-based method. We also found that Bi-LSTM was better than MLP and thus confirmed the importance of learning dependencies between adjacent lines. Since we concluded from these results that the proposed Bi-LSTM is the best for the chorus-section detection task, in the subsequent experiments reported here we used only Bi-LSTM.

4.4 Importance of Using Both Structural and Linguistic Features

To investigate the effectiveness of structural and linguistic features, we compared their use individually and in combination. Table 3 summarizes the results. We found that the model with only the structural features sim_{all} greatly outperformed the model with only the linguistic features $ling_{all}$. Using both kinds of features further improved the performance. This not only confirms the importance of using SSMs, as had been shown for the audio-based detection of chorus sections, but also confirms that the additional use of linguistic features is helpful for detecting chorus sections, which has not been shown before.

4.5 Reliability of Generated Annotations

As stated in Section 4.2, we used JA_man for the proposed training comparison. Table 4 clearly shows that the model trained using JA_auto, automatically generated data the amount of which can be large, outperformed the model trained using JA_man, manually annotated data, the amount

---

[3] In our experiments, English and Japanese lyrics text as well as the start time of every lyric line were provided by a lyrics distribution company. Automatic lyrics-to-audio synchronization [7–17] could also be used to estimate such start times.

[4] The main genres are Rock (33%), Pop (25%) and Alternative (12%) for EN_auto, and J-Pop (53%), Rock (20%) and Anime (9%) for JA_auto.

[5] To investigate the accuracy of the automatic annotation method we used for generating EN_auto and JA_auto, we applied the same method to the songs (audio signals and corresponding manually time-aligned lyrics) in JA_man. The accuracy of the generated annotations was $F = 68.0\%$, thus the automatic annotation method seems to work decently well.

[6] The `chorus` and `not-chorus` labels were annotated only on the lyrics. No audio signal is available for these test data.

Training data	F	p-F	n-F	V
JA_auto (91,459 songs)	**83.4**	**83.5**	**64.9**	**61.4**
JA_man (1,103 songs)	80.3	77.3	53.3	50.4

Table 4. Experimental result: Reliability of automatically generated annotations.

Training data	Testing data	F	p-F	n-F	V
EN_auto (9,313 songs)	EN_test	77.9	76.1	48.6	45.5
JA_auto (91,459 songs)	EN_test	80.3	80.6	58.1	54.4
EJ_auto (100,772 songs)	EN_test	**81.0**	**82.3**	**60.7**	**57.4**

Table 5. Experimental result: Can the Japanese model detect English chorus sections?

of which is usually very limited because of the laborious manual effort its creation requires. The results also confirm that even if annotations generated automatically are not perfect they are reliable enough for training the model.

4.6 Training Data Size and Language Dependency

Tables 2 and 3 also show that the performances for English lyrics were worse than those for Japanese lyrics. Since the amount of Japanese training data was about 10 times than that of English training data, we think that the amount of training data greatly affects the performance of the proposed model. We are thus interested in answering the question "Can a model trained on a large amount of Japanese data detect English chorus sections?" In fact, although linguistic features are language dependent and the process of computing SSMs is also language dependent, structural features based on the resulting SSMs can be language independent because our SSMs simply represent patterns of repeating lyric lines, which could be universal in music.

As shown in the upper half of Table 5, which shows results obtained without using linguistic features, we found that the structural-feature-based model trained on Japanese data JA_auto succeeded in detecting English chorus sections in EN_test and its performance was better than that of the model trained on the smaller dataset EN_auto. This result indicates that the SSM-based model trained on a large amount of data can detect chorus sections regardless of the language of the test set. Moreover, this result is further evidence that Japanese and English SSMs (i.e., patterns of repeating lyric lines) have similar structures.

Obviously, the above result raises another question: "Can a model trained on both EN_auto and JA_auto perform better than one trained on only EN_auto or JA_auto?" To answer this question, we created training data EJ_auto by including both EN_auto and JA_auto and constructed yet another structural-feature-based model with EJ_auto. As shown in the lower half of Table 5, we found that the model trained on both languages performed better than the model trained on only one.

These results confirm that chorus sections can be detected by a model trained on data in another language, that patterns of repeating lyric lines are language-independent and that mixing different language data allows the model to learn the general structure of chorus sections and thereby

perform better. This could have an impact on low-resource languages because large-scale training data can be created by mixing other available language resources.

5. RELATED WORK

Previous work in the MIR community has addressed musical structure analysis and chorus-section detection based on repeated patterns in musical audio signals [23–41]. Studies in the chorus-section detection for audio signals typically used SSMs to capture repeated structures, and we share this motivation. Our approach differs from those audio-based approaches in that it exploits multiple lyrics-based SSMs and linguistic features within chorus sections.

On the other hand, recent work in the NLP community has tackled lyrics segmentation and summarization tasks by exploiting SSMs. Fell et al. and Watanabe et al. proposed a neural network model and logistic regression model for segmenting paragraphs (sections) without labeling them by using SSMs as features [18, 19]. Those tasks, however, are essentially different from detecting all chorus sections that are the most representative sections in lyrics text. Addressing a task similar to chorus-section detection, Fell et al. [53] proposed a method of summarizing lyrics by combining general document summarization methods with audio thumbnailing methods. They focus on extracting individual informative lines as a summary from lyrics text, not redundant repeated lines. On the other hand, the focus of our paper is to detect chorus sections whose successive lines are often repeated in lyrics text.

6. CONCLUSION

This paper has addressed the novel task of detecting chorus sections in English and Japanese lyrics. We proposed a neural-network-based sequence labeling model that learns structural (i.e., phrase-repetition) and linguistic features to detect lyric lines of chorus sections. We also generated over 100,000 training data with chorus-section annotations. No previous work has ever conducted chorus-section detection for text-only lyrics with this much data.

The contributions of this study are summarized as follows: (1) We designed a variety of features to capture structural and linguistic properties of chorus sections. (2) We proposed a sequence labeling model that can detect chorus sections in lyrics. (3) We showed how to generate a large training dataset of lyrics with chorus-section annotations. (4) We demonstrated that our Bi-LSTM-based method outperforms alternative baseline methods. (5) We thoroughly investigated this detection task and the nature of chorus sections of lyrics from different perspectives such as the importance of features, the amount of training data, and language dependency.

We plan to extend our method to detect other sections, such as verse and bridge sections. Future work will also develop MIR applications using our method, such as those discussed in Section 1.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] K. Watanabe and M. Goto, "Query-by-Blending: A music exploration system blending latent vector representations of lyric word, song audio, and artist," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, 2019, pp. 144–151.

[2] K. Tsukuda, K. Ishida, and M. Goto, "Lyric Jumper: A lyrics-based music exploratory web service by modeling lyrics generative process," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, 2017, pp. 544–551.

[3] S. Sasaki, K. Yoshii, T. Nakano, M. Goto, and S. Morishima, "LyricsRadar: A lyrics retrieval system based on latent topics of lyrics," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 585–590.

[4] A. Tsaptsinos, "Lyrics-based music genre classification using a hierarchical attention network," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, 2017, pp. 694–701.

[5] R. Mayer and A. Rauber, "Music genre classification by ensembles of audio and lyrics features," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, 2011, pp. 675–680.

[6] R. Mayer, R. Neumayer, and A. Rauber, "Rhyme and style features for musical genre classification by song lyrics," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, 2008, pp. 337–342.

[7] B. Sharma, C. Gupta, H. Li, and Y. Wang, "Automatic lyrics-to-audio alignment on polyphonic music using singing-adapted acoustic models," in *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP 2019)*, 2019, pp. 396–400.

[8] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP 2019)*, 2019, pp. 181–185.

[9] C. Gupta, E. Yilmaz, and H. Li, "Acoustic modeling for automatic lyrics-to-audio alignment," in *Proceedings of the 20th Annual Conference of the International Speech Communication Association (Interspeech 2019)*, 2019, pp. 2040–2044.

[10] C. Gupta, R. Tong, H. Li, and Y. Wang, "Semi-supervised lyrics and solo-singing alignment," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, 2018, pp. 600–607.

[11] S. Chang and K. Lee, "Lyrics-to-audio alignment by unsupervised discovery of repetitive patterns in vowel acoustics," *IEEE Access*, vol. 5, pp. 16 635–16 648, 2017.

[12] S. W. Lee and J. Scott, "Word level lyrics-audio synchronization using separated vocals," in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP 2017)*, 2017, pp. 646–650.

[13] Y. Chien, H. Wang, and S. Jeng, "Alignment of lyrics with accompanied singing audio based on acoustic-phonetic vowel likelihood modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 1998–2008, 2016.

[14] G. B. Dzhambazov and X. Serra, "Modeling of phoneme durations for alignment between polyphonic audio and lyrics," in *Proceedings of the 12th Sound and Music Computing Conference (SMC 2015)*, 2015.

[15] M. Mauch, H. Fujihara, and M. Goto, "Integrating additional chord information into HMM-based lyrics-to-audio alignment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 200–210, 2012.

[16] H. Fujihara, M. Goto, J. Ogata, and H. G. Okuno, "LyricSynchronizer: Automatic synchronization system between musical audio signals and lyrics," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1252–1261, 2011.

[17] M. Kan, Y. Wang, D. Iskandar, T. L. Nwe, and A. Shenoy, "LyricAlly: Automatic synchronization of textual lyrics to acoustic music signals," *IEEE Trans. Speech Audio Process.*, vol. 16, no. 2, pp. 338–349, 2008.

[18] M. Fell, Y. Nechaev, E. Cabrio, and F. Gandon, "Lyrics segmentation: Textual macrostructure detection using convolutions," in *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, 2018, pp. 2044–2054.

[19] K. Watanabe, Y. Matsubayashi, N. Orita, N. Okazaki, K. Inui, S. Fukayama, T. Nakano, J. B. L. Smith, and M. Goto, "Modeling discourse segments in lyrics using repeated patterns," in *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, 2016, pp. 1959–1969.

[20] A. Baratè, L. A. Ludovico, and E. Santucci, "A semantics-driven approach to lyrics segmentation," in

Proceedings of the 8th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP 2013), 2013, pp. 73–79.

[21] J. P. G. Mahedero, A. Martinez, P. Cano, M. Koppenberger, and F. Gouyon, "Natural language processing of lyrics," in *Proceedings of the 13th ACM International Conference on Multimedia (ACM Multimedia)*, 2005, pp. 475–478.

[22] K. Watanabe, Y. Matsubayashi, K. Inui, S. Fukayama, T. Nakano, and M. Goto, "Modeling storylines in lyrics," *IEICE Transactions on Information and Systems*, vol. 101-D, no. 4, pp. 1167–1179, 2018.

[23] G. Shibata, R. Nishikimi, E. Nakamura, and K. Yoshii, "Statistical music structure analysis based on a homogeneity-, repetitiveness-, and regularity-aware hierarchical hidden semi-markov model," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, 2019, pp. 268–275.

[24] A. Maezawa, "Music boundary detection based on a hybrid deep model of novelty, homogeneity, repetition and duration," in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP 2019)*, 2019, pp. 206–210.

[25] G. Sargent, F. Bimbot, and E. Vincent, "Estimating the structural segmentation of popular music pieces under regularity constraints," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 2, pp. 344–358, 2017.

[26] T. Cheng, J. B. L. Smith, and M. Goto, "Music structure boundary detection and labelling by a deconvolution of path-enhanced self-similarity matrix," in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP 2018)*, 2018, pp. 106–110.

[27] J. B. L. Smith and M. Goto, "Using priors to improve estimates of music structure," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, 2016, pp. 554–560.

[28] T. Grill and J. Schlüter, "Music boundary detection using neural networks on combined features and two-level annotations," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, 2015, pp. 531–537.

[29] B. McFee and D. Ellis, "Analyzing song structure with spectral clustering," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 405–410.

[30] G. Peeters and V. Bisot, "Improving music structure segmentation using lag-priors," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 337–342.

[31] H. Grohganz, M. Clausen, N. Jiang, and M. Müller, "Converting path structures into block structures using eigenvalue decompositions of self-similarity matrices," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, 2013, pp. 209–214.

[32] O. Nieto and T. Jehan, "Convex non-negative matrix factorization for automatic music structure identification," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP 2013)*, 2013, pp. 236–240.

[33] F. Kaiser and G. Peeters, "A simple fusion method of state and sequence segmentation for music structure discovery," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, 2013, pp. 257–262.

[34] J. Serrà, M. Müller, P. Grosche, and J. L. Arcos, "Unsupervised detection of music boundaries by time series structure features," in *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2012.

[35] M. Müller, P. Grosche, and N. Jiang, "A segment-based fitness measure for capturing repetitive structures of music recordings," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, 2011, pp. 615–620.

[36] J. Paulus, M. Müller, and A. Klapuri, "State of the art report: Audio-based music structure analysis," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010, pp. 625–636.

[37] J. Paulus and A. Klapuri, "Music structure analysis using a probabilistic fitness measure and a greedy search algorithm," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1159–1170, 2009.

[38] M. Müller and S. Ewert, "Joint structure analysis with applications to music annotation and synchronization," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, 2008, pp. 389–394.

[39] M. Goto, "A chorus section detection method for musical audio signals and its application to a music listening station," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1783–1794, 2006.

[40] M. Cooper and J. Foote, "Summarizing popular music via structural similarity analysis," in *Proceedings of the 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2003)*, 2003, pp. 127–130.

[41] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Proceedings of the 2000 IEEE International Conference on Multimedia and Expo (IEEE ICME 2000)*, 2000, p. 452.

[42] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NeurIPS 2013)*, 2013, pp. 3111–3119.

[43] O. Melamud, J. Goldberger, and I. Dagan, "context2vec: Learning generic context embedding with bidirectional LSTM," in *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 51–61.

[44] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Popular, classical and jazz music databases," in *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, vol. 2, 2002, pp. 287–288.

[45] Y. Li and B. Liu, "A normalized Levenshtein distance metric," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1091–1095, 2007.

[46] S. Bird, "NLTK: the natural language toolkit," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL 2006)*, 2006.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of Workshop on Knowledge Discovery in Databases*, 1994, pp. 359–370.

[49] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP 2013)*, 2013, pp. 6645–6649.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, vol. 37, 2015, pp. 448–456.

[51] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mir_eval: A transparent implementation of common MIR metrics," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 367–372.

[52] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019.

[53] M. Fell, E. Cabrio, F. Gandon, and A. Giboin, "Song lyrics summarization inspired by audio thumbnailing," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, 2019, pp. 328–337.

CHORD JAZZIFICATION:
LEARNING JAZZ INTERPRETATIONS OF CHORD SYMBOLS

Tsung-Ping Chen[1] **Satoru Fukayama**[2] **Masataka Goto**[2] **Li Su**[1]

[1] Institute of Information Science, Academia Sinica, Taiwan
[2] National Institute of Advanced Industrial Science and Technology (AIST), Japan

`{tearfulcanon, lisu}@iis.sinica.edu.tw`
`{s.fukayama, m.goto}@aist.go.jp`
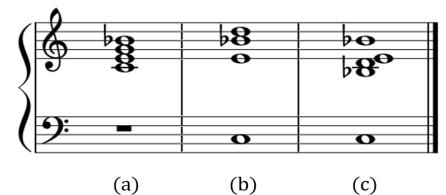
ABSTRACT

Chord symbols, typically notating the root note and the chord quality, are extensively used yet oversimplified representation of tonal harmony and chord progressions in popular music. In spite of its convenience, the chord symbol notation only provides basic information about the chordal configuration, and leaves much room for interpretation. With such limitations, an algorithm generating merely chord symbols is usually insufficient for a wide range of music genres such as jazz. To solve this problem, we propose *chord jazzification*, a process to generate realistic chord configurations in jazz style. With deep learning approaches, we decompose chord jazzification into *coloring* and *voicing*. Coloring concerns the choice of color tones, while voicing concerns the configurations of chords. We also create a new dataset featuring interpretations of chord symbols in pop-jazz compositions. By conducting experiments on the new dataset, we show that 1) the two-stage process outperforms an end-to-end generation approach in modeling chord configurations, and 2) attention-based models are better at capturing the structure of chord sequences in comparison with recurrent neural networks.

1. INTRODUCTION

Harmony and chords are the central topic in the study of tonal music. To facilitate the study, researchers in the fields of music information retrieval (MIR) and computational musicology have developed various techniques, such as automatic chord recognition [1–5], chord similarity and tonal distance [6–9], harmonic analysis [10–13], and chord generation [14–16].

Most of these aforementioned techniques process the chord data using *chord symbol* representation, i.e., a symbolic notation system indicating the root note, quality, and other additional information of the chord. For example, the chord symbol `C:7` stands for the C dominant seventh chord in root position, whose theoretical configuration is

Figure 1: Three configurations of the C dominant seventh chord. In music theory, the C dominant seventh chord in root position is configured as (a). (b) and (c) are two alterations of the chord.

depicted in Figure 1a. This symbol, however, does not explicitly point out the actual configuration, e.g., every specific note performed by a musician. Hence, the notation system is limited in describing the nuances in real-world performance. Learning to interpret the chord symbols is challenging in two aspects. First, expert musicians often *color* a chord by adding notes to, or omitting notes from the chord according to the musical context, although the chord symbol itself does not specify such alterations. Second, chords composed of the same set of pitch classes can be *voiced* differently by spacing and doubling the chord tones. Take the *comping* technique in jazz music as an instance.[1] Instead of sticking to the typical configuration of a chord, a jazz pianist may play the C dominant seventh chord with the configuration shown in Figure 1b, in which the 5th of the root, `G4`, is omitted, and the 9th of the root, `D5`, is added; while another jazz pianist may arrange these notes in a totally different way as demonstrated in Figure 1c, where `Bb` is doubled and `D` is spaced a register lower. In fact, the way how musicians interpret chord symbols in a given context involves not only their thorough understanding of various musical styles, but also their personal tastes. Therefore, to generate the realization of chord symbols through MIR approaches is a challenging yet valuable task, despite this topic is rarely discussed possibly because of the lack of data.

In this paper, we propose *chord jazzification*, a process to realize chord symbols with jazz harmony through deep learning approaches. Based on the two aforementioned aspects of interpreting the chord symbols, the chord jazz-

---

[1] Comping means accompanying or complementing a soloist by playing the chords to fill the harmonic and rhythmical vacancies in the music.

Chord Symbol	Db:M	$\frac{\text{Db:M}}{\text{F}}$	Gb:M7	$\frac{\text{D:M}}{\text{E}}$	$\frac{\text{Bb:m7}}{\text{Eb}}$	Eb:m7	Ab:7	G:dim	F:m7
Time: Onset	55	57	59	62	63	64	65	66	67
Time: Duration	2	2	3	1	1	1	1	1	2
Voicing	(Db2,Db3,F4,Ab4,Eb5) (F2,F3,Ab4,Db5,Eb5)		(Gb2,Db3,Gb3,Bb4,F5)	(Eb2,Ab4,C5,Db5,F5) (E2,D4,F#4,Bb4)	(Ab2,Gb3,Bb4,Db5) (Eb3,Gb4,Db5)		(F2,Eb3,C4,Ab4) (Gb2,Gb3,A4,Eb5)		
Coloring	(2)	(2)	-	(o5,b6)	(o1,9)	(o5)	(o5,d7) (o3,o5,9,11)		-
Roman Numeral Analysis	Db: I	VI^6	IV^7	bIII$_7$	ii$^7_4$ $-$ $_3$		V$_7$	vii$^o_4$$_3$	iii^7
Structure: Phrase	B1	B1	B1	B1	B1	B1	B1	B1	B1
Structure: Measure	15	15	16	16	17	17	17	17	18
Structure: Metrical Position	0	2	0	3	0	1	2	3	0

Figure 2: Annotations of the proposed dataset. *Chord Symbol* specifies the root and the quality of each chord (a bass note is explicitly notated with a slash when it is not the root note), e.g., `Db:M/F` stands for a Db major triad with the bass note `F`. *Time* indicates the *onset* and the *duration* (measured in beats) of each chord. *Voicing* represents the configuration of each chord with a set of scientific pitch notations. *Coloring* indicates the chord degrees which appear in the voicing but are not specified by the chord symbol, and vice versa, e.g., `(o1,9)` indicates that the root is omitted and the 9th is added. Note that `7` is explicitly qualified by {`M`, `m`, `d`} (major, minor, diminished) to disambiguate its interval. *Roman Numeral Analysis* denotes the scale degree on which a chord is built, as well as the quality and the inversion information, e.g., `V7` stands for a dominant seventh chord in root position. *Structure* includes the annotations of *phrase*, *measure* and *metrical position*. To specify the phrase a chord belongs to, each chord is labeled with a letter plus a number in a way similar to musical form analysis. Metrical position shows the position of a chord with respect to the metric grid (starting with 0).

ification task is formulated as two subtasks, namely the *chord coloring* and the *chord voicing*. The chord coloring task decides which pitch classes are to be played for elaborating a chord symbol, while the chord voicing task deals with the spacing and the doubling of the pitch classes to be played. Although the jazzification of chord progressions is not limited to coloring and voicing, we focus on the two aspects for the primary study. To facilitate the research on chord jazzification, we also compile a new dataset consisting of chord symbols and corresponding chord configurations in pop-jazz compositions. In comparison with other jazz-related datasets, such as the Charlie Parker's Omnibook data [17], the Jazz Audio-Aligned Harmony (JAAH) dataset [18], the JazzCorpus [19], and the Weimar Jazz Database [20], our dataset includes more detailed information of chords, especially the chordal configuration. With the newly compiled dataset, we conduct several experiments to verify the two-stage framework for chord jazzification. Experiment results indicate that it is effective to decompose chord jazzification into coloring and voicing, rather than to adopt an end-to-end approach.

The current work is different from the accompaniment or harmonization tasks [21,22], in terms of that such works require melodies as prior knowledge and regard chords as appendages to melodies. Besides, our work concerns both the voicings of a chord sequence and the interpretation of each single chord, thus is distinct from the voice leading generation task [23,24]. The jazzification of a chord progression is part of the music composition process to expand

a tonal structure, i.e., the *prolongation* described in Heinrich Schenker's music theory [25]. Chord jazzification can advance other MIR-related tasks such as style analysis and chord generation [26–29]. We hope that our work can draw more attention to the musical knowledge regarding the implicit relations between the notated chord symbols and the actual harmony being performed.

In summary, our contribution is threefold. First, we address the issue of interpreting chord symbols by chord jazzification. Second, we compile a new dataset for the generation of jazzy harmony and for the study of chord embellishments. Finally, a deep learning framework is proposed to generate jazz-style chord progressions. In the following, we will first present the new dataset (Section 2), and then formulate the framework of chord jazzification (Section 3); based on the framework, several experiments are introduced thereon (Section 4).

2. THE CHORD JAZZIFICATION DATASET

The corpus is composed of 50 musical pieces selected from published Japanese pop-jazz piano solos, in which chord symbols are explicitly specified.[2] To obtain the corresponding voicing of each chord symbol, we manually perform harmonic reduction for each piece. Concretely, notes within the region of a chord symbol are selected to build the configuration of the chord symbol. With the chord

---

[2] The dataset is available at `https://github.com/Tsung-Ping/Chord-Jazzification`.

Figure 3: Chord qualities and chord colorings in the proposed dataset (the long tail of the coloring distribution is left out). Chord degrees of compound intervals are merged with simple ones in the coloring figure for simplicity.

symbols and the transcribed voicings, coloring and harmonic information are annotated. Specifically, there are six types of annotations in the dataset: *Chord Symbol*, *Time*, *Voicing*, *Coloring*, *Roman Numeral Analysis* [10, 30], and *Structure*. Figure 2 gives an example.

A brief introduction of the chord qualities and colorings used in the corpus is illustrated in Figure 3. Not surprisingly, the three seventh chords, major seventh (M7), minor seventh (m7), and dominant seventh (7), account for more than half of the chords, as jazz harmony is notable for the use of seventh chords. On the other hand, characteristic colorings of pop-jazz music can also be found in many chords, such as adding a major second or a compound major second (2) and omitting the perfect fifth (o5).

In summary, 796 musical phrases amounting to 6700 chord labels are included in the dataset. These annotations provide information concerning the relationship between the symbolic notation and the actual configuration of chords in human compositions. It therefore has the potential to be applied to many MIR-related research topics, such as corpus-based study of tonal harmony in music practice, generation of colorful chord progressions, and computer-aided composition, to name but a few. It has to be acknowledged that the dataset has some limitations in describing an actual interpretation of chord symbols, in the sense that the transcription of chords eliminates the harmonic and rhythmical variances within the region of each chord symbol. Nevertheless, the dataset can be a starting point for performers and composers to learn to elaborate and develop a tonal structure through the chord jazzification process, which is relatively easier compared to learning the elaboration directly from a complete musical piece.

3. CHORD JAZZIFICATION

The goal of chord jazzification is to endow plain chords (e.g., a sequence of triads) with jazz harmony. We tackle the chord jazzification task through two successive steps, that is, coloring and voicing. The coloring part functions as an intermediate state which specifies chords in terms of pitch classes, and the voicing part assigns pitch heights to the specified pitch classes.

3.1 Coloring

In this paper, 48 triads in root position ({major, minor, augmented, diminished} by 12 semitones) are considered for coloring. We define the *chord coloring* task as predicting the bass note and the pitch classes to render each triad in a given sequence.

Formally, the input of the coloring task is a triad sequence $\{\mathbf{x}_i\}_{i=1}^T$ and a duration sequence $\{\mathrm{d}_i\}_{i=1}^T$, where i denotes time steps, T is the length of the sequence, $\mathbf{x}_i \in \mathbb{R}^{12}$ is a chroma representation of the ith triad, and d_i is the duration of $\mathbf{x}_i$. The coloring task predicts the bass sequence $\{\mathbf{b}_i^c\}_{i=1}^T$ and the pitch class sequence $\{\mathbf{p}_i^c\}_{i=1}^T$ for the input sequence, where c stands for coloring, $\mathbf{b}_i^c \in \mathbb{R}^{12}$ is a softmax-activated chroma vector indicating the 12 pitch classes' probabilities to be the bass of colored $\mathbf{x}_i$, and $\mathbf{p}_i^c \in \mathbb{R}^{12}$ is a sigmoid-activated chroma vector indicating the 12 pitch classes' probabilities to be the constituent notes (except the bass) of colored $\mathbf{x}_i$. For example, $\mathbf{b}_i^c = [0.8, 0, 0, 0, 0.2, 0, 0, 0, 0, 0, 0, 0]$ indicates the pitch classes C and E respectively have a probability of 80% and 20% to be the bass note, and $\mathbf{p}_i^c = [0, 0.9, 0, 0, 0.9, 0, 0, 0, 0, 0, 0.9, 0]$ suggests that there is a 90% chance that Db, E, and Bb are activated to render $\mathbf{x}_i$.

We employ a basic sequential learning architecture for the coloring task. As shown in Figure 4a, the architecture is composed of three layers: 1) an input embedding layer, 2) a sequential modeling layer, and 3) an output layer. Specifically, the three layers are formulated as follows:

$$
\begin{aligned}
\mathbf{e}_i^c &= \mathbf{W}^{e^c}(\mathrm{d}_i \mathbf{x}_i), \quad \text{(Input Embedding)} \\
\mathbf{h}_i^c &= f^c(\mathbf{e}_i^c \mid \mathbf{e}_{1:T}^c), \quad \text{(Sequential Modeling)} \\
\mathbf{p}_i^c &= \mathrm{sigmoid}(\mathbf{W}^{p^c}\mathbf{h}_i^c), \quad \text{(Output)} \\
\mathbf{b}_i^c &= \mathrm{softmax}(\mathbf{W}^{b^c}\mathbf{h}_i^c),
\end{aligned}
\tag{1}
$$

where $\mathbf{W}^{e^c} \in \mathbb{R}^{d\times 12}$, $\mathbf{W}^{p^c} \in \mathbb{R}^{12\times d}$, and $\mathbf{W}^{b^c} \in \mathbb{R}^{12\times d}$ are learnable parameters, $f^c : \mathbb{R}^d \to \mathbb{R}^d$ denotes a trainable neural network, and d is a hyperparameter indicating the dimensions of the embedding space. Two candidate networks are employed for the sequential modeling layer:

- Bi-directional Recurrent Neural Network with Long Short-Term Memory (BLSTM):

$$
\begin{aligned}
\mathbf{h}_i^c &= \overrightarrow{\mathbf{h}}_i^c \oplus \overleftarrow{\mathbf{h}}_i^c, \\
\overrightarrow{\mathbf{h}}_i^c &= \mathrm{LSTM}(\mathbf{e}_i^c \mid \mathbf{e}_{1:i-1}^c), \\
\overleftarrow{\mathbf{h}}_i^c &= \mathrm{LSTM}(\mathbf{e}_i^c \mid \mathbf{e}_{i+1:T}^c),
\end{aligned}
\tag{2}
$$

where $\oplus$ denotes vector concatenation.

Figure 4: (a) The coloring model. (b) The voicing model. (c) The two-stage chord jazzification model.

- Multihead Self-attention Network (MHSA):

$$\mathbf{h}_i^{c(l)} = \mathbf{W}^{outer}\sigma(\mathbf{W}^{inner}\mathbf{u}_i + \mathbf{b}_1) + \mathbf{b}_2,$$

$$\mathbf{u}_i = \mathbf{W}^u(\mathbf{u}'_{i1} \oplus \cdots \oplus \mathbf{u}'_{iJ}) + \mathbf{h}_i^{c(l-1)},$$

$$\mathbf{u}'_{ij} = \mathbf{V}_j \text{ softmax}\left(\frac{\mathbf{K}_j^\top \mathbf{q}_{ij}}{\sqrt{d}}\right),$$

$$\mathbf{q}_{ij} = \mathbf{W}_j^Q \mathbf{h}_i^{c(l-1)}, \qquad (3)$$

$$\mathbf{K}_j = \mathbf{W}_j^K[\mathbf{h}_1^{c(l-1)}, \cdots, \mathbf{h}_T^{c(l-1)}],$$

$$\mathbf{V}_j = \mathbf{W}_j^V[\mathbf{h}_1^{c(l-1)}, \cdots, \mathbf{h}_T^{c(l-1)}],$$

where l denotes the iteration step, and the initial value $\mathbf{h}_i^{c(0)} = \mathbf{e}_i^c$; σ represents the ReLU activation function; J is the number of heads; $\mathbf{W}_j^{outer} \in \mathbb{R}^{d \times 4d}$, $\mathbf{W}_j^{inner} \in \mathbb{R}^{4d \times d}$, $\mathbf{W}^u \in \mathbb{R}^{d \times d}$, and $\mathbf{W}_j^Q, \mathbf{W}_j^K, \mathbf{W}_j^V \in \mathbb{R}^{\frac{d}{J} \times d}$ are learnable parameters. This network is equivalent to the encoder of the Transformer [31], while we leave out layer normalization and position encoding terms for simplicity. In this paper, we set $d = 512$, $l = 2$, and $J = 8$.

The binary cross entropy (BCE) and the categorical cross entropy (CCE) are used to calculate the losses. Let $\mathbf{p}_i^{c*}$ and $\mathbf{b}_i^{c*}$ denote the ground truths of $\mathbf{p}_i^c$ and $\mathbf{b}_i^c$; the total loss of the coloring model $\mathcal{L}^c$ is defined as:

$$\mathcal{L}^c = \sum_{i=1}^T \left[\text{BCE}(\mathbf{p}_i^{c*}, \mathbf{p}_i^c) + \text{CCE}(\mathbf{b}_i^{c*}, \mathbf{b}_i^c)\right]. \qquad (4)$$

3.2 Voicing

We define *chord voicing* as a task which predicts the voicings of a chord sequence. Formally, given a chord sequence of T time steps in terms of their basses $\{\mathbf{b}_i^v\}_{i=1}^T$, constituent pitch classes $\{\mathbf{p}_i^v\}_{i=1}^T$, and durations $\{d_i\}_{i=1}^T$, the task predicts the voicings $\{\mathbf{v}_i\}_{i=1}^T$ for the chord sequence, where v stands for voicing, $\mathbf{b}_i^v \in \mathbb{R}^{12}$ is a one-hot chroma vector indicating the bass of the ith chord,

$\mathbf{p}_i^v \in \mathbb{R}^{12}$ is a multi-hot chroma vector representing the pitch classes of the ith chord except the bass note, and $\mathbf{v}_i \in \mathbb{R}^{88}$ is a voicing vector indicating the 88 tones' probabilities to be played on the piano.

Similar to the coloring task, we employ a 3-layer architecture for the voicing task, as shown in Figure 4b. The three layers are formulated as follows:

$$\mathbf{e}_i^v = \mathbf{W}^{e^v}(d_i(\mathbf{p}_i^v \oplus \mathbf{b}_i^v)), \quad \text{(Input Embedding)}$$

$$\mathbf{h}_i^v = f^v(\mathbf{e}_i^v \mid \mathbf{e}_{1:T}^v), \quad \text{(Sequential Modeling)} \qquad (5)$$

$$\mathbf{v}_i = \text{sigmoid}(\mathbf{W}^v \mathbf{h}_i^v), \quad \text{(Output)}$$

where $\mathbf{W}^{e^v} \in \mathbb{R}^{d \times 24}$ and $\mathbf{W}^v \in \mathbb{R}^{88 \times d}$ are learnable parameters, and $f^v : \mathbb{R}^d \to \mathbb{R}^d$ is a neural network. Likewise, the BLSTM and the MHSA networks are two options for the sequential modeling layer.

Let $\mathbf{v}_i^* \in \mathbb{R}^{88}$ denote the target voicing of the ith chord; we define the loss as:

$$\mathcal{L}^v = \sum_{i=1}^T \text{BCE}(\mathbf{v}_i^*, \mathbf{v}_i). \qquad (6)$$

As the voicing task is to arrange the constituent notes of chords on an 88-key piano based on the given basses and the sets of pitch classes, the outcome of $\mathbf{v}_i^*$ can be known to a certain degree. More precisely, a note in $\mathbf{v}_i^*$ can be activated only if its pitch class is activated in $\mathbf{b}_i^v$ or $\mathbf{p}_i^v$. With this consideration, we design corresponding masks to modify the loss computation. Let $\mathbf{b}_i^{v'} \in \mathbb{R}^{88}$ and $\mathbf{p}_i^{v'} \in \mathbb{R}^{88}$ be the extensions of $\mathbf{b}_i^v$ and $\mathbf{p}_i^v$ to all octaves of the piano. Then, the loss constrained by the masks becomes:

$$\mathcal{L}^{v'} = \sum_{i=1}^T \text{BCE}(\mathbf{v}_i^*, \mathbf{m}_i \odot \mathbf{v}_i),$$

$$\mathbf{m}_i = \mathbf{b}_i^{v'} \vee \mathbf{p}_i^{v'}, \quad \text{(Mask)} \qquad (7)$$

where $\odot$ stands for the Hadamard product, and $\vee$ denotes the logical OR operator.

363

3.3 Two-stage Chord Jazzification

We stack the chord voicing model on the top of the chord coloring model by setting $\mathbf{b}_i = \mathbf{b}_i^v = \text{onehot}(\arg\max \mathbf{b}_i^c)$ and $\mathbf{p}_i = \mathbf{p}_i^v = \text{round}(\mathbf{p}_i^c)$, as illustrated in Figure 4c. In other words, the outputs of the coloring model are first converted into binary vectors, and then taken as inputs by the voicing model. Such an integrated model jazzifies chord progressions in two stages: first, for a given sequence of triads $\{\mathbf{x}_i\}_{i=1}^T$ and the corresponding durations $\{\mathbf{d}_i\}_{i=1}^T$, the coloring model generates a colored sequence represented by $\{\mathbf{b}_i\}_{i=1}^T$ and $\{\mathbf{p}_i\}_{i=1}^T$; based on the colored sequence, the voicing model subsequently generates a voiced chord progression $\{\mathbf{v}\}_{i=1}^T$.

4. EXPERIMENTS

4.1 Chord Jazzification with Supervised Learning

With the formulations of chord jazzification as multi-class classification (for $\mathbf{b}_i^c$) and multi-label classification (for $\mathbf{p}_i^c$ and $\mathbf{v}_i$) problems, we train the coloring and voicing models using the new dataset, and perform 4-fold cross validation. For the coloring task, the input triad sequences and the input duration sequences are respectively derived from the *Chord Symbol* and the *Time* annotations of the dataset, while the ground truths of the bass sequences and the pitch class sequences are obtained from the *Voicing* labels. As for the voicing task, the duration sequences and the ground truth labels of the coloring task are taken as the inputs, while the *Voicing* labels are used as the ground truths of the output sequences. We augment the training set through transposing the data from 4 semitones down to 5 semitones up (within the valid range of the piano), leading to 10 times the training data. As a result, there are 5970 and 199 sequences for training and testing respectively.

Evaluation results are shown in Table 1. For both the coloring and voicing tasks, the employment of either the BLSTM or the MHSA as the sequential modeling layers yields comparable performance to the other, while the MHSA appears to surpass the BLSTM in cases of multi-label classification, i.e., the predictions of pitch classes and voicing. When the input embedding layers in the two sub-tasks are removed, all the performances decrease by from 3.19% to 4.69%. This indicates that the transformation to dense vectors benefits the learning process when the input data is sparsely represented. Moreover, the introduction of the input-related masks to the loss calculation in Eqn (7) also improves the modeling of voicing; precisely, the F1 score increases 2.66% if the masks are utilized. It is worth noting that the amount of training data is quite limited, and therefore the performance seems to be satisfactory in the current experimental setting.

4.2 End-to-end Chord Jazzification

To motivate the decomposition of chord jazzification into 2 stages, we train a chord jazzification model in an end-to-end manner for comparison. Technically, we replace the output module of the coloring model with that of the voicing model; and we employ a 2-layer BLSTM, rather than

Model	Coloring		Voicing
	Bass	Pitch Classes	
BLSTM	**81.87**	76.52	63.64
MHSA	80.78	**77.02**	**64.86**
BLSTM w/o E	77.18	73.33	60.12
BLSTM w/o M	-	-	60.98
End-to-End	-	-	37.87

Table 1: Results of the coloring and the voicing tasks. The lower part shows the ablation tests without the embedding layer (w/o E) and without masks (w/o M), as well as the result using end-to-end training. All the values indicate the average F1 scores (%) over 4 validations.

a 1-layer BLSTM as defined in Eqn (2), for the sequential modeling layer in order to make the number of parameters comparable to the two-stage chord jazzification model. The evaluation result is shown in Table 1.

In this end-to-end architecture, the performance drops substantially to nearly half of the value. This result conversely validates the two-stage approach. Given the fact that the prediction of polyphony is often challenging, it turns out to be beneficial to generate an intermediate stage, that is, the chroma representations with respect to coloring, before the overall jazzification of chords.

4.3 Consistency of Chord Jazzification

Chord progressions often have a repetitive structure, therefore it is important for a model to preserve this property and generate chord sequences of self-consistency. To measure the consistency of a model's generations, we compute the self-similarity matrices (SSMs) of each generated voicing sequence and corresponding label sequence, and then calculate the difference between each generation-label SSM pair. Let $\overline{\mathbf{V}} = [\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_T]$ denote the normalized voicing sequence generated by a model, and $\overline{\mathbf{V}}^* = [\overline{\mathbf{v}}_1^*, \cdots, \overline{\mathbf{v}}_T^*]$ denote the normalized label sequence, where $\overline{\mathbf{v}}_i = \frac{\text{round}(\mathbf{v}_i)}{\|\text{round}(\mathbf{v}_i)\|}$ is a binarized and normalized voicing, and $\overline{\mathbf{v}}_i^* = \frac{\mathbf{v}_i^*}{\|\mathbf{v}_i^*\|}$ is a normalized target voicing; we define the consistency score (CS) of a generated sequence as follows:

$$\begin{aligned}
\text{CS} &= 1 - \text{reduce\_mean}(\Delta\text{SSM}), \\
\Delta\text{SSM} &= |\text{SSM}_{pred} - \text{SSM}_{label}|, \\
\text{SSM}_{pred} &= \overline{\mathbf{V}}^\top \overline{\mathbf{V}}, \\
\text{SSM}_{label} &= \overline{\mathbf{V}}^{*\top} \overline{\mathbf{V}}^*.
\end{aligned} \quad (8)$$
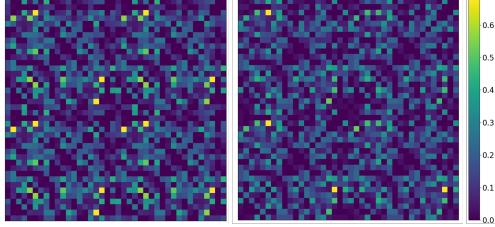
The more similar the structures of the two sequences are, the higher the CS score is.

Table 2 shows the average consistency score over 4 cross-validation folds. To provide a benchmark for the consistency measure, we also show the CS score computed from label sequences and randomly-generated sequences (denoted as *RANDOM*). Both the two models get higher scores than the random condition, indicating that they learn some structural information. Besides, the MHSA outperforms the BLSTM due to an essential difference between them: the BLSTM processes each time step of a

Model	BLSTM	MHSA	RANDOM
CS Score (%)	86.91	**87.68**	72.80
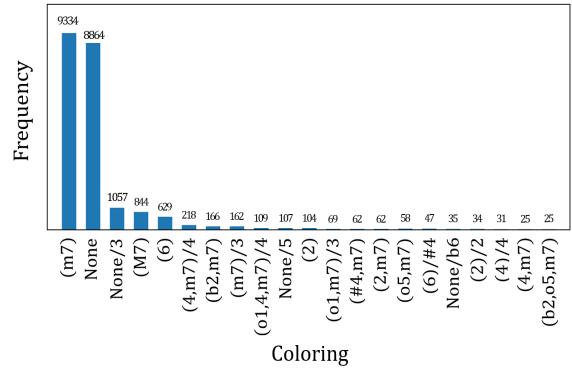
Table 2: Consistency measure of chord jazzification.

Figure 5: The difference of self-similarity matrices. Left: ΔSSM of the BLSTM. Right: ΔSSM of the MHSA. The origin of the matrices is at the lower left corner.

sequence recurrently, while the MHSA accesses the entire sequence simultaneously. As a result, the MHSA can capture more structural features than the BLSTM does, leading to more consistent generations. Two examples of the ΔSSM are demonstrated in Figure 5. Evidently, there are fewer bright regions in the ΔSSM of the MHSA, indicating that the generation by the MHSA is structurally closer to the ground truth than that by the BLSTM.

4.4 Generating Jazz Harmony

We train the two-stage chord jazzification model using the proposed dataset, and generate jazzified chord progressions with input triad sequences derived from the JAAH dataset. [3] In total, 2210 sequences with 23199 chords were generated. To examine the effect of jazzification, we quantitatively analyze the difference between each input triad and its jazzified counterpart. Particularly, we are interested in changes with respect to chord coloring: 1) what notes are added to or omitted from a triad? 2) Is a note other than the root being chosen as the bass note?

The result is represented in Figure 6. Around 40% of the input triads are embellished with a minor seventh (`m7`). And the addition of a major seventh (`M7`) also accounts for around 3.6%, ranked in the top fourth. These frequent colorings with a major or minor seventh reflect the characteristics of jazz music in which most triads that appear in lead sheets or fake books can have sevenths added to them. Moreover, extended chords and inverted chords can also be found. For instance, the coloring (`b2,m7`) for a major triad will lead to a dominant seventh flat ninth chord; and the coloring `None/3` indicates the first inversion of triads. It is worth mentioning that some generated slash chords are not inverted chords. An example is shown in the bottom of Figure 6. With the coloring (`2,M7`)`/2`, the last triad `Db:M` becomes `Db:M7/Eb`, which can be interpreted as an Eb dominant thirteenth chord—the dominant chord of the relative major mode (assuming `F` is the tonic). In other words, this coloring not only breaks the repetitive structure

---
[3] https://github.com/MTG/JAAH

Figure 6: Top: the coloring distribution of the generated chords (part of the distribution is omitted). Bottom: a generated example. A number after the slash symbol indicates the degree of the bass note relative to the root note.

of the input triad sequence, but also implies a new tonality. In addition to coloring, it can be observed that the linear progressions of voices are quite smooth, showing that the model also learns the knowledge of voice leading.

5. CONCLUSION

To learn the interpretation of chord symbols from musical data, we proposed chord jazzification, a process of generating realistic jazz-style chord progressions through two musical techniques: chord coloring and chord voicing. Chord coloring decides a bass and a set of pitch classes for elaborating a triad, while chord voicing arranges the bass and the set of pitch classes on the piano. We correspondingly built a dataset which includes coloring and voicing annotations, and hence can be used as the training data of the chord jazzification task. By formulating the chord coloring and chord voicing tasks as classification problems, we experimentally showed that the two-stage framework is capable of generating plausible chord configurations from a sequence of chord symbols.

Chord jazzification has the potential to be applied to two different yet related practices in music: performing and composing. For music performing, it is practical and desirable to interpret the chord symbols on lead sheets through jazzification. For music composing, the generated sequence by the chord jazzification model can be regarded as an intermediate product with which a musician can further create a human-machine collaborative musical work. In future research, we are planning to include more rhythmical and structural information, such as metrical position, to better represent the harmonic features, and apply more advanced deep learning techniques to improve the chord jazzification task.

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] F. Korzeniowski and G. Widmer, "Improved chord recognition by combining duration and harmonic language models," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 10–17.

[2] F. Korzeniowski, D. R. W. Sears, and G. Widmer, "A large-scale study of language models for chord prediction," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 91–95.

[3] J. Deng and Y. Kwok, "Large vocabulary automatic chord estimation with an even chance training scheme," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 531–536.

[4] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 188–194.

[5] F. Korzeniowski and G. Widmer, "A fully convolutional deep auditory model for musical chord recognition," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2016, pp. 1–6.

[6] W. B. de Haas, M. Robine, P. Hanna, R. C. Veltkamp, and F. Wiering, "Comparing approaches to the similarity of musical chord sequences," in *Exploring Music Contents - 7th International Symposium (CMMR)*, 2010, pp. 242–258.

[7] W. B. de Haas, R. C. Veltkamp, and F. Wiering, "Tonal pitch step distance: a similarity measure for chord progressions," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008, pp. 51–56.

[8] C. Harte, M. Sandler, and M. Gasser, "Detecting harmonic change in musical audio," in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, 2006, pp. 21–26.

[9] J. Paiement, D. Eck, and S. Bengio, "A probabilistic model for chord progressions," in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 312–319.

[10] T. Chen and L. Su, "Functional harmony recognition of symbolic music data with multi-task recurrent neural networks," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 90–97.

[11] N. Condit-Schultz, Y. Ju, and I. Fujinaga, "A flexible approach to automated harmonic analysis: multiple annotations of chorales by Bach and Prætorius," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 66–73.

[12] P. B. Kirlin and P. E. Utgoff, "A framework for automated Schenkerian analysis," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008, pp. 363–368.

[13] T. Chen and L. Su, "Harmony Transformer: Incorporating chord segmentation into harmony recognition," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 259–267.

[14] H. Lim, S. Rhyu, and K. Lee, "Chord generation from symbolic melody using BLSTM networks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 621–627.

[15] K. Choi, G. Fazekas, and M. Sandler, "Text-based LSTM networks for automatic music composition," in *the 1st Conference on Computer Simulation of Musical Creativity*, 2016.

[16] K. Kosta, M. Marchini, and H. Purwins, "Unsupervised chord-sequence generation from an audio example," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 481–486.

[17] K. Déguernel, E. Vincent, and G. Assayag, "Using multidimensional sequences for improvisation in the OMax paradigm," in *Proceedings of the 13th Sound and Music Computing Conference (SMC)*, 2016, pp. 481–486.

[18] V. Eremenko, E. Demirel, B. Bozkurt, and X. Serra, "Audio-aligned jazz harmony dataset for automatic chord transcription and corpus-based research," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 483–490.

[19] M. Granroth-Wilding, "Harmonic analysis of music using combinatory categorial grammar," Ph.D. dissertation, University of Edinburgh, 2013.

[20] M. Pfleiderer, K. Frieler, J. Abeßer, W.-G. Zaddach, and B. Burkhart, Eds., *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.

[21] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a steerable model for Bach chorales generation," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017, pp. 1362–1371.

[22] T. Kitahara, M. Katsura, H. Katayose, and N. Nagata, "Computational model for automatic chord voicing based on Bayesian network," in *Proceedings of the 10th International Conference on Music Perception and Cognition (ICMPC)*, 2008, pp. 395–398.

[23] D. Hörnel, "ChordNet: Learning and producing voice leading with neural networks and dynamic programming," *Journal of New Music Research*, vol. 33, no. 4, pp. 387–397, 2004.

[24] P. M. C. Harrison and M. T. Pearce, "A computational cognitive model for the analysis and generation of voice leadings," *Music Perception: An Interdisciplinary Journal*, vol. 37, pp. 208–224, 2020.

[25] O. Jonas, *Introduction to the Theory of Heinrich Schenker*, 2nd ed. Musicalia Press, 2005.

[26] D. Conklin, M. Gasser, and S. Oertl, "Creative chord sequence generation for electronic dance music," *Journal of Applied Sciences*, vol. 8, no. 9, p. 1704, 2018.

[27] D. Scanteianu, E. Jackson, and R. M. Keller, "A fluid chord voicing generator," in *Proceedings of the International Computer Music Conference (ICMC)*, 2016, pp. 171–175.

[28] R. Dias, C. Guedes, and T. Marques, "A computer-mediated interface for jazz piano comping," in *Music Technology meets Philosophy - From Digital Echos to Virtual Ethos: Joint Proceedings of the 40th International Computer Music Conference (ICMC), and the 11th Sound and Music Computing Conference (SMC)*, 2014, pp. 558–564.

[29] I. Simon, D. Morris, and S. Basu, "MySong: automatic accompaniment generation for vocal melodies," in *Proceedings of the 2008 Conference on Human Factors in Computing Systems (CHI)*, 2008, pp. 725–734.

[30] S. Kostka, D. Payne, and B. Almen, *Tonal Harmony*, 3rd ed. McGraw-Hill, 1995.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," in *Neural Information Processing Systems (NIPS)*, 2017, pp. 5998–6008.

367

PIANOTREE VAE: STRUCTURED REPRESENTATION LEARNING FOR POLYPHONIC MUSIC

Ziyu Wang[1] **Yiyi Zhang**[2] **Yixiao Zhang**[1] **Junyan Jiang**[1]
Ruihan Yang[1] **Junbo Zhao (Jake)**[3] **Gus Xia**[1]

[1] Music X Lab, Computer Science Department, NYU Shanghai
[2] Center for Data Science, New York University
[3] Computer Science Department, Zhejiang University

`{ziyu.wang, yz2092, yixiao.zhang, jj2731, ry649, j.zhao, gxia}@nyu.edu`
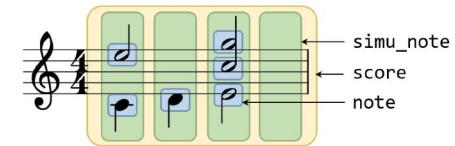
ABSTRACT

The dominant approach for music representation learning involves the deep unsupervised model family *variational autoencoder* (VAE). However, most, if not all, viable attempts on this problem have largely been limited to monophonic music. Normally composed of richer modality and more complex musical structures, the polyphonic counterpart has yet to be addressed in the context of music representation learning. In this work, we propose the PianoTree VAE, a novel tree-structure extension upon VAE aiming to fit the polyphonic music learning. The experiments prove the validity of the PianoTree VAE via (i)-semantically meaningful latent code for polyphonic segments; (ii)-more satisfiable reconstruction aside of decent geometry learned in the latent space; (iii)-this model's benefits to the variety of the downstream music generation.[1]

1. INTRODUCTION

Unsupervised machine learning has led to a marriage of symbolic learning and vectorized representation learning [1–3]. In the computer music community, the MusicVAE [4] enables the interpolation in the learned latent space to render some smooth music transition. The EC^2-VAE [5] manages to disentangle certain interpretable factors in music and also provides a manipulable generation pathway based on these factors. Pati *et al*. [6] further utilizes the recurrent networks to learned music representations for longer-term coherence.

Unfortunately, most of the success has been limited to monophonic music. The generalization of the learning frameworks to polyphonic music is not trivial, due to its much higher dimensionality and more complicated musical syntax. The commonly-adopted MIDI-like event sequence modeling or the piano-roll formats fed to either recurrent or convolutional networks have fell short in learn-

---

[1] Code and demos can be accessed via `https://github.com/ZZWaang/PianoTree-VAE`

Figure 1: An illustration of the proposed polyphonic syntax.

ing good representation, which usually leads to unsatisfied generation results [7–9]. In this paper, we hope to pioneer the development of this challenging task. To begin with, we conjecture a proper set of **inductive bias** for the desired framework: (i)-a sparse encoding of music as the model input; (ii)-a neural architecture that incorporates the hierarchical structure of polyphonic music (i.e., musical syntax).

Guided by the aforementioned design principles, we propose PianoTree VAE, a hierarchical representation learning model under the VAE framework. We adopt a tree structured musical syntax that reflects the hierarchy of musical concepts, which is shown in Figure 1. In a top-down order: we define a `score` (indicated by the yellow rectangle) as a series of `simu_note` events (indicated by the green rectangles), a `simu_note` as multiple `note` events sharing the same onset (indicated by blue rectangles), and each `note` has several attributes such as *pitch* and *duration*. In this paper, we focus on a simple yet common form of polyphonic music—piano score, in which each note has only pitch and duration attributes. For future work, this syntax can be generalized to multiple instruments and expressive performance by adding extra attributes such as voice, expressive timing, dynamics, etc.

The whole neural architecture of PianoTree VAE can be seen as a tree. Each node represents the embedding of either a `score`, `simu_note`, or `note`, where a higher level representation has larger receptive fields. The edges are bidirectional where a recurrent module is applied to either encode the children into the parent or decode the parent to generate its children.

Through extensive evaluations, we show that PianoTree VAE yields semantically more meaningful latent representations and further downstream generation quality gains, on top of the current state-of-the-art solutions.

2. RELATED WORK

The complex hierarchical nature of music data has been studied for nearly a century (e.g. GTTM [10], Schenklerian Analysis [11], and their follow-up works [12–15]). However, the emerging deep representation-learning models still lack the compatible solutions to deal with the complex musical structure. In this section, we first review different types of polyphonic music generation in Section 2.1. After that, we discuss some popular deep music generative models indexed by their compatible data structure from Section 2.2 to Section 2.4.

2.1 Different Types of Polyphony

In the context of deep music generation, *polyphony* can refer to three types of music: 1) multiple monophonic parts (e.g., a four-part chorus), 2) a single part of a polyphonic instrument (e.g., a piano sonata), and 3) multiple parts of polyphonic instruments (e.g., a symphony).

The first type of polyphonic music can be created by simply extending the number of voices in monophonic music generation with some inter-voice constraints. Some representative systems belonging to this category include DeepBach [16], XiaoIce [17], and Coconet [18]. Music Transformer [19] and the proposed PianoTree VAE both focus on the generation of the second type of polyphony, which is a much more difficult task. Polyphonic pieces under the second definition no longer have a fixed number of "voices" and consist of more complex textures. The third type of polyphony can be regarded as an extension of the second type, and we leave it for future work.

2.2 Piano-roll and Compatible Models

Piano-roll and its variations [7, 20–22] view polyphonic music as 3-D (one-hot) tensors, in which the first two dimensions denote time and pitch and the third dimension indicates whether the token is an onset, sustain or rest. A common way for deep learning models to encode/decode a piano-roll is to use recurrent layers along the time-axis while the pitch-axis relations are modeled in various ways [20, 21, 23]. Another method is to regard a piano-roll as an image with three channels (onset, sustain and rest) and apply convolutional layers [7, 22].

Through the proposal of PianoTree VAE, we argue that a major way to improve the current deep learning models is to utilize the built-in priors (intrinsic structure) in the musical data. In our work, we primarily use the sparsity and the hierarchical priors.

2.3 MIDI-like Event Sequence and Compatible Models

MIDI-like event sequence is first used in deep music generation in performanceRNN [24] and Multi-track Music-VAE [9], and then broadly applied in transformer-based generation [19, 25, 26]. This direction of work leverages the sparsity of polyphonic data to efficiently flatten polyphonic music into an array of events. The vocabulary size of events usually tripples the vocabulary size of MIDI pitches, including "note-on" and "note-off" events for 128 MIDI pitches, "time shifts", and so on.

However, the format of MIDI-like events lacks the proper flexibility. A few operations are made difficult due to its very nature. For instance, during addition or deletion of notes, often numerous "time shift" tokens must be merged or split with the "note-on" or "note-off" tokens being changed all-together. This has caused the model being trained inefficient for the potential generation tasks. In addition, this format has a risk of generating illegal sequences, say a "note on" message without a paired "note off" message.

Similarly, we see the note-based approaches [27, 28], in which polyphonic music is represented as a sequence of note tuples, as an alternative to the MIDI-like methods. The representation has resolved the illegal generation problem but still not revealed much of the intrinsic music structure. We argue that our work improves on the note-based approaches by utilizing deeper musical structures implied by the data. (See Section 3.1 for details.)

2.4 GNN as a Novel Structure

Recently, we see a trend in using graph neural networks (GNN) [29] to represent polyphonic score [30], in which each vertex represents a note and the edges represent different musical relations. Although the GNN-based model offers sparse representation learning capacity, it is limited by the specification of the graph structure design and it is nontrivial to generalize it for score generations.

3. METHOD

3.1 Data Structure
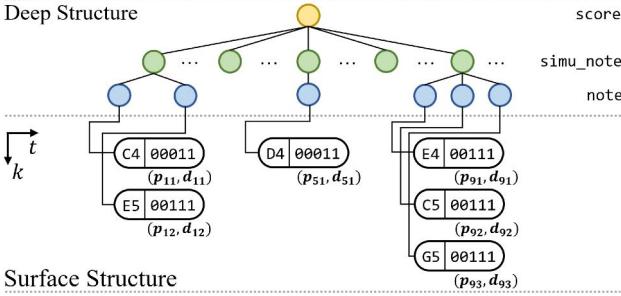
We first define a data structure to represent a polyphonic music segment, which contains two components: 1) *surface structure*, a data format to represent the music observation, and 2) *deep structure*, a tree structure (containing `score`, `simu_note` and `note` nodes) showing the syntactic construct of the music segment.

Each music segment lasts T time steps with $\frac{1}{4}$ beat as the shortest unit. We further use K_t, where $1 \leq t \leq T$ to denote the number of notes having the same onset t. The current model uses $T = 32$, i.e., each music segment is 8-beat long.

3.1.1 Surface Structure

The surface structure is a nested array of *pitch-duration* tuples, denoted by $\{(p_{t,k}, d_{t,k}) | 1 \leq t \leq T, 1 \leq k \leq K_t\}$. Here, $(p_{t,k}, d_{t,k})$ is the kth lowest note starting at time step t. The pitch attribute $p_{t,k}$ is a 128-D one-hot vector corresponding to 128 MIDI pitches. The duration attribute $d_{t,k}$ encodes the duration ranging from 1 to T using a $\log_2 T$-bit binary vector. For example, when $T = 32$ ($\log_2 T = 5$), '00000' represents a 16th note, '00001' is an 8th note, '00010' is a dotted 8th note, and so on so forth. The base-2 design is inspired by the similar binary relation among different note values in western musical notation.

The bottom part of Figure 2 illustrates the surface structure of the music example in Figure 1. We see that the data structure is a sparse encoding of music, and it eliminates illegal tokens since every possible nested array has a correspondent music.

Figure 2: An illustration of PianoTree data structure to encode the music example in Figure 1.

3.1.2 Deep Structure

We further build a syntax tree to reveal the hierarchical relation of the observation. First, for $1 \leq t \leq T, 1 \leq k \leq K_t$, we define $\texttt{note}_{t,k}$ as the summary (i.e., embedding) of $(p_{t,k}, d_{t,k})$, which are the bottom layers of the tree. Then, for $1 \leq t \leq T$, we define $\texttt{simu\_note}_t$ as the summary of $\texttt{note}_{t,1 \leq k \leq K_t}$, which are the middle layers of the tree. Finally, we define the $\texttt{score}$ as the summary of $\texttt{simu\_note}_{1 \leq t \leq T}$, which is the root of the tree. The upper part of Figure 2 illustrates the deep structure built upon its surface structure.
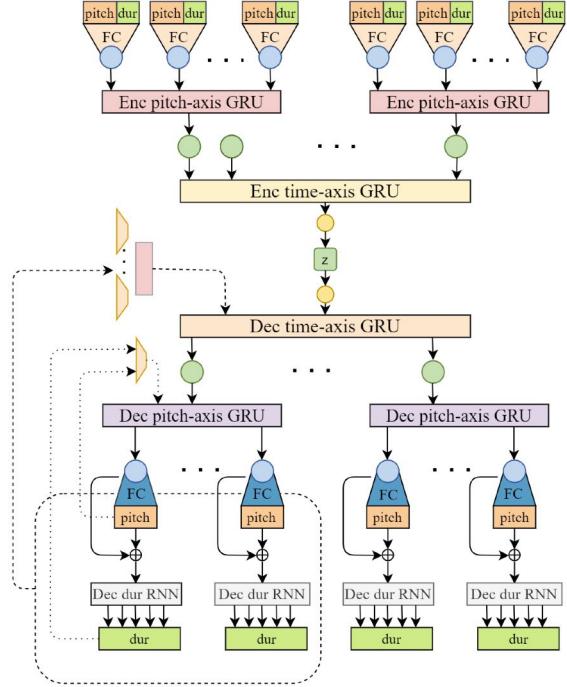
The syntax tree, so-called the deep structure has both musical and linguistic consideration. In terms of music, $\texttt{note}$, $\texttt{simu\_note}$ and $\texttt{score}$ roughly reflect the musical concept of a note, chord and grouping. In terms of linguistics, the tree is analogous to a constituency tree, with surface structure being the terminal nodes and deep structure being the non-terminals. Recent studies in natural language processing have revealed that incorporating natural language syntax results in better semantics modeling [31, 32].

3.2 Model Structure

We use the surface structure of polyphonic music as the model input. The VAE architecture is built upon the deep structure.

We denote the music segment in the proposed surface structure as x and the latent code as z, which conforms to a standard Gaussian prior denoted by $p(z)$. The encoder models the approximated posterior $q_\phi(z|x)$ in a bottom-up order of the deep structure. First, $\texttt{note}$ embeddings are computed through a linear transform of pitch-duration tuples. Second, the $\texttt{note}$ embeddings (sorted by pitch) are then embedded into $\texttt{simu\_note}$ using a bi-directional GRU [33] by concatenating the last hidden states on both ends. With the same method, the $\texttt{simu\_note}$ embeddings (sorted by onsets) are summarized into $\texttt{score}$ by another bi-directional GRU. We assume an isotropic Gaussian posterior, whose mean and log standard deviation are computed by a linear mapping of $\texttt{score}$. Algorithm 1 shows the details.

The decoder models $p_\theta(x|z)$ in a top-down order of the deep structure, almost mirroring the encoding process. We use a uni-directional time-axis GRU to decode $\texttt{simu\_note}$, another uni-directional (pitch-axis) GRU to decode $\texttt{note}$, a fully connected layer to decode pitch at-

Figure 3: An overview of the model architecture. The recurrent layers are represented by rectangles and the fully-connected (FC) layers are represented by trapezoids. The $\texttt{note}$, $\texttt{simu\_note}$ and $\texttt{score}$ events are represented by circles.

tributes, and finally another GRU to decode duration attribute starting from the most significant bit. Algorithm 2 shows the details.

We use the ELBO (evidence lower bound) [34] as our training objective. Formally,

$$\mathcal{L}(\phi, \theta; x) = -\mathbb{E}_{z \sim q_\phi} \log p_\theta(x|z) + \beta \mathrm{KL}\Big(q_\phi || p(z)\Big), \quad (1)$$

where β is a balancing parameter used in β-VAE [35].

We denote the embedding size of $\texttt{note}$, $\texttt{simu\_note}$ and $\texttt{score}$ as e_n, e_sn and e_sc; the dimension of latent space as d_z; and the hidden dimensions or pitch-axis, time-axis and dur GRUs as h_p, h_t and h_d respectively. In this work, we report our result on the following model size: $e_\text{n} = 128$, $e_\text{sn} = h_\text{p,dec} = 2 \times h_\text{p,enc} = 512$, $e_\text{sc} = h_\text{t,dec} = 2 \times h_\text{t,enc} = 1024$, $h_\text{d,dec} = 64$ and $d_z = 512$.

Algorithm 1: The **PianoTree Encoder**. n, sn, sc are short for $\texttt{note}$, $\texttt{simu\_note}$, $\texttt{score}$.

```
/* gru(·): passes a sequence to
   bi-directional GRU and ouputs the
   concatenation of hidden states from
   both ends.                          */
```
input: PianoTree
  $x = \{(p_{t,k}, d_{t,k}), 1 \leq t \leq T, 1 \leq k \leq K_t\}$
foreach t, k **do** $n_{t,k} \leftarrow \text{emb}_\text{enc}(p_{t,k}, d_{t,k})$;
foreach t **do** $sn_t \leftarrow \text{gru}_\text{enc}^\text{pitch}(n_{t,1:K_t})$;
$sc \leftarrow \text{gru}_\text{enc}^\text{time}(sn_{1:T})$;
$\mu \leftarrow \text{fc}_\mu(sc); \sigma \leftarrow \exp(\text{fc}_\sigma(sc))$;
return $q(z|x) = N(\mu, \sigma^2)$;

Algorithm 2: The **PianoTree Decoder**. We still use the abbreviation n, sn, and sc, defined in Algorithm 1

```
/* gru(·), same as Algorithm 1.           */
/* grucell(·,·):  updates the hidden state
   using the current input and the
   previous hidden state.  The output is
   replicated.                            */
```
input: latent representation z
$sc \leftarrow z$;
$\tilde{sn}_0, \tilde{n}_{:,0}, d_{:,:,0} = \text{<SOS>}$;
for $t = 1, 2, ...T$ **do**
$\quad [sn_t, sc] \leftarrow \text{grucell}^{\text{time}}_{\text{dec}}(\tilde{sn}_{t-1}, sc)$;
$\quad$ **for** $k = 1, 2, ...$ **do**
$\quad\quad [n_{t,k}, sn_t] \leftarrow \text{grucell}^{\text{pitch}}_{\text{dec}}(\tilde{n}_{t,k-1}, sn_t)$;
$\quad\quad p_{t,k} \leftarrow \text{softmax}(\text{fc}(n_{t,k}))$;
$\quad\quad$ **for** $r = 1, 2, ..., 5$ **do**
$\quad\quad\quad h = [n_{t,k}, p_{t,k}]$;
$\quad\quad\quad [y_{t,k,r}, h] = \text{grucell}^{\text{dur}}_{\text{dec}}(d_{t,k,r-1}, h)$;
$\quad\quad\quad d_{t,k,r} \leftarrow \text{softmax}(y_{t,k,r})$;
$\quad\quad$ **end**
$\quad\quad d_{t,k} = [d_{t,k,1:5}]$;
$\quad\quad$ **if** $p_{t,k} \neq \text{<EOS>}$ **then** $K_t \leftarrow k$; break;
$\quad\quad \tilde{n}_{t,k} \leftarrow \text{emb}_{\text{enc}}(p_{t,k}, d_{t,k})$;
$\quad$ **end**
$\quad \tilde{sn}_t \leftarrow \text{gru}^{\text{pitch}}_{\text{enc}}(n_{t,1:K_t})$;
end
return $\{(p_{t,k}, d_{t,k}), 1 \leq t \leq T, 1 \leq k \leq K_t\}$;

4. EXPERIMENTS

In this section, we compare PianoTree VAE with several baseline models. We present the dataset in Section 4.1, baseline models in Section 4.2,and the training details in Section 4.3. We present the objective evaluation on reconstruction accuracy in Section 4.4. In Section 4.5, we inspect and visualize the latent space of note and simu_note. After that, we present the subjective evaluation on latent space traversal in Section 4.6. Finally, we apply the learned representation to downstream music generation task in Section 4.7.

4.1 Dataset

We collect around 5K classical and popular piano pieces from Musicalion [2] and the POP909 dataset [36]. We only keep the pieces with $\frac{2}{4}$ and $\frac{4}{4}$ meters and cut them into 8-beat music segments (i.e., each data sample in our experiment contains 32 time steps under sixteenth note resolution). In all, we have 19.8K samples. We randomly split the dataset (at song-level) into training set (90%) and test set (10%). All training samples are further augmented by transposing to all 12 keys.

4.2 Baseline Model Architectures

We train four types of baseline models in total using piano-roll (Section 2.2) and MIDI-like events (Section 2.3) data structures. As a piano-roll can be regarded as either a sequence or a 2-dimensional image, we couple it with

---

[2] Musicalion: https://www.musicalion.com.

three neural encoder-decoder architectures: a recurrent VAE (**pr-rnn**), a convolutional VAE (**pr-cnn**), and a fully-connected VAE (**pr-fc**). For the MIDI-like events, we couple it with a recurrent VAE model (**midi-seq**). All models share the same latent space dimension ($d_z = 512$). Specifically,

- The piano-roll recurrent VAE (**pr-rnn**) model is similar to a 2-bar MusicVAE proposed in [4]. The hidden dimensions of the GRU encoder and decoder are both 1024.
- The piano-roll convolutional VAE (**pr-cnn**) architecture adopts a convolutional–deconvolutional architecture. The encoder contains 8 convolutional layers with kernel size 3×3. Strided convolution is performed at the 3rd, 5th, 7th and 8th layer with stride size $(2 \times 1), (2 \times 3), (2 \times 2)$ and (2×2) respectively. The decoder adopts the deconvolution operations in a reversed order.
- The piano-roll fully-connected VAE (**pr-fc**) architecture uses a time-distributed 256-dimensional embedding layer, followed by 3 fully-connected layers with the hidden dimensions [1024, 768] for the encoder. The decoder adopts the counter-operations in a reversed order.
- The MIDI-like event recurrent VAE (**midi-seq**) adopts the recurrent model structure similar to **pr-rnn**. Here, the event vocabulary contains 128 "note-on", 128 "note-off" and 32 "time shift" tokens. The embedding size of a single MIDI event is 128. The hidden dimensions of the encoder GRU and decoder GRU are 512 and 1024 respectively.

4.3 Training
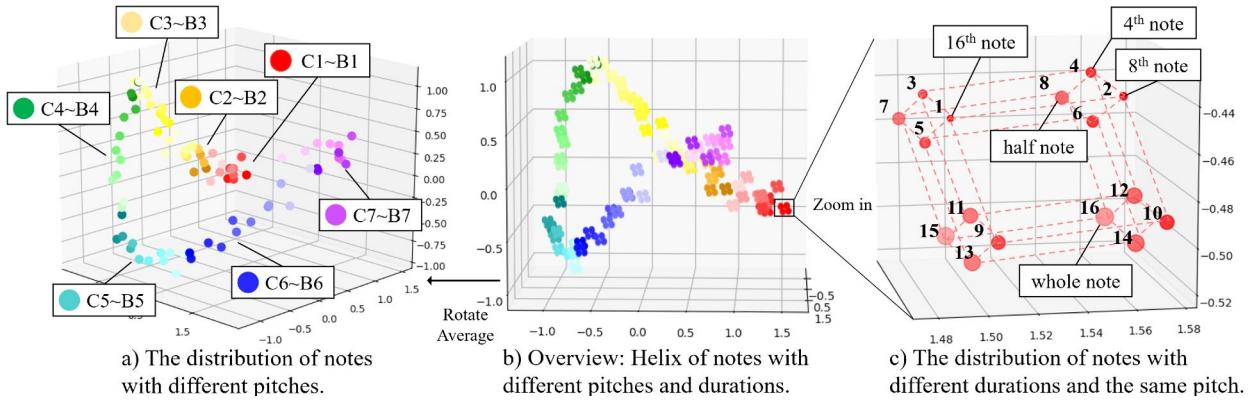
For all models, we set batch size = 128 and use Adam optimizer [37] with a learning rate starting from 1e-3 with exponential decay to 1e-5. For PianoTree VAE, we use teacher forcing [38] for decoder time-axis and pitch-axis GRU and for other recurrent-based baselines, we use teacher forcing in the decoders. The teacher forcing rates start from 0.8 and decay to 0.0. PianoTree VAE converges within 6 epochs, and the baseline models converge in approximately 40 to 60 epochs.

Models	PianoTree	midi-seq	pr-rnn	pr-cnn	pr-fc
Onset Precision	**0.9558**	0.8929	0.9533	0.9386	0.9211
Onset Recall	**0.9532**	0.6883	0.9270	0.8818	0.8827
Onset F1	**0.9545**	0.7774	0.9399	0.9093	0.9015
Duration Precision	**0.9908**	0.3826	0.9777	0.9757	0.9688
Duration Recall	0.9830	**0.9899**	0.9891	0.9796	0.9743
Duration F1	**0.9869**	0.5519	0.9834	0.9777	0.9715

Table 1: Objective evaluation results on reconstruction criteria. PianoTree is our proposed method. Other columns correspond to the baseline models described in Section 4.2.

4.4 Objective Evaluation of Reconstruction

The objective evaluation is performed by comparing different models in terms of their reconstruction accuracy of pitch onsets and note duration [39, 40], which are commonly used measurements in music information retrieval tasks. For note duration accuracy, we only consider the

a) The distribution of notes with different pitches.

b) Overview: Helix of notes with different pitches and durations.

c) The distribution of notes with different durations and the same pitch.

Figure 4: A visualization of `note` embeddings after dimensionality reduction using PCA.

notes whose onset and pitch reconstruction is correct. Table 1 summarizes the results where we see that the PianoTree VAE (the 1st column) is better than others in terms of F1 score for both criteria.

4.5 Latent Space Visualization

Figure 4 shows the *latent note space* by plotting different `note` embeddings after dimensionality reduction by PCA (with the three largest principal components being reserved). Each colored dot is a `note` embedding and a total of 1344 samples are displayed; note pitch ranges from C-1 to C-8 and note duration from a sixteenth note to a whole note.

We see that the `note` embeddings have the desired geometric properties. Figure 4 (a) & (b) show that at a macro level, notes with different pitches are well sorted and form a "helix" in the 3-D space. Figure 4 (c) further shows that at a micro level, 16 different note durations (with the same pitch) form a "fractal parallelogram" due to the binary encoding of duration attributes. One of the advantages of the encoding method is the translation invariance property. For example, the duration difference between the upper left cluster and the lower left cluster is 8 semiquavers, so is the difference between the upper right and lower right cluster. The same property also applies to the four smaller-scale parallelograms.

a) The relative relationship of different major chords.

b) Ring structure of chords with different degree scales in C major.

Figure 5: A visualization of `simu_note` embeddings after dimensionality reduction using PCA.

Figure 5 is a visualization of the latent chord space by plotting different `simu_note` embeddings under PCA di-

mensionality reduction. Each colored cluster corresponds to a chord label realized in 343 different ways (we consider all possible pitch combinations within three octaves, with a minimum of 3 notes and a maximum of 9 notes). The duration for all chords is one beat.
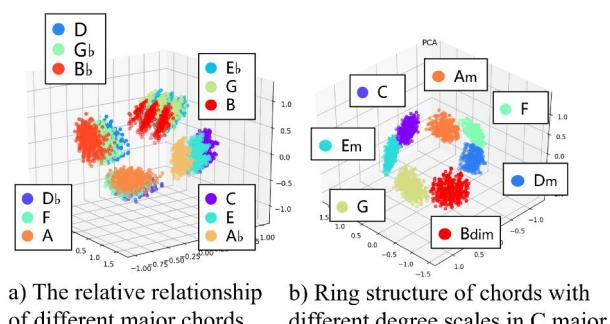
The geometric relationships among different chords are consistent and human interpretable. In specific, Figure 5 (a) shows the distribution of 12 different major chords, which are clustered in four different groups. By unfolding the circle in a counterclockwise direction, we can observe the existence of *the circle of the fifth*. Figure 5 (b) is the visualization of seven C major triad chords: forming a ring in the order of 1-3-5-7-2-4-6 degree in the counterclockwise direction.

4.6 Subjective Evaluation of Latent Space Interpolation

Latent space traversal [4, 5, 41] is a popular technique to demonstrate model generalization and the smoothness of the learned latent manifold. When interpolating from one music piece to another in the latent space, new pieces can be generated by mapping the representations back to the signals. If a VAE is well trained, the generated piece will sound natural and form a smooth transition.

To this end, we invite people to subjectively rate the models through a double-blind online survey. During the survey, the subjects first listen to a pair of music, and then listen to 5 versions of interpolation, each generated by a model listed in Table 1. Each version is a randomly selected pair of music segments, and the interpolation is achieved using SLERP [42]. Since the experiment requires careful listening and a long survey could decrease the quality of answers, each subject is asked to rate only 3 pairs of music, i.e., $3 \times 5 = 15$ interpolations in a random order. After listening to the 5 interpolations of each pair, subjects are asked to select two best versions: one in terms of the *overall musicality*, and the other in terms of the *smoothness of transition*.
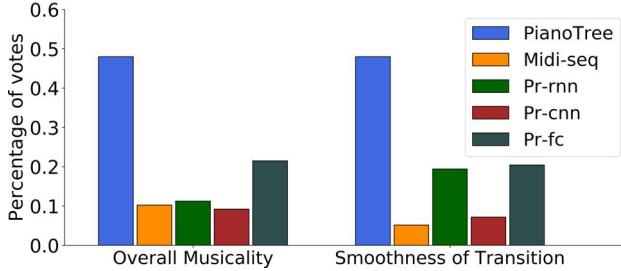
A total of n = 33 subjects (12 females and 21 males) with different music backgrounds have completed the survey. The aggregated result (as in Figure 6) shows that the interpolations generated by our model are better than the ones generated by baselines, in terms of both overall musi-

cality and smoothness of transition. Here, different colors represent different models (with the blue bars being our model and other colors being the baselines), and the height of the bars represent the percentage of votes (on the best candidate).

Figure 6: Subjective evaluation results of latent space interpolation.

4.7 Downstream Music Generation
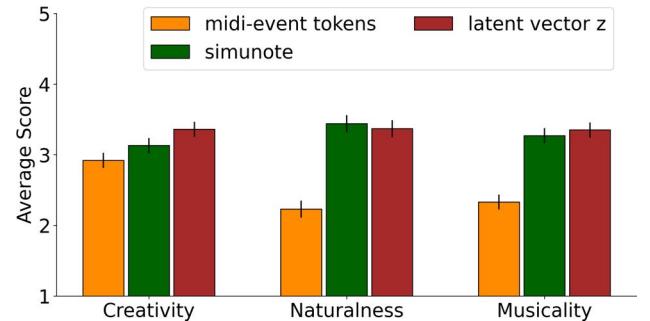
In this section, we further explore whether the polyphonic representation helps with *long-term music generation* when coupled with standard downstream sequence prediction models. (Similar tasks have been applied to *monophonic music* in [43] and [6].)

The generation task is designed in the following way: given 4 measures of piano composition, we predict the next 4 measures using a Transformer decoder (as in [44]). We compare three different music representations: MIDI-like event sequence (Section 2.1), pretrained (decoder) `simu_note` embeddings, and latent vector z for every 2-measure music segment (without overlap). Here z is the mean of the approximated posterior from the encoder. For all three representations, we use the same Transformer decoder architecture (outputs of dimension = 128, number of layers = 6 and number of heads = 8) with the same training procedure. Only the loss functions are correspondingly adjusted based on different representations: cross entropy loss is applied to midi-event tokens and MSE loss is applied to both `simu_note` and latent vector z. We use the same datasets mentioned in Section 4.1 and cut the original piano pieces into 8-measure subsequent clips for generation purposes. We still keep 90% for training and 10% for testing.

We then invited people to subjectively rate different music generations through a double-blind online survey (similar to the one in Section 4.6). Subjects are asked to listen to and rate 6 music clips, each of which contains 3 versions of 8-measure generation using different note representations. Subjects are told that the first 4 measures are given and the rest are generated by the machine. For each music clip, subjects rate it based on *creativity*, *naturalness* and *musicality*.
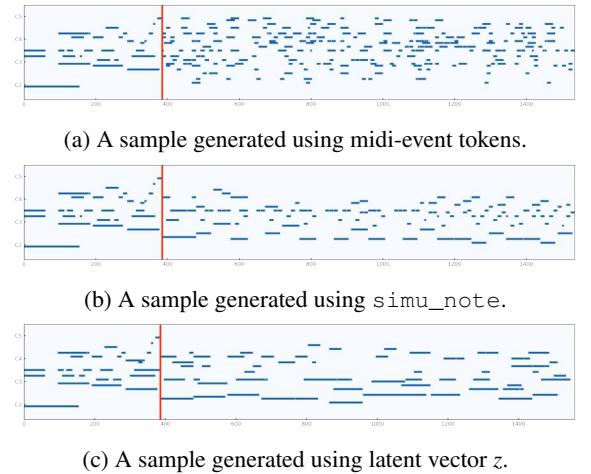
A total of n = 48 subjects (20 females and 28 males) with different music backgrounds have participated in the survey. Figure 7 summarizes the survey results, where the heights of bars represent means of the ratings and the error bars represent the confidence intervals computed via within-subject ANOVA [45]. The result shows that `simu_note` and latent vector z perform significantly better

than the midi-event tokens in terms of all three criteria (p < 0.005).

Figure 7: Subjective evaluation results of downstream music generation.

Besides the aforementioned generation task, we also iteratively feed the generated 4-measure music clips into the model to get longer music compositions. Figure 8 shows a comparison of 16-measure generation results using all three representations. The first 4 bars are selected from the test set, and the subsequent 12 bars are generated by the models. Generally speaking, using `simu_note` and latent vector z as data representations yields more coherent music compositions. Furthermore, we noticed that long generations using the `simu_note` representation tend to repeat previous steps in terms of both chords and rhythms, while those generations using the latent vector z usually contain more variations.

(a) A sample generated using midi-event tokens.

(b) A sample generated using `simu_note`.

(c) A sample generated using latent vector z.

Figure 8: Long music generations given first 4 measures.

5. CONCLUSION AND FUTURE WORK

In conclusion, we proposed PianoTree VAE, a novel representation-learning model tailored for polyphonic music. The key design of the model is to incorporate both the music data structure and model architecture with the sparsity and hierarchical priors. Experiments show that with such inductive biases, PianoTree VAE achieves better reconstruction, interpolation, downstream generation, and strong model interpretability. In the future, we plan to extend PianoTree VAE for more general musical structures, such as motif development and multi-part polyphony.

6. REFERENCES

[1] T. Zhao, R. Zhao, and M. Eskenazi, "Learning discourse-level diversity for neural dialog models using conditional variational autoencoders," *arXiv preprint arXiv:1703.10960*, 2017.

[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[3] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in neural information processing systems*, 2015, pp. 2980–2988.

[4] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," *arXiv preprint arXiv:1803.05428*, 2018.

[5] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," *arXiv preprint arXiv:1906.03626*, 2019.

[6] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," *arXiv preprint arXiv:1907.01164*, 2019.

[7] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[8] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.

[9] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, "Learning a Latent Space of Multitrack Measures," *arXiv e-prints*, p. arXiv:1806.00195, Jun 2018.

[10] F. Lerdahl and R. S. Jackendoff, *A generative theory of tonal music*. MIT press, 1996.

[11] J. Rothgeb, *Introduction to the theory of Heinrich Schenker: the nature of the musical work of art*. New York: Longman, 1982.

[12] M. Hamanaka, K. Hirata, and S. Tojo, "Implementing "a generative theory of tonal music"," *Journal of New Music Research*, vol. 35, no. 4, pp. 249–277, 2006.

[13] ——, "σgttm iii: Learning-based time-span tree generator based on pcfg," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2015, pp. 387–404.

[14] S. W. Smoliar, "A computer aid for schenkerian analysis," in *Proceedings of the 1979 annual conference*, 1979, pp. 110–115.

[15] A. Marsden, "Schenkerian analysis by computer: A proof of concept," *Journal of New Music Research*, vol. 39, no. 3, pp. 269–289, 2010.

[16] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1362–1371.

[17] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2837–2846.

[18] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *International Society for Music Information Retrieval (ISMIR)*, 2017.

[19] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "Music transformer: Generating music with long-term structure," *arXiv preprint arXiv:1809.04281*, 2018.

[20] G. Brunner, Y. Wang, R. Wattenhofer, and J. Wiesendanger, "Jambot: Music theory aware chord based generation of polyphonic music with lstms," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2017, pp. 519–526.

[21] H. H. Mao, T. Shin, and G. Cottrell, "Deepj: Style-specific music generation," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. IEEE, 2018, pp. 377–382.

[22] E. S. Koh, S. Dubnov, and D. Wright, "Rethinking recurrent latent variable model for music composition," in *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2018, pp. 1–6.

[23] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.

[24] I. Simon and S. Oore, "Performance rnn: Generating music with expressive timing and dynamics," https://magenta.tensorflow.org/performance-rnn, 2017.

[25] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," *arXiv preprint arXiv:1907.04868*, 2019.

[26] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Generating music with rhythm and harmony," *arXiv preprint arXiv:2002.00212*, 2020.

[27] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

[28] C. Hawthorne, A. Huang, D. Ippolito, and D. Eck, "Transformer-nade for piano performances."

[29] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[30] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Graph neural network for music score data and modeling expressive piano performance," in *International Conference on Machine Learning*, 2019, pp. 3060–3070.

[31] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith, "Recurrent neural network grammars," *arXiv preprint arXiv:1602.07776*, 2016.

[32] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[33] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[35] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[36] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proceedings of 21st International Conference on Music Information Retrieval (ISMIR), virtual conference*, 2020.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] N. B. Toomarian and J. Barhen, "Learning a trajectory using adjoint functions and teacher forcing," *Neural networks*, vol. 5, no. 3, pp. 473–484, 1992.

[39] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," *arXiv preprint arXiv:1710.11153*, 2017.

[40] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.

[41] R. Yang, T. Chen, Y. Zhang, and G. Xia, "Inspecting and interacting with meaningful music representations using vae," *arXiv preprint arXiv:1904.08842*, 2019.

[42] A. Watt and M. Watt, "Advanced animatidn and bendering technidues," 1992.

[43] K. Chen, G. Xia, and S. Dubnov, "Continuous melody generation via disentangled short-term representations and structural conditions," *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pp. 128–135, 2020.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017.

[45] H. Scheffe, *The analysis of variance*. John Wiley & Sons, 1999, vol. 72.

HIERARCHICAL MUSICAL INSTRUMENT SEPARATION

Ethan Manilow
Northwestern University
`ethanm@u.northwestern.edu`

Gordon Wichern   Jonathan Le Roux
Mitsubishi Electric Research Laboratories (MERL)
`wichern@merl.com    leroux@merl.com`
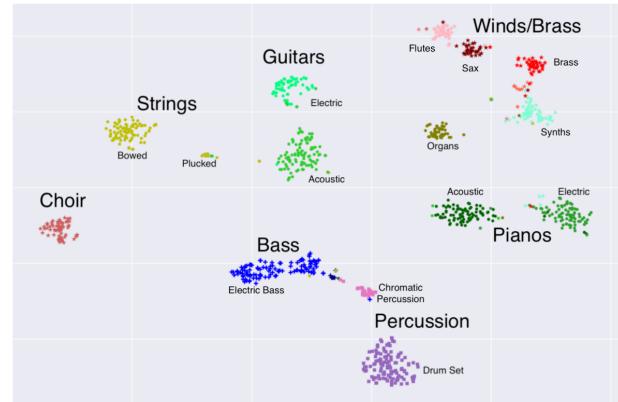
ABSTRACT

Many sounds that humans encounter are hierarchical in nature; a piano note is one of many played during a performance, which is one of many instruments in a band, which might be playing in a bar with other noises occurring. Inspired by this, we re-frame the musical source separation problem as hierarchical, combining similar instruments together at certain levels and separating them at other levels. This allows us to deconstruct the same mixture in multiple ways, depending on the appropriate level of the hierarchy for a given application. In this paper, we present various methods for hierarchical musical instrument separation, with some methods focusing on separating specific instruments (like guitars) and other methods that determine what to separate based on a user-supplied audio example. We additionally show that separating all hierarchy levels is possible even when training data is limited at fine-grained levels of the hierarchy.

1. INTRODUCTION

The field of source separation has seen notable performance improvements with the introduction of deep learning techniques, most notably in the areas of speech enhancement [1–4], speech separation [5–8], and music separation [9–12]. These techniques succeed in cases where the notion of a source is well defined; in the case of speech enhancement or separation, the target is always defined as the speech of a single speaker. However, real-world scenarios can have more complicated definitions of a source. Consider the case where a band is playing on the radio while two people are having a conversation: how does one segment this audio scene? Is the radio one source and the talkers each a source? Or are each of the instruments in the band on the radio a source as well? Clearly, there are many correct answers to this question, but one way to understand this auditory scene is to apply a hierarchical structure to its parts. In this work, we re-frame the source separation problem as hierarchical, focusing on the example of music source separation, where we use musical instruments as elements in a complex hierarchical auditory scene.

When considering music separation, determining what

Figure 1. Annotated t-SNE [13] projection of the learned anchors from a hierarchical query-by-example separation model on a test set.

constitutes a target source is not well defined. Even in a well-studied problem like singing voice separation [9–11], in which the singer is isolated from non-vocal background music, the definition of what is "singing voice" is somewhat muddled. Many popular songs often contain a lead vocal part, possibly several additional background vocal parts, and sometimes additional vocal effect tracks. This is a simple case; when we consider instrument categories with a larger variety of possible timbres, like synthesizers or guitars, deciding what particular instrument part to isolate can become even harder to nail down. One may want to go even further and separate each instrument into unique notes or chord instances.

Framing a musical scene as hierarchical has precedent in fields that study human audition. Evidence shows that human auditory perception has many hierarchical characteristics [14–17]. As Bregman notes in *Auditory Scene Analysis* [18]: "It makes sense [...] to think of the auditory perceptual organization of [a musical] duet as having a hierarchical structure [...]. This argument implies that there are levels of perceptual belongingness intermediate between 'the same thing' and 'unrelated things'". While perceptual auditory hierarchies can involve timing, timbre, rhythm, and much more, in this paper we focus on the task of building hierarchical source separation systems via an instrument hierarchy.

In the field of musicology, musical instruments have long been thought of as hierarchical. Almost all human cultures throughout history have created musical instrument classification systems [19], many of which are inherently hierarchical. One prominent example is the

Hornbostel-Sachs system [20], which classifies musical instruments by their sound production mechanisms in a hierarchical manner similar to the Dewey Decimal System [21]. Another system widely used in Western music classifies instruments by their musical range, with terms named after singing voice classifications: *soprano*, *alto*, etc. We use a hierarchy inspired by both of these approaches for hierarchical source separation.

There is also an element of a musical instrument hierarchy when making recordings in the recording studio. Each track is assumed to be an isolated recording of a single instrument, or part of one instrument. At the mixing board, a sound engineer can mix together multiple tracks into a "submix", which acts as a single unit in the recording session, having effects and other signal routing configurations be specific to the submix rather than the individual tracks therein [22]. The submixes are then manipulated alongside other tracks, which may contain only a single instrument. For example, a standard practice is to record every separate piece of a drum kit with a single microphone and then combine those into a drum submix. This configuration is hierarchical; the engineer can choose to manipulate all of the drum sounds (the drum submix) or manipulate individual drum tracks within it.

In this paper, we re-frame the problem of musical source separation as hierarchical. We propose two main strategies for hierarchical source separation, one solely based on the well-studied source-specific mask inference approach to source separation [3], and another based on more recently proposed query-by-example source separation systems [23, 24]. In both cases, we learn to simultaneously separate submixes of instruments corresponding to multiple levels of an instrument label hierarchy. By learning to separate sources at multiple levels of granularity, we observe performance benefits even in cases where training data is limited for the most fine-grained source types.

2. RELATED WORK

Music source separation has recently seen a great deal of success. Most of this success is owed to the availability of the MUSDB18 [25] dataset. This dataset has avoided the "source definition ambiguity" by grouping all audio within a track into four target sources: vocals, bass, drums, and other. The "other" source contains a variety of different instruments, like guitars, pianos, strings, and synthesizers. While MUSDB18 has undoubtedly helped to advance the field of music source separation, its source groupings remain overly coarse for many real-world remixing applications. In this work, we propose systems to separate sources historically grouped as the "other" source.

Our proposed work is related to source separation algorithms that attempt to estimate multiple musical sources with one network. Some works accomplish this by outputting a set of masks for each target source, improving performance via specialized training techniques [26, 27] or by giving the networks additional tasks to solve, like music transcription [28]. Other works accomplish this by conditioning a network to output different sources depending on

the desired source [23, 29, 30]. None of these approaches have any requirements that the sources they separate have any inherent structure in relation to other sources, especially not in a hierarchical manner as we propose here.

This work also draws inspiration from query-by-example (QBE) networks. Within the speech separation literature, the task of using a query to separate a specific speaker from a mixture with many speakers is called speaker extraction, and this task has garnered much attention recently [31–33]. Specifically, this work builds off of work [34] that extends deep attractor networks [35] for the QBE case. Deep attractor networks have been successfully used for music separation [23, 36], where QBE music separation was considered as an auxiliary benefit of the learned embedding space. Although systems specifically tailored to QBE separation of musical instruments have also been proposed [24], none of these systems assume or enforce any hierarchical structure on an auditory scene.

3. AUDITORY HIERARCHIES

In this work, we are interested in hierarchies of sound producing objects, where top levels of the hierarchy correspond to broad groups (e.g., midrange stringed instruments) and lower levels are more specific (e.g., acoustic guitar). With regard to source separation, we can define an auditory hierarchy such that sources at higher levels in the hierarchy are composed of mixtures of sources at lower levels of the hierarchy. Each source node can potentially be further separated into child sources and combined with its siblings to create parent sources. Considering a hierarchy with L levels, we denote by $\mathcal{S}_{l,c}$ the c-th source type node at hierarchy level l, for $l = 1, \ldots, L$, where we assume that the set of leaf source types $\mathcal{S}_{1,c}$ cannot be decomposed into further source types, and $\mathcal{S}_{L,1}$ is the sole source type at the top of the hierarchy and includes all source types. Further denoting by $\mathcal{C}_{l,c}$ the set of indices of the child sources at level $l - 1$ of $\mathcal{S}_{l,c}$, the hierarchy can be defined as

$$\mathcal{S}_{l,c} = \bigcup_{c' \in \mathcal{C}_{l,c}} \mathcal{S}_{l-1,c'}, \forall l = 2, \ldots, L. \qquad (1)$$

We define a *path* down the hierarchy as a sequence of source types from a beginning source type node $\mathcal{S}_a$ to a destination source type node $\mathcal{S}_b$ at a lower level.

When using this hierarchy to decompose a mixture x, we denote by $\mathbf{S}_{l,c}$ the corresponding source component in x whose source type is $\mathcal{S}_{l,c}$, where the submix of all signals of the same type are considered as a single component. By definition, $\mathbf{S}_{L,1} = x$. Each c-th source component $\mathbf{S}_{l,c}$ at a level l can be decomposed into source components $\mathbf{S}_{l-1,c'}$, such that $\mathbf{S}_{l-1,c'}$ is the signal corresponding to all sources belonging to the child source type $\mathcal{S}_{l-1,c'}$:

$$\mathbf{S}_{l,c} = \sum_{c' \in \mathcal{C}_{l,c}} \mathbf{S}_{l-1,c'}, \text{ s.t. } \mathbf{S}_{l-1,c'} \in \mathcal{S}_{l-1,c'}, \qquad (2)$$

for $l = 2, \ldots, L$. For simplicity, we use the sum operator to denote mixing, although the mixing process is often more complex than a simple summation of signals.

In this paper, we specifically examine auditory hierarchies composed of mixtures of musical instruments, but

we note that this hierarchical formulation can be applied to mixtures with any type of source content.

4. HIERARCHICAL SOURCE SEPARATION

In general, source separation is formulated as trying to estimate C complex spectrograms, $\boldsymbol{S}_c \in \mathbb{C}^{F \times T}$ for $c = 1, \ldots, C$, that represent a set of desired sources within the spectrogram $X \in \mathbb{C}^{F \times T}$ of an audio mixture. In this general formulation, there is no requirement that source $\boldsymbol{S}_c$ have any relationship to source $\boldsymbol{S}_d$, for $c \neq d$.

Given an audio mixture X, a hierarchical separation algorithm under a given hierarchy may attempt to extract a submix of all sources belonging to some source type $\mathcal{S}_{l,c}$ at a level l. For instance, separating out all guitars (acoustic and electric) from a mixture that includes electric guitar, acoustic guitar, piano, and drums (as depicted in Fig. 2).

4.1 Hierarchical Source-Specific Separation
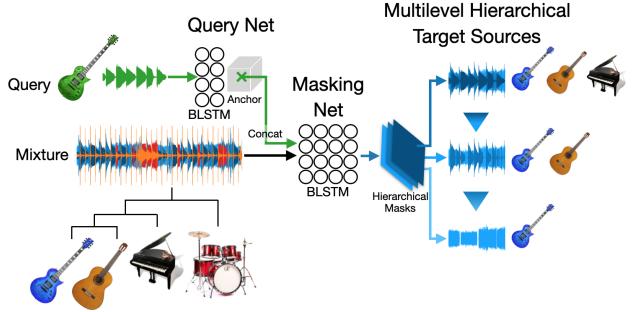
Conventional source-specific separation (SSS) networks based on mask inference typically attempt to estimate a real-valued mask $\hat{M}_c \in \mathbb{R}^{F \times T}$ for a single target source c by minimizing some distortion measure between the source estimate obtained from the mask and a reference $\boldsymbol{S}_c$. A commonly used example of such an objective function, which we use in this work, is the truncated phase sensitive approximation (tPSA) objective [3]:

$$\mathcal{L}_{\text{tPSA}} = \left\| \hat{M}_c \odot |X| - \text{T}_0^{|X|} \left(|\boldsymbol{S}_c| \odot \cos(\angle \boldsymbol{S}_c - \angle X) \right) \right\|_1, \tag{3}$$

where $\odot$ denotes element-wise product, $|Y|$ and $\angle Y$ denote the magnitude and phase of a spectrogram Y, and $\text{T}_0^{|X|}(x) = \min(\max(x,0),|X|)$ is a truncation function ensuring the target can be reached with a sigmoid activation function. The estimated mask $\hat{M}_c$ is element-wise multiplied with the original mixture spectrogram X to obtain an estimate for the target source $\boldsymbol{S}_c$.

As a first naive strategy for building hierarchical SSS networks, we can train single networks which output a single node $\mathcal{S}_{n,c}$ at a given level of the hierarchy. Each such single-level network can be trained to minimize the tPSA objective above, where the target source is $\boldsymbol{S}_{n,c}$, the component corresponding to the targeted source type in the hierarchy within the mixture X. Each of these networks outputs one mask $\hat{M}_{n,c}$ for its targeted source type, and they are trained independently of each other.

In order to make further use of the hierarchical structure of the data, we propose a multi-level strategy in which we train a network to output multiple levels of the hierarchy at once. A potential advantage of this strategy is that the network may be able to leverage learned knowledge about a mask $\hat{M}_{n,c}$ to reinforce and improve its estimate for another mask $\hat{M}_{n',c'}$ in the hierarchy. A trivial implementation of this strategy would be to output a mask for each leaf node in the hierarchy, and recompose the leaf sources as we travel through the hierarchy, training the network by combining loss functions for all nodes

Figure 2. One of the proposed methods for hierarchical source separation. We assume that a mixture contains a hierarchy of musical instruments (bottom left), and use an audio query (the green electric guitar, top left) to separate instruments at multiple levels of the hierarchy, with the closest target at the lowest level (blue electric guitar).

in the hierarchy. However, any sufficiently realistic hierarchy likely contains dozens of leaf nodes, leading to memory and computation issues as well as difficulties balancing the contributions of all the losses. To avoid these issues, we consider instead a single network that outputs N masks for N levels along a single path down the hierarchy, e.g., [strings/keys] → [guitars] → [clean guitars] ("Clean" indicates acoustic and electric guitars with no overdrive or distortion applied).

4.2 Hierarchical Query-by-Example

The approaches described above cannot capture many instruments in an instrument hierarchy: using one network per level only allows the network to learn one node in the hierarchy at a time, and using a multilevel network only learns one path down the instrument hierarchy. If we want to capture relationships between different instruments in a hierarchy, we need a method for separating multiple instruments at different levels with a single network.

A successful recent strategy involves query-by-example (QBE) networks that ingest a mixture and an example of the desired source to separate from the mixture [24]. By extending this to a hierarchical case, we can model an entire instrument hierarchy for source separation. Note that, instead of conditioning on a query, we could alternatively condition the separation on the leaf node label, leading to a hierarchical extension of conditional source separation methods [23, 29, 30]. We here focus on QBE, as an audio query can be considered as a generalization of a class label, and QBE may further provide the ability to interpolate to unseen source types during inference.

Our proposed realization of hierarchical QBE relies on two networks, a query net and a masking net. The query net calculates a query anchor $A_q \in \mathbb{R}^k$ for some input query $Q \in \mathbb{R}^{F \times T_q}$ as a weighted sum of k-dimensional query embeddings $V_{q,i}$ produced by the network at each time-frequency bin $i = (f,t)$ of the query spectrogram space:

$$A_q = \frac{\sum_i P_{q,i} V_{q,i}}{\sum_i P_{q,i}}, \tag{4}$$

where $P_q \in \mathbb{R}^{FT_q}$ is a query presence vector for query Q, defined such that $P_{q,i} = 1$ if the magnitude at bin $i = (f,t)$ is above a threshold (set to -60 dB from the maximum in our experiments), and 0 otherwise. The query anchor A_q is concatenated with the frequency vector of the mixture X_t at each frame t, and used as input to the masking network, which produces, for each hierarchy layer n of interest, a mask $\hat{M}_{n,c}$ for a target source $S_{n,c}$ which is in the same node $\mathcal{S}_{n,c}$ of the hierarchy as the query Q. This architecture is depicted in Fig. 2.

This QBE system is trained to minimize the tPSA objective in Eq. 3 based on a target source $S_{n,c}$, where the target source used to train the network is here determined both by the query and a given level in the hierarchy. Other QBE systems [24] apply a loss directly on the query embedding space; while we leave this direction to future work, we note that we are already able to learn some form of hierarchical structure without introducing a specific loss on the embedding space, as exemplified in Fig. 1.

Using an acoustic guitar query as example, the training procedure for a hierarchical QBE system is as follows: an acoustic guitar query is used to train a network that attempts to extract the corresponding sources at the leaf node level, in which case the target will consist of the submix of all clean guitars in the mixture. Note that we leave the problem of separating instruments of the same fine-grained type (e.g., multiple clean guitars) using techniques such as permutation-invariant training [5, 6] for future work. The same acoustic guitar query may also be used to train a network that attempts to extract the corresponding sources one level up, in which case the target will consist of the submix of all guitars in the mixture, regardless of whether they are clean guitars or not. When there is no target in the mixture corresponding to the query at the given level of the hierarchy, the target is set to silence.

As with hierarchical SSS networks, we can make a single-level QBE network for each separate level in the hierarchy and only separate instruments at that level, as described in the above example, or we can make a single hierarchical multi-level QBE network that returns multiple (or even all) levels of the hierarchy. For the latter strategy, we can consider enforcing a hierarchical constraint on the masks, as described below.

4.3 Constraints on Hierarchical Masks

Assuming the components of a mixture exist in some hierarchy, we can leverage knowledge about its structure to impart constraints on the network. For instance, we can use the relationship defined in Eq. 2 to require the set of masks produced by a multi-level hierarchical network to follow the same structure as the hierarchy, namely that masks at higher levels be composed of masks at lower levels.

However, this would require us to output masks for every node in the hierarchy, which is infeasible for any sufficiently realistic hierarchy. Instead, we consider imposing a hierarchical constraint that does not depend on knowledge of the whole hierarchy. This hierarchical constraint requires that masks at higher levels in the hierarchy must

Level	Submixes to be separated
3	Keyboards, guitars, and orchestral strings
2	All guitars (both clean and effected)
1	Only clean guitars (both electric and acoustic)

Table 1. Contents of hierarchical levels used for training and testing the hierarchical single-instrument source-specific separation (SSS) networks[1]. Hierarchical SSS can only learn one path down the hierarchy at a time.

apportion at least the same amount of energy as masks at lower levels. More precisely, the mask at level l is set as

$$\hat{M}_l = \max(\hat{M}'_l, \hat{M}_{l-1}), \qquad (5)$$

where $\max$ is applied element-wise to every TF bin, and $\hat{M}'_l$ is the mask estimate output by the network for level l.

5. EXPERIMENTAL DESIGN

We design a set of experiments to determine the validity of our hierarchical source separation methods outlined above. We want to understand how well the proposed methods work in a hierarchical scenario. We look specifically at the case of a musical instrument hierarchy.

5.1 Dataset and Evaluation

To test the proposed methods in this paper, we required a large dataset with isolated sources of many instruments that could be combined in a hierarchical way. Specifically, we required a dataset with a wide variety of granular source labels, i.e., not only "guitars", but "acoustic guitars", "electric guitars", "effected guitars", and so on for every instrument in the dataset. Because of this, we chose Slakh2100 [37], which contains 2,100 musical mixtures along with isolated sources. This dataset has 145 hours of mixture data split into 34 instrument categories.

Before selecting excerpts from the dataset, we created a musical instrument hierarchy from Slakh's included instrument categories [1]. For these experiments, we define a hierarchy with three levels (excluding the trivial level consisting of the mixtures of all sources). The top level contains four categories: mid-range strings and keys (guitars, keyboards, and orchestral strings), bass instruments (acoustic and electric basses), winds (flutes, reeds, and brass), and percussion (drum sets and chromatic percussion). The middle level has seven categories (e.g., from mid-range strings: orchestral strings, guitars, keyboards, and electric keyboards), and the lowest level has eighteen categories (e.g., from guitars: clean guitars, and effected guitars). We note that this is just one of many possible hierarchies and almost all of the instruments described here would be classified as "other" in MUSDB18 [25].

To select examples from the dataset, we create multiple instantaneous submixes for each track, corresponding to the different levels of the hierarchy. As an example illustrated in Table 1, at the highest level, all pianos, guitars, and strings are considered one source, while at the next

---

[1] The full hierarchy can be seen at: `https://git.io/JJ4gx`

level all guitars are considered one source, and at the lowest level only clean guitars are considered one source. For each mixture in the dataset, we compute the saliency of each hierarchical submix in 10 second chunks, with a hop size of 2.5 seconds. If the source in the submix has energy above -30 dB in that chunk, it is considered salient. For the experiments involving multiple levels, we ensure that for a given node, its parent (or grandparent) has energy from child nodes other than itself. In other words, we want to make sure that a parent is not exactly the same as the child, meaning that some of the child node's siblings or cousins are also salient.

For our experiments, we use the *Slakh2100-split2* stratification and downsample the audio to 16 kHz. We do the mixing on the fly and select chunks randomly from the pool of salient examples for the specific experiment. For training, the networks see 20,000 examples per epoch ($\approx$55.5 h), and we use 3,000 examples ($\approx$8.3 h) for the validation and test sets. To ensure we have enough examples and a rich enough hierarchy to train, for the hierarchical SSS experiments we choose to separate sources down a path of the hierarchy as shown in Table 1, although the proposed methods can be extended to other paths down this or other hierarchies. For the QBE networks, we separate every instrument type in the hierarchy. Query chunks are selected from the pool of salient chunks such that they are always leaf nodes along the same path as the target regardless of the target level, but originate from different tracks.

For all experiments, we use the scale-invariant source-to-distortion ratio (SI-SDR) [38] to determine the output quality of our models. For reference, we also report the SI-SDR when doing no processing on the mixes.

5.2 Experiments and Model Configurations

In this paper, we evaluate four types of hierarchical source separation models. We vary models along two dimensions: whether they are single-instrument (i.e., source-specific separation, or SSS) or multi-instrument (i.e., query-by-example, or QBE), and whether they output a single level, or multiple levels. We describe each configuration below:

- **Single-instrument, Single-level:** A trio of instrument-specific SSS models each corresponding to one level of the hierarchy along one hierarchical path.

- **Single-instrument, Multi-level:** One SSS model that outputs a hierarchical set of masks, separating at all levels of a single hierarchical path simultaneously.

- **Multi-instrument, Single-level:** A trio of multi-instrument QBE models outputting one mask at one level of the hierarchy as determined by an input query.

- **Multi-instrument, Multi-level:** One QBE model that outputs a hierarchical set of masks for every level of the hierarchy along a path determined by an input query.

For the single-instrument models, we separate along one path of the hierarchy as referenced in Table 1. The multi-instrument, multi-level model is trained to separate a source based on a query, and thus can learn the full hierarchy (i.e., all instruments) instead of just one path as in the single-instrument, multi-level case.

Model Type	HC	Level 3	Level 2	Level 1
SSS (Guitar)		3.5	4.0	4.0
SSS (Guitar)	✓	3.2	3.6	3.8
QBE		3.2	2.4	0.2
QBE	✓	3.3	2.1	1.6

Table 2. Improvement in SI-SDR (dB) for hierarchical SSS (Guitar) and QBE models. Each model is trained either with the hierarchical constraint (HC) described in Section 4.3 or with no constraints on the masks produced for sources at different levels of granularity.

For the multi-level models, we test the effect of the hierarchical constraint proposed in Section 4.3. We can also test how well they learn with limited data about the leaf source. To do this, we train the three-level SSS and QBE models under the assumption that the leaf ground truth is unavailable either 50% or 90% of the time, in which cases only the upper levels are directly involved in the objective function. For comparison, we also evaluate models where *all* nodes are missing either 50% or 90% of the time during training. These experiments can tell us how well the multi-level network can leverage higher (i.e., coarser) levels of the hierarchy at the leaf node. Such an ability would be particularly advantageous as it is typically more difficult to collect data with fine-grained ground truth sources compared to data with a mixture and only a few source components gathered in broad categories, and could potentially help breaking open the "other" category of MUSDB18 with limited annotations.

All single-level and multi-level networks we test have the same architecture. The SSS models are composed of 4 bidirectional long short-term memory (BLSTM) layers with 600 hidden units in each direction and dropout of 0.3, followed by a fully connected layer with sigmoid activation function that outputs a mask. As described in Section 4.2, the QBE models are composed of two sub-networks, a query net and a masking net. The query net is composed of 2 BLSTM layers with 600 nodes in each direction and dropout of 0.3, followed by a fully-connected layer with linear activation that maps each time-frequency bin to an embedding space with 20 dimensions. The masking net is the same as the SSS models, with a larger input feature vector to accommodate the concatenated query anchor.

All models were trained with the Adam optimizer at a learning rate of 1e-4 for 100 epochs and a batch size of 25. The learning rate was halved if the loss on the validation set did not decrease for 5 straight epochs. The gradient was clipped to the 10th percentile of historical gradient norms if the norm of the minibatch was above that value [39].

6. RESULTS

In Table 2, we examine the effect of the hierarchical constraint (HC) on multi-level hierarchical networks. We observe that, for the source-specific separation network (which in this case only separates guitars), the HC slightly diminishes performance at all levels, indicating that SSS models are able to learn the specific hierarchical relation-

Model Type	# lvls	All Levels			Level 3			Level 2			Level 1		
		Mix	SI-SDR	Δ	Mix	SI-SDR	Δ	Mix	SI-SDR	Δ	Mix	SI-SDR	Δ
SSS (Guitar)	1	−3.9	−2.1	1.8	0.9	4.1	3.2	−5.9	−3.2	2.7	−6.6	−7.3	−0.7
SSS (Guitar)	3	−3.9	0.0	3.9	0.9	4.3	3.4	−5.9	−1.9	4.0	−6.6	−2.6	4.0
QBE	1	−4.9	−3.9	1.0	−1.3	2.0	3.3	−5.3	−3.9	1.4	−8.0	−9.8	−1.9
QBE	3	−4.9	−2.5	2.3	−1.3	2.0	3.3	−5.3	−3.2	2.1	−8.0	−6.4	1.6

Table 3. SSS and QBE model results in terms of SI-SDR (dB), where Δ denotes improvement over the noisy mix. SSS networks are only trained to separate sources in the hierarchy containing clean guitars (See Table 1), whereas QBE networks separate any source in the hierarchy. Here we compare single-level networks (denoted by a "1") to multi-level networks (denoted "3"). There is only one multi-level network for all three levels, but three single-level networks (one for each level).

	Data Reduction		Levels			
	%	type	All	Level 3	Level 2	Level 1
SSS (Guitar)	0	-	3.8	3.5	4.0	4.0
	50	all	3.3	3.1	3.4	3.4
	50	leaf	3.5	3.3	3.6	3.6
	90	all	0.1	1.5	−0.7	−0.5
	90	leaf	3.6	3.4	3.7	3.7
	Mix		−3.9	0.9	−5.9	−6.6
QBE	0	-	2.3	3.3	2.1	1.6
	50	all	−1.5	−2.1	−1.4	−1.1
	50	leaf	2.2	3.4	2.1	1.1
	90	all	−1.8	−2.1	−1.8	−1.5
	90	leaf	1.9	3.1	1.7	0.8
	Mix		−4.9	−1.3	−5.3	−8.0

Table 4. SI-SDR improvement (dB) over the unprocessed mix ("Mix") for hierarchical SSS and QBE models (separated by the thick broken line). Each model is trained while removing either just the leaf ("leaf") or the whole example ("all") for a specified percentage of the data. Reducing just leaf nodes up to 90% shows only a 0.3 dB drop for SSS and 0.8 dB drop for QBE compared to using all of the leaves.

ship for a single source (in this case, guitar) at different levels without additional help. For the query-by-example network (which separates all types of instruments), the HC marginally hinders performance at Level 2, but helps considerably for the leaf node (Level 1). We hypothesize that QBE networks benefit more because they are unable to learn the specific mask "shapes" of any individual source, and thus need the additional help offered by the HC. Therefore, in all subsequent experiments we include the HC for QBE networks, but omit it for the SSS networks.

In Table 3, we expand on the results from Table 2 and compare the results from single-level and multi-level hierarchical models for both SSS and QBE separation models. In both cases, the multi-level hierarchical networks improve over the single-level models, with the largest gains occurring at lower hierarchy levels. This implies that the networks can leverage their shared knowledge of the hierarchy to aid themselves at the lower levels, where individual instruments are more difficult to discern in the mix.

From the Level 1 results in Table 3, we see that sepa-

rating sources at this fine level of detail (e.g., clean electric guitars vs. distorted electric guitars) is extremely difficult, especially with a MIDI-synthesized data set such as Slakh2100, where several different instrument types may sound similar. In fact, when trying to train a single network to only separate these fine-grained sources, we are unsuccessful as noted by the negative SI-SDR improvements in the # lvls=1 (single level) rows for Level 1 sources. Training networks on multiple levels simultaneously mitigates this to some extent, although we have informally noticed the multi-level network sometimes outputting nearly identical separated sources between Level 1 and Level 2. We also note that the highest output SI-SDR values are obtained when separating Level 3 sources in Table 3, and we mention that Level 3 sources can be considered similar to the "other" source class in MUSDB18 [25]. Therefore, separating sources at the more fine-grained Levels (1 and 2) is more difficult than what is typically attempted in musical source separation.

In Table 4, we can observe the effect of removing leaf sources (Level 1 sources, see Table 1 for guitar example) from the training set. Compared to reducing *all* of the data by 50% or 90%, the performance of reducing only the leaves degrades very minimally. In cases where we have rich data at higher levels but sparse data at lower levels, hierarchical multi-level networks can do a respectable job at separating lower levels. We see the same story for both SSS and QBE networks: even a small amount of leaf data can help ward off a large drop in performance.

7. CONCLUSIONS

In this paper, we re-framed the source separation problem as hierarchical, and demonstrated the benefit of learning to simultaneously separate sources at different levels of granularity. In the present work, we considered network architectures that output masks for source separation at all relevant levels together. We showed that in doing so, we are still able to separate out the most granular source types when training data is severely limited. A major drawback of this work is the need for a large quantity of labeled and curated data, a limitation that we hope future work can address. Other future directions include architectures that output relevant levels sequentially, such as cascaded models [40], or directions inspired by hierarchical audio classification models [41, 42].

8. REFERENCES

[1] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 65–68, 2014.

[2] F. J. Weninger, J. R. Hershey, J. Le Roux, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation," in *GlobalSIP Machine Learning Applications in Speech Processing Symposium*, Dec. 2014.

[3] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 708–712.

[4] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 26, no. 10, pp. 1702–1726, 2018.

[5] J. R. Hershey, Z. Chen, and J. Le Roux, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 31–35.

[6] M. Kolbæk, D. Yu, Z.-H. Tan, and J. Jensen, "Multi-talker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, pp. 1901–1913, 2017.

[7] Z.-Q. Wang, J. Le Roux, and J. R. Hershey, "Alternative objective functions for deep clustering," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018.

[8] Y. Luo and N. Mesgarani, "TasNet: Surpassing ideal time-frequency masking for speech separation," *arXiv preprint arXiv:1809.07454*, Sep. 2018.

[9] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep U-Net convolutional networks," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, Oct. 2017.

[10] N. Takahashi, N. Goswami, and Y. Mitsufuji, "MM-DenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *Proc. IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*, Sep. 2018.

[11] Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, and N. Mesgarani, "Deep clustering and conventional networks for music separation: Stronger together," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 61–65.

[12] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - a reference implementation for music source separation," *Journal of Open Source Software*, 2019. [Online]. Available: https://doi.org/10.21105/joss.01667

[13] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov., pp. 2579–2605, 2008.

[14] C. M. Wessinger, J. W. VanMeter, B. Tian, J. Van Lare, J. Pekár, and J. P. Rauschecker, "Hierarchical organization of the human auditory cortex revealed by functional magnetic resonance imaging," *Journal of cognitive neuroscience*, vol. 13, no. 1, pp. 1–7, 2001.

[15] J. E. Peelle, I. Johnsrude, and M. H. Davis, "Hierarchical processing for speech in human auditory cortex and beyond," *Frontiers in human neuroscience*, vol. 4, p. 51, 2010.

[16] G. G. Parras, J. Nieto-Diego, G. V. Carbajal, C. Valdés-Baizabal, C. Escera, and M. S. Malmierca, "Neurons along the auditory pathway exhibit a hierarchical organization of prediction error," *Nature communications*, vol. 8, no. 1, pp. 1–17, 2017.

[17] M. M. Farbood, D. J. Heeger, G. Marcus, U. Hasson, and Y. Lerner, "The neural processing of hierarchical structure in music and speech at different timescales," *Frontiers in neuroscience*, vol. 9, p. 157, 2015.

[18] A. S. Bregman, *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[19] M. J. Kartomi, *On concepts and classifications of musical instruments*. University of Chicago Press Chicago, 1990.

[20] E. M. Von Hornbostel and C. Sachs, "Classification of musical instruments: Translated from the original German by Anthony Baines and Klaus P. Wachsmann," *The Galpin Society Journal*, pp. 3–29, 1961.

[21] M. Dewey, *A classification and subject index, for cataloguing and arranging the books and pamphlets of a library*. Brick row book shop, Incorporated, 1876.

[22] B. Owsinski, *The mixing engineer's handbook*. Nelson Education, 2013.

[23] P. Seetharaman, G. Wichern, S. Venkataramani, and J. Le Roux, "Class-conditional embeddings for music source separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 301–305.

[24] J. H. Lee, H.-S. Choi, and K. Lee, "Audio query-based music source separation," *arXiv preprint arXiv:1908.06593*, 2019.

[25] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[26] C. S. Doire and O. Okubadejo, "Interleaved multitask learning for audio source separation with independent databases," *arXiv preprint arXiv:1908.05182*, 2019.

[27] V. S. Kadandale, J. F. Montesinos, G. Haro, and E. Gómez, "Multi-task U-Net for music source separation," *arXiv preprint arXiv:2003.10414*, 2020.

[28] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 771–775.

[29] O. Slizovskaia, G. Haro, and E. Gómez, "Conditioned source separation for music instrument performances," *arXiv preprint arXiv:2004.03873*, 2020.

[30] G. Meseguer-Brocal and G. Peeters, "Conditioned-U-Net: Introducing a control mechanism in the U-Net for multiple source separations," *arXiv preprint arXiv:1907.01277*, 2019.

[31] X. Xiao, Z. Chen, T. Yoshioka, H. Erdogan, C. Liu, D. Dimitriadis, J. Droppo, and Y. Gong, "Single-channel speech extraction using speaker inventory and attention network," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 86–90.

[32] Q. Wang, H. Muckenhirn, K. Wilson, P. Sridhar, Z. Wu, J. Hershey, R. A. Saurous, R. J. Weiss, Y. Jia, and I. L. Moreno, "Voicefilter: Targeted voice separation by speaker-conditioned spectrogram masking," *arXiv preprint arXiv:1810.04826*, 2018.

[33] M. Delcroix, K. Zmolikova, K. Kinoshita, A. Ogawa, and T. Nakatani, "Single channel target speaker extraction and recognition with speaker beam," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, pp. 5554–5558.

[34] J. Wang, J. Chen, D. Su, L. Chen, M. Yu, Y. Qian, and D. Yu, "Deep extractor network for target speaker recovery from single channel speech mixtures," *arXiv preprint arXiv:1807.08974*, 2018.

[35] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 246–250.

[36] R. Kumar, Y. Luo, and N. Mesgarani, "Music source activity detection and separation using deep attractor network." in *Proc. ISCA Interspeech*, Sep. 2018, pp. 347–351.

[37] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2019.

[38] J. Le Roux, S. T. Wisdom, H. Erdogan, and J. R. Hershey, "SDR – half-baked or well done?" in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019.

[39] P. Seetharaman, G. Wichern, B. Pardo, and J. Le Roux, "AutoClip: Adaptive gradient clipping for source separation networks," in *Proc. International Workshop on Machine Learning for Signal Processing (MLSP)*, Oct. 2020.

[40] M. Maciejewski, G. Wichern, E. McQuinn, and J. Le Roux, "WHAMR!: Noisy and reverberant single-channel speech separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 696–700.

[41] J. Cramer, V. Lostanlen, A. Farnsworth, J. Salamon, and J. P. Bello, "Chirping up the right tree: Incorporating biological taxonomies into deep bioacoustic classifiers," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 901–905.

[42] H. Shrivaslava, Y. Yin, R. R. Shah, and R. Zimmermann, "Mt-gcn for multi-label audio tagging with noisy labels," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 136–140.

TAG2RISK: HARNESSING SOCIAL MUSIC TAGS FOR CHARACTERIZING DEPRESSION RISK

Aayush Surana[1]   **Yash Goyal**[1]   **Manish Shrivastava**[1]
Suvi Saarikallio[2]   **Vinoo Alluri**[1]

[1] International Institute of Information Technology, Hyderabad, India
[2] Department of Music, Art and Culture Studies, University of Jyväskylä, Finland

{aayush.surana, yash.goyal}@research.iiit.ac.in, suvi.saarikallio@jyu.fi
{m.shrivastava, vinoo.alluri}@iiit.ac.in

ABSTRACT

Musical preferences have been considered a mirror of the self. In this age of Big Data, online music streaming services allow us to capture ecologically valid music listening behavior and provide a rich source of information to identify several user-specific aspects. Studies have shown musical engagement to be an indirect representation of internal states including internalized symptomatology and depression. The current study aims at unearthing patterns and trends in the individuals at risk for depression as it manifests in naturally occurring music listening behavior. Mental well-being scores, musical engagement measures, and listening histories of Last.fm users (N=541) were acquired. Social tags associated with each listener's most popular tracks were analyzed to unearth the mood/emotions and genres associated with the users. Results revealed that social tags prevalent in the users at risk for depression were predominantly related to emotions depicting *Sadness* associated with genre tags representing *neo-psychedelic-, avant garde-, dream-pop*. This study will open up avenues for an MIR-based approach to characterizing and predicting risk for depression which can be helpful in early detection and additionally provide bases for designing music recommendations accordingly.

1. INTRODUCTION

According to reports from the World Health Organization, an estimated 322 million people worldwide were affected from depression, the leading cause of disability [1]. Recent times have witnessed a surge in studies on using social multimedia content, such as those from Facebook, Twitter, Instagram, to detect mental disorders including depression [2–6]. Music plays a vital role in mental well-being by impacting moods, emotions and other affective states [7]. Musical preferences and habits have been associated with the individual's need to satisfy and reinforce their psychological needs [8, 9]. Empirical evidence exists linking musical engagement strategies to measures of ill-health including internalized symptomatology and depression [10, 11]. Also, increased emotional dependency on music during periods of depression has been reported [12]. Specifically, the Healthy-Unhealthy Music Scale (HUMS), a 13-item questionnaire was developed for assessing musical engagement strategies that identified maladaptive ways of using music. Such strategies are characterized by using music to avoid other people, resort to ruminative thinking and ending up feeling worse after music engagement. Such unhealthy musical engagement was found to correlate with higher psychological distress and was indicative of depressive tendencies [13]. Furthermore, the high predictive power observed from the machine learning models in predicting risk for depression from HUMS further bolsters its efficacy as an indirect tool for assessing mental states [14]. Research suggests that such musical engagement does not always lead to alleviating depressive symptoms [15]. This indeed calls for developing intervention strategies that allow for altering music listening behavior to suit the individual's state, traits, and general musical preferences which may lead to a positive outcome. Thus, it is of vital importance not only to identify individuals with depressive tendencies but also to unearth music listening habits of such individuals that will provide bases for designing music recommendations accordingly.

Past research studying the link between music listening habits and depression has been done using self-reported data and controlled listening experiments wherein participants may have wished to conform to social expectations, or their responses might be influenced by how they want other people to perceive them thereby resulting in demand characteristics [16]. This has also been identified as a limitation by Nave et al. [8], who have proposed collecting data in more ecologically valid settings, such as user listening histories from music streaming platforms which are a better reflection of the users' true preferences and behaviours. To date no studies have looked at the link between active music listening and depression using the music listening histories of users which motivates us for this study.

In this age of big data, online music streaming platforms such as Last.fm, Spotify, and Apple Music provide access to millions of songs of varying genres and this has allowed

Figure 1: Methodology

for assessing users' features from their digital traces on music streaming platforms. To the best of our knowledge, Last.fm is the only platform that makes it possible to extract the listening history of users and other metadata describing their listening behavior using its own public API. Last.fm has been used extensively by researchers for various purposes such as music emotion classification, user behavior analysis, and social tag categorization [17, 18]. Last.fm has an abundance of social tags that provide a wide range of information about the musical tracks including audio low- and high-level feature description, emotions and experiences evoked, genre, etc. These tags have been found to predict short-term user music preferences [19] and in successfully predicting next played songs in the design of a recommendation system [20]. Our aim is to identify the tags and their respective occurrences in the listening behavior of individuals at risk for depression, which makes Last.fm an apt choice for this study. The data was collected using an online survey comprising of Last.fm music listening histories, in addition to music engagement strategies (HUMS), and mental well-being scores of the participants. Specifically, each track in the data was semantically represented by the tags assigned to it. We leverage these representations of tags as social descriptors of music to uncover emotional experiences and concepts that are associated with users with risk for depression.

1.1 Research Objectives and Hypotheses

In this study we investigated whether people's music listening history, in terms of social tags, could be used to predict a risk for depression. Our research questions were:

- What are the social tags associated with music chosen by At-Risk users?

- What emotions do these tags signify in the context of musically evoked emotions?

- What genres are mostly associated with At-Risk users?

- How well can we classify users as At-Risk given user-specific social tags?

We expected the social tags linked with At-Risk listeners to contain emotions with low arousal and low valence, being typical of depressive mood. Owing to the lack of research

associating music genres and risk for depression [15], this part of the study was exploratory.

2. METHODOLOGY

The methodological approach and procedure of our study is illustrated in Figure 1. The steps of data collection, processing, and analysis are described below.

2.1 Data Collection

An online survey was designed wherein participants were asked to fill their Last.fm usernames and demographics followed by standard scales for assessing their mental well-being, musical engagement strategies and personality. Participants were solicited on the Last.fm groups of social media platforms like Reddit and Facebook. The inclusion criterion required being an active listener on Last.fm for at least a year prior to filling the survey. The survey form required the users' consent to access their Last.fm music history.

2.1.1 Participants

A total of 541 individuals (Mean Age = 25.4, SD = 7.3) were recorded to be eligible and willing to participate in the study consisting of 444 males, 82 females and 15 others. Most of them belonged to the United States and the United Kingdom accounting for about 30% and 10% of the participants respectively. Every other country contributed to less than 5% of the total participants.

2.1.2 Measure of Well-Being, Musical Engagement, and Personality

The Kessler's Psychological Distress Scale (K10) questionnaire [21] was used to assess mental well-being. It is a measure of psychological distress, particularly assessing anxiety and depression symptoms. Individuals scoring 29 and above on K-10 are likely to be at severe risk for depression and hence, constitute the "At-Risk" group. Those scoring below 20 are labeled as the "No-Risk" group [22] as they are likely to be well. There were 193 participants in the No-Risk group and 142 in the At-Risk group. The HUMS survey was administered to assess musical engagement strategies which resulted in two scores per participant, *Healthy* and *Unhealthy*. Personality information was obtained using the Mini-IPIP questionnaire [23]

which results in scores for the Big Five traits of Personality namely *Openness*, *Conscientiousness*, *Extraversion*, *Agreeableness* and *Neuroticism*. HUMS and personality data were collected in order to identify if specific personality traits engage more in *Unhealthy* music listening and as additional measures to assess internal validity.

2.1.3 Music Listening History

Each participant's music listening history was extracted using a publicly available API. The data included tracks, artists, and social tags associated with the tracks. For each participant, the top n (n=500,200,100) tracks based on play-counts were extracted centered around the time t (t = $\pm$ 3 months,2 months) they filled in the questionnaire. The reason for varying n and t was to find converging evidence in music listening behavior in order to make our results more robust. For each track, the top 50 social tags based on tag weight (number of times the tag has been assigned to the track) were chosen for subsequent analysis.

2.2 Social Tags Processing

2.2.1 Tag Filtering

Music-related social tags are known to be descriptors of genre, perceived emotion, artist and album amongst others. It is therefore important to filter them to organize them according to some structure and interpretable dimensions for the task at hand. The purpose of this preprocessing step was to retrieve tags that could be mapped onto a semantic space representing music-evoked emotions. To this end, we used four filtering stages: first, include lower-casing, removal of punctuation and stop-words, spell-checking and checking for the existence of tag words in the English corpus; second, retain tags that are most frequently assigned adverbs or adjectives via POS (Part Of Speech) tagging since POS tags representing nouns and pronouns do not have emotion relevance in this context; third, remove tags containing 2 or more words to avoid valence shifters [24] and sentence-like descriptions from our Last.fm corpus; fourth, manually filter them by discarding tags without any mood/emotion associations.

2.2.2 Tag Emotion Induction

To project the tags onto an emotion space, we used dimensional models that represent the emotions. Multiple research studies have shown the usefulness of both two-dimensional and three-dimensional models to represent emotions [25–27]. We therefore used both these models for further analysis in order to check for trends and the effect of the third dimension when dealing with emotions.

The first model is one of the most popular dimensional models, the Russell's Circumplex Model of Affect [28], where an emotion is a point in a two-dimensional continuous space representing *Valence* and *Arousal* (VA). *Valence* reflects pleasantness and *Arousal* describes the energy content of the emotion. The second model is an extension of the Russell's model with an added *Dominance*

dimension (VAD), which represents control of the emotional state. The VAD model has been a popular framework used to construct emotion lexicons in the field of Natural Language Processing. The projection in the VAD space is based on semantic similarity and has been largely used to obtain affective ratings for large corpora of English words [29] [30]. Another common emotion model is the VAT model wherein the third dimension represents *Tension* (VAT) and has been used in retrieving mood information from Last.fm tags [17]. However, Saari et. al.'s [17] approach was based on tag co-occurrence rather than semantic similarity. Moreover, a subsequent study by the same authors reported a positive correlation (r=0.85) between *tension* and *dominance* [31]. Also, multiple studies have supported the use of the VAD space for analysing emotions in the context of music [32, 33]. We therefore have chosen the VAD framework for the purpose of our study. Since VA dimensions alone were found to sufficiently capture musical emotions [26], we also repeat our analysis based on the VA model to observe the effect of the omitted *Dominance* dimension.

The tags were projected onto the VAD space using a word-emotion induction model introduced by Buechel and Hahn [29]. We used the FastText embeddings of the tags as input to a 3-layer multi-layer perceptron that produced VAD values ranging from 1 to 9 on either of the dimensions. FastText has shown better accuracy for word-emotion induction [29] when compared to other commonly used models like Word2vec and GloVe. Moreover, FastText embeddings incorporate sub-word character n-grams that enable the handling of out-of-vocabulary words. This results in a large advantage over the other models [34]. In addition, FastText works well with rarely occurring words because their character n-grams are still shared with other words. This made it a suitable choice since some of the user-assigned tags may be infrequent or absent in the training corpus of the embedding model. We used the same approach to project the tags onto the VA space by changing the number of nodes in the output layer from 3 to 2.

Both the models were trained using the EN+ dataset which contains *valence*, *arousal* and *dominance* ratings (on a 9-point scale) for a majority of well-known English words [35]. This module resulted in an n-dimensional vector (n=3 for VAD, n=2 for VA) for each tag. The remainder of the pipeline describes the 3-dimensional VAD vector processing. The same procedure is repeated for the VA scores.

2.2.3 Tag Emotion Mapping

The social tags were grouped into broader emotion categories. These categories consisted of 9 first-order factors of Geneva Emotional Music Scale (GEMS) [36]. These were *Wonder, Transcendence, Nostalgia, Tenderness, Peacefulness, Power, Joyful activation, Tension and sadness.* Table 1 in the supplementary material displays the factor loadings for these first-order factors of GEMS. GEMS contains 40 emotion terms that were consistently chosen to describe musically evoked emotive states across

a wide range of music genres. These were subsequently grouped to provide a taxonomy for music-evoked emotions. This scale has outperformed other discrete and dimensional emotion models in accounting for music-evoked emotions [37]. In order to project these 9 emotion categories onto the VAD space, we first obtained the VAD values for the 40 emotion terms. Next, the VAD values were weighted and summed according to the weights provided in the original GEMS study to finally obtain VAD values for each of the emotion categories. Figures 1 & 2 in supplementary material display the projections of these emotion categories onto the VAD and VA spaces. Each of the tags are then assigned the emotion category based on the proximity in the VAD space as evaluated by the euclidean distance.

2.3 User-Specific Emotion Prevalence Score

After every user's tags had been mapped onto the 9 emotion categories, we calculated an *Emotion Prevalence Score* $S_{u,c}$ for every user. This represents the presence of tags belonging to that particular emotion category in the user's listening history.

$$S_{u,c} = \frac{\sum_{j \epsilon V_{\text{tr}}} (N_{j,c} \times tr_{\text{u,j}})}{\sum_{i \epsilon T_{\text{u}}} tr_{\text{u,i}}} \quad (1)$$

where

$$N_{j,c} = \sum_{k \epsilon Tags_{\text{c}}} \frac{tw_{\text{j,k}}}{\sum_{l \epsilon V_{\text{tg}}} tw_{\text{j,l}}} \quad (2)$$

c : emotion category
$N_{j,c}$: the association of track j with c
T_u : all tracks for user u
V_{tg} : all tags obtained after tag filtering
V_{tr} : all tracks having at least one tag from V_{tg}
$tr_{u,i}$: playcount of track i for user u
$tw_{j,k}$: tag weight of tag k for track j
$Tags_c$: all tags in V_{tg} which belong to c

Since the objective of this work was to identify which of the 9 categories are most characteristic of At-Risk individuals when compared to No-Risk individuals, we performed group-level statistical tests of difference as described in the following section.

2.4 Emotion-based Analysis : Group Differences and Bootstrapping

For each emotion category, we performed a two-tailed Mann-Whitney U (MWU) Test on the *Emotion Prevalence Scores* between the No-Risk and At-Risk groups. For a category, the group having higher mean rank from MWU Test indicates a stronger association of the category with that group. For the emotion categories that exhibited significant differences ($p < .05$), we further performed bootstrapping to account for Type I error and ensure that the observed differences are not due to chance. Bootstrapping (random sampling) with replacement was performed with 10,000 iterations. Each iteration randomly assigned participants to the At-Risk or No-Risk group. The U-statistic

for each iteration was calculated. As a result, we obtain a bootstrap distribution for the U-statistic from which we estimate the significance of the observed statistic.

2.5 Genre-Prevalence Analysis
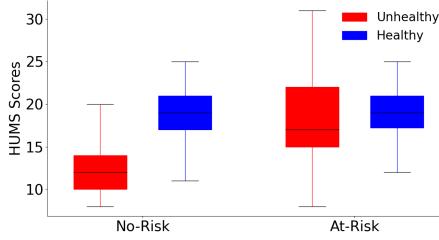
To further analyse the types of music associated with the tags of emotion categories, we explored genre-related social tags. In order to select the genre-related tags from our data, we collected the results of the multi-stage model proposed by Ferrer et al. [38] which assigned tags of Last.fm to different semantic layers namely genre, artist, affect, instrument, etc. In order to understand the underlying genre tag structure and obtain broader genre categories, we employed the approach described by Ferrer et al. [39] to cluster genre tags (details in Equation 1 in the Supplementary material). In this, music tags were hierarchically organized revealing taxonomy of music tags by means of latent semantic analysis. The clusters thus obtained were labelled based on the genre-tags constituting the core points of the cluster [40].

For the emotion categories that exhibited significant group differences, the genre tags co-occurring with its tags were used to calculate a user-specific *Genre Prevalence Score* for each genre-tag cluster. The formula used was similar to *Emotion Prevalence Scores* with the change in definition of the following terms: c represents the genre cluster, T_u is the set of all tracks for user u which have a tag belonging to the particular emotion category and V_{tg} is the set of all genre tags. Finally, we performed a biserial correlation between *Genre Prevalence Scores* for each genre-tag cluster and the users' risk for depression (represented as a dichotomous variable with 0 = No-Risk; 1 = At-Risk).

3. RESULTS

3.1 Internal Consistency and Criterion Validity

The Cronbach's alphas for *Unhealthy* scores obtained from HUMS and K10 scores were found to be relatively high at 0.80 and 0.91 respectively. A significant correlation (r=0.55, df=539, p<0.001) between *Unhealthy* Score and K10 was found which is in concordance with past research studies in the field [13]. Also, in line with previous research [41, 42], a significant positive correlation was observed between K10 score and Neuroticism (r=0.68, p<0.0001) adding to the internal consistency of the data and confirming construct validity. As can be seen in Figure 2, the At-Risk group displayed higher mean and median *Unhealthy* score compared to No-Risk while *Healthy* scores were comparable. Partial correlations between *Unhealthy*, *Healthy*, and K10 are presented in Table 1. K10 scores exhibit significant positive correlation only with *Unhealthy* for both the groups. The moderate correlation between *Healthy* and *Unhealthy* scores for the No-Risk population indicates that both of these subscales capture a shared element, most likely active music listening.
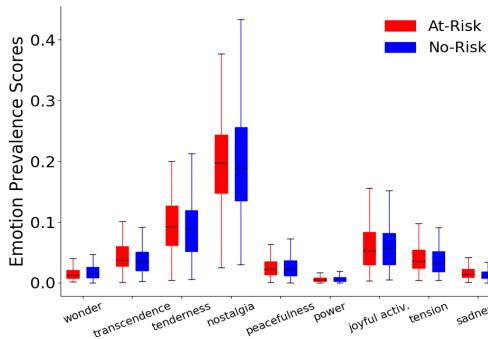
Figure 2: Boxplot of HUMS scores for No-Risk and At-Risk Groups.

	No-Risk		At-Risk	
Scales	Healthy	Unhealthy	Healthy	Unhealthy
Healthy	1.0	0.36**	1.0	-0.14
Unhealthy	0.36**	1.0	-0.14	1.0
K10	0.07	0.26**	-0.11	0.22*

Table 1: Partial Correlation Values between HUMS & K10. (*p<0.01 & **p<0.001)

3.2 Emotion-based Results
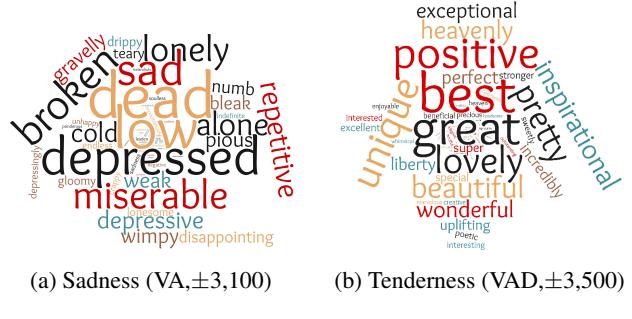
The data ($t=\pm3$,$n=500$) consisted of 3,80,261 social tags. The tag filtering process resulted in a final set of 1254 unique tags (Mean=109, SD = 24 tags per user) which were then mapped onto the VA and VAD emotion spaces. Figure 3 in supplementary material displays the tags closest to each of the Emotion Categories based on VA and VAD models.

Figure 3: Boxplot of Emotion Prevalence Scores for No-Risk and At-Risk based on VA.

Figure 3 illustrates the *Emotion Prevalence Scores* for both groups for VA mapping (Supplementary Figure 4 displays the same for VAD, showing a similar distribution). The overall pattern appears similar between both the groups with minor observable differences for the emotion categories *wonder, transcendence, tenderness, tension, and sadness*. Table 2 displays the emotion categories that exhibited significant differences between the groups (MWU U-statistic and bootstrap p-values in Table 2 of Supplementary material). The At-Risk group consistently exhibits higher Prevalence Scores in *Sadness* while the No-Risk group vacillates between *Wonder* and *Transcendence*. The most significant difference was observed in *Sadness* (VA model,$t=\pm3$,$n=100$) with a significantly greater *Emotion Prevalence Score* for the At-Risk group (Median

= 0.0117) than the No-Risk group (Median = 0.0091), U=11414.5, p=0.009. Significant difference was also observed for *Tenderness* with greater *Emotion Prevalence Score* for the At-Risk group (Median = 0.1271) than the No-Risk group (Median = 0.1189), U=11905.0, p=0.04. On the other hand, the *Emotion Prevalence Score* in *Wonder* (VA model,$t=\pm2$,$n=100$) was significantly greater for the No-Risk group (Median = 0.0131) than the At-Risk group (Median = 0.0086), U=16270.0, p=0.003. The word-clouds of tags comprising *Sadness* and *Tenderness* are displayed in Figure 4a and Figure 4b. A score per tag is computed for each group (Equation 2 in Supplementary material). A rank was assigned to the tag based on the absolute difference of the tag scores between No-Risk & At-Risk groups. The size of the tag in the word-cloud is directly proportional to its rank in the category. Supplementary figures 5 and 6 depict word-clouds for *Transcendence* and *Wonder*.

(a) Sadness (VA,±3,100)   (b) Tenderness (VAD,±3,500)

Figure 4: Wordclouds for emotion categories associated with At-Risk group.

We also assessed the predictive power of social tags for risk of depression by classifying participants into At-Risk or No-Risk groups using their tag information (feature details in Equation 4 in Supplementary material). The SVM model with 'rbf' kernel (C=2301, gamma=101) gave the best results with a 5-fold cross-validation accuracy of 66.4%.

3.3 Genre-Prevalence Results

Out of the 5062 tags assigned to the genre layer in [38], 94% (4766) of the tags were present in our data. The clustering of the genre tags resulted in 17 clusters and is displayed in Table 3 of Supplementary material. Figure 7 in Supplementary material displays mean genre prevalence scores between both groups for these 17 clusters. Overall, genre-cluster representing *indie-,alternative-pop/rock* represented by Cluster 4 is predominant in both groups. Genre prevalence scores were then evaluated specific to the tracks associated with the emotion categories that exhibited most significant group differences, that is, *Wonder* and *Sadness* (VA model, $t=\pm3$, $n=100$). For *Sadness*-specific tracks, the highest correlation (r=0.2, p<0.01) was observed between the Genre-Prevalence scores in the cluster representing *neo psychedelic-, avant garde-, dream-pop* and K-10 scores. Also, genre clusters representing *electronic rock* (r=0.17, p<0.01), *indie-, alternative-pop/rock* (r=0.12, p<0.05), and

Group	Top Tracks	VAD		VA	
		$t=\pm3$	$t=\pm2$	$t=\pm3$	$t=\pm2$
At-Risk	$n=100$	Sadness*		Sadness**	
	$n=200$	Sadness*	Sadness*, Tenderness*	Sadness*	Sadness*, Transcendence*
	$n=500$	Sadness*, Tenderness*	Tenderness*	Sadness*	Sadness*, Transcendence*
No-Risk	$n=100$		Transcendence*		Wonder**
	$n=200$	Transcendence*	Transcendence*	Wonder*	Wonder**
	$n=500$	Transcendence*	Transcendence*	Wonder*	Wonder**

Table 2: Emotion Categories with Significant Differences between At-Risk and No-Risk groups.*p<0.05; **p<0.01

world music (r=-0.11, p<0.05) demonstrated significant correlations for *Tenderness*. For *Wonder* (VA model, $t=\pm2$, n=100), the K-10 scores exhibited significant negative correlation with *Genre Prevalence Scores* of clusters representing *black metal* (r=-0.11, p<0.05) and *neo-progressive rock* (r=-0.13, p<0.05).

4. DISCUSSION

This study is the first of its kind to examine the association between risk for depression and social tags related to music listening habits as they occur naturally as opposed to self-reported or lab-based studies. A clear difference in the music listening profiles was observed between the At-Risk group and the No-Risk group, particularly in terms of the emotional content of the tags. *Sadness* was significantly more prevalent in the At-Risk group and the word-cloud of sadness was highly illustrative of other low-arousal, low-valence emotions such as *dead, low, depressed, miserable, broken*, and *lonely*. The stronger association of the At-Risk group with sadness is in concordance with the past research studies in the field [43] and confirms our hypothesis. The At-Risk group is attracted to music that reflects and resonates with their internal state. Whether this provides emotional consolation as an adaptive resource or whether it only worsens repetitive negative feelings and fuels rumination, remains an open question. Nonetheless, statistically, such listening style can be seen as a highly predictive factor of psychological distress.

In addition, *Tenderness*, which represents low-arousal and high-valence, was also more prevalent in the At-Risk group, especially for shorter-term (± 2 months) music listening habits. *Tenderness* appears to be more significant in the shorter time period in addition to *Sadness*, possibly indicating that At-Risk people tend to oscillate between positive and negative states within a general state of low arousal. These findings appear to be very much in line with the results found by Houben et al. [44] who found high levels of emotional inertia and emotional variability to be linked with depression and ill-being. The consistent results related to *Sadness* in our study reflect the overall inert states in which the At-Risk tend to be. On the other hand, the *Tenderness* results reflect their tendency to jump to positive affective states while retaining low arousal, thereby demonstrating emotional variability. Furthermore, the omission of the *Dominance* dimension causes most of the tags to shift from *Tenderness* to *Transcendence* and *Transcendence* to *Wonder*, which explains

the results in a reversal of the group association as evidenced in the results. Nevertheless, *Sadness* appears to be the predominant state as it is largely consistent for ±3 months as well as for ±2 months of music listening histories.

The At-Risk group also exhibited a tendency to gravitate towards music with genre tags such as *neo-psychedelic-, avant garde-, dream-pop* co-occurring with *Sadness*. Such genres are characterized by ethereal-sounding mixtures that often result in a wall of sound comprising electronic textures with obscured vocals. Similarly, the genres co-occurring with *Tenderness* (VAD model) or *Transcendence* (VA model) comprise similar mixtures with heavy synthesizer-based sounds (such as mellotron) which result in sounds that seem otherworldly. Such out-of-this world soundscapes have been also associated with transcendent druggy and mystical imagery and immersive experiences [45]. These results strengthen the claim that depression may foster musical immersion as an escape from a reality that is perceived to be adverse. This is somewhat in line with prior research that has linked depression with the use of music for avoidant coping [46]. On the other hand, music listening history of the No-Risk group was characterized by an inclination to listen to music tagged by positive valence and higher arousal as characterized by *Wonder* with a predilection for *Heavy metal* and *Progressive Rock* genres.

The use of only single word tags in the third stage of tag filtering is one limitation of this study which is due to lack of compatibility of the word emotion induction model with multi-word tags. Our results could potentially be extended to find significant differences in emotional concepts after considering multi-word social tags. We achieve a decent classification accuracy of 66.4% which is significantly above the chance level which indicate that social tags indeed may be indicative of At-Risk behavior. This may further be improved by considering additional descriptors of music such as acoustic features and lyrical content of the tracks. Another future direction is to incorporate the temporal evolution of these emotion categories in the listening histories to characterize depression, since past research suggests depression to be a result of gradual development of daily emotional experiences [47]. This study is intended to be one of many to come that will be helpful in early detection of depression and other potential mental disorders in individuals using their digital music footprints.

5. AUTHOR CONTRIBUTIONS

Aayush Surana and Yash Goyal are joint first authors with equal contribution.

6. REFERENCES

[1] WHO, "Depression and other common mental disorders: global health estimates," World Health Organization, Tech. Rep., 2017.

[2] M. De Choudhury, M. Gamon, S. Counts, and E. Horvitz, "Predicting depression via social media," in *Seventh international AAAI conference on weblogs and social media*, 2013.

[3] G. Coppersmith, M. Dredze, and C. Harman, "Quantifying mental health signals in twitter," in *Proceedings of the workshop on computational linguistics and clinical psychology: From linguistic signal to clinical reality*, 2014, pp. 51–60.

[4] M. De Choudhury, S. Counts, E. J. Horvitz, and A. Hoff, "Characterizing and predicting postpartum depression from shared facebook data," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, 2014, pp. 626–638.

[5] M. De Choudhury, E. Kiciman, M. Dredze, G. Coppersmith, and M. Kumar, "Discovering shifts to suicidal ideation from mental health content in social media," in *Proceedings of the 2016 CHI conference on human factors in computing systems*, 2016, pp. 2098–2110.

[6] A. G. Reece and C. M. Danforth, "Instagram photos reveal predictive markers of depression," *EPJ Data Science*, vol. 6, no. 1, pp. 1–12, 2017.

[7] M. Baltazar and S. Saarikallio, "Toward a better understanding and conceptualization of affect self-regulation through music: A critical, integrative literature review," *Psychology of Music*, vol. 44, no. 6, pp. 1500–1521, 2016.

[8] G. Nave, J. Minxha, D. M. Greenberg, M. Kosinski, D. Stillwell, and J. Rentfrow, "Musical preferences predict personality: evidence from active listening and facebook likes," *Psychological Science*, vol. 29, no. 7, pp. 1145–1158, 2018.

[9] L. Qiu, J. Chen, J. Ramsay, and J. Lu, "Personality predicts words in favorite songs," *Journal of Research in Personality*, vol. 78, pp. 25–35, 2019.

[10] K. S. McFerran, S. Garrido, and S. Saarikallio, "A critical interpretive synthesis of the literature linking music and adolescent mental health," *Youth & Society*, vol. 48, no. 4, pp. 521–538, 2016.

[11] S. Garrido, T. Eerola, and K. McFerran, "Group rumination: Social interactions around music in people with depression," *Frontiers in psychology*, vol. 8, p. 490, 2017.

[12] K. S. McFerran, "Contextualising the relationship between music, emotions and the well-being of young people: A critical interpretive synthesis," *Musicae Scientiae*, vol. 20, no. 1, pp. 103–121, 2016.

[13] S. Saarikallio, C. Gold, and K. McFerran, "Development and validation of the healthy-unhealthy music scale," *Child and adolescent mental health*, vol. 20, no. 4, pp. 210–217, 2015.

[14] R. Agarwal, R. Singh, S. Saarikallio, K. McFerran, and V. Alluri, "Mining mental states using music associations," *depression*, vol. 2, p. 6, 2019.

[15] J. Stewart, S. Garrido, C. Hense, and K. McFerran, "Music use for mood regulation: self-awareness and conscious listening choices in young people with tendencies to depression," *Frontiers in psychology*, vol. 10, p. 1199, 2019.

[16] D. M. Greenberg and P. J. Rentfrow, "Music and big data: a new frontier," *Current opinion in behavioral sciences*, vol. 18, pp. 50–56, 2017.

[17] P. Saari and T. Eerola, "Semantic computing of moods based on tags in social media of music," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2548–2560, 2013.

[18] C. Laurier, M. Sordo, J. Serra, and P. Herrera, "Music mood representations from social tags." in *International Society for Music Information Retrieval (ISMIR) Conference*, 2009, pp. 381–386.

[19] K. Gupta, N. Sachdeva, and V. Pudi, "Explicit modelling of the implicit short term user preferences for music recommendation," in *European Conference on Information Retrieval*. Springer, 2018, pp. 333–344.

[20] M. Polignano, P. Basile, M. de Gemmis, and G. Semeraro, "Social tags and emotions as main features for the next song to play in automatic playlist continuation," in *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, 2019, pp. 235–239.

[21] R. C. Kessler, G. Andrews, L. J. Colpe, E. Hiripi, D. K. Mroczek, S.-L. Normand, E. E. Walters, and A. M. Zaslavsky, "Short screening scales to monitor population prevalences and trends in non-specific psychological distress," *Psychological medicine*, vol. 32, no. 6, pp. 959–976, 2002.

[22] L. S. Sakka and P. N. Juslin, "Emotion regulation with music in depressed and non-depressed individuals: Goals, strategies, and mechanisms," *Music & Science*, vol. 1, p. 2059204318755023, 2018.

[23] M. B. Donnellan, F. L. Oswald, B. M. Baird, and R. E. Lucas, "The mini-ipip scales: tiny-yet-effective measures of the big five factors of personality." *Psychological assessment*, vol. 18, no. 2, p. 192, 2006.

[24] L. Polanyi and A. Zaenen, "Contextual valence shifters," in *Computing attitude and affect in text: Theory and applications.* Springer, 2006, pp. 1–10.

[25] R. Trnka, A. Lačev, K. Balcar, M. Kuška, and P. Tavel, "Modeling semantic emotion space using a 3d hypercube-projection: an innovative analytical approach for the psychology of emotions," *Frontiers in psychology*, vol. 7, p. 522, 2016.

[26] T. Eerola and J. K. Vuoskoski, "A comparison of the discrete and dimensional models of emotion in music," *Psychology of Music*, vol. 39, no. 1, pp. 18–49, 2011.

[27] Z. Zhu, J. Li, X. Deng, Y. Hu *et al.*, "An improved three-dimensional model for emotion based on fuzzy theory," *Journal of Computer and Communications*, vol. 6, no. 08, p. 101, 2018.

[28] J. A. Russell, "A circumplex model of affect." *Journal of personality and social psychology*, vol. 39, no. 6, p. 1161, 1980.

[29] S. Buechel and U. Hahn, "Word emotion induction for multiple languages as a deep multi-task learning problem," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1907–1918.

[30] M. M. Bradley and P. J. Lang, "Measuring emotion: the self-assessment manikin and the semantic differential," *Journal of behavior therapy and experimental psychiatry*, vol. 25, no. 1, pp. 49–59, 1994.

[31] P. Saari, M. Barthet, G. Fazekas, T. Eerola, and M. Sandler, "Semantic models of musical mood: Comparison between crowd-sourced and curated editorial tags," in *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, 2013, pp. 1–6.

[32] M. Buccoli, M. Zanoni, G. Fazekas, A. Sarti, and M. B. Sandler, "A higher-dimensional expansion of affective norms for english terms for music tagging." in *ISMIR*, 2016, pp. 316–322.

[33] F. H. Rachman, R. Sarno, and C. Fatichah, "Music emotion classification based on lyrics-audio using corpus based emotion." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 8, no. 3, 2018.

[34] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[35] A. B. Warriner, V. Kuperman, and M. Brysbaert, "Norms of valence, arousal, and dominance for 13,915 english lemmas," *Behavior research methods*, vol. 45, no. 4, pp. 1191–1207, 2013.

[36] M. Zentner, D. Grandjean, and K. R. Scherer, "Emotions evoked by the sound of music: characterization, classification, and measurement." *Emotion*, vol. 8, no. 4, p. 494, 2008.

[37] J. K. Vuoskoski and T. Eerola, "Domain-specific or not? the applicability of different emotion models in the assessment of music-induced emotions," in *Proceedings of the 10th international conference on music perception and cognition*, 2010, pp. 196–199.

[38] R. Ferrer and T. Eerola, "Looking beyond genres: Identifying meaningful semantic layers from tags in online music collections," in *2011 10th International Conference on Machine Learning and Applications and Workshops*, vol. 2. IEEE, 2011, pp. 112–117.

[39] ——, "Semantic structures of timbre emerging from social and acoustic descriptions of music," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2011, no. 1, p. 11, 2011.

[40] P. Langfelder, B. Zhang, and S. Horvath, "Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r," *Bioinformatics*, vol. 24, no. 5, pp. 719–720, 2008.

[41] D. N. Klein, R. Kotov, and S. J. Bufferd, "Personality and depression: explanatory models and review of the evidence," *Annual review of clinical psychology*, vol. 7, p. 269, 2011.

[42] S. H. McKenzie, U. W. Jayasinghe, M. Fanaian, M. Passey, D. Lyle, G. P. Davies, and M. F. Harris, "Socio-demographic factors, behaviour and personality: associations with psychological distress," *European journal of preventive cardiology*, vol. 19, no. 2, pp. 250–257, 2012.

[43] S. Garrido and E. Schubert, "Music and people with tendencies to depression," *Music Perception: An Interdisciplinary Journal*, vol. 32, no. 4, pp. 313–321, 2015.

[44] M. Houben, W. Van Den Noortgate, and P. Kuppens, "The relation between short-term emotion dynamics and psychological well-being: A meta-analysis." *Psychological bulletin*, vol. 141, no. 4, p. 901, 2015.

[45] M. Goddard, B. Halligan, and N. Spelman, *Resonances: noise and contemporary music.* A&C Black, 2013.

[46] D. Miranda and M. Claes, "Music listening, coping, peer affiliation and depression in adolescence," *Psychology of music*, vol. 37, no. 2, pp. 215–233, 2009.

[47] D. Miranda, P. Gaudreau, R. Debrosse, J. Morizot, and L. J. Kirmayer, "Music listening and mental health: Variations on internalizing psychopathology," *Music, health, and wellbeing*, pp. 513–529, 2012.

PROGRAMMING INEQUALITY: GENDER REPRESENTATION ON CANADIAN COUNTRY RADIO (2005-2019)

Jada Watson

School of Music, University of Ottawa

`jada.watson@uOttawa.ca`

ABSTRACT

In May 2015, a consultant for country radio revealed a decades' long practice of limiting space for songs by female artists. He encouraged program directors to avoid playing songs by women back-to-back and advocated for programming their songs at 13-15% of station playlists. His words sparked debate within the industry and drew attention to growing inequalities on radio and within the genre. The majority of these discussions have centered on US country radio, with limited attention to the growing imbalance on the format in Canada. While country format radio in both countries subscribe to a practice of gender-based programming, Canadian program directors are governed by the federal *Broadcasting Act,* which regulates dissemination of Canadian content. Using metadata extracted from one of the main radio monitoring services – Mediabase, this paper examines gender-related trends on Canadian country format radio between 2005 and 2019. Through data-driven analysis of Mediabase's weekly reports, this paper shows declining representation of songs by women on Canadian country radio and addresses the impact of Canadian content regulations on this process.

1. INTRODUCTION

Gender has long been a central dynamic of programming in the country music industry. In the early days of the genre, male artists were associated with "public work" and commercial success, while women were tucked away in domestic, administrative and musically supporting roles [1-2]. Despite their significant contributions to the genre on the stage and behind the scenes, female artists have historically been limited on radio playlists, tours, television programs, and label rosters [3-5]. Since the 1960s, radio programmers have employed a form of gender-based scheduling as a means to structure and balance playlists; claiming they had fewer songs by women, programmers "spread out" their songs to avoid repetition [6]. This practice formalized in the late 1990s through the work of a radio consultant who developed a system for programming songs by women at 13-15% of playlists [7]. In May 2015, after nearly two decades of promoting his formula, the consultant spoke openly about his method, arguing that the format's majority female fanbase prefers male voices and advocated employing his formula to make ratings [7].

In the 5 years following this interview, research has addressed the growing imbalance on terrestrial radio. Data-driven studies have challenged the industry's growing "anti-female" myth and evaluated the decline in representation on industry charts [8-11]. Using airplay reports to investigate programming practices, recent studies revealed a 66% decline in the number of songs by women within the Top 150 on Mediabase's Yearend reports [12] and an increase in the ratio of spins for songs by men and women from 2 to 1 in 2000 to 10 to 1 by 2018. They also evaluated the distribution of spins for songs by women and found that women were underrepresented across all dayparts in the 24-hour cycle [13]. Taken together, these studies reveal the long-term impact of the gender-based programming on US country format radio.

While gender representation on US country format has been widely discussed, growing inequalities on terrestrial radio in Canada has not received the same attention. A September 2019 report took preliminary steps toward evaluating representation on Canadian country format radio [14], finding similar disparity as in the US but did not fully address the impact on Canadian artists. Given the federal laws regulating Canadian Content, it is important to consider representation through a geo-cultural lens. The present study seeks to extend this framework to consider compliance with federal content regulations and its impact on programming female artists on Canadian country radio.

Adopting methods for Big Data research in the humanities and social sciences [15-18], and influenced by prosopography [19-21], this paper presents an approach for using music industry data to study what airplay and biographic metadata can tell us about socio-cultural and institutional frameworks that govern popular cultures. To do so, I extracted weekly reports of aggregated radio airplay metadata from the Mediabase radio monitoring service's database and built a prosopography of all artists whose songs were played on Canadian country radio between 2005 and 2019. Using RapidMiner, these datasets were then joined to enable data-driven analysis of the community of artists whose songs are programmed on Canadian country radio. This study aims to answer several inter-related questions about representation. First, does the number of songs by male artists exceed those by women on Canadian country radio as it does in the US? Second, how often are songs by women programmed and at what time of day? Finally, how does this programming impact contention on weekly charts? Through each of these questions, the geo-origins of artists are considered in order to facilitate a deeper understanding of how Canadian Content regulations might help or hinder representational issues.

2. CONTEXT – CANADIAN RADIO

Broadcast radio in Canada began in the early 1920s (like the USA), with licenses for purchase from the national Department of Marine and Fisheries by groups that operated limited daytime programming [22-23]. As Canadian radio developed through the first half of the century, programming did not reflect the country's cultural heritage. By the late 1960s, as a renewed sense of nationalism emerged surrounding the country's centenary celebrations, there was a growing concern of Americanization of Canadian culture in general, and on broadcast radio specifically. Terrestrial radio was dominated by musical imports: by 1968 only 4% to 7% of the songs broadcasted were Canadian, with the remaining songs programmed primarily by musicians from the USA [22]. With little support from radio, Canadian artists struggled to break through the national market – which limited their reach outside of the country [22-23].

Debate came to a head in 1971 with the passage of the *Broadcasting Act*, which outlined Canadian content regulations that would be overseen by the newly formed Canadian Radio-Television Commission (CRTC) [24-25]. These regulations sought to carve out space for Canadian music on AM and (later) FM radio, governing the minimum percentage of songs aired and their time of day – to ensure that Canadian songs are not ghettoized in the overnights. As of 1998, stations are required to play 35% Canadian content [26-27].

In addition to a content quota, the *Broadcasting Act* also outlines criteria for evaluating a song's "Canadianness". Known by its acronym **MAPL**, the system includes the following criteria:

1. **M**usic is composed entirely by a Canadian;
2. **A**rtist performing the music or lyrics is Canadian;
3. **P**erformance is (i) recorded wholly in Canada or (ii) performed and broadcast wholly in Canada; and
4. **L**yrics are written entirely by a Canadian [28].

A musical work must fulfill two of these conditions to qualify as Canadian and count toward a station's quota.

3. DATASET

The dataset for this study was curated from the weekly airplay reports generated by Mediabase, a music industry service that monitors airplay in the US and Canada. The dataset contains 319,369 records capturing the weekly activity of the 6,675 unique songs played on Canadian country format radio between 2005 and 2019. The reports include descriptive metadata about the songs played on Canadian country radio (artist, featured artist, title, label and release year) as well as information about their weekly activity (report date, weekly ranking and status, as well as distribution of spins overall and across all 5 dayparts). The 780 weekly reports were downloaded directly from Mediabase's database, cleaned and merged, and then structured to discover gender- and race-related trends that characterize programming on country format radio. As such, they were augmented with a metadata capturing the biographic details of the artists and ensembles, including genre, ensemble type, gender, race/ethnicity and country of origins.

Country radio programmers use just two labels to code artists in their scheduling software: "male" and "female," and apply the latter designation to male-female ensembles

– even if the group has a male lead [29]. Moving outside of the industry's strict binary coding system, this study follows the practice employed in previous studies, using three codes to define artists their biological and sociological status: M for men, W for Women, and M-F for male-female ensembles [11-16]. While this coding system still works with gender binaries, the aggregated radio reports do not include transgender and gender non-binary artists. Thus, while more nuanced categories could be used, the absence of LGBTQ artists in the dataset is suggestive of larger socio-culture issues within programming that are in dire need of attention. This study thus acknowledges their absence and advocates for more inclusive programming.

4. RESULTS

The stations reporting to country format radio play contemporary country music (sometimes referred to as "country-pop"), not what one might call "classics" (i.e. songs by older generations often credited with developing the sound and culture of the genre). The reports include both current singles vying for chart contention and songs in "recurrent" status, which are comprised of songs that have exited the chart but still played regularly on radio that have become part of a station's back catalogue of standards. Generally, these songs include only those that were released within the preceding five years. Sections 4.1 to 4.3 evaluate representation across all songs played on radio, and Section 4.4 focuses just on the charting activity of current singles.

4.1 Gender representation

Over the course of this 15-year study period, songs by 1,309 artists were played on Canadian country format radio, 59.6% were men, 34.0% were women, 6.4% were male-female ensembles. While this amounts to a 60/40 split for male-only acts and acts that include women (as per radio coding), just 27.9% of all artists are played regularly on country format radio and with enough daily support to make the weekly charts. As such, even fewer female artists are heard regularly on radio. The overwhelming majority of these artists – 94.9% – are white, while 1.4% are Black, 0.8% are Indigenous, 1.1% are multi-racial, and 0.7% identify as having Filipino, Columbian, Hispanic, Latin, and Portuguese heritage. The ethnicity of 1.2% of the artists was unverifiable. This preliminary level of evaluating the identity of the artists reveals gender imbalance, to be sure, but also an overwhelming racial inequality that privileges white artists and excludes Black, Indigenous, Musicians of Colour (hereafter as BIMOC).

Table 1 parses data for the 6,675 songs played on country radio by gender and geo-origins, summarizing the number and percentage of unique songs by Canadians and non-Canadians played on Canadian country format radio between 2005 and 2019. Songs by Canadian artists consistently occupy 46% of the weekly programming. The remaining 54% of the songs are by non-Canadian artists (mostly from the USA). The majority of the songs were by non-Canadian men (37.9%) with those by Canadian men (29.4%) coming in second. Songs by Canadian and non-Canadian women and male-female ensembles occupy approximately the same percentage (16.3%) of songs.

This picture changes when evaluating representation of all songs on the weekly reports, which reveals how many times these songs are included on weekly playlists across the country. As Table 2 summarizes, 75.5% of the songs were by men, 19.3% were by women, and 5.1% were by male-female ensembles. Here, too, non-Canadian men dominate the Canadian country format soundscape, with both Canadian and non-Canadian women occupying just under 10% of weekly radio playlists. These results show that there are not just fewer songs by women (Table 1), but that they are less frequently included on station playlists (Table 2) than those by their male colleagues. The gap thus increases from 59.2% unique songs by men and women to 74.4% when evaluating the rate at which those songs appear on weekly playlists.

	Canadians	Non-Canadians
Men	1,965 (29.4%)	2,528 (37.9%)
Women	942 (14.1%)	892 (13.4%)
Male-female ens.	162 (2.4%)	186 (2.8%)
Total songs	3,069 (45.9%)	3,606 (54.1%)

Table 1. Gender representation of unique songs played on Canadian country format radio

	Canadians	Non-Canadians
Men	105,166 (32.9%)	136,052 (42.6%)
Women	31,486 (9.9%)	30,230 (9.5%)
Male-female ens.	6,747 (2.1%)	9,688 (3.0%)
Total Songs	143,399 (44.9%)	175,970 (55.1%)
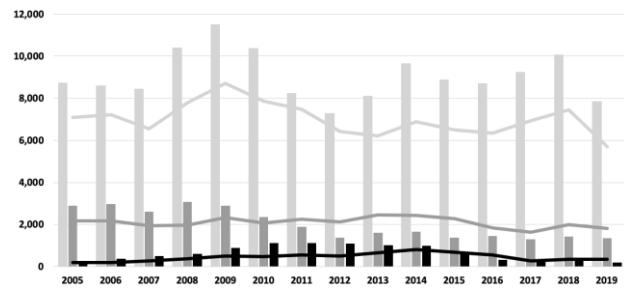
Table 2. Gender representation of all songs played on Canadian country format radio

As with the number of unique artists, 96.4% of the unique titles are by white artists. This figure does not change when evaluating the full representation of all songs on the weekly reports. In both perspectives, songs by Black artists make up just 1.3% of the songs played on Canadian country radio, with 1.0% by Indigenous artists, 0.4% by multi-racial artists and all other ethnicities performing the final 0.9%. Few titles by artists who are BIMOC are included on Canadian country radio station playlists.

Figure 1 graphs distribution of songs on the weekly reports by artists' gender and geographic origins. Songs by Canadian artists are displayed in the line graph with those by non-Canadians in the columns, with light grey representing men, dark grey for women, and black for male-female ensembles. This graph shows that all male artists were programmed at a higher rate every year in this study period. Although the number of songs by men fluctuates, they consistently average 75.6% of the songs played and increase to 82% by 2017, as the percentage of songs by women decline. Canadian male artists consistently occupy 33% of the weekly reports, with songs by non-Canadian men receiving averaging 43% of the weekly playlists.

At the start of this period, women were responsible for an average of 22.7% of the songs played. But between 2009 and 2012, there was a 33% decline in the number of songs by men and women on the weekly airplay reports. Despite the fact that the change appears to have impacted men more in this period, both suffered a 33% loss in songs

on the reports. The gap between men and women remains relatively stable through this period of decline, but steadily widens to a high of 69.2 percentage points by 2017. As the number of songs by women declined between 2009 and 2012, male-female ensembles increased from 5.3% to 9.1%, and then declined back to 2.9% by 2019. This shows more clearly the impact of coding male-female ensembles as "females" in programming, as the ensembles simply replace solo female and all-female ensembles in playlists.

Figure 1. Distribution of songs by Canadians (lines) and non-Canadians (columns)

While male artists maintain their strong placement on Canadian country format playlists, songs by female artists declined from 23.8% in 2005 down to 14.9% by 2017 – a period low for women on Canadian country radio. However, as revealed in Figure 1, this decline had a greater impact on non-Canadian women. Songs by non-Canadian women dropped 13.6% of the weekly reports to a 7.3% average for the last seven years of the study period. Canadian women, though certainly underrepresented, are not impacted to the same degree as non-Canadians, as their songs consistently make up 10% of the weekly reports.
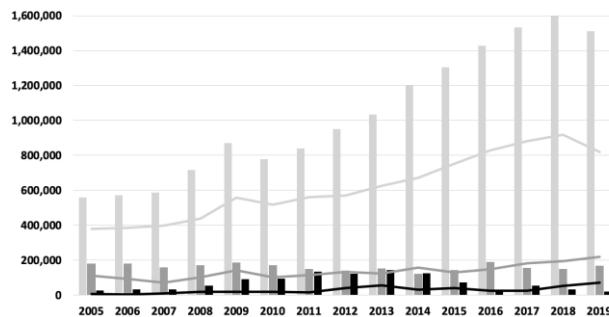
4.2 Distribution of spins

The results above show that an average of 19.3% of the songs played on Canadian country radio are by women, but that picture begins to change when evaluating how often songs are played at the level of accumulated "spins". Not only does this provide a sense of the space available for songs by women on radio, but it also aids in understanding how often the average listener actually *hears* women's voices on radio.

Figure 2 maps the distribution of all songs played on radio according to the total spins accumulated weekly, using the same geo-cultural and gender distribution as Figure 1. This figure paints a strikingly different picture of Canadian country radio culture. The number of spins for songs on country radio increased 55% overall between 2005 and 2019, with the majority of spins are granted to songs by non-Canadian men (averaging 50% of the spins), with those by Canadian men coming in second (averaging 30%). Combined, male artists average 80% of the spins over the course of this period. This leaves the remaining 20% for female artists and male-female ensembles. Songs by Canadian women average 7% of the spins on weekly playlists, while airplay for non-Canadian women declines steadily from 14% in 2005 down to 6% by 2019.

Here, too, as with the number of titles included on playlists, songs by artists who are BIMOC are drastically

underrepresented. White artists received a majority of the annual airplay, with more than 90% annually. While this level of racial inequity is disheartening, the data shows that the percentage of spins for Black and Multi-racial artists increased from 0.2% to 8.3% by 2019. These spins are divided between 15 country artists, 10 of which are men who receive 7.8% of the spins. The 4 women of colour played on radio received a combined 0.5% of the annual spins.

Figure 2. Distribution of spins for songs by Canadians (lines) and non-Canadians (columns)

The picture that emerges in Figure 2 is one of growing inequality and cultural imbalance. While the number of spins for songs by women *increases* between 2005 and 2019, their representation within the full ecosystem *decreases,* dropping 10 percentage points over the course of this period. The decline in overall representation of women can be explained by an increase in the number of country format stations from 21 stations reporting to Mediabase in 2005 to 36 by 2019. As such, songs aren't being played more, they are spread out across more stations.

4.3 Time of Day
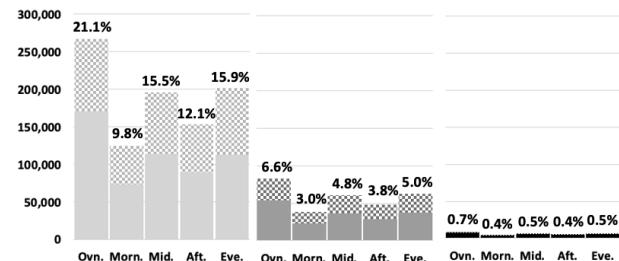
It is also important to understand the time of day that these songs are heard by the majority of radio listeners. The 24-hour period at radio is divided into 5 dayparts:

- Overnights (12:00 a.m. to 6:00 a.m.),
- Morning (6:00 a.m. to 10:00 a.m.),
- Midday (10:00 a.m. to 3:00 p.m.),
- Afternoon (3:00 p.m. to 7:00 p.m.), and
- Evening (7:00 p.m. to 12:00 p.m.) [30].
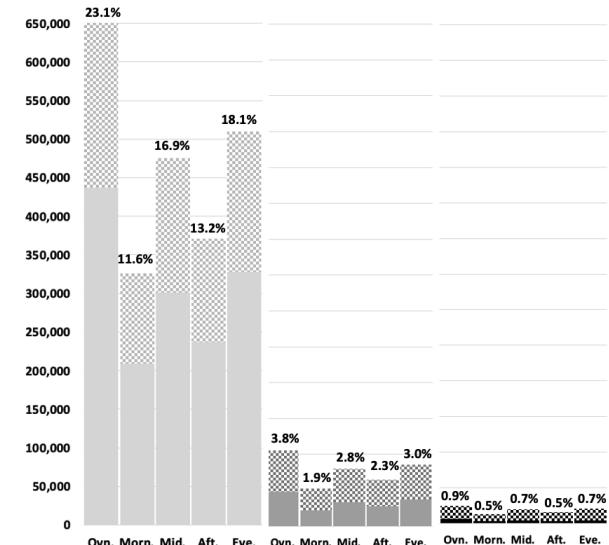
The evening and overnight dayparts register the smallest listening audiences (8% and 4% of the listening audience [31]), but this is when the majority of songs are played with 22% of the spins in the evening and 28% in the overnights. The morning, midday and afternoons periods have the highest percentages of tuned-in listeners (21%, 26% and 21%, respectively [31]); and yet, these three dayparts have the lowest percentage of songs played. Just 13% of the daytime spins occur in the morning, with 20% in the midday and 16% in the afternoon.

Figures 3a and b map the distribution of spins for men, women and male-female ensembles in 2005 and 2019 according to the time of day that songs are played and by country of origins, drilling further into the distribution of spins reported in Figure 2. The columns are presented in temporal order from the overnights (left) to the evening (right) for each category, with shading to represent geo-origins of the artists (non-Canadians in a solid shade, with

Canadians in a patterned shade). While the amount of spins for songs by women has increased (as in Figure 2), there is an overall decline in representation because of the significant increase for songs by men. In 2005, songs by men received 74.4% of the daytime spins, with 23.1% for songs by women and 2.5% for male-female ensembles. By 2019 (Figure 3b), male artists were receiving 83.3% of the total daytime spins. Despite an increase in spins for songs by women over this period, by 2019 songs by women occupy a smaller percentage of the daytime spins.

Figure 3a. Distribution of spins by daypart in 2005

Figure 3b. Distribution of spins by daypart in 2019

What is more disconcerting is the distribution across all five dayparts: songs by women register nearly the same percentage overall as men do in a single daypart for both periods. What this distribution shows is that songs by women are barely heard in daytime hours (6:00 a.m. to 7:00 p.m.), periods with the most listeners. Graphing distribution in this manner also shows clearly how the increase in spins for Canadian women factors on a 24-hour cycle. While they occupy the same space percentage of spins over this period, the number of spins distributed to Canadian women increases, while spins for non-Canadian women decreases.

Taking into account the increase in the number of stations reporting to Mediabase between 2005 and 2019, the number of songs per day can then be mapped to each daypart. As summarized in Tables 2a and 2b, songs by women are not being heard more on Canadian country format radio, they are spread across *more stations*. Thus, this type

of programming marginalizes female artists and ghettoizes their songs to a time of day when the majority of the listening audience is sleeping.

	Canadians			Non-Canadians		
	M	W	MF	M	W	MF
Overnight	25	8	1	45	14	2
Morning	13	4	0	19	6	1
Midday	21	6	0	30	9	1
Afternoon	17	5	0	23	7	1
Evening	23	7	1	30	10	1
	99	29	2	146	47	6
	30%	9%	1%	44%	14%	2%

Table 2a. Number of songs played by non-Canadian and Canadian artists in 2005

	Canadians			Non-Canadians		
	M	W	MF	M	W	MF
Overnight	25	7	2	51	6	1
Morning	14	4	1	25	3	0
Midday	20	5	2	35	4	0
Afternoon	17	4	1	28	3	0
Evening	21	6	1	38	4	1
	96	26	8	177	20	2
	29%	8%	2%	54%	6%	1%

Table 2b. Number of songs played by non-Canadian and Canadian artists in 2019

4.4 Women on the Charts

How often a song is played on radio significantly impacts its chart contention. Songs that enter and climb the weekly airplay charts receive the most weekly spins across Canada's reporting stations and are most often heard by audiences. Just 32.7% of the unique songs played on country radio appeared on weekly airplay charts, 75% by men, 19% by women and 6% by male-female ensembles. While the charts are certainly not a measure of "quality", they are an industry standard of measuring a song's success and are integral to an artist's development: success on the airplay chart is linked to opportunities within the industry, including tours, festivals and eligibility for industry awards.
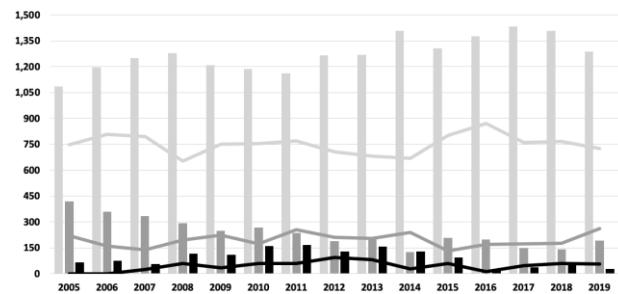
Between 2005 and 2019, the average amount of spins needed for a song to break into the 50-position chart increased 50% from 100 to 150 weekly spins. As this base number increased, the number of charting songs by women decreased. Figure 4 maps the distribution of these songs on the 50-position weekly airplay chart, showing the dominance of non-Canadian men (46.7%), followed by Canadian men (28%). Representation of female artists declines from 25.2% to 12.2% in 2018 before increasing to 17.7% (with 10.2% for Canadians and 7.5% for non-Canadians).

Given the racial inequity noted at the level of the full weekly reports, it is not surprising that the 50-position chart is dominated by white artists. Though significantly underrepresented, there have been increasingly more songs by artists who are BIMOC – from 1.0% in 2005 to 9.2% by 2019. These songs, however, are performed by just 14 unique artists, 10 of which are men and who are responsible for 80% of the 83 charting songs by BIMOC.
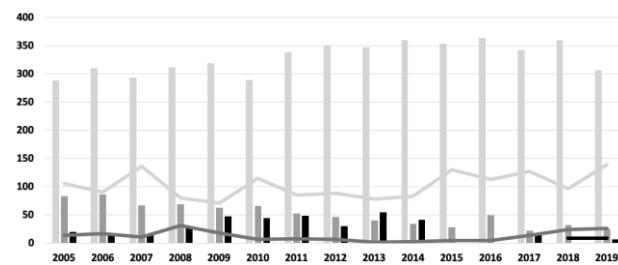
Representation worsens when drilling into the top positions on the weekly airplay charts. The bar for entering the

Top 10 positions on the weekly chart increased 59%, from 390 to 950 weekly spins by 2019. This, coupled with declining number of songs by women, resulted in a gradual disappearance of their songs from the top positions. As visible in Figure 5, the same general geo-cultural pattern is maintained: non-Canadian men dominate the chart and Canadian men come in second. Unlike above, however, Canadian women are significantly underrepresented in the Top 10, with an average of just 2.4% of the songs against 9.8% by non-Canadian women. By 2019, songs by both groups make up 5% of the annual Top 10.

Representation in the top positions of the chart is particularly dire when evaluating racial equity: just 3.5% of the Top 10 songs are by artists who are BIMOC. Despite the noted absence overall, there percentage of Top 10 songs by artists who are BIMOC increases from 0% in 2005 to 8.6% by 2019. Just two women of colour – Ojibwe artist Crystal Shawanda and Columbian-Canadian Kira Isabella – were responsible for five Top 10 songs between 2008 and 2014, none of which reached the top of the chart. Women of colour are thus absent from the top 10 positions of the chart, significantly limiting their exposure and shutting them out of opportunities within the industry.

Figure 4. Distribution of songs in the Top 50 chart by Canadians (lines) and Non-Canadians (columns)
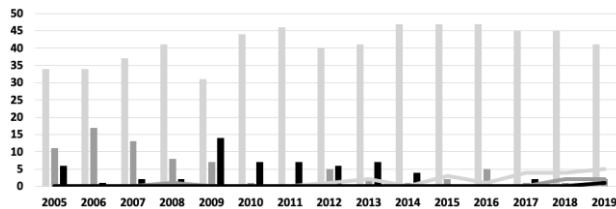
Figure 5. Distribution of Top 10 songs in the chart by Canadians (lines) and non-Canadians (columns)

The picture is most bleak at the top of the chart. Reaching the #1 position on an airplay chart requires a significant amount of weekly spins. As with other benchmark positions, the number of spins required for a song to reach #1 more than doubled between 2005 and 2019, increasing from 550 to 1,370 weekly spins.

Figure 6 reveals that the majority of the #1 songs in 2005 were by men (66%), increasing to a high of 96% in 2015 before declining to 90% by 2019 – and almost exclusively for non-Canadian artists. There is a period high of 36% for songs by women in the #1 spot in 2006, but this is followed by a steady decline to no songs in 2011 and an

average of 3% in the final years. Canadian artists are nearly absent from the top of the chart in the first 7 years of the period. With the exception of one #1 song for Terri Clark in 2008, Canadians were completely shut out of the top position of the chart until 2012, when Canadian male artists begin to achieve #1 songs – averaging three a year. Given the trends discussed thus far, Canadian women (and male-female ensembles) are the most underrepresented group, with just five #1 songs over 15 years. As the space available for women on station playlists declines, that accolade becomes increasingly unattainable for female artists.

Figure 6. Distribution of #1 songs on the weekly chart by Canadians (lines) and non-Canadians (columns)

5. DISCUSSION

The findings of this study illustrate that gender and racial inequality plagues Canadian country format radio. Not only do white, male artists have more songs on Canadian country radio overall and annually, but their songs are also played more often throughout the 24-hour cycle. The findings here echo the inequality identified in studies of US country format radio [12-13]. In Canada, as in the US, the gap between songs by men and women increases from 67.9% in 2005 to 76.5% by 2019 – a disparity that holds true (and indeed worsens) at the level of weekly airplay, time of day programming, and on the weekly charts. Racial inequity, though not surprising for this genre, is particularly problematic: while there has been an increase in representation of artists who are BIMOC over this period, their voices are nearly absent from radio. Only 14 artists who are BIMOC had enough airplay to have charting songs, just 4 of whom are women. Thus, over the course of this 15-year period, not only have songs by women started to disappear from the Canadian country charts and from the #1 position on the weekly reports, but women of colour are excluded from participation.

The Canadian content regulations offer another layer to this discussion. While this study revealed that 44% of the songs played on radio are performed by Canadians, those songs receive just 38.2% of the annual spins, with 30.3% songs by men, and just 6.6% by women and 1.3% by male-female ensembles. Thus, even though stations are fulfilling their content requirements, they average just above the quota and privilege non-Canadian artists. Indeed, non-Canadian men dominate at all levels of programming: they have the most songs, the most spins, the highest percentage of daytime programming and the most charting songs. Canadian men are second to non-Canadian men at all levels, with the exception of #1 songs – where non-Canadian women rank second overall in the number of chart-topping songs. While Canadian women are underrepresented, to be sure, the trend at all levels of analysis (with the exception

of the #1 position) is one of decline for non-Canadian women against an increase in songs, spins and daytime activity for Canadian women. This is not simply a result of the dwindling number of songs by women in the broader country music market but is suggestive of a trend in which program directors are creating more opportunities for the Canadian female artists – even if marginal.

Drilling into these weekly reports in this manner reveals the dominance and success of non-Canadians. Canadian artists receive enough spins to fulfill federal regulations on programming but are not favored enough within daily programming to have greater successes within the top positions of the weekly charts. Their near absence from the top of the chart– especially in the case of Canadian women – reveals a limit on success for the country's most prominent country artists on Canadian radio. While this of course makes business sense when considering that Nashville, Tennessee has been the centre of the industry since the genre formed, this is a missed cultural opportunity to further develop the industry north of the 49th parallel.

6. CONCLUSION AND FUTURE WORK

Music industry data offers a unique opportunity to evaluate the changing dynamics of a genre's culture and gain a better understanding of the composition of its cultural ecosystem. What emerges in the results of this study is a feedback loop that has slowly eliminated opportunities (in the form of daily airplay) for female artists, and gradually erases them from the industry's ecosystem. Women of colour are most impacted by this practice and excluded from participation. Program directors use this same airplay data to make programming decisions for their stations and then use the absence of songs by women on weekly reports and popularity charts to justify and maintain a gender-based programming practice. This is what data scientists refer to as "digital redlining" [32-33], a system by which data indirectly or directly uses criteria like gender, sexuality, and ethnicity to make assessments and recommendations.

Beyond the impact that these practices have on the livelihood of female artists, gender-based programming is also culturally damaging. Current practices, which focus heavily on repetition of songs by white, male voices, creates increased familiarity with those male artists and results over time in a more homogenized sound [34]. In this context, female voices – especially of artists who are BIMOC, have become increasingly *unfamiliar* to country radio audiences. This type of programming completely alters the public's perception of who is contributing to country music culture and contributes to a growing crisis of inequality.

Radio airplay is only one part of the story regarding gender inequality. The lack of radio airplay for songs by women deters labels and publishers from investing in female artists, and songwriters are discouraged from writing songs for them [35]. As such, next steps in this project will evaluate representation across label and publishing rosters to better understand how they operate within this system. In this regard, gender-identity and sexuality must be incorporated into future studies to investigate how these practices have been structured to establish and maintain the white, male, heteronormative discourse that pervades country music's narrative and culture.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] M.A. Bufwack and R.K. Oermann, *Finding Her Voice: Women in Country Music, 1800-2000*. Nashville, Tennessee: The Country Music Foundation Press & Vanderbilt University Press, 2004.

[2] K.M. McCusker, "Gendered stages: Country music, authenticity, and the performance of gender," in *The Oxford Handbook to Country Music,* T. D. Stimeling, Ed. New York, New York: Oxford University Press, 2017, pp. 355-74.

[3] K. Heidemann, "Remarkable women and ordinary gals: Performance of identity in songs by Loretta Lynna and Dolly Parton," in *Country Boys and Redneck Women: New Essays in Gender and Country Music,* D. Pecknold and K.M. McCusker, Eds. Jackson, Mississippi: University of Mississippi Pres, 2016, pp. 166-88.

[4] K.M. McCusker, *Lonesome Cowgirls and Honky Tonk Angels: The Women of Barn Dance Radio.* Champaign, Illinois: University of Illinois Press, 2008.

[5] E. Weisbard, "Country radio: The dialectic of genre and format," in *The Oxford Handbook to Country Music,* T. D. Stimeling, Ed. New York, New York: Oxford University Press, 2017, pp. 229-48.

[6] B. Keel, "Sexist 'tomato barb launches food fight on Music Row," *The Tennessean,* 25 May 2015, https://www.tennessean.com/story/entertainment/music/2015/05/27/sexist-tomato-barb-launches-food-fight-music-row/28036657, retrieved April 11, 2020.

[7] R. Penuell, Interview with Keith Hill on "Music scheduling," *Country Aircheck,* vol. 499, 2015, pp. 8.

[8] D. Ghosh, "The meaningless Florida-Georgia Line Billboard country songs record: Who really has the biggest country hit?" *Mjs Big Blog,* 2 August 2013, https://www.mjsbigblog.com/the-meaningless-florida-georgia-line-billboard-hot-country-songs-record-who-really-has-the-biggest-country-hit.htm, retrieved April 11, 2020.

[9] D. Ghosh, "Country radio & the anti-female myth: a data-based look," *Mjs Big Blog,* 29 May 2015, https://www.mjsbigblog.com/country-radio-the-anti-female-female-myth.htm, retrieved April 11, 2020.

[10] Trigger, "Billboard changes country chart rules, boosts 'crossover' songs," *Saving Country Music,* 11 October 2012, https://www.savingcountrymusic.com/billboard-changes-country-chart-rules-boosts-crossover-songs/, retrieved April 11, 2020.

[11] J. Watson, "Gender on the *Billboard* Hot Country Songs chart, 1996-2016," vol. 42, no. 5, pp. 538-660, 2018.

[12] J. Watson, "Gender representation on country format radio: A study of published reports from 2000-2018," *SongData Reports,* 26 April 2019, https://songdata.ca/wp-content/uploads/2019/04/SongData-Watson-Country-Airplay-Study-FullReport-April2019.pdf, retrieved April 11, 2020. Prepared in consultation with WOMAN Nashville.

[13] J. Watson, "Gender representation on country format radio: A study of spins across dayparts (2002-2018)," *SongData Reports,* 6 December 2019. https://songdata.ca/wp-content/uploads/2019/12/SongData-Watson-Country-Airplay-TODStudy-December2019.pdf, retrieved April 11, 2020. Prepared in consultation with WOMAN Nashville.

[14] J. Watson, "Gender representation on Canadian country format radio: A study of published reports from 2005-2018," *SongData Reports,* 6 September 2020, https://songdata.ca/wp-content/uploads/2019/09/SongData-Watson-Country-Airplay-Canada-FullReport-September2019.pdf, retrieved April 11, 2020.

[15] M. Lafrance, L. Worcester, and L. Burns, "Gender and the *Billboard* Top 40 Charts between 1997 and 2007," *Popular Music & Society,* vol. 34, no. 5, pp. 557-70, 2011.

[16] M. Lafrance, C. Scheibling, L. Burns, and J. Durr, "Race, gender, and the *Billboard* Top 40 Charts between 1997 and 2007," *Popular Music & Society,* vol. 41, no. 5, pp. 522-38, 2018.

[17] F. Moretti, *Graphs, Maps and Trees: Abstract Models for Literary History.* New York, New York: Verso, 2005.

[18] S. Rose, S. Tuppen, and L. Drosopoulou, "Writing a Big Data history of music," *Early Music,* vol. XLIII, no. 4, 649-660, 2014.

[19] A. Cameron, *Fifty Years of Prosopography: The Later Roman Empire, Byzantium and Beyond.* New York, New York: Oxford University Press, 2003.

[20] C. Crompton and M. Schwartz, *Lesbian and gay liberation in Canada prosopography,* (n.d.), https://prosopography.lglc.ca/about, retrieved April 11, 2020.

[21] K. Verboven, M. Carlier, and J. Dumolyn, "A short manual of the art of prosopography," in *Prosopography Approaches and Applications: A Handbook,*

K. Keats-Rohan, Ed. New York, New York: Oxford University Press, pp. 35-70, 2007.

[22] P. Audley, *Canada's Cultural Industries: Broadcasting, Publishing, Records and Film.* Toronto, Ontario: J. Lorimer in association with the Canadian Institute for Economic Policy, 1983.

[23] E. Spalding, "Turning point: The origins of Canadian content requirement for commercial radio," *Journal of Canadian Studies,* vol. 50, no. 3, pp. 669-690, 2016.

[24] Canada, Broadcasting Act (S.C. 1991, c. 11), *Gov of Canada,* last amended 11 July 2019, https://laws-lois.justice.gc.ca/eng/acts/B-9.01/, retrieved April 11, 2020.

[25] Canada, "Chapter XVIII: Radio broadcasting," *Royal Commission on National Development in the Arts, Letters and Sciences: Report,* Ottawa, King's Printer, 1951, http://www.collectionscanada.gc.ca/2/5/h5-440-e.html, retrieved April 11, 2020.

[26] Canadian Radio-Television and Telecommunications Commission, "Content made by Canadians," *Gov. of Canada,* last modified 12 March 2020, https://crtc.gc.ca/eng/cancon.htm, retrieved April 11, 2020.

[27] Unknown, "The history of Canadian broadcast regulation," *History of Canadian broadcasting, Canadian communications foundation,* 2020, https://www.broadcasting-history.ca/history-canadian-broadcast-reulation, retrieved April 11, 2020.

[28] Canadian Radio-Television and Telecommunications Commission, "The MAPL system – defining a Canadian song," *Gov. of Canada,* last modified 10 August 2009, https://crtc.gc.ca/eng/info_sht/r1.htm, retrieved April 11, 2020.

[29] L. Gifford, "Interview with Keith Hill," *Radio Stuff Podcast*, 2 June, https://omny.fm/shows/radio-stuff-podcast/keith-hill-country-radio-consultant, retrieved July 31, 2020.

[30] Mediabase, Dayparts, *Mediabase* (website). http://www.mediabase.com/mmrweb/mmrhelp/Daypart.htm, retrieved April 11, 2020.

[31] P. Bouvard, "Perception vs. reality: Drive time isn't the only time for AM/FM Radio," *WestwoodOne,* 19 February 2019, https://www.westwoodone.com/2019/02/19/perception-vs-reality-drive-time-isnt-the-only-time-for-am-fm-radio/, retrieved April 11, 2020.

[32] C. D'Ignazio and L.F. Klein, *Data Feminism*. Cambridge, Massachusetts: MIT Press, 2020.

[33] S.U. Noble, *Algorithms of Oppression: How Search Engines Reinforce Racism.* New York, New York: New York University Press, 2018.

[34] P.A. Russell, "Effects of repetition on familiarity and likability of popular music recordings," *Psychology of Music*, vol. 15, pp. 187-97, 1987.

[35] L. Liebig, "Carrie Underwood says, 'songwriters aren't writing for women,' but she understands why," *The Boot,* February 24, 2020, https://theboot.com/carrie-underwood-supporting-women-crs-2020/, retrieved April 11, 2020.

DANCE BEAT TRACKING FROM VISUAL INFORMATION ALONE

Fabrizio Pedersoli   **Masataka Goto**

National Institute of Advanced Industrial Science and Technology (AIST), Japan

{fabrizio.pedersoli, m.goto}@aist.go.jp

ABSTRACT

We propose and explore the novel task of *dance beat tracking*, which can be regarded as a fundamental topic in the *Dance Information Retrieval* (DIR) research field. Dance beat tracking aims at detecting musical beats from a dance video by using its visual information without using its audio information (i.e., dance music). The visual analysis of dances is important to achieve general machine understanding of dances, not limited to dance music. As a sub-area of Music Information Retrieval (MIR) research, DIR also shares similar goals with MIR and needs to extract various high-level semantics from dance videos. While audio-based beat tracking has been thoroughly studied in MIR, there has not been visual-based beat tracking for dance videos.

We approach dance beat tracking as a time series classification problem and conduct several experiments using a Temporal Convolutional Neural Network (TCN) using the AIST Dance Video Database. We evaluate the proposed solution considering different data splits based on either "dancer" or "music". Moreover, we propose a periodicity-based loss that considerably improves the overall beat tracking performance according to several evaluation metrics.

1. INTRODUCTION
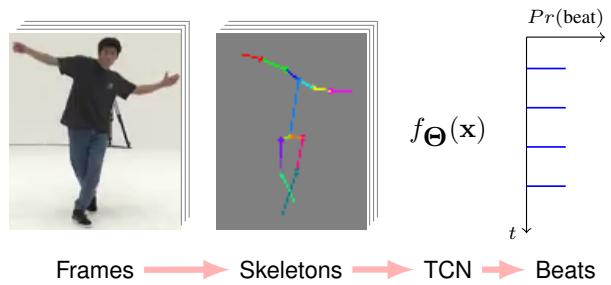
One of core tasks of *Dance Information Retrieval (DIR)* [1] is to extract high-level semantics from dance videos, which could be similar to what Music Information Retrieval (MIR) tasks attempt to detect from music. For instance, some common tasks among the two research fields are: beat tracking, structure analysis, genre recognition, and automatic tagging. Although DIR shares similar objectives with MIR, DIR tasks are typically solved by analyzing video frames of dance motions (visual information). Of course, those tasks could also be solved by analyzing audio signals of dance music (audio information) when such

---

[1] Dance Information Retrieval is almost the same as *Dance Information Processing* [1] as Music Information Retrieval often means Music Information Processing/Research, but in this paper we use the term Dance Information Retrieval to focus on tasks analyzing dance information, which could be either audio or video.

Figure 1. Our approach for dance beat tracking.

signals are given or by analyzing both visual and audio information (multimodal information). As research on dance motions has not yet received much attention in the MIR community [1], the goal of this paper is to initiate *dance beat tracking* as a novel DIR task. Dance beat tracking is named to differentiate it from a standard task of music beat tracking and is defined as the task of detecting musical beats by using only visual analysis of video frames. Figure 1 shows an overview of our approach for dance beat tracking.

Since dance motions are usually related to the accompanying dance music, several dance characteristics can be inferred by joint analysis of motion and music. In fact, various researchers have already worked on multimodal aspects of dance music and motions [2–5]. In the MIR community, dance music such as traditional dances [6–9], electronic dance music [10–14], and ballroom dance music [15–17] has been a popular target of research. The literature on analysis of dance motions by using only video frames, however, is rather limited [1, 18, 19]. To the best of our knowledge, no work has focused on dance beat tracking using visual information of dance videos and evaluated its performance.

As music beat tracking is one of the most fundamental MIR tasks, dance beat tracking is also one of the most fundamental DIR tasks. *Beat* is the basic unit of time and can be used as a basis for further processing. For example, beat-synchronous analysis is effective and frequently used in the MIR community: music audio signals and dance videos could be divided into temporal sections associated with beats, which are then used to obtain beat-synchronous or beat-wise representations for various higher-level tasks [20–24]. Some direct applications of dance beat tracking systems would include automatic synchronization of dancing with music. Although dance videos usually have video frames synchronized with mu-

sic audio signals, there are irregular video files such as those in which the timing of video frames is out-of-sync with audio signals, and those in which a dancer is dancing without music or at different tempi. Dance beat tracking is useful for synchronizing and temporally-aligning (time-stretching) such video frames, or even identifying such out-of-sync videos.

Whether it is possible to automatically track beats of a dance video using only video frames is an open question [25]. To answer this question, we developed a dance beat tracking system that extracts skeletal body keypoints of a dancer from each video frame and uses Temporal Convolutional Neural Network (TCN) architectures to classify each frame as either a "beat" frame or a "non-beat" frame. In our experiments with a shared large-scale dance database, the AIST Dance Video Database [1], we found that it is possible to achieve dance beat tracking with the best F-measure performance of 61.20% and there is still large room for improvement. We also found that TCN architectures are more effective than architectures based on bidirectional Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs), and that the use of an additional loss term based on periodicity, which we propose in this paper, considerably improves beat tracking performances.

2. RELATED WORK

2.1 Audio-based music beat tracking

Initial work on music beat tracking for audio signals was based on spectral features, such as onset strength. By relying on these features, previous studies proposed multiple beat tracking agents [26, 27]. Further research on beat tracking was based on a dynamic programming framework [28–30]. Moreover, solutions based on the Kalman filter for detecting the beat locations were studied as well [31, 32]. Another popular way of approaching beat tracking was through the bar pointer model, originally proposed by Whiteley et al. [33], and improved by others [34, 35].

More recently, beat tracking research has largely adopted deep learning models to predict the beat positions, mainly by means of RNNs. The core idea of these deep learning solutions is to feed spectrogram frames, or features extracted from them, into an LSTM RNN. The network outputs the beat activation function, which is post-processed, usually by HMM decoding, to obtain the final beat locations. Previous work which adopted this processing setup is described in the papers of Durand et al. [36], Krebs et al. [37], Böck and Schedl [34], Böck et al. [38], and Cheng et al. [39].

RNN architectures have recently started to be replaced by more computationally efficient deep learning models such as Temporal Convolutional Networks (TCNs) [40, 41]. TCNs have also been used for beat tracking, as in the papers of Davies and Böck [42] and Böck et al. [43].

2.2 Visual or multimodal dance analysis related to beats

Guedes et al. [44] developed Max/MSP objects to control the tempo and rhythm of a music performance by using dance movements instead of a conducting baton. Although those objects extracted musically relevant rhythmic information from the video frames, they did not detect any dancer and their purpose was different from dance beat tracking. Ho et al. [45] developed a multimodal system that evaluated a street dance performance by estimating how well dance motions from a Kinect device correlated with musical beats. The system detected motion velocity drops as candidate motion beats, which did not necessarily have regular intervals, and then evaluated the synchronization between their candidates and musical beats.

Automatic dance motion generation for artificial characters, such as robot dancers and computer-graphics animation dancers, often needs beat-related audio-visual dance analysis. Ohkita et al. [46] presented a multimodal audio-visual beat tracking algorithm that enabled a robot to dance in synchronization with music and a human dancer. Audio-visual features were also used in the work of Shiratori et al. [47]. In their work, the authors proposed a method that synthesized a robot dance performance imitating the performance of a human dancer listening to the same music.

In addition, Davis et al. [18] presented a method that extracts visual rhythm from motions in video and aligns it with its musical counterpart. Visual rhythm was also at the base of a Xie et al. [48] article. In their work, the authors extracted several features from video frames, and proposed the use of an attention network to align them to the correspondent audio onsets through sequence labeling layers.

Although the above references do not directly deal with the task of dance beat tracking, they show its potential applications.

2.3 Orchestra conductor analysis

In the work of Huang et al. [49] body movements of the conductor were analyzed with the goal of inferring musical expressiveness. The authors proposed a multi-task learning model based on a bidirectional RNN to jointly identify dynamic, articulation, and phrasing cues of music from conducting movements.

A similar study on orchestral director movement was described in Schmidt et al. [50]. Musical beats were detected in correspondence of a director's hand's velocity peaks, where the movements were recorded by using a wrist e-watch. The paper of Schramm et al. [51] also focused on a director's hand gestures acquired by Kinect with the purpose of detecting duple, triple, and quadruple patterns by using a probabilistic Dynamic Time Warping framework.

Although motion analysis of dancers and conductors could have some technical similarities, conductor motions tend to give more explicit cues for musical beats. On the other hand, since dancer motions do not necessarily corre-

late with musical beats, dance beat tracking is harder than conductor analysis in general.

3. PROPOSED SYSTEM

Inspired by the success attained in audio-based music beat tracking, we address dance beat tracking as a sequence classification problem. According to this framework, a classifier takes as input a sequence of observations $x_{1:T} = \{x_1, \dots, x_T\}$, and produces an output of the same length $y_{1:T} = \{y_1, \dots, y_T\}$ where each observation is classified into "beat" ($y = 1$) or "non-beat" ($y = 0$), by taking into account past observations. In our application each observation consists of a video frame.

Two main technical challenges of dance beat tracking are modeling long time sequences and extracting meaningful descriptors from video frames. The former challenge has been successfully tackled in recent years initially by using RNNs and then by using TCNs [41]. TCNs, as explained in Section 3.1, are deep learning architectures based on stacks of causal dilated convolutions [40] which serve the same purpose as an RNNs while offering several advantages over them. TCNs are more computationally efficient since convolutions can be easily parallelized. In addition, TCNs do not suffer from exploding/vanishing of gradients, which is a major drawback when dealing with long time sequences such as the one used in this application. The latter challenge can be dealt with by using standard computer vision convolutional networks to extract meaningful features from the video frames.

We thus chose to use a TCN as a sequence classifier. From each video frame (60 fps) we extract the (x, y) position of dancer body keypoints by using the OpenPose framework [52]. Thus, we represent a video as a sequence of keypoints (Section 4.2). Although extracting the body keypoints requires preprocessing of the dance video, we found this description to be at the same time powerful for effectively modeling dance movements. We do not directly use video frame pixels since it is difficult to prepare a training dataset with sufficient variations of dancer clothes, colors, and backgrounds, and using video frame pixels limits generalization of the model.

3.1 Temporal convolutional networks

TCNs process the input x_n by taking into account only the past information, and produces an output y_n of the same length as the input. To achieve this goal, 1D causal[2] convolutional layers with the "same" padding[3] are used. In order to model long time sequences, the network must have a large receptive field. We therefore use a stack of *dilated* convolutional layers so that we can increase the receptive field while maintaining the same (small) kernel size of each layer. Each layer of the stack has the same number of features. The dilation factor increases in an exponential way

at each convolution stack. More precisely, at a particular network level i, the dilation factor d is 2^i.

Stacking more dilated convolutional layers to model longer time sequences results in a deeper network, which is harder to train. It has been shown that for very deep networks, training on residual connections ensures a better gradient flow which allows more effective training [53].

In our work we use a TCN in the configuration proposed by Bai et al. [41]. The authors proposed a deep learning architecture which is composed of several TCN residual blocks. Each TCN residual block is composed of two dilated causal convolutional layers (of same dilation factor) and Rectified Linear Units activation (ReLU [54]). In order to accelerate the training of the model, a weight normalization layer [55] is placed after each dilated causal convolutional layer. In addition, a spatial dropout layer [56] is utilized after activations so that overfitting is mitigated. Finally, an optional 1×1 convolution is used on the identity path to match the feature map size of the input to the output when these two differ.

3.2 Network training

We train our model by using the Adam optimizer [57] with default PyTorch parameters, a learning rate of 0.5×10^{-3}, and batch size of 32. Training is stopped when the loss on a validation dataset does not improve for subsequent 30 epochs. The best model is then selected according to the best performance achieved on the validation dataset. For data augmentation, Gaussian noise $\mathcal{N}(0, 1)$ is added to the keypoint delta values before feeding them to the network.

3.2.1 Loss function

The basic loss criterion used to train the network is the cross-entropy, L_{ce}. Giving an input sequence of M observations, for each observation $m = 0, 1, \dots, M - 1$, the model outputs a softmax distribution $\hat{\mathbf{y}}_m$ over two classes: "beat" and "non-beat". Since probabilities of "beat" and "non-beat" in the ground truth are largely unbalanced, we weight the cross-entropy loss with a weight vector $\mathbf{w}$ of empirically chosen values: 1 and 0.1 respectively for "beat" and "non-beat". Thus, given a particular sample pair n of true sequence $\mathbf{y}^{(n)}$ and predicted sequence $\hat{\mathbf{y}}^{(n)}$, the weighted cross-entropy loss is defined as follows:

$$L_{ce}^{(n)} = -\mathbf{w} \sum_{m=0}^{M-1} \mathbf{y}_m^{(n)} \log\left(\hat{\mathbf{y}}_m^{(n)}\right). \qquad (1)$$

We also propose an additional loss term that takes account of *periodicity*, L_p. It is reasonable to assume that a dance is characterized by repeating patterns which are correlated to the music beats. By using the periodicity loss we inform the model that predictions made in correspondence of multiples of the music tempo T should be considered similar. In a training dataset the music tempo is known a priori and constant throughout the audio clips. Note that this additional loss term is used only during training. The periodicity loss is simply the summation of the absolute difference of predictions made at multiples of the music

---

[2] The result of the convolution at time $t = T$ is obtained using inputs at and before $t \leq T$.

[3] For a kernel $h_1, \dots, h_M$ the "same" convolution padding length is $M - 1$.

BPM	Street Dance Genre									
	BR	PO	LO	WA	MH	LH	HO	KR	JS	JB
T_0	21	23	23	23	24	25	23	23	21	22
T_1	23	23	23	23	24	23	23	23	23	24
T_2	23	24	23	24	24	23	24	23	23	22
T_3	23	24	23	24	23	24	24	25	21	23
T_4	24	23	25	23	23	23	23	23	23	23
T_5	24	24	24	24	23	23	22	24	23	24

Table 1. Video counts of the AIST Dance Video Database subset used in our work.

tempo kT:

$$L_p^{(n)} = \left\| \sum_{\substack{k=0 \\ k'=k+1 \\ k''=k'+1}}^{N_b-3} \hat{y}_{kT:k'T}^{(n)} - \hat{y}_{k'T:k''T}^{(n)} \right\|_1, \qquad (2)$$

where N_b is the number of beats contained in the ground truth sequence. The output sequence is zero-padded to a multiple of ground truth tempo.

We get the total loss by adding together the weighted cross-entropy and a scaled version of the periodicity loss. The scale parameter α is empirically chosen by grid-search. Finally, we take the average among the training batch of N samples:

$$L = \frac{1}{N} \sum_{n=0}^{N-1} L_{ce}^{(n)} + \alpha L_p^{(n)}. \qquad (3)$$

4. DATASET AND DATA PROCESSING

The AIST Dance Video Database [1] is a large-scale collection of dance videos. This database includes 10 street dance genres: "Break" (BR), "Pop" (PO), "Lock" (LO), "Waack" (WA), "Middle Hip-Hop" (MH), "LA-style Hip-Hop"(LH), "House" (HO), "Krump" (KR), "Street Jazz" (JS), and "Ballet Jazz" (JB). For each dance genre, 6 musical pieces of different tempi are used. In particular, the music tempi are: $T_0 = 80$, $T_1 = 90$, $T_2 = 100$, $T_3 = 110$, $T_4 = 120$, and $T_5 = 130$ beats per minute (bpm) for all the genres, except the "House" genre whose tempi are : $T_0 = 110$, $T_1 = 115$, $T_2 = 120$, $T_3 = 125$, $T_4 = 130$, and $T_5 = 135$ bpm.

In this work we consider dance videos where a single dancer is performing ("Basic Dance" and "Advanced Dance" in the database), and we use the frontal camera as the source of information [4]. The total number of dance videos considered in our experimental evaluation is 1396. The resolution of the videos is 1920×1080 pixels. Table 1 shows a detailed breakdown of our dataset.

4.1 Data splits

We split the data according to "music" and "dancer". For each of the split configurations, we randomly split

[4] The frontal view of the dancer is the most reliable for detecting the body keypoints because body part occlusions are minimized by this visual perspective

the data samples in training, validation, and test datasets with percentages of 70%, 20%, and 10%, respectively. In order to make balanced splits, we adopt the following strategy. In the case of the "music" data split, for each of the 60 music clips, we select the correspondent videos, and randomly split them according to the aforementioned train/validation/test percentages. We then concatenate the individual micro splits to obtain the final "music" train/validation/test splits. The same process is executed according to the individual dancer for compiling the "dancer" data splits.

4.2 Data preprocessing

All the dance videos are preprocessed by extracting the body keypoints by using the OpenPose framework [52]. We use the `BODY_25` pose model which represents the human body by 25 skeletal keypoints. However, in our application a subset of the `BODY_25` keypoints was problematic to detect with high reliability, and therefore it was discarded from the body pose. These problematic keypoints correspond to "eyes", "ears", "nose", "heels", and "big/small toe". After removing these keypoints we end up with a total of 13 keypoints: "neck", "shoulders", "elbows", "wrists", "mid hip", "hips", "knees", and "ankles". However, missed detections of body keypoints can still happen. The missed detections are recovered by spline interpolation.

The body keypoints are defined by their pixel position (x, y) within a video frame on the basis of the OpenPose output. However, this representation has the drawback of not being invariant to the dancer's position and of being dependent on the dancer's body size. To overcome these issues, we convert the absolute $(x_n, y_n)^{(i)}$ position of the i-th keypoint at time n into its displacement (delta values) in time:

$$(\Delta x_n, \Delta y_n)^{(i)} = (x_n - x_{n-1}, \ y_n - y_{n-1})^{(i)}. \qquad (4)$$

4.3 Beat activation processing

The output of network is the beat activation function; i.e., for each video frame, the model predicts its probability of being a "beat" frame. To obtain the final beat positions, a postprocessing of the beat activation function is required. In our work we use the algorithm proposed by Krebs et al. [35], which is based on HMM decoding.

5. EXPERIMENTAL SETUP

We report performance results by using several beat tracking metrics typical of music and by following the practice described in Davies et al. [58]. For our experimental evaluation, we make use of the `mir_eval` [59] software package [5].

For the evaluation we consider the first 420 frames (7 s) of each video. In addition, the first 1 s of the predicted beat sequence is discarded when computing the performance results.

[5] https://github.com/craffel/mir_eval

5.1 Model selection

In order to select the best-performing configuration of the model, we conduct hyperparameter grid-search on the number of stacks $N_{\text{stack}} \in \{3, \ldots, 12\}$ and the number of convolutional features $N_{\text{feat}} \in \{32, 64, 128, 256\}$. Since no pooling is involved in the TCN architecture, the number of convolutional features is kept constant among the entire stack. The extreme values (minimum and maximum) of these hyperparameters are chosen in a way that the model would respectively under-fit and over-fit the validation dataset.

The training is stopped when the loss on the validation dataset does not decrease for 30 successive epochs. According to the considered performance metric, the best-performing model on the validation dataset is selected. This model is then evaluated on the test dataset.

The hyperparameter grid-search reported that $N_{\text{stack}} = 7$ and $N_{\text{feat}} = 128$ yields, in the majority of the cases, the best TCN. The evaluation is done for both "dancer" and "music" data splits. We denote this configuration as TCN$_{128}^7$ and we use it as our baseline.

5.2 Periodicity loss ablation study

With the purpose of assessing the usefulness of the proposed periodicity loss term (L_p), we conduct an ablation study using TCN$_{128}^7$ as the baseline.

Additional hyperparameter grid-search is done for finding the best value of α, see equation (3). This parameter weights the contribution of L_p to the overall loss and must be carefully chosen by empirical evaluation. The tested values are: $\alpha \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$. The grid-search is done for both "dancer" and "music" data splits by training the TCN$_{128}^7$ according to the procedure previously described.

6. EXPERIMENTAL RESULTS

In the first part of this section we report the baseline performance of TCN$_{128}^7$. We also compare the baseline performance between TCN and LSTM architectures and show that the TCN architecture is the best in our application. Finally, we elaborate on the results of the ablation study of the periodicity loss term by showing its effectiveness.

We report the performance in terms of different metrics. More specifically we consider: F-measure (F), Cemgil's score (Cem), and continuity base scores [58]. The continuity scores are: Correct Metrical Level continuity required/not required (CML$_{c/t}$), and Allowed Metrical Level continuity required/not required (AML$_{c/t}$).

6.1 Baseline loss

Table 2 reports the performance results of TCN$_{128}^7$ model trained with L_{ce}. Results are subdivided according to the "dancer" and "music" data splits and are reported as percentages correct.

Examining the overall results, we acknowledge at first look the difficulty of this new task. A similar conclusion

Split	CML$_c$	CML$_t$	AML$_c$	AML$_t$	Cem	F
Dancer	44.28	46.93	47.27	49.04	52.92	55.02
Music	40.14	39.71	44.84	47.53	47.43	53.02

Table 2. TCN$_{128}^7$ results trained with the baseline loss.

Split	CML$_c$	CML$_t$	AML$_c$	AML$_t$	Cem	F
Dancer	38.66	41.49	56.23	53.03	32.17	46.56
Music	27.62	32.45	43.13	46.32	28.35	39.18

Table 3. bLSTM$_{128}^4$ results trained with the baseline loss.

about music beat tracking is drawn in when percussion is not present in the analyzed audio signal. We find that in a dance, the body movements cannot stress the tempo as efficiently as percussive sounds do. Therefore, a lower performance is reasonably expected in dance beat tracking than in music beat tracking (Section 6.3).

In addition, we notice that the performance on the dancer data split is $\approx 4\%$ higher (for all the metrics) than the performance on the music data split. The dancing style that characterizes each dancer seems to be easier to capture by the network, while the difference in choreographies for the same music piece is less effectively captured by the model.

In more detail, for both of the data splits, we see that the continuity scores obtain lower performance. Specifically, CML is the least performing metric and is followed by AML. The performance gap between CML and AML ($\approx 3\%$ for dancer and $\approx 6\%$ for music) suggests that the model tends to detect beats at half or double the correct tempo. An improvement of CML/AML scores is achieved when continuity is not required (CML$_t$ and AML$_t$). The F-measure attains the highest performance for both of the data splits, while the Cemgil's score shows a slight decrease, which is more evident for the music data split. This performance decrease indicates that the model tends to detect beats with a slight offset with respect to the ground truth position.

The main conclusions of this experimental section are as follows. (1) Detecting beats is more difficult for the "music" split. (2) Detected beats are prone to errors such as beat positions with less continuity, with a half or double tempo, and with a small deviation from the correct beats.

6.1.1 Comparison with LSTM

We provide a baseline comparison with a bidirectional LSTM RNN, whose performance results are reported in Table 3. Also in this case, we conduct similar hyperparameter grid-search as done for the TCN. In particular, we tested $N_{\text{stack}} \in \{1, 2, \ldots, 6\}$ and $N_{\text{units}} \in \{32, 64, 128, 256\}$, and found that in average for the different performance metrics, the best performing configuration is $N_{\text{stack}} = 4$ and $N_{\text{units}} = 128$. We refer to this model specification as bLSTM$_{128}^4$.

From Table 3 we notice that for both "dancer" and "music" data splits the bLSTM$_{128}^4$ performs worse than the

Loss	CML_c	CML_t	AML_c	AML_t	Cem	F
L_{ce}	44.28	46.93	47.27	49.04	52.92	55.02
$L_{ce}+\alpha L_p$	**53.05**	**54.30**	**55.23**	**57.64**	**59.02**	**61.20**

Table 4. Performance results for the "dancer" data split using the proposed loss with $\alpha = 0.05$.

Loss	CML_c	CML_t	AML_c	AML_t	Cem	F
L_{ce}	40.14	39.71	44.84	47.53	47.43	53.02
$L_{ce}+\alpha L_p$	**46.50**	**48.33**	**48.27**	**50.87**	**54.27**	**58.25**

Table 5. Performance results for the "music" data split using the proposed loss with $\alpha = 0.1$.

TCN^7_{128} in terms of almost all the considered metrics. The performance gap is quite significant for Cemgil's score and the F-measure. Notably, the AML scores are comparable, or even better ("dancer" split), than the TCN. In this particular instance, the recurrent nature of the tested model is helpful in detecting beat locations that are regularly spaced according to musical metric (half/double).

6.2 Proposed loss

From Tables 4 and 5 we assess the benefit introduced by the proposed periodicity loss term. Indeed, the proposed loss considerably improves each of the performance metrics and does so for both of the data splits. The improvement averages $\approx 7.5\,\%$ points for the dancer split and about $\approx 5.5\,\%$ points for the music split. We found by means of grid-search that the best value for the hyperparameter α is: 0.05 for the "dancer" split and 0.1 for the "music" split.

The performance gap between "dancer" and "music" data splits is also present in this case. Moreover, a similar trend of the performance scores also occurs in the combined loss experiments. In fact, sorted in ascending order of the achieved performances, the metrics are: CML, AML, Cemgil's score, and F-measure.

In the case of the "dancer" data split (see Table 4), the continuity metrics are the most improved metrics. With an improvement of $\approx 8\,\%$ points, the periodicity loss term is beneficial for detecting beats at the correct time spacing. Although less improved, Cemgil's score and the F-measure still indicate an important boost in performance by $\approx 6\,\%$ points. This means that the proposed loss is helpful for obtaining more accurate detection of beats.

In the case of the "music" data split (see Table 5), the performance improvement, although slightly less evident than in the case of the dancer split, is still consistent. For the music data split, the proposed loss improves all the performance metrics by an average of $\approx 5.5\,\%$ points. The behavior similar to the results of the dancer data split is also observed in this case. However, for the music data split we see that improvement for AML metrics is relatively low: $\approx 3\,\%$ points. In this case CML and AML results are more aligned, with the latter being $\approx 2\,\%$ points better.

To summarize, the main conclusions of this experimen-

CML_c	CML_t	AML_c	AML_t	Cem	F
81.34	81.34	94.27	94.89	73.78	88.98

Table 6. Average performance results on the audio clips.

tal section are as follows. (1) The proposed loss based on periodicity improves performance considerably. (2) The proposed loss helps in detecting beat locations which are more aligned with the correct music metric (or half/double of it). (3) The performance gap between the two data splits is still present, although slightly mitigated.

6.3 Audio-based music beat tracking

In Table 6, we report the beat tracking performance achieved on the audio clips of the AIST Dance Video Database [1] for comparison. We use the model of Böck et al. [34] in combination with the same HMM postprocessing module used for dance beat tracking.

Since the results are much better than those in Tables 4 and 5, music beat tracking is an easier task than dance beat tracking in our experiments. This is expected since the audio clips chosen for the dancing purpose have usually distinctive beats.

7. CONCLUSION

Our main contributions are as follow. (1) We propose the task of dance beat tracking which is characterized by the novelty of using visual information, in the form of motion patterns, for detecting musical beats. (2) We provide a baseline evaluation on the AIST Dance Video Database [1] considering data splits based on "music" and "dancer". By comparing the results based on those data splits, we gain deeper insights about the dance beat tracking task. In addition, we also provide a performance comparison of deep learning architectures commonly used for time series classification. In this regard, we show that TCNs outperform bidirectional LSTMs for dance beat tracking. (3) We propose the periodicity loss term, which is scaled and added to the baseline cross-entropy loss. This novel loss term takes into account motion repetitions in relationship to beats and considerably improves the beat tracking performance.

Detecting musical beats from video frames revealed to be a challenging task, and it encourages further research in this direction in order to improve the performance results. In fact, the relationship between dancer body movements and musical beats is difficult to capture due to high variability of motion patterns among different dancers and different choreographies. This challenge is similarly faced in MIR when trying to detect beats from non-percussive music. In future work we plan to investigate deep learning architectures that can directly process video frames without needing to extract the body keypoints ahead of time. Future work will also include investigation of whether it is possible for human beings to visually track beats of a dance video without listening to the accompanying sounds, and will compare machine and human performances.

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] S. Tsuchida, S. Fukayama, M. Hamasaki, and M. Goto, "AIST Dance Video Database: Multi-genre, multi-dancer, and multi-camera database for dance information processing," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 501–510.

[2] J. Li, H. Peng, H. Hu, Z. Luo, and C. Tang, "Multi-modal information fusion for automatic aesthetics evaluation of robotic dance poses," *International Journal of Social Robotics*, pp. 1–16, 2019.

[3] F. Ofli, E. Erzin, Y. Yemez, and A. M. Tekalp, "Multi-modal analysis of dance performances for music-driven choreography synthesis," in *Proceedings of the 34th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010, pp. 2466–2469.

[4] A. Camurri, B. Mazzarino, M. Ricchetti, R. Timmers, and G. Volpe, "Multimodal analysis of expressive gesture in music and dance performances," in *Proceedings of the 4th International Gesture Workshop*. Springer, 2003, pp. 20–39.

[5] G. Qian, F. Guo, T. Ingalls, L. Olson, J. James, and T. Rikakis, "A gesture-driven multimodal interactive dance system," in *Proceedings of the 5th IEEE International Conference on Multimedia and Expo (ICME)*, vol. 3, 2004, pp. 1579–1582.

[6] P. Beauguitte, B. Duggan, and J. D. Kelleher, "A corpus of annotated Irish traditional dance music recordings: Design and benchmark evaluations," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 53–59.

[7] B. Duggan, B. O'Shea, M. Gainza, and P. Cunningham, "Machine annotation of sets of traditional Irish dance tunes," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008, pp. 401–406.

[8] A. Holzapfel and E. Benetos, "The sousta corpus: Beat-informed automatic transcription of traditional dance tunes," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 531–537.

[9] L. Risk, L. Mok, A. Hankinson, and J. Cumming, "Melodic similarity in traditional french-canadian instrumental dance tunes." in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 93–99.

[10] N. Collins, "Influence in early electronic dance music: An audio content analysis investigation," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 1–6.

[11] T. Kell and G. Tzanetakis, "Empirical analysis of track selection and ordering in electronic dance music using audio feature extraction," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 505–510.

[12] M. Panteli, N. Bogaards, and A. K. Honingh, "Modeling rhythm similarity for electronic dance music." in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 537–542.

[13] H. Schreiber and M. Müller, "A crowdsourced experiment for tempo estimation of electronic dance music." in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 409–415.

[14] K. Yadati, M. Larson, C. C. Liem, and A. Hanjalic, "Detecting drops in electronic dance music: Content based approaches to a socially significant music event," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 143–148.

[15] S. Dixon, E. Pampalk, and G. Widmer, "Classification of dance music by periodicity patterns," in *Proceedings of the 4th International Conference of Music Information Retrieval (ISMIR)*, 2003.

[16] F. Gouyon and S. Dixon, "Dance music classification: A tempo-based approach," in *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, 2004.

[17] P. Knees, A. Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. Le Goff, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections." in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 364–370.

[18] A. Davis and M. Agrawala, "Visual rhythm and beat," *ACM Transaction on Graphics*, vol. 37, no. 4, pp. 122–1, 2018.

[19] Y. Xie, H. Wang, Y. Hao, and Z. Xu, "Visual rhythm prediction with feature-aligning network," in *Proceedings of the 11th International Conference on Machine Vision Applications (MVA)*, 2019, pp. 1–6.

[20] G. Griffin, Y. E. Kim, and D. Turnbull, "Beat-sync-mash-coder: A web application for real-time creation of beat-synchronous music mashups," in *Proceedings of the 35th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010, pp. 437–440.

[21] R. Nishikimi, E. Nakamura, K. Itoyama, and K. Yoshii, "Musical note estimation for f0 trajectories of singing voices based on a bayesian semi-beat-synchronous hmm." in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 461–467.

[22] S.-C. Pei and N.-T. Hsu, "Instrumentation analysis and identification of polyphonic music using beat-synchronous feature integration and fuzzy clustering," in *Proceedings of the 34th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 169–172.

[23] Y. Panagakis and C. Kotropoulos, "Music structure analysis by ridge regression of beat-synchronous audio features." in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 271–276.

[24] D. P. Ellis, C. V. Cotton, and M. I. Mandel, "Cross-correlation of beat-synchronous representations for music similarity," in *Proceedings of the 33rd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 57–60.

[25] C. C. Stauffer, J. Haldemann, S. J. Troche, and T. H. Rammsayer, "Auditory and visual temporal sensitivity: evidence for a hierarchical structure of modality-specific and modality-independent levels of temporal information processing," *Psychological research*, vol. 76, no. 1, pp. 20–31, 2012.

[26] M. Goto and Y. Muraoka, "A beat tracking system for acoustic signals of music," in *Proceedings of the 2nd ACM International Conference on Multimedia*, 1994, pp. 365–372.

[27] M. Goto, "An audio-based real-time beat tracking system for music with or without drum-sounds," *Journal of New Music Research*, vol. 30, no. 2, pp. 159–171, 2001.

[28] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[29] A. M. Stark, M. E. Davies, and M. D. Plumbley, "Real-time beat-synchronous analysis of musical audio," in *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx)*, 2009, pp. 299–304.

[30] F.-H. F. Wu, T.-C. Lee, J.-S. R. Jang, K. K. Chang, C.-H. Lu, and W.-N. Wang, "A two-fold dynamic programming approach to beat tracking for audio music with time-varying tempo," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 191–196.

[31] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing, "On tempo tracking: Tempogram representation and Kalman filtering," *Journal of New Music Research*, vol. 29, no. 4, pp. 259–273, 2000.

[32] Y. Shiu, N. Cho, P.-C. Chang, and C.-C. J. Kuo, "Robust on-line beat tracking with Kalman filtering and probabilistic data association (kf-pda)," *IEEE transactions on consumer electronics*, vol. 54, no. 3, pp. 1369–1377, 2008.

[33] N. Whiteley, A. T. Cemgil, and S. J. Godsill, "Bayesian modelling of temporal structure in musical audio." in *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, 2006, pp. 29–34.

[34] S. Böck and M. Schedl, "Enhanced beat tracking with context-aware neural networks," in *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx)*, 2011, pp. 135–139.

[35] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking." in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 72–78.

[36] S. Durand, J. P. Bello, B. David, and G. Richard, "Downbeat tracking with multiple features and deep neural networks," in *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 409–413.

[37] F. Krebs, S. Böck, M. Dorfer, and G. Widmer, "Downbeat tracking using beat synchronous features with recurrent neural networks." in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 129–135.

[38] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 255–261.

[39] T. Cheng, S. Fukayama, and M. Goto, "Convolving Gaussian kernels for RNN-based beat tracking," in *Proceedings of the 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 1919–1923.

[40] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[41] S. Bai, Z. J. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[42] M. E. Davies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," in *Proceedings of the 27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[43] S. Böck, M. E. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *Proceedings of the 20th International Society*

for Music Information Retrieval Conference (ISMIR), 2019, pp. 486–493.

[44] C. Guedes and C. Branco, "Extracting musically-relevant rhythmic information from dance movement by applying pitch-tracking techniques to a video signal," in *Proceedings of the 12th Sound and Music Computing Conference (SMC)*, 2006, pp. 25–33.

[45] C. Ho, W.-T. Tsai, K.-S. Lin, and H. H. Chen, "Extraction and alignment evaluation of motion beats for street dance," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 2429–2433.

[46] M. Ohkita, Y. Bando, Y. Ikemiya, K. Itoyama, and K. Yoshii, "Audio-visual beat tracking based on a state-space model for a music robot dancing with humans," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5555–5560.

[47] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Synthesizing dance performance using musical and motion features," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006, pp. 3654–3659.

[48] G. Xia, J. Tay, R. Dannenberg, and M. Veloso, "Autonomous robot dancing driven by beats and emotions of music," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 205–212.

[49] Y.-F. Huang, T.-P. Chen, N. Moran, S. Coleman, and L. Su, "Identifying expressive semantics in orchestral conducting kinematics," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[50] D. Schmidt, A. Smailagic, R. B. Dannenberg, D. P. Siewiorek, and B. Bugge, "Learning an orchestra conductor's technique using a wearable sensor platform," in *Proceedings of the 11th IEEE International Symposium on Wearable Computers*, 2007, pp. 113–114.

[51] R. Schramm, C. R. Jung, and E. R. Miranda, "Dynamic time warping for music conducting gestures evaluation," *IEEE Transactions on Multimedia*, vol. 17, no. 2, pp. 243–255, 2014.

[52] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[54] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[55] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 901–909.

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[58] M. E. Davies, N. Degara, and M. D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," *Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06*, 2009.

[59] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common MIR metrics," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

SESQUIALTERA IN THE COLOMBIAN BAMBUCO: PERCEPTION AND ESTIMATION OF BEAT AND METER

Estefanía Cano[1]   **Fernando Mora-Ángel**[2]   **Gustavo A. López Gil**[2]

José R. Zapata[3]   **Antonio Escamilla**[3]   **Juan F. Alzate**[2]   **Moisés Betancur**[2]

[1] Fraunhofer IDMT, Germany

[2] Universidad de Antioquia, Colombia

[3] Universidad Pontificia Bolivariana , Colombia

`cano@idmt.fraunhofer.de`

ABSTRACT

The bambuco, one of the national rhythms of Colombia, is characterized by the presence of sesquialteras or the superposition of rhythmic elements from two meters. In this work, we analyze sesquialteras in bambucos from two perspectives. First, we analyze the perception of beat and meter by asking 10 Colombian musicians to perform beat annotations in a dataset of bambucos. Results show great diversity in the annotations: a total of five different meters or meter combinations were found in the annotations, with each bambuco in the study being annotated in at least two different meters. Second, we perform a beat tracking analysis in a dataset of bambucos with two state-of-the-art algorithms. Given that the algorithms used in the analysis were designed to deal with the rhythmic regularity of a single meter, it is not surprising that tracking performance is not very high ($\simeq$42% mean F-measure). However, a deeper analysis of the onset detection functions used for beat tracking, indicate that there is enough information on the signal level to characterize the bi-metric behavior of bambucos. With this in mind, we highlight possibilities for computational analysis of rhythm in bambucos.

1. INTRODUCTION
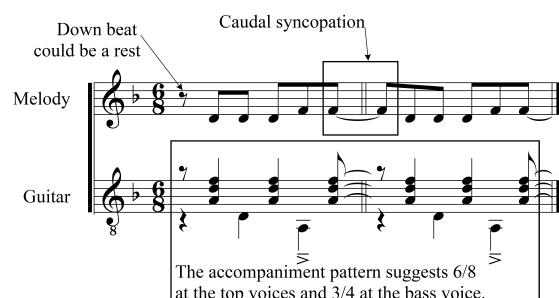
The focus of this work is the bambuco, one of the national rhythms of Colombia characterized by the superposition of musical elements in two meters, 3/4 and 6/8. This phenomenon is called *sesquialtera*, and while it is not unique to the bambuco [1, 2], this work focuses on perceptual and computational aspects particular to the Colombian bambuco. Our goal is to better understand how meter in bambuco is perceived by cultural insiders. To do so, we conducted a study where Colombian musicians were asked to

tap the beat of a selection of bambucos (Section 2.1). We then investigate whether computational tools can help ethnomusicological investigations on tendencies of bambucos to follow a given meter. We extracted beat information from a bambuco dataset using state-of-the-art algorithms and evaluate tracking performance (Section 2.5).

The contributions of this work are summarized as follows: (1) To the authors' knowledge, this paper presents the first study on meter perception in bambucos. (2) We present an objective evaluation of computational tools for rhythm analysis on bambucos, and highlight analysis possibilities for future research. (3) All the data including audios, transcriptions, annotations, and code have been made publicly available to enable future research on the topic.

1.1 The bambuco

There are references about the presence of bambuco in Colombia dating back to the mid 19th century; however, despite numerous discussions about its origin and musicological characteristics, there is no clarity today about the real origin of this music: Is it indigenous, African or Hispanic? Is it urban or peasant mestizo? Despite this uncertainty, the reality is that little by little bambuco became a regional and musical symbol of identity. Like all the great Latin American genres that fulfilled this purpose towards the end of the 19th century and the first half of the 20th (e.g. Habanera, Tango, Chacarera), to become a worthy representative of this imagined regional identity and of those who coined it, the bambuco had to undergo a

Figure 1: Bambuco example showing the downbeat, caudal syncopation and a guitar accompaniment pattern.
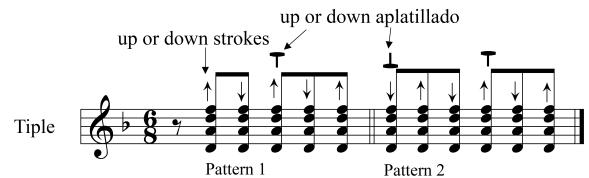
transformation process referred to as "whitening" [3]. This whitening can be understood as a progressive adherence to the bourgeois ideal of chamber music. This particular process has been studied by the Colombian ethnomusicologist Santamaria in [4]: "When relocating to the city since the mid-19th century, the bambuco progressively stopped being popular dance music and became music to be performed and listened to in an atmosphere of literary or concert gatherings".

Bambucos show musical elements typical of ancient Spanish-Iberian and Colombian peasant dances, typified as sesquialteras, whose main characteristic is a bi-metric behavior (3/4 - 6/8) within the melodic line or between the melodic line and the bass line. This behavior can be observed in the example in Figure 1 where the guitar accompaniment has elements from both 6/8 and 3/4. Another characteristic element of bambucos is the presence of *caudal* syncopation in its phrases (sixth eighth note tied to the first eighth note of the following measure -see Fig. 1) which can result in the perception of a delay or a harmonic anticipation [5]. Another element of bambucos which adds to its rhythmic complexity is the characteristic accentuation of the third pulse in the accompaniment patterns in 3/4, which leads to the perception of a downbeat that is not the first pulse of the bar.

Of the instruments that usually participate in the performance of this type of bambucos [1] (such as guitars, tiples, and bandolas [2]), the main role of the rhythmic accompaniment is usually delegated to the tiple. The tiple is a plucked string instrument slightly smaller than a guitar, with 12 strings grouped in four tripled courses. One of the instrument's most characteristic idiomatic playing techniques is the *aplatillado* which is achieved by bringing the nails closer to the strings to alter their timbre. With an alternating up and down strumming and the *aplatillado* (see Figure 2), textural elements are generated that can sometimes interfere with rhythm perception. This is similar to what happens in the charango (traditional string instrument) in certain Bolivian music [6].

Ramón y Rivera [7], proposed the term "free rhythm" in the context of Latin American music to refer to a certain elasticity in the unit of time, in breathing and in the execution of rhythmic groups, as opposed to a rhythmic reference subject to a measure or bar. This rhythmic freedom is observed in the set of recordings that are part of this study and that account not only for particularities of the genre, but also for a historical moment of the recordings not rigorously subject to a metronomic guide. Additionally, tempo and micro-timing in bambuco appear to work in general in a flexible way, with even subtle differences between timing of the melody and that of the various elements of the accompaniment. These freedoms could be associated with the rubato of European music or with the floating rhythm of jazz; however, it is a different phenomenon that contributes to the rhythmic complexity of bambuco and its per-

Figure 2: Tiple accompaniment patterns
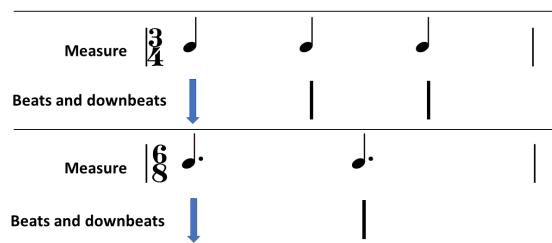
ception [3].

The different levels of complexity described in this section become critical elements when developing computational tools for musicological analysis of these music traditions.

1.2 Beat and meter perception

The perception of meter and beat in music is directly associated with the perception of regularity. It is precisely this regularity that allows the listener to create expectations about the musical events in a given time span [8]. While beat perception is mostly linked to a perceived periodicity, meter is additionally linked to an accentuation pattern that differentiates, for example, beats from downbeats. Based on these ideas, Western music theory defines a hierarchical relationship between beats, measures (bars), and meter (see Figure 3). For certain musical traditions where a unique meter cannot always be clearly defined (such as the bambuco but also Bolivian Easter songs [6], the Southern Eve dance drumming of the Guinea Coast [2], among others), Western music theory (and music notation) can often fall short in providing an accurate representation of these traditions. In the particular case of the Colombian bambuco, its correct music notation has been the source of many academic discussions [3]. Besides the superposition of 3/4 and 6/8 meters, bambuco's characteristic accentuation pattern (due to caudal syncopation and the accentuation of the third beat in 3/4 by the accompaniment - see Section 1.1) adds another layer of complexity as the traditional definition of downbeats (Figure 3) do not hold in the case of bambuco.

Of particular interest in this context is the work by Stobart et al. [6] on rhythm perception of Bolivian Easter songs. The study outlines how cultural outsiders perceived

Figure 3: Hierarchical relationship between meter, measures (bars) and beats as defined in Western music theory. In both 3/4 and 6/8, the beats are indicated with vertical lines, and the downbeat with a blue arrow.

these songs as anacrustic 6/8 rhythms, while footfalls of locals dancing to the rhythm of the music indicated a 2/4 rhythmic perception. The authors highlight that accentuation patterns of the charango (traditional string instrument) accompaniment as well as stress patterns in the local language *Quechua* in which the songs are sung, are possible causes of the differences in perception.

1.3 Music Information Retrieval (MIR) approaches for rhythm analysis

The computational analysis of musical beat has been widely addressed in the literature, predominately applied to Western popular music [9] but also applied to non-Western music [10, 11]. While beat tracking accuracy for Western popular music can already be very high, beat tracking of non-Western music presents many more challenges, and performance highly depends on the rhythmic complexity of each music tradition. In the particular case of Latin American music, work on computational analysis of rhythm has either focused on understanding characteristic patterns in micro-timing that implant local rhythms their unique rhythmic feel (e.g. Brazilian Samba [12], and Uruguayan Candombe [13]), on using rhythmic pattern templates for beat tracking (e.g. Afro-Cuban rhythms [14], and Uruguayan Candombe [15]) or on genre classification [16].

To the authors' knowledge, an in-depth computational analysis of rhythm in the Colombian bambuco has never been performed. This motivated the preliminary beat tracking evaluation where the goal was to understand how state-of-the-art tools for beat tracking perform when meters superpose in music. However, we approach this evaluation not with the expectation that the algorithms will succeed in tracking rhythmic patterns they were not originally designed to track; we approach this evaluation with the goal of understanding the potential of these techniques to be expanded into meaningful musicological analysis tools for bambucos and music from the Andes in general.

2. BAMBUCO ANALYSIS

2.1 Dataset

The data used in this study is part of the ACMUS-MIR dataset (V1.1),[3] a collection of annotated music from the Andes region in Colombia [17]. To evaluate beat tracking performance, all the bambucos in the **Rhythm Set** of the ACMUS-MIR dataset were used (N=73). From the 73 bambucos, a smaller selection of 10 bambucos were used in the perceptual study (see Table 1 for details).[5] The 10 bambucos in the perceptual study were chosen as they clearly exemplify the bi-metric behaviour of the bambuco genre, and include a diversity of instrumental formats (duets, trios, wind orchestra). Additionally, the majority of the tracks were composed by Luis Uribe Bueno, a representative composer and performer of bambuco in Colombia.

2.2 Participants

A total of 10 Colombian participants took part in the perceptual study (8 male, 2 female, ages 25-50), all of whom had been exposed to bambuco music throughout their lives (cultural insiders). All the participants had musical training, and were either university music students or professional musicians: five guitarists, two bandola players, two pianist, one flutist, one singer. The majority of the participants had previous experience performing bambucos within their musical practices.

2.3 Survey

As part of the perceptual study, each participant also answered a short survey consisting of three questions: (1) Which musical elements guided you when tapping the beat? (2) Was there any element that made the annotation process difficult? and (3) Do you have any observations about the tempo in these bambucos?[1]

2.4 Annotations

2.4.1 Beat

For the perceptual study, the 10 participants were asked to tap the beat to the selection of 10 bambucos using the computer keyboard in Sonic Visualiser.[4] Participants were given freedom to tap the beats that felt more natural to them. No indications about meter were given to the participants to avoid biasing them. Two sets of annotations were recorded: (1) Beat annotations tapped while listening to the audio (without any visual information) without allowing corrections by the participants (***Audio Only***). (2) Participants were allowed to modify their initial beat annotations in Sonic Visualiser using both audio information and a visual representation of the audio waveform. Participants were allowed to make as many corrections as necessary for them to be satisfied with their annotations (***Audiovisual + corrections***). If participants were satisfied with the ***Audio Only*** annotations, the correction step was not performed.[5]

For the computational beat tracking analysis, the annotations from the **Rhythms Set** of the ACMUS-MIR (V1.1) dataset[3] were used. With the awareness that in many cases a unique meter in bambucos cannot be defined, beat annotations in the dataset were performed independently for the two predominant meters, 3/4 and 6/8. For the 73 bambucos, these two sets of annotations were used, each assuming a unique underlying meter [17].

2.4.2 Melody, bass and chord annotations
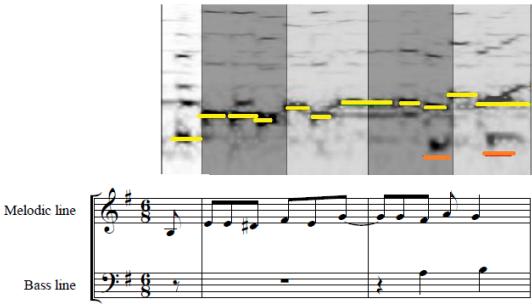
The melody line and the bass of each bambuco in the perceptual study were transcribed by four professional musicians in Colombia. The transcriptions in MIDI format were manually aligned to the audio signal resulting in time-aligned transcriptions. The chord progression of each bambuco was also annotated (see Fig. 4 for an example).[5]

Title	Composer	Tempo 3/4 [bpm]	Tempo 6/8 [bpm]	Duration [sec]	BLIND RE-VIEW IDs
Mimí	Unknown	181	121	19.4	rh_0001
Campanitas de mi pueblo	Luis Uribe Bueno	154	102	18.8	rh_0002
El espinaluno	Carlos A. Rosso Manrique	213	142	16.4	rh_0003
El marco de tu ventana	Luis Uribe Bueno	130	89	13.9	rh_0038
Baile de ranas	Luis Uribe Bueno	153	102	16.5	rh_0039
Bambuco instrumental	Luis Uribe Bueno	192	128	15.1	rh_0067
Bambuco instrumental	Luis Uribe Bueno	195	128	20.1	rh_0079
Bambuco instrumental	Luis Uribe Bueno	199	132	17.0	rh_0080
Bambuco instrumental	Luis Uribe Bueno	169	113	25.5	rh_0100
Bochicaniando	Luis Uribe Bueno	184	123	25.6	if_0172

Table 1: Selection of bambucos from the ACMUS-MIR dataset used in the perceptual study. Each segment corresponds to a complete musical idea or phrase taken from the original recording. Due to the superposition of 3/4 and 6/8 meters in these bambucos, tempo annotations for both meters are presented.

Figure 4: Example transcription of the first two measures of track rh_0067. Conventional music notation and their MIDI representation is displayed.

2.5 Automatic beat tracking

For beat tracking evaluation, two state-of-the-art algorithms were used to predict the beat positions. The first set of beat tracking estimations was obtained using Madmom. [6] In the context of the Madmom library, we specifically used a multi-model approach that uses recurrent neural networks to track beats [18]. The second algorithm used for beat estimation was the Multi-Feature Beat tracker (MultiBT) [19] implemented in Essentia. [7] This algorithm selects between beat estimations from a single beat tracking model with diverse input features. Given the bi-metric characteristics of bambucos, independent ground-truth annotations assuming either a 3/4 or 6/8 meter were used (see Section 2.4.1).

For evaluation we use a subset of metrics from the standard evaluation methods described in [20]. Among all the proposed metrics, we chose the F-measure (F1), along with the continuity measures originally defined in [21, 22]. This allows us to analyze both the ambiguity associated with the annotated metrical level and the continuity in the beat estimates. The F-measure (F1) is a generic score often used in information retrieval. For beat tracking, it is common practice to use a ± 70 ms tolerance window around annotations to consider a beat prediction as correct. The F-measure takes into consideration the number of correct beats, the

number of false positives (extra detections), and the number of false negatives (missed detections). Under this metric, completely unrelated beat sequences typically score around 25% by virtue of beats arbitrarily falling within the range of tolerance windows.

Continuity-based evaluation considers regions of continuously correct beat estimates relative to the length of the audio signal. This is the case of the Correct Metrical Level Continuity (**CMLc**) measure, which computes the ratio of the longest continuously correct segment to the length of the input. By definition, continuity is defined using a tolerance window of $\pm 17.5\%$ around each annotation, considering an estimation as correct if it falls within this window. To include the effect of beats in other segments, a less strict measure considers the total number of correct beats at the correct metrical level without the continuity criteria (**CMLt**) [20]. Lastly, to account for ambiguity in the metrical level, two additional metrics consider beats tapped at double or half the annotated metrical level, with the same continuity criteria as before. This conditions are considered *allowed* metrical levels resulting in the Allowed Metrical Level Continuity (**AMLc**) metric and its less strict alternative (**AMLt**) [20]. [8]

3. RESULTS AND DISCUSSION
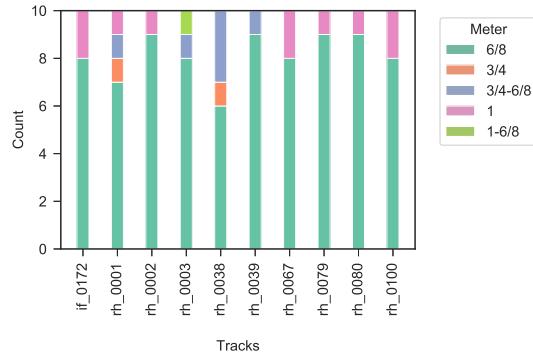
3.1 Meter perception in bambucos

The beat annotations obtained from the 10 participants (Section 2.4.1) were analyzed by three musicologists in Colombia to determine the underlying meter(s) perceived by each participant in each track. [5] Even though participants were given freedom to tap beats that felt natural to them, each annotation can be directly mapped back to a given meter. This can be understood by looking at Figure 3: If a participant taps three beats per bar, these annotations are mapped back to a 3/4 meter. Conversely, if a participant taps two beats per bar, the underlying meter is assumed to be 6/8. In total, five different meters or meter combinations were observed: 3/4 meter, 6/8 meter, a combination of 3/4 and 6/8, "one count" annotations where participants annotated the first beat of the measure (blue arrows in Fig. 3

---

[6] https://madmom.readthedocs.io/en/latest/
[7] https://essentia.upf.edu/

[8] Code available: https://github.com/ACMUS-MIR/publications-resources/tree/master/ISMIR2020
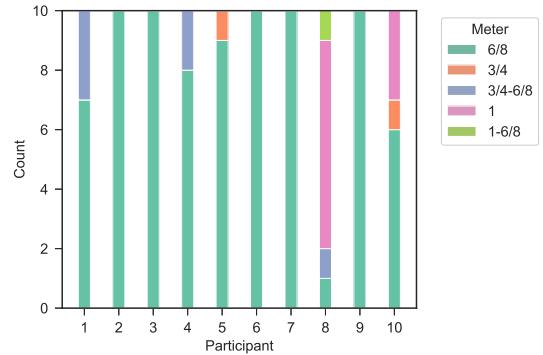
(a) Perceived meter aggregated per track

(b) Perceived meter aggregated per participant

Figure 5: (a) Perceived meter aggregated per track over the 10 participants. (b) Perceived meter aggregated per participant over the 10 tracks. Five distinct meters or meter combinations were observed.

which correspond to the downbeats in Western traditions but are not the accentuated beat in bambucos), and a combination between 6/8 and "one count" annotations. These five alternatives are denoted "3/4", "6/8", "3/4-6/8", "1", and "1-6/8", respectively.

Figure 5a shows a summary of the annotations aggregated per track from the revised annotations (*Audiovisual + corrections*). It can be seen that for each of the 10 bambucos, at least two different meters or meter combinations were perceived. The 6/8 meter proved to be predominant in the annotations. It should be noted, that as of today, bambuco is written as a convention in 6/8, and hence, there might be a tendency in trained musicians to default to 6/8.

Similarly, Figure 5b shows a summary of the revised annotations aggregated per participant. It can be seen that five of the 10 participants annotated all the tracks in 6/8 meter. Two of the participants perceived "6/8" and the "3/4-6/8" combination, and two participants perceived a "3/4" meter. Of particular interest in participant eight (p8), who predominantly annotated the bambucos in "1". This is interesting in the sense that this is the only type of annotation that removes the ambiguity in meter perception as the first beat coincides in "6/8" and "3/4" (see Fig. 3). The practice of counting music in "1" is often related to music in fast tempi, where counting all beats in a bar might no longer be comfortable. However, this is not the case here. Table 1 shows the tempo distribution of our bambuco dataset. The fastest bambuco in our dataset is rh_0003, which is mostly annotated in "6/8", with p8 choosing the "1-6/8" alternative in this case. Participant p8 annotated seven bambucos in "1", all of them with slower tempi than rh_0003.

From the 100 annotation instances in this study (10 tracks x 10 participants), a total of 10 instances showed different meters when comparing the (*Audio Only*) annotations with the revised annotations (*Audiovisual + corrections*). Three instances were modified from "3/4-6/8" to "6/8", two instances were modified from "6/8" to "3/4-6/8", two from "3/4" to "6/8", two from "6/8" to "1", and one from "6/8" to "3/4". These results further indicate the dynamic nature of meter perception in bambucos.

The responses from the participants in the survey show
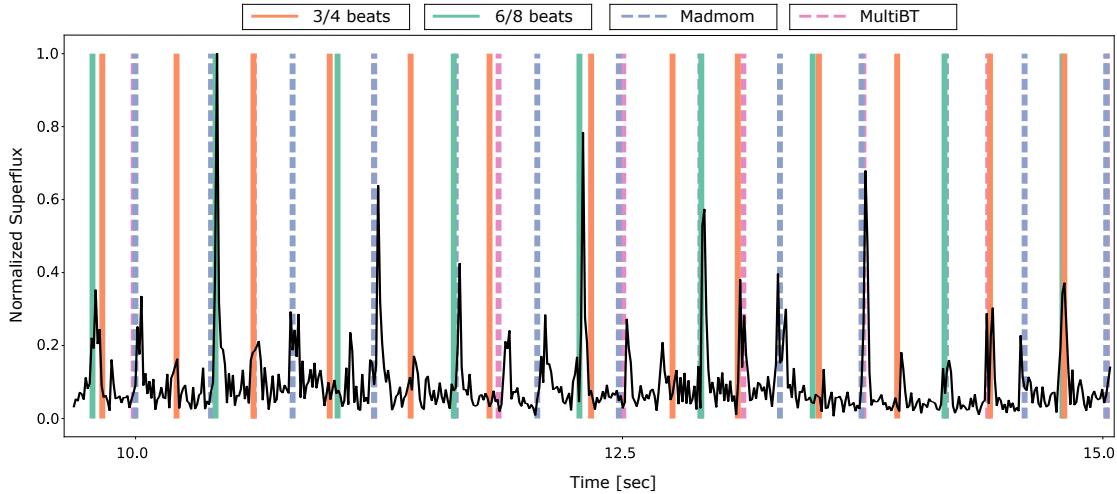
	Algorithm	F1	AMLc	AMLt	CMLc	CMLt
$\frac{3}{4}$	Madmom	75.06	60.76	77.05	50.89	64.27
	MultiBT	42.79	23.32	25.24	12.43	14.33
$\frac{6}{8}$	Madmom	41.13	9.23	10.71	5.64	5.72
	MultiBT	45.15	42.87	51.76	32.38	35.54

Table 2: Beat tracking evaluation metrics obtained with Madmom and MultiBT. Results are presented using two sets of ground-truths: 3/4 and 6/8. All metrics presented have a maximum score of 100%.

a tendency to use harmony, as well as a tendency to rely on parts of the musical discourse that are close to their personal experience (guitar or tiple players, for example, focused more on the accompaniment patterns of the guitar and the tiple). According to the participants, the main difficulties of the analysis process, in addition to flexibility in tempo, were the conception of the phrasing present in the sample, the ritardandos and accelerandos performed between different parts of the musical texture (melody and accompaniment), and the quality of the recordings. Finally, the participants observed that the tempo in these recordings behaves in an organic way, far from the metric rigor typical of the practices of current academic musicians. This is no longer frequent in the way the bambuco is performed today. This transformation could be related to the changes in recording techniques and the prescriptive function of the academic institutions and the musical events in which this type of music circulates.

3.2 Beat tracker performance on bambucos

Two independent evaluations are presented in Table 2 for each of the two beat tracking algorithms. The top row presents results obtained when ground-truth annotations assuming an underlying 3/4 meter are used. The bottom row presents results with ground-truth annotations in 6/8. Metrics that enforce continuity (CMLc and AMLc) are in all cases lower than their less strict counterparts (CMLt and AMLt). Additionally, metrics that allow estimation in different metrical levels (AMLc and AMLt) are also higher than the ones that enforce a correct one (CMLc and CMLt).

Figure 6: Onsets detection function extracted using *Superflux* on a segment of track rh_0002. Ground-truth annotations in 3/4 and 6/8 are shown, as well as beat estimations obtained with Madmom and MultiBT.

These results indicate that in certain occasions, the algorithms are tracking a higher metrical level, detecting the first beat of the bar as the underlying beat (similar to the "1" annotations in the perceptual study). As previously mentioned, this is the only beat where 6/8 and 3/4 coincide. When focusing on those metrics that only consider the correct estimations and not the false positives and false negatives, namely AMLc, AMLt, CMLc and CMLt, Madmom appears to be consistently better at estimating beats in 3/4 than in 6/8. In contrast, MultiBT shows better performance for 6/8 for the same set of metrics.

Evaluation results confirmed our initial hypothesis that the bi-metric nature of our dataset can be challenging for the beat tracking algorithms. However, to better understand the potential of beat tracking algorithms when working with our dataset, we analyzed the onset detection functions, as obtained by the spectral flux or the superflux, [6] of the 10 bambucos in our dataset. [8] Onset detection functions are intermediate signal representations often used in beat tracking algorithms that highlight time instants of the signal where onsets might be present. A peak in the onset detection function suggests that there is a high probability of an onset occurring in that position. With this analysis, the goal was to understand whether enough information could be found on the signal level to characterize the bi-metric behaviour of bambucos. Figure 6 shows a segment of the onset detection function obtained with Superflux on track rh_0002. The ground-truth beat annotations in 6/8 and in 3/4 are also displayed for reference. It should be noted that the annotations in 3/4 and 6/8 were extracted independently by different annotators, and hence the downbeats (which in theory should coincide) do not exactly overlap in all cases. Strong peaks in the onset detection function can be observed in most beat positions from the ground-truth annotations (solid orange line (3/4) and solid green line (6/8) lines). This suggests that regardless of the rhythmic complexity, there is information that can be exploited to characterize the metric behavior of bambucos. For reference, the beat estimations obtained by Madmom

and MultiBT (dashed lines) are also shown in the figure. The complexity of the task is further confirmed by the fact that, not surprisingly, the estimations obtained by the Madmom and MuiltiBT also tend to overlap with peaks in the onset detection function.

4. CONCLUSIONS

This work presented an analysis of beat and meter in the Colombian bambuco, a rhythm characterized by the presence of musical elements from two different meters. Our perceptual study confirmed that even for human listeners, there is not an unique understanding of the rhythmic structures of the genre. Even though current conventions assume a 6/8 meter when writing bambucos, our perceptual study confirmed that reality is much more complex than that. A total of five metric alternatives were found in the annotations produced by the participants in the study. Not surprisingly, results from the computational analysis confirmed that beat tracking models developed to deal with the regularity of a unique meter, do not fully characterize the complex rhythmic interactions in bambucos. However, our analysis of onset detection functions suggests that there is relevant information in these signal representations that could be leveraged for musicological analysis of bambucos. It is clear from the findings in this study that the development of tools for rhythm analysis of bambucos –or of any other music tradition that shares similar rhythmic properties– cannot be approached from a binary decision (right/wrong) perspective. This calls for rhythm analysis tools with an exploratory nature, where the existence of several truths is permitted, and the choice of the most relevant one is both task- and context-dependent. Our hope is that this study as well as the data and annotations collected in it, will serve as a preliminary step in the development of computational tools for musicological analysis of bambucos and Andean music.

5. REFERENCES

[1] R. Brandel, "The African Hemiola Style," *Ethnomusicology*, vol. 3, no. 3, pp. 106–117, 2006.

[2] D. Locke, "Principles of offbeat timing and cross-rhythm in southern eve dance drumming," *Ethnomusicology*, vol. 26, no. 2, pp. 217–246, 1982.

[3] C. Santamaría Delgado, *Vitrolas, rocolas y radioteatros: hábitos de escucha de la música popular en Medellín.* Editorial Pontificia Universidad Javeriana y Banco de la Republica, 2014.

[4] P. Wade, *Gente negra, nación mestiza: dinámicas de las identidades raciales en Colombia*, ser. Antropología social. Ediciones Uniandes, 1997.

[5] A. Tovar and J. Urrea, *Rítmica y melódica del folclor chocoano.* Universidad Nacional de Colombia, 1961.

[6] H. Stobart and I. Cross, "The Andean anacrusis? rhythmic structure and perception in Easter songs of Northern Potosí, Bolivia," *British Journal of Ethnomusicology*, vol. 9, no. 2, pp. 63–92, 2000.

[7] L. F. Ramón y Rivera, "Fenomenología de la etnomúsica del área latinoamericana," *Inidef 3 – Conac*, pp. 30–31, 1980.

[8] E. W. Large and J. F. Kolen, "Resonance and the perception of musical meter," *Connection science*, vol. 6, no. 2-3, pp. 177–208, 1994.

[9] S. Böck, M. E. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, Delft, The Netherlands, 2019, pp. 486–493.

[10] A. Holzapfel, F. Krebs, and A. Srinivasamurthy, "Tracking the "odd": meter inference in a culturally diverse music corpus," in *15th International Society for Music Information Retrieval (ISMIR) Conference*, Taipei, Taiwan, 2014, pp. 425–430.

[11] A. Srinivasamurthy, A. Holzapfel, and X. Serra, "Informed automatic meter analysis of music recordings," in *18th International Society for Music Information Retrieval (ISMIR) Conference*, Suzhou, China, 2017, pp. 679–685.

[12] L. Naveda, F. Gouyon, C. Guedes, and M. Leman, "Microtiming patterns and interactions with musical properties in samba music," *Journal of New Music Research*, vol. 40, no. 3, pp. 225–238, 2011.

[13] L. Lure and M. Rocamora, "Microtiming in the rhythmic structure of candombe drumming patterns," in *Proceedings of the Fourth International Conference on Analytical Approaches to World Music AAWM*, 2016, pp. 1–5.

[14] M. Wright, W. Schloss, and G. Tzanetakis, "Analyzing afro-cuban rhythms using rotation-aware clave template matching with dynamic programming." in *9th International Society for Music Information Retrieval (ISMIR) Conference*, Philadelphia, USA, 2008, pp. 647–652.

[15] L. Nunes, M. Rocamora, L. Jure, and L. Biscainho, "Beat and downbeat tracking based on rhythmic patterns applied to the uruguayan candombe drumming," in *16th International Society for Music Information Retrieval (ISMIR) Conference*, Málaga, Spain, 2015, pp. 264–270.

[16] T. Völkel, J. Abeßer, C. Dittmar, and H. Großmann, "Automatic genre classification of latin american music using characteristic rhythmic patterns," in *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound*, ser. AM '10. New York, NY, USA: Association for Computing Machinery, 2010.

[17] F. Mora-Ángel, G. A. L. Gil, E. Cano, and S. Grollmisch, "ACMUS-MIR: An annotated data set of Andean Colombian music," in *In 7th International Conference on Digital Libraries for Musicology*, Delft, The Netherlands, 2019.

[18] S. Böck, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles." in *15th International Society for Music Information Retrieval (ISMIR) Conference*, 2014, pp. 603–608.

[19] J. R. Zapata, M. E. P. Davies, and E. Gómez, "Multi-feature beat tracking," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 816–825, 2014.

[20] M. E. Davies, N. Degara, and M. D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," *Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06*, 2009.

[21] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 342–355, 2006.

[22] S. Hainsworth, "Techniques for the automated analysis of musical audio," Ph.D. dissertation, Dept. Eng., Cambridge University, 2004.

AUTOMATIC RANK-ORDERING OF SINGING VOCALS WITH TWIN-NEURAL NETWORK

Chitralekha Gupta[1]   **Lin Huang**[1]   **Haizhou Li**[1,2]

[1] Department of Electrical and Computer Engineering, National University of Singapore, Singapore
[2] Machine Listening Lab, University of Bremen, Germany
`chitralekha@nus.edu.sg, lin.huang@u.nus.edu, haizhou.li@nus.edu.sg`

ABSTRACT

When making judgements, humans are known to be better at choosing a preferred option amongst a small number of options, rather than giving an absolute ranking of all the options. This preference-based judgment rank-ordering method is called Best-Worst Scaling (BWS). Inspired by this concept, we propose a preference-based framework to generate a relative rank-ordering of singing vocals, and therefore, singers. We adopt a twin-neural network (Siamese) that learns to choose a preferred candidate in terms of singing quality between two inputs. With a few such pairwise comparisons, this method generates a relative rank-order of a complete list of singers. Additionally, we incorporate a knowledge-based musically-relevant pitch histogram representation, as a conditioning vector, to provide explicit musical information to the network. The experiments show that this method is able to reliably evaluate singing quality and rank-order singing vocals, independent of the song or the singer. The results suggest that the twin-neural network learns the underlying discerning properties relevant to singing quality, instead of being specific to the content of a song or singer.

1. INTRODUCTION

Singing is a popular form of entertainment and a desirable skill to develop [1]. In recent times, many online applications that provide a platform to showcase singing talent as well as socially engage through music have become popular, such as Smule Sing! [1] , Starmaker [2] , Quanmin K Ge [3] , and SoundCloud [4] . With high volumes of singing performances on such online platforms, there is a need to explore automated methods of assessing the quality of singing for the purpose of identifying singing talent as well as providing meaningful feedback to amateur and aspiring singers.

---

[1] https://www.smule.com/
[2] https://www.starmakerstudios.com/
[3] https://kg.qq.com/
[4] https://soundcloud.com/

For example, such an automated evaluation method would be useful for screening of the singers for popular singing talent reality shows such as American Idol and The Voice. In this work, we provide a data-driven and preference-based framework for evaluating singing quality.

Previous work on automatic singing quality evaluation has focused on comparing a test singing rendition against the known musical notes of the song [2, 3] or against an ideal singing rendition of the song by a professional singer [4–6]. These methods extract audio features such as pitch contour and mel-frequency cepstral coefficients that are relevant to perceptual parameters used by music experts to evaluate singing quality such as intonation accuracy, rhythm consistency, and timbre brightness [7,8]. However, such methods are constrained by the need for a reference or ideal singing rendition for every song. Moreover, the choice of an "ideal" reference singer introduces a bias of subjective choice.

Another approach is the assessment of singing quality without a reference singer. Studies have shown that music experts can evaluate singers with a high level of consensus even when the song is unknown to them [9, 10], which implies that there are underlying inherent characteristics of singing quality that differentiate between preferred and amateur singing. Previously, Nakano et al. [10] designed features such as pitch interval accuracy that measure the offset of the pitch contour from the musical semitone grid to evaluate singing quality without a reference rendition. Gupta et al. [11, 12] designed hand-crafted features that characterize the shape of the pitch histogram and inter-singer distances to evaluate singing quality without a reference. Such methods provide insight and explanation to the objective evaluation, such as the measurement of the sharpness of peaks in a pitch histogram correlating with the consistency of hitting musical notes. However, such hand-crafted features provide an approximate representation of singing quality, that depend on manual thresholds that are determined empirically. They do not capture all aspects of singing and therefore are limited.

Previously in [11], the authors showed that since a song can be sung correctly in one or a few similar ways, but incorrectly in many different and dissimilar ways, it implies that the quality of a singer is proportional to his/her similarity with other singers. However, to obtain a relative rank-order based on this idea, they needed to calculate the

distance of every singer in the dataset with respect to every other singer, which becomes computationally demanding as the size of the dataset increases. Moreover, this distance calculation made sense only if the singers were singing the same song, making the algorithm song-dependent.
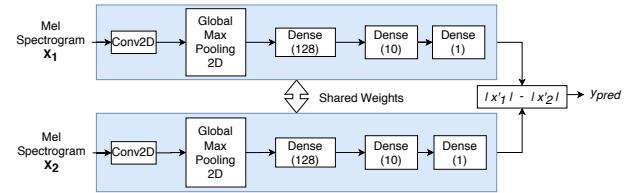
Humans are known to be better at relative judgments, i.e. choosing the most preferred singer among a small set of singers, than giving an absolute rating [13, 14]. This is the basis of the best-worst scaling (BWS) method used for consumer value preference surveys [15]. Motivated by this human behavior, we would like to develop a singing evaluation framework that is song-independent. The task is to rank-order a list of singing vocals without the need of any singing reference. We achieve a rank-ordering of a long list of candidates through a number of pairwise decisions.

2. TWIN-NETWORK FOR RELATIVE SINGING QUALITY EVALUATION

Twin-neural networks (or Siamese networks) have been previously used to measure similarity between two audio inputs, for example for vocal imitation [16–18], singing style identification [19, 20], and singing query retrieval [21]. The idea behind using twin-neural network for the task of singer style identification is to map different singing and song renditions of the same singer closer to each other than those of different singers. However, to the best of our knowledge, twin-neural network has not be explored for the task of singing quality assessment.

In this work, we modify a twin-neural network such that it learns which of the two given singing inputs is *more preferable* in terms of singing quality. We then obtain the rank-ordering of singing vocals by counting the number of times a singing input is preferred in many such pairwise comparisons across different singers, based on the concept of BWS. A similar approach has been discussed by Niu et al. [22] where a twin-neural network is applied for the task of image quality assessment. The network learns to rank the quality scores between the two input image patches, where it applies cross entropy as the loss function. Our work differs from [22] in that we propose a novel and intuitive preference metric and comparative loss function for training a siamese neural network to predict ranking.

Additionally, we include explicit musical knowledge in this framework, by using the pitch histogram as a conditioning vector. The two arms of the twin-network share the same architecture as well as parameters, i.e., the two inputs pass through exactly the same networks for feature learning. Singers share a similar underlying singing vocal production mechanism, however they differ in quality due to prosodic characteristics such as the ability to consistently hit the right notes. We hypothesize that the two arms of the network should be able to project each singing vocal to a compressed latent space that only represents the discriminatory singing quality properties independent of the song or the singer, thus making it suitable for the task of singing quality comparison of two singing vocals. Furthermore, BWS rank-ordering method is known to provide a reliable rank-ordering with fewer number of comparisons,

Figure 1. Twin-neural network modified for preference-based singing quality judgment.

which is helpful when the dataset size increases.

2.1 Model

A twin neural network consists of two identical sub-networks that have the same configuration with the same parameters and shared weights. During training, the parameter update is mirrored across both the sub-networks. The two sub-networks extract features from the two inputs, and then similarity between the two feature vectors is computed by a distance metric. In general, in a twin neural network, for a pair of inputs (x_1, x_2), the distance metric of the output of the two sub-networks $f(x_1)$ and $f(x_2)$ is given by Euclidean distance

$$D = ||f(x_1) - f(x_2)||_2 \qquad (1)$$

The contrastive loss function, that needs to be minimized, is defined as

$$L = y_t \cdot max(1 - D, 0)^2 + (1 - y_t) \cdot D^2 \qquad (2)$$

where y_t is the ground truth label, such that $y_t = 1$ whenever x_1 and x_2 are from the same class and $y_t = 0$ otherwise. This framework has been successfully used for similarity detection tasks such as sound search and vocal imitation detection [16, 18].

We modify this framework such that it learns to choose the better singer amongst the two input singers, as shown in Figure 1. To do this, we propose to replace (1) the distance metric with a *preference metric*, and (2) the contrastive loss with a *comparative loss*.

2.2 Preference Metric

We define the preference metric as the difference between the L1 norm of the feature vectors,

$$D_p = |f(x_1)| - |f(x_2)| \qquad (3)$$
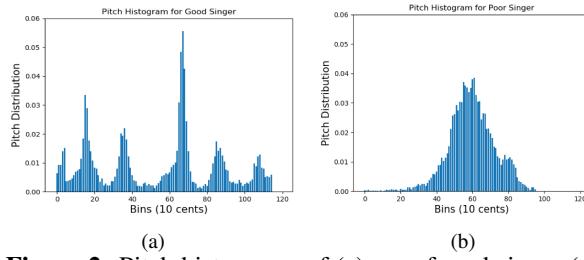
where $|f(.)|$ is the L1 norm of the feature vector. This provides a direction to the comparison, i.e. if $D_p \geq 0$ implies singer 1 input rendition x_1 is better than or similar to singer 2 input rendition x_2, and $D_p < 0$ implies singer 2 is better than singer 1. In contrast, a distance metric can only provide the magnitude, but not the direction of the difference.

2.3 Comparative Loss

Given the preference metric D_p, we compute the comparative loss function to be minimized, as

$$L_c = y_t \cdot max(1 - D_p, 0)^2 + (1 - y_t) \cdot (D_p + 1)^2 \quad (4)$$

where, y_t is the ground truth label, such that $y_t = 1$ whenever x_1 is better than or similar to x_2, and $y_t = 0$ otherwise. Note that the modification in comparative loss compared to contrastive loss is to accommodate for the directional or signed property of the preference metric D_p. Let's

(a)   (b)

Figure 2. Pitch histograms of (a) a preferred singer (rank 1) and (b) an amateur singer (rank 99) from the song *Let it go* of dataset 1 (Section 4.1.1). (1 bin = 10 cents).

examine this equation closely. If x_1 is better than x_2, then $y_t = 1$, so equation 4 will become

$$L_c = max(1 - D_p, 0)^2 \qquad (5)$$

Minimizing this loss function, makes the preference metric D_p close to 1. On the other hand, if x_2 is better than x_1, then $y_t = 0$, so equation 4 will become

$$L_c = (D_p + 1)^2 \qquad (6)$$
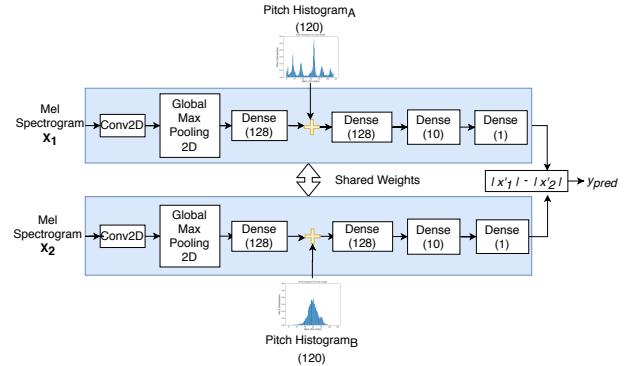
For this loss function to be zero, the preference metric D_p should be optimized to -1, thus preserving the signed property of the preference metric D_p.

3. HYBRID TWIN-NEURAL NETWORK WITH PITCH HISTOGRAM CONDITIONING VECTOR

We use mel-spectrogram as the input time-frequency representation of the input audio waveforms. However, measuring pitch correctness is a vital component of singing quality evaluation. Therefore, we condition the twin-network with pitch information in the form of pitch histogram. This unburdens the network from learning pitch-related information from the input representation.

The pitch histogram represents the distribution of pitch values in a sung rendition [23]. As demonstrated by [11], a pitch histogram is a strong indicator of the quality of singing. A pitch histogram is computed as the count of the pitch values (calculated in the unit of cents) folded on to the 12 semitones in an octave, where one semitone represents 100 cents on equi-tempered octave. The melody of a song typically consists of a set of dominant musical notes (or pitch values). These are the notes that are hit frequently in the song and sometimes are sustained for long duration. In the pitch histogram of a preferred singing rendition, there are several narrow, sharp, and well-defined spikes that indicate that the dominant notes are hit repeatedly and consistently (Figure 2(a)). On the other hand, an amateur singing rendition has a dispersed distribution of pitch values, that reflect that the singer is unable to hit the dominant notes of the song consistently (Figure 2(b)).

Due to its strong relevance to singing quality, we condition the twin-neural network by concatenating the pitch histogram vectors of the two inputs, ph_A and ph_B to the output vector of their respective sub-network intermediate layer, as shown in Figure 3. Such a configuration, called the hybrid twin-neural network, improves the performance

Figure 3. Hybrid Twin-neural network conditioned on pitch histogram.

of the network for singing quality evaluation, by providing explicit pitch-related information to the network releasing degrees of freedom to the network to learn other non-pitch related properties.

4. EXPERIMENTAL SETUP

We conduct experiments to evaluate the performance of the twin-neural network and the hybrid twin-neural network for the task of automatic rank-ordering of singing vocals. We analyse the performance of the framework when the input pair of singing vocals belong to the same song, as well as when they belong to different songs. We also compare the performance and capabilities of this framework with previous similar work in literature.

4.1 Dataset

4.1.1 Singing voice dataset 1

We use the subset of DAMP dataset[5] that was curated by [11] for the purpose of singing quality evaluation. It consists of solo-singing recordings (16 kHz sampling rate, mono) of 4 popular Western songs each sung by 100 unique singers (50 male, 50 female). There were no common singers across different songs. The selection of songs was based on the available number of unique singers in the DAMP dataset, and equal distribution between males and females, to avoid gender bias. The 4 popular songs are *Let it go (Idina Menzel), Cups (Anna Kendrick), When I Your Man (Bruno Mars), Stay (Rihanna)*. All the songs are rich in steady notes and rhythm, as summarized in Table III of [11].

We use one 20-30 seconds long snippet from each singing rendition. This snippet is a common section of the song for all the singers singing that song. The ground-truth subjective ranking provided with this dataset was a BWS score obtained through a crowd-sourcing platform by asking listeners to choose the best and the worst amongst a few singers singing the same song. This score resulted in a rank-order of the singers of each song from 1 to 100, where rank 1 means the best singer, and rank 100 means the worst singer. We divide this dataset into a train set that has 80%, i.e. 80 singers per song, and validation and test sets, each

---

Dataset	Division	#songs	#singers per song	#singer pairs per same song	#singer pairs singing same+different songs
1	Train	4	80	4 x 80 x 79 = 25,280	80x4=320 singers 320x319=102,080 pairs
	Validation	4	10	4 x10 x 9 / 2 = 180	10x4=40 singers 40 x 39 / 2 = 780 pairs
	Test	4	10	4 x 10 x 9 / 2 = 180	10x4=40 singers 40 x 39 / 2 = 780 pairs
2	Test	2	10	2 x 10 x 9 / 2 = 90	10x2=20 singers 20 x 19 / 2 = 190 pairs

Table 1. Summary of the number of singer pairs from the different datasets used in this work.

consisting of 10%, i.e. 10 singers per song. To ensure similar distribution of singing quality in all of these subsets, we pick the singers with ranks [1,11,21,...91] for the test set, [2,12,22,...92] for the validation set, and the rest for the train set. Note that the test set consists of singers that are not present in training or validation sets. However, the songs in all three sets are the same.

4.1.2 Singing voice dataset 2

To test the trained models on unseen songs, we use a small test dataset provided by [4] that consists of solo-singing recordings (16 kHz sampling rate, mono) of 2 Western pop songs (*I have a dream (ABBA), Edelweiss (Sound of Music)*) each sung by 10 singers. Since this dataset was recorded in a lab-controlled environment, the entire spectrum of singing ability - from amateur singers to professionally trained excellent singers - was covered. The ground-truth singing quality annotations provided in this dataset are absolute ratings on a scale of 1-5, 5 being the best, provided by professional music teachers and/or performers in a lab-controlled environment. Additionally, the music experts evaluate and score the quality of pitch and rhythm separately.

4.1.3 Singer pair inputs

The number of singer pairs singing the same song and different songs in the two datasets is summarized in Table 1. There exist a total of 25,280 ordered pairs of singers singing the same song in the training set, and 102,080 ordered pairs of singers singing different as well as same song. We treat ordered pairs, i.e. singer pairs (A,B) and (B,A) as different training samples for the purpose of data augmentation. Also this is helpful because the preference metric is asymmetric. The validation set consists of 180 unordered pairs of singers singing the same, 780 unordered pairs of singers singing the same and different songs. We have two test sets, one each from singing vocals datasets 1 and 2. The number of pairs in test dataset 1 is same as the validation set. The test dataset 2 consists of 90 unordered pairs of singers singing the same song, and 190 unordered pairs of singers singing the same and different songs.

4.1.4 Ground-truth Labels

The ground-truth label for each pairwise comparison of singers is derived from the human scores provided in the singing datasets 1 and 2. In dataset 1, the BWS score b provided for every singer that ranges between -1 and 1 is first normalized between 0 and 1. We then label a pair of singers A and B as

$$y_{true} = \begin{cases} 1, & \text{if } b_A \geq b_B \\ 0, & \text{otherwise} \end{cases}$$

which implies that if singer A is better or similar to singer B, the label is 1, and it is 0 otherwise. Similarly, in dataset 2, the absolute human ratings between 1 and 5 is normalized between 0 and 1, and the same method is applied to give a binary label every pair of singers.

4.2 Setup

The overall structure of the twin network is shown in Figure 1, and the hybrid twin network is a modified version of the twin network, as shown in Figure 3.

4.2.1 Pre-processing

Both the input audio waveforms are converted to the 2-D mel-spectrogram representations with 96 mel-bins over the frequency range of 0–8 kHz, a window length of 512 and a hop-size of 256. The input waveforms are singing vocal snippets of approximately 20 seconds in duration. The number of frames of the two inputs are made equal by appending zeros to the shorter spectrogram.

4.2.2 Twin-network

Each arm of the twin network consists of a 2D Convolutional Neural Network (CNN) with 1 convolutional layer having 64 filters with a kernel size of 3x3 and stride size of 1x1, followed by a sigmoid activation function. They are then each followed by a 2D global max-pooling layer, and three fully-connected dense layers. There are 128 neurons in the first dense layer, 10 in the second layer, and 1 in the third. The sigmoid activation function is used in all of these layers, as it squashes the output of the layers between 0 and 1. Empirically, we observe that applying the sigmoid activation at all the layers results in convergence while training.

The preference metric is computed using the outputs of these two arms, as discussed in Section 2.2. This value is viewed as the preference judgment value between the input singer pair, i.e. which of the two singers is better.

4.2.3 Hybrid twin-network

In order to incorporate musical relevance into the network, we concatenate the normalized 120 dimensional pitch histogram vectors of the two inputs, at the output of the first dense layer in both the arms of the twin network. We chose to inject the pitch histogram information here because of the comparable number of dimensions of the latent space and the histogram. Empirically, an additional fully-connected layer was needed to gradually project the dimensions of the output to 1, and for training to converge.

4.2.4 Training

Training the network requires positive (singer A better than or similar to singer B) and negative pairs (singer B better than A) of singer inputs. In our training, the ground truth label is 1 for positive pairs and 0 for negative pairs.

The loss function we minimize is the comparative loss which is a function of the probability output of the network and the binary ground-truth label, as given in equation 4. We use the Adam optimization algorithm [24]. The learning rate is 0.0001. The batch size is 10. Maximum number of epochs is set to 250, though early stopping based on training loss with patience of 5 epochs is employed for training termination. Back-propagation is car-

ried out through the twin-net arms. We choose the model that shows minimum loss in the validation set.

4.2.5 Prediction

The preference judgment value, i.e. the preference metric D from equation 3 of the twin network lies between -1 and 1. If this value is >=0, it implies singer A is preferred over singer B, thus the verdict is 1, and vice versa.

After all the pairwise comparisons, the singers can be rank-ordered according to the aggregate scores of each singer, given by the BWS score defined as

$$B = \frac{n_{best} - n_{worst}}{n} \qquad (7)$$

where n_{best} and n_{worst} are the number of times the singer is marked as preferred and not preferred respectively, and n is the total number of times the singer appears.

4.3 Evaluation Metrics

We use three kinds of metrics to evaluate the performance of the framework with respect to the human ground-truth labels as described in section 4.1:

Pair prediction accuracy: This is defined as the percentage of input singer pairs for which the preference prediction from the network is correct.

Pearson's Score Correlation: This is the correlation between the machine BWS scores and human BWS scores.

Spearman's Rank Correlation: This is the correlation between the machine and human annotated singer rank-orders based on the respective BWS scores.

5. EXPERIMENTS AND RESULTS

The inter-judge correlation between ratings from music experts is 0.82 [4], which means that experts do not always agree with each other, and there is, in general, an upper limit of the achievable performance of any machine-based singing quality evaluation.

5.1 Twin-Net vs. Hybrid Twin-Net

We test our hypothesis that twin neural network can be applied for the task of learning singing quality preference in pairwise comparisons of singers to predict rank-ordering of singers. We train the twin-network and the hybrid-twin network on the 25,280 *same song* singer pairs from the training set of dataset 1. Since the two singers sing the same sequence of words, the twin arms focus on learning the discriminatory characteristics from the input representations which lie in the differences in the prosodic properties such as pitch harmonics of the two singing renditions. The hybrid network further helps in this process as the pitch histogram provides a direct singing quality discriminatory representation, as discussed in section 3.

From Table 2, we see that the both the twin-networks are able to converge on the training dataset with a high pair prediction accuracy and score correlation with humans. This validates our hypothesis and technique of the adaptation of a Siamese network for preference-based judgment and hence rank-ordering of singers. We also observe that the hybrid-twin network outperforms the twin-network on the test set from dataset 1. This implies that conditioning

Dataset	%Accuracy		Pearson Corr.	
	Twin	**Hybrid**	**Twin**	**Hybrid**
Train	88.3	81.3	0.91	0.82
Validation	73.8	73.3	0.63	0.62
Test Dataset 1	72.7	76.1	0.61	0.68

Table 2. Performance of twin-network and hybrid twin-network in terms of pair classification accuracy and Pearson correlation between machine BWS scores and human BWS scores. All correlation values are statistically significant with $pvalue \ll 0.05$.

the network on pitch histogram frees degrees of freedom to model non-pitch related information via the network.

5.2 Comparison with Prior Studies

The prior studies that are closest to this work are the ones by Gupta et al. [11] and Pati et al. [25]. In the former, the authors studied various hand-crafted features to generate rank-ordering of singers, such as pitch histogram-based absolute measures and inter-singer distance based relative measures. They also performed late-fusion of these ranks to get a good correlation with human annotations. Pati et al. trained a supervised regression DNN model that uses mel spectrograms of pitched wind instruments as input representation to predict their subjective human scores.

In this experiment, we compare the performance of our proposed hybrid twin-network against the relative measures performance of [11]. Both these techniques involve same-song pair comparisons, and hence are conceptually similar. Additionally, we train the absolute score prediction network of [25] on our dataset. The ground-truth, in this case, are the raw human BWS scores of every singer that was provided with this dataset. This prediction network is similar to the absolute measures prediction from [11] in the sense that both involved direct assessment of singers. Finally, in late-fusion, we compute the average of the rank-order obtained from our hybrid twin network and the absolute score prediction network, similar to [11].

In Table 3, we see that the proposed hybrid-twin network performs better than the relative measures of [11]. Moreover, hybrid-twin outperforms the absolute score prediction network. This implies that pairwise comparisons in combination with pitch histogram representation results in better modeling of singing quality, than hand-crafted features. The late-fusion performances are comparable.

The inter-singer distances of relative measures in [11] compare the features from one singer with that of the rest of the singers in the dataset singing the same song. Thus, the major drawback of this method is that the relative measures will make sense only if all the singers are singing the same song. Moreover, for a new unseen song, there needs to be a large number singers singing that song for the thresholds designed for relative measures to be reliable. The above drawbacks make the relative measures highly song dependent. Moreover, any new test singer needs to be compared to all the singers in the dataset to get a reliable ranking. This becomes computationally cumbersome with increasing size of dataset. In the next sections, we show how our proposed framework overcomes these drawbacks.

Gupta et al. [11]		This work	
Framework	Corr	Framework	Corr
Relative Measures	0.64	Hybrid Twin-network	0.68
Absolute Measures	0.48	Absolute score prediction network [25]	0.62
Late-Fusion	0.71	Late-Fusion	0.71

Table 3. Comparison of the Spearman's rank correlation performance of the proposed hybrid twin network on dataset 1 with that from a recent previous work on the same dataset. All correlation values are statistically significant with $pvalue \ll 0.05$.

Framework	%Accuracy	Pearson's Score Corr	Spearman's Rank Corr
Twin-net	65.9	0.39	0.41
Hybrid twin-net	77.7	0.63	0.65

Table 4. The performance of twin-net and hybrid twin-net models on unseen songs from test dataset 2. The models are trained on the same song input training pairs from dataset 1. All correlation values are statistically significant with $pvalue \ll 0.05$.

5.3 Performance on Unseen Songs

To test the performance of the trained model on unseen songs, we evaluate its performance on test dataset 2 (Table 1). These songs and singers were not present in training set. The dataset consists of 90 same song singer pairs. From Table 4, we observe that the hybrid twin net outperforms the twin-net by a significant margin. This shows that the pitch histograms are a powerful representation of singing quality that reduces the dependency of the network on the identity of the song, thus confirming that our proposed framework can reliably evaluate unseen songs.

5.4 Comparing Different Songs
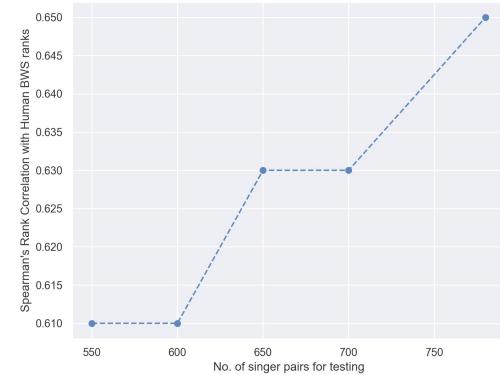
We further test if our proposed framework can compare singing vocals of different singing content. For this, we train the hybrid-twin net singer pairs singing same as well as different songs, for which we use the 102,080 ordered singer pairs of the training dataset (Table 1). In Table 5, we observe the performance of this model on the different song singer pairs from both test dataset 1, where the songs are seen by the trained model, and test dataset 2, where the songs are not seen by the trained model. Rank-ordering singing vocals with different-song singer-pair inputs (Table 5, row 1 and row 3) shows comparable results to same-song singer-pair comparisons (Table 3 row 1 and Table 4 row 2). Moreover, when rank-ordering is done using both different-song and same-song pair comparisons, the results on unseen songs (Table 5, row 4) significantly outperforms that from the same-song pair trained model (Table 4 row 2). This experiment shows that our proposed preference-based framework is able to learn discerning properties of singing quality such that given any two singers singing the same or different songs, it learns to choose the better singer.

5.5 Effect of Number of Comparisons

BWS method is known to be able to reliably rank-order with fewer number of comparisons. We tested this idea by

Test Dataset	No. of diff. songs singer pairs	No. of same songs singer pairs	%Accuracy	Pearson's Score Corr	Spearman's Rank Corr
1	600	0	72.3	0.64	0.64
	600	180	72.7	0.65	0.65
2	100	0	77	0.68	0.68
	100	90	78.6	0.70	0.73

Table 5. Performance of hybrid twin network trained on the same and different song input pairs. All correlation values are statistically significant with $pvalue \ll 0.05$.

Figure 4. Spearman's rank correlation as the number of pairwise comparisons is reduced.

reducing the number of paired comparisons in the test set, while ensuring that each singer appears at least once. Out of the 780 pairs, we randomly selected x number of unique pairs three times, and calculated the average of the performance of the three random trials. These average values of Spearman's rank correlation are plotted in Figure 4, where the number of pairs selected ranged from 550 to all of the 780 pairs. We observe that for a reduction of 30% in the number of pairs for comparison, there is a very small drop in the correlation value, approximately 6%. This computational advantage will become more significant when the size of the dataset increases.

6. CONCLUSIONS

In this work, we propose a preference-based framework in which we adapt the twin neural network (Siamese) such that given two input singers, it learns to choose the better singer. We incorporate structural changes in the Siamese network framework such as preference metric instead of distance metric and comparative loss instead of contrastive loss, so that it is able to learn a preference instead of similarity. We show that with a few pairwise comparisons, this modified Siamese network effectively gives a reliable rank-order of singers. We also incorporate the musically relevant pitch histogram representation in a hybrid twin network framework, which shows to provide reliable singing quality predictions in a singer and song independent way on unseen data.

7. ACKNOWLEDGMENT

8. REFERENCES

[1] C. Gupta, "Comprehensive evaluation of singing quality," *PhD Thesis, National University of Singapore*, 2019.

[2] W.-H. Tsai and H.-C. Lee, "Automatic evaluation of karaoke singing based on pitch, volume, and rhythm features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1233–1243, 2012.

[3] E. Molina, I. Barbancho, E. Gómez, A. M. Barbancho, and L. J. Tardón, "Fundamental frequency alignment vs. note-based melodic similarity for singing voice assessment," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 744–748.

[4] C. Gupta, H. Li, and Y. Wang, "Perceptual evaluation of singing quality," in *APSIPA Annual Summit and Conference*, vol. 2017, 2017, pp. 12–15.

[5] C.-H. Lin, Y.-S. Lee, M.-Y. Chen, and J.-C. Wang, "Automatic singing evaluating system based on acoustic features and rhythm," in *Orange Technologies (ICOT), 2014 IEEE International Conference on*. IEEE, 2014, pp. 165–168.

[6] C. Gupta, H. Li, and Y. Wang, "A technical framework for automatic perceptual evaluation of singing quality," *APSIPA Transactions on Signal and Information Processing*, vol. 7, 2018.

[7] C. Cao, M. Li, J. Liu, and Y. Yan, "A study on singing performance evaluation criteria for untrained singers," in *Signal Processing, 2008. ICSP 2008. 9th International Conference on*. IEEE, 2008, pp. 1475–1478.

[8] J. M. Oates, B. Bain, P. Davis, J. Chapman, and D. Kenny, "Development of an auditory-perceptual rating instrument for the operatic singing voice," *Journal of Voice*, vol. 20, no. 1, pp. 71–81, 2006.

[9] T. Nakano, M. Goto, and Y. Hiraga, "Subjective evaluation of common singing skills using the rank ordering method," in *Ninth International Conference on Music Perception and Cognition*. Citeseer, 2006.

[10] ——, "An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features," in *Ninth International Conference on Spoken Language Processing*, 2006.

[11] C. Gupta, H. Li, and Y. Wang, "Automatic leaderboard: Evaluation of singing quality without a standard reference," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 13–26, 2020.

[12] ——, "Automatic evaluation of singing quality without a reference," in *Proceedings of APSIPA Annual Summit and Conference*, 2018.

[13] J. J. Louviere, T. N. Flynn, and A. A. J. Marley, *Best-worst scaling: Theory, methods and applications*. Cambridge University Press, 2015.

[14] A. Marley, T. N. Flynn, and V. Australia, "Best worst scaling: theory and practice," *International Encyclopedia of the Social & Behavioral Sciences*, vol. 2, no. 2, pp. 548–552, 2015.

[15] J. Louviere, I. Lings, T. Islam, S. Gudergan, and T. Flynn, "An introduction to the application of (case 1) best–worst scaling in marketing research," *International Journal of Research in Marketing*, vol. 30, no. 3, pp. 292–303, 2013.

[16] Y. Zhang, B. Pardo, and Z. Duan, "Siamese style convolutional neural networks for sound search by vocal imitation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 2, pp. 429–441, 2018.

[17] Y. Zhang and Z. Duan, "Visualization and interpretation of siamese style convolutional neural networks for sound search by vocal imitation," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2406–2410.

[18] A. Gresse, M. Quillot, R. Dufour, V. Labatut, and J.-F. Bonastre, "Similarity metric based on siamese neural networks for voice casting," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6585–6589.

[19] C.-i. Wang and G. Tzanetakis, "Singing style investigation by residual siamese convolutional neural networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 116–120.

[20] M. Panteli, R. Bittner, J. P. Bello, and S. Dixon, "Towards the characterization of singing styles in world music," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 636–640.

[21] K. Lee and J. Nam, "Learning a joint embedding space of monophonic and mixed music signals for singing voice," *arXiv preprint arXiv:1906.11139*, 2019.

[22] Y. Niu, D. Huang, Y. Shi, and X. Ke, "Siamese-network-based learning to rank for no-reference 2d and 3d image quality assessment," *IEEE Access*, vol. 7, pp. 101 583–101 595, 2019.

[23] G. Tzanetakis, A. Ermolinskyi, and P. Cook, "Pitch histograms in audio and symbolic music information retrieval," *Journal of New Music Research*, vol. 32, no. 2, pp. 143–152, 2003.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] K. A. Pati, S. Gururani, and A. Lerch, "Assessment of student music performances using deep neural networks," *Applied Sciences*, vol. 8, no. 4, p. 507, 2018.

NEURAL LOOP COMBINER: NEURAL NETWORK MODELS FOR ASSESSING THE COMPATIBILITY OF LOOPS

Bo-Yu Chen[1], Jordan B. L. Smith[2], and Yi-Hsuan Yang[1]

[1] Academia Sinica, Taiwan, [2] TikTok, London, UK

`bernie40916@citi.sinica.edu.tw, jordan.smith@tiktok.com, yang@citi.sinica.edu.tw`

ABSTRACT

Music producers who use loops may have access to thousands in loop libraries, but finding ones that are compatible is a time-consuming process; we hope to reduce this burden with automation. State-of-the-art systems for estimating compatibility, such as AutoMashUpper, are mostly rule-based and could be improved on with machine learning. To train a model, we need a large set of loops with ground truth compatibility values. No such dataset exists, so we extract loops from existing music to obtain positive examples of compatible loops, and propose and compare various strategies for choosing negative examples. For reproducibility, we curate data from the Free Music Archive. Using this data, we investigate two types of model architectures for estimating the compatibility of loops: one based on a Siamese network, and the other a pure convolutional neural network (CNN). We conducted a user study in which participants rated the quality of the combinations suggested by each model, and found the CNN to outperform the Siamese network. Both model-based approaches outperformed the rule-based one. We have opened source the code for building the models and the dataset.

1. INTRODUCTION

The emergence of digital audio editing techniques and software such as digital audio workstations (DAWs) has changed the way people make music. In a DAW, producers can easily produce music by making use of pre-existing audio as loops. Loops are used in many styles, especially Electronic and Hip-Hop music. Perhaps noticing the business value of such loop-based music, many companies such as Splice and LANDR have built up databases of loops. Community efforts have flourished too, including Looperman, an audio community that provides free loops for music producers to use. But having so many resources to choose from leaves a separate problem: how to navigate the library efficiently and how to choose which loops to combine. These tasks require human practise, expertise, and patience: to recognize the characteristics of the

loop and to determine whether it is suitable for their song. However, thanks to recent advances in music information retrieval (MIR), we believe it is possible to make loop selection from large-scale loops database easier.

A few existing systems could potentially solve this problem. Kitahara *et al.* [1] presented an intelligent loop sequencer that chooses loops to fit a user-defined 'excitement' curve, but, excitement only accounts for part of what makes two loops compatible. The AutoMashUpper system [2] could also be applied to the loop selection process: it involves estimating the 'mashability' of two songs (i.e., how compatible they would be if played at the same time). It is a rule-based system that computes the harmonic and rhythmic similarity and spectral balance, and it has proven useful in other applications [3,4]. However, AutoMashUpper has two limitations that could be improved by current machine learning methods. First, it regards the harmonic and rhythmic similarity as part of mashability, while it is actually possible that two music segments match perfectly despite having different harmonies and rhythms. Second, hand-crafted representations cannot fully describe all features in the music segment.

To capture the more complicated compatible relationship between two music segments, we propose to employ modern machine learning models to learn to predict the compatibility of loops. A major obstacle preventing the development of such a model is the lack of a dataset with sufficient labelled data. While there are many datasets of loops, none provide ground truth compatibility values.

We make two main contributions. First, we propose a data generation pipeline that supplies labelled data for model-based compatibility estimation (see Section 3). This is done by using an existing loop extraction algorithm [5] to yield positive examples of loops that have been used together in real loop-based music. We also propose procedures to ensure the quality of the positive data, and investigate different strategies to mine negative data from the result of loop separation.

Second, we develop and evaluate two neural network based models for loop compatibility estimation (see Section 4), one based on convolutional neural network (CNN) and the other Siamese neural network (SNN) [6]. The two approaches perform loop compatibility estimation using different approaches: the CNN directly evaluates the combination of loops in a time-frequency representation, whereas SNN processes the two loops to be evaluated for compatibility separately. We report both objective and sub-

jective evaluations (see Section 5) to study the performance of these approaches, and to compare the model-based approaches with AutoMashUpper.

The audio data we employ to build the neural network is from the Free Music Archive (FMA) [7] (see Section 3.1), which make data re-distribution easier. Moreover, we have open-sourced the code implementing the proposed CNN and SNN models at `https://github.com/mir-aidj/neural-loop-combiner`.

2. RELATED WORK

Along with the growing interest in loop-based music, academic studies focusing on assisting loop-based music production have become popular. An early study [8] proposed two ways to help create loop-based music: automatic loop extraction and assisted loop selection. This laid the foundation for this field.

For **loop extraction**, Shi and Mysore [9] proposed an interface with automatic and semi-automatic modes for the producer to find the most suitable segment in a piece of music to excerpt and use as a loop, by cropping directly. These algorithms estimate similarity with handcrafted features: harmony, timbre, and energy [8, 9]. However, the segments are excerpted without any attempt to isolate one part of a potentially multi-part piece of music. One solution to this used a common quality of loop-based music—that loops are often introduced one at a time—to recover the source tracks [10], but not all pieces have this form. To tackle both problems, Smith *et al.* [5, 11] proposed to extract loops by taking into account how they repeat. The algorithm they proposed can directly extract the loops from songs using nonnegative tensor factorization (NTF).

For **loop selection**, Kitahara *et al.* [1] proposed to select the loops according to the level of excitement entered by the user. However, we see three limitations in this work. First, the study focuses on Techno music only—while excitement is certainly highly relevant to this genre, the approach may not generalize well to other genres. Second, the level of excitement has to be manually entered by a user, which limits its usability. Third, the study does not take compatibility into consideration. As a result, even though a user can find loops with the desired excitement level, the loops may not be compatible with one another.

To the best of our knowledge, the work of Davies *et al.* [2, 12] represents the first study to investigate mashability estimation. Their AutoMashUpper system represents each music segment (not necessarily a loop) with a chromagram, a rhythmic representation, and a spectral band representation, each made to be beat-synchronous. Given two songs, it computes the similarity between the chromagrams and the rhythmic representations, and the spectral balance between the songs, to obtain the final mashability estimate. While AutoMashUpper is developed for general mashability estimation of longer music pieces, it can also be applied to loop selection.

Lee *et al.* [13] extended AutoMashUpper, proposing that two parts should be more compatible if they have complimentary amounts of harmonic instability. They gener-

Data type	# loops	# loop pairs	# songs
Training set	9,048	12,774	2,702
Validation set	2,355	3,195	7,06
Test set	200	100	100
$\sum$	11,603	16,069	3,508

Table 1: Statistics of the dataset, which is derived from Hip-Hop songs in the Free Music Archive (FMA) [7] using the data generation pipeline shown in Figure 1.

ated mashups that were preferred by listeners to the output of AutoMashUpper, but they focused on a particular type of mash-up (combinations of vocal and backing tracks) whereas we focus on general loop compatibility. Bernardes *et al.* [14,15] proposed several harmonic mixing approaches based on psychoacoustic principles and "tonal interval space indicators." Xing *et al.* [16] used paired audio, MIDI, and lyrics data for mashability estimation, taking into account similarity in melody, rhythm, and rhyming lyrics. Among these works, AutoMashUpper stands out as a general-purpose system requiring only audio data, so we consider it as our baseline.

3. DATA GENERATION PIPELINE
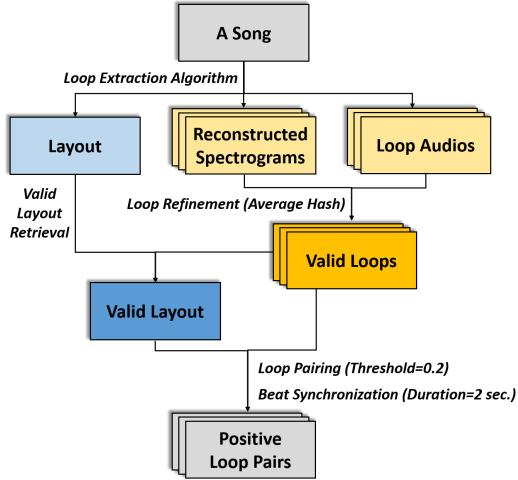
To create the labeled data needed for training our models, we obtained a dataset of loop-based music and used loop extraction [5] to obtain audio loops. We use a new loop refinement procedure to reduce the number of duplicate loops per song, and a loop pairing procedure to get pairs of loops that co-occur in songs.

3.1 Dataset

To extract a collection of loops, we first need a large set of songs that use loops. We chose to use music from the Free Music Archive (FMA) [7] so that we could redistribute the audio data. We restricted ourselves to the genre of Hip-Hop for three reasons: First, we could not manually verify whether each song used loops, so we needed to use a genre known for using loops. Second, for the loop extraction step to work, we needed the music to have a steady tempo. Third, we expected that a Hip-Hop subset would provide a useful variety of loops since the genre is known for incorporating loops from many other genres

We collected 6,868 Hip-Hop songs in total from FMA by searching for the tag "Hip-Hop." We passed these songs through the proposed data generation pipeline, and kept the 3,508 songs from which we could find at least one valid loop pair. Specifically, from these 3,508 songs, we obtained 11,603 valid loops and 16,069 valid loop pairs. From the full set of loops, we reserved 200 loops (1 pair each from 100 different songs) for the evaluations (see Section 5), and then split the rest into train and validation sets in a 4-to-1 ratio (see Table 1). [1]

---

[1] To facilitate follow-up research, we plan to share publicly the loops we extracted from FMA songs. While FMA has looser copyright re-

Figure 1: The proposed data generation pipeline for creating positive loop pairs.

Figure 2: Illustration of various loop-pair 'negative sampling' (NS) strategies explored in the paper.
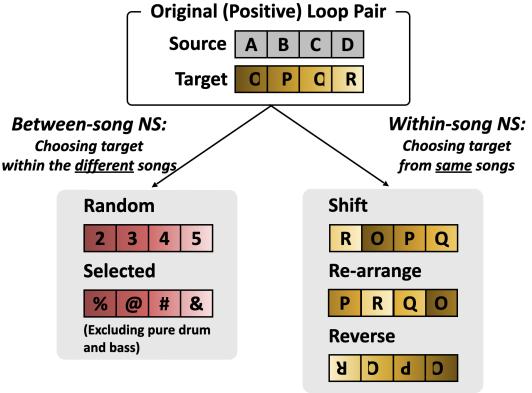
3.2 Data Generation

Figure 1 depicts the overall data generation pipeline. We firstly use the algorithm proposed by Smith and Goto [5] for loop extraction. The algorithm uses NTF to model a song as a set of sound templates and rhythm templates, a set of "recipes" for combining these templates to recreate the loop spectrograms, and a loop layout that defines when each loop occurs in the song. In later work, the authors described how to choose the best instance of a loop to extract from a piece, and a second factorization step that can be applied to reduce the redundancy of the loops [11]. As a result of this process, we get out of a song the loop layout, the reconstructed spectrograms, and the audio files of loops, as depicted in the first half of Fig. 1.

Despite this extra step, we still found many redundant, similar-sounding loops from the Hip-Hop dataset. To further deduplicate the loops, we introduce a new loop refinement step. The main idea is to consider the reconstructed spectrograms obtained from the loop extraction algorithms as an image, and apply the average hash algorithm [17] used in the identification or integrity verification of images to detect the duplicate loops. We first construct the hash from each spectrogram extracted from the same song, then count the number of bit positions that are different in every pair of spectrograms. If a pair of spectrograms has fewer than five bit positions that are different, we regard them as duplicates and remove one of them.

After this, we refine the loop layout by two steps. First, we combine all activation values from the duplicate loop in the loop layout into a single activation value. Second, we normalize all the activation values in each bar. This leads a valid loop layout that is ready for loop pairs creation.

Finally, we have to process the real-valued loop layout to obtain pairs of loops that do co-occur. A straightforward approach is to threshold the loop layout; we found

a threshold of 0.2 to work reasonably well. Please note that, to make all the loops in our dataset comparable, we time-stretch each to be 2 seconds long.

4. PROPOSED LEARNING METHODS

4.1 Negative Sampling Strategies

We get abundant positive data after the data generation pipeline. However, we also need negative examples (pairs of clips known to not match well) to train our mashability predictor. While there are straightforward ways to collect such examples, proper and domain-specific negative sampling has been shown to be important [18–22]. We experiment with the two types of methods of negative sampling. See Figure 2 for an overview of such methods.
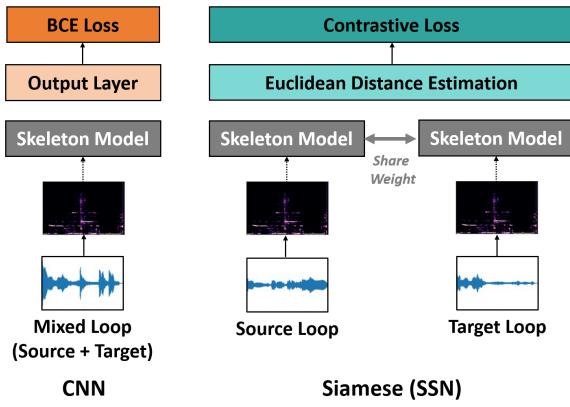
4.1.1 Between-Song Negative Sampling

A naive approach to negative sampling, dubbed the **random** method, is to take any two loops from two different songs, and call that a negative pair. But, it is hard to ensure the loops collected in this way clash with one another.

We therefore also study a slightly more sophisticated method that takes *instrumentation* into account, dubbed the **selected** method. Specifically, we noted that many loops are pure drum or pure bass loops, and they tend to be compatible with any loops. Therefore, we use [23] to detect pure drum or bass loops and avoid selecting them in the process of random negative sampling. This way may help emphasize the harmonic compatibility of the loops. With this strategy, we can experiment whether putting more emphasis on *harmonic compatibility* can indeed improve the result, or other feature's compatibility is still crucial.

4.1.2 Within-Song Negative Sampling

Negative data can also be created by editing a loop in a positive loop pair. We come up with three methods for making conflicting loop pairs: 'reverse,' 'shift,' and 'rearrange.' Given a positive loop pair, we view one of them as the *source loop*, and the other as the *target loop*. With the **reverse** method, the target loop is played backward to create the rhythmically and harmonically conflict. The **shift**

Figure 3: Architectures of the CNN and SNN models studied in this paper. The CNN takes a pair of loops (in the time domain) as the input, whereas the SNN takes the two loops as separate inputs and concatenate the intermediate abstractions only in the middle of its network.

method cycle-shifts a target loop by 1 to 3 beats at random to make the source loop and manipulated target loop unbalanced. The **rearrange** method cuts the target loop into beats and reorders them randomly. The combination of the source loop and the manipulated target loop is considered as a negative loop pair. See Figure 2 for illustrations.

4.2 Model and Training

Figure 3 shows the two models (CNN and SNN) proposed to learn the compatibility of two loops. To make a fair performance comparison, we train them with the same skeleton in a different way. Therefore, we first introduce the skeleton model and then explain how it is incorporated into the specific model architectures for the training process.

4.2.1 Skeleton model

A two-layer 2-D convolutional neural network (CNN) and 3-layer fully connected neural network is the skeleton of networks used in this paper. There are 16 filters (with kernel size 3) and 4 filters (also with kernel size 3) in the first and second convolutional layers. The 3-Layer fully-connected neural network is constructed by 256, 128, 16 output features, respectively. Furthermore, batch normalization [24], 10% dropout [25], and PReLU [26] are applied to all convolutional and fully-connected layers. All the models are trained using stochastic gradient descent with batch size 128. The input for the skeleton model is a 2-second loop audio. We compute the spectrograms by sampling the songs at 44k Hz and using a 2,048-point Hamming window and 512-point hop size. We then transform the spectrograms into 128-bin log mel-spectrograms. The resulting input has shape 173 frames by 128 bins.

4.2.2 CNN Model

A convolutional neural network is well-known for its ability as feature extraction. In our system, in order to learn the compatibility of two loops, we propose to use a CNN

as a classifier by combining two loops into a single input, and then train the CNN model to learn to distinguish whether loop combinations would sound harmonious (high compatibility) or incompatible (low compatibility). For the classification purpose, we stack the skeleton model with a fully-connected output layer to get a single value as the output. Then, we compute the binary cross entropy loss (BCELoss) to update the parameters of the whole model. We note that its output is a value between 0 and 1, with values closer to 1 indicating a higher probability that the pair of loops are compatible, and closer to 0 when they are not. Therefore, we can later use its output to estimate the compatibility of any two loops.

4.2.3 Siamese Model

A Siamese neural network (SNN) [6] consists of twin networks that share weights and configurations. SNN has been shown effective in image retrieval [27] and various MIR tasks alike [4, 28–32]. Aiming to test its applicability to loop compatiblity estimation, we train an SNN using the labeled data we created through the data generation pipeline as follows. The outcome of an SNN is a mapping function from the input features to the output vectors in an embedding space. During the training process, our SNN first transforms the input Mel-spectrograms into a vector by a skeleton model and then optimize in contrastive loss [33] directly. After training, we have a mapping function that can map any loop to the embedding space. To select the compatible loops, we compute the Euclidean distance between two loops in that embedding space. If the distance for two loops is close, then we assume they may be compatible, and vice versa.

5. EXPERIMENTS AND EVALUATION

5.1 Objective Evaluation

In the objective evaluation, we aim to evaluate the performance of different combinations of model architectures (i.e., CNN and SNN) and negative sampling methods. In doing so, we consider two types of objective metrics.

The first type of evaluation entails a *classification* task. It assesses a model's ability to distinguish compatible loops from incompatible ones. To create a test set for this evaluation, we used the positive data from the validation data and collected the negative data by using all the negative sampling methods equally, in order not to favor any of them in this evaluation. To make the test set balanced, we set the ratio of negative to positive data to 1:1.

The second type of evaluation, on the other hand, involves a *ranking* task. Given a query loop and a certain number of candidate loops, a model has to rank the candidate loops in descending order of compatibility with the query. We created the set of candidate loops for a query loop such that we knew one and only one of the candidate loops formed a positive loop pair with the query loop. We could then evaluate the performance of a model by checking the position of this "target loop" in the ranked list. The closer the rank is to 1, the better. This evaluation task

Model	Negative sampling	Classification-based metric		Ranking-based metric			
		Accuracy	F1 score	Avg. rank	Top 10	Top 30	Top 50
CNN	Random	0.60	0.59	43.0	0.13	0.35	0.59
	Selected	0.59	0.59	43.1	0.13	0.29	0.62
	Reverse	**0.63**	**0.62**	41.2	0.19	0.42	0.62
	Shift	0.57	0.56	49.0	0.11	0.34	0.54
	Rearrange	0.57	0.57	47.7	0.10	0.31	0.57
Siamese NN	Random	0.51	0.47	**34.2**	**0.27**	**0.52**	**0.74**
	Selected	0.52	0.47	42.8	0.18	0.39	0.59
	Reverse	0.53	0.48	42.7	0.16	0.37	0.62
	Shift	0.53	0.52	43.0	0.16	0.41	0.65
	Rearrange	0.53	0.53	44.2	0.16	0.40	0.60

Table 2: Objective evaluation result of different combinations of models (CNN or SNN; see Section 4.2) and negative sampling strategies (see Section 4.1). We highlight the best result for each metric in bold.

aligns well with the real-world use case of the proposed model: to find loops compatible with a query loop from a pool of loop libraries. However, we note that it may not be always possible to rank the target loop high in this evaluation, because the other loops in the candidate pool may also be compatible with the query.

In our experiment, we set the number of candidate loops to 100. We computed four metrics for the ranking task: top-10 accuracy, top-30 accuracy, and top-50 accuracy—which evaluate whether the target loop was placed in the top-10, top-30, and top-50 of the ranked list, respectively—as well as the average rank. We report the average of these values for the 100 different query loops.

Classification Result—Table 2 shows that, regardless of the negative sampling method, the CNN models outperform the SNN models for the classification-based metrics. While the CNN learns to rate the compatibility of the loop combinations directly, the SNN learns to place the loops in a space, with nearby loops being more compatible. The training data were the same, so the difference in performance suggests that the space learned was not effective for the classification task.

Ranking Result—Table 2 also shows CNN and SNN models seem to perform comparably with respect to the ranking-based metrics. Yet, the best scores in the four ranking-based metrics are all obtained by SNN with 'random' negative sampling. When an SNN is used, there appears to be a large performance gap between 'random' and all the other negative sampling methods. This suggests that focusing on harmonic compatibility alone (e.g., as done by using the 'selected' negative sampling method) is not optimal; compatibility in other dimensions of music is also important. On the other hand, when a CNN is used, 'reverse' appears to perform the best among the five negative sampling methods.

5.2 Subjective Listening Test
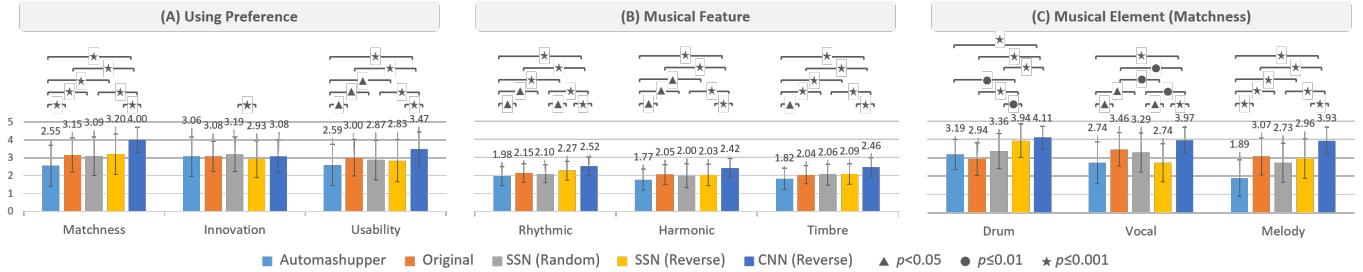
We note that even if a model obtains the highest classification or ranking result, we still cannot guarantee that the model also works well in real-world applications. Accordingly, we deployed a user study by releasing an online questionnaire to get user feedback. As the users' time is

precious, it is not possible to test the result of all combinations of the models and negative sampling methods. Therefore, we considered the result of the objective evaluation and picked five methods for subjective evaluation:

- 'CNN + reverse', 'SNN + random,' and 'SNN + reverse,' three instances of the proposed methods that performed well in the objective evaluation;
- 'AutoMashUpper' [2], as it represents the state-of-the-art for the task. Our implementation is based on the open source code from `https://github.com/migperfer/AutoMashupper`. As the rhythmic compatibility part of [2] is missing in this repo, we implement it ourselves following [2].
- 'Original,' which stands for the real loop combinations observed in FMA and extracted by the procedures described in Section 3.2. Specifically, an 'original' loop combination is one of the 100 positive loop pairs from the 'test set' listed in Table 1.

The loop combinations, or *test clips*, presented to users for evaluation are created as follows. As in Section 4.1.2, for each positive loop pair, we view one as the source loop, and the other as the target loop. Accordingly, we have 100 source loops and 100 target loops in the test set. A test clip is a combination of two loops. For the 'original' method, we pair the source loop with its true target. For the other methods, the 99 target loops (excluding the true target) are ranked, and the one predicted to match the source best is combined with it to form a test clip. The 5 resulting test clips for each source loop were considered as a *group*. This way, we have 100 groups in total. Among them, we picked 6 groups for evaluation: they have 2 vocal loops, 2 drum loops, and 2 loops of instrumental melody as the source loop, respectively.

We designed a subjective listening test that could be administered to users through an anonymous online questionnaire, and advertised it on social media websites related to EDM and Hip-Hop. A participant was randomly directed to a questionnaire containing one group of test clips, and was then asked to listen to the 5 test clips and rate each clip, on a 5-point Likert scale, in terms of: i) whether they sounded *matched*, ii) whether they were an *innova-*

Figure 4: The subjective evaluation results of comparing the preference, musical feature and musical element among 5 methods. A method is considered to show low abilities to manipulate loops if its MOS is under 3 in (A) 'using preference'-related and (C) 'musical elements'-related metrics, and under 2 in (B) 'musical feature'-related metrics.

tive combination, iii) and whether they were *usable* in future compositions (see Figure 4A). They were also asked to indicate whether the loops matched according to 3 musical features: rhythm, harmony, and timbre (see Figure 4B). Finally, to see how consistent the models are at recommending loops given different query types, Figure 4C breaks down the results for Matchness (Figure 4A(i)) according to source loop type: drum, vocal or melody loop.

5.2.1 Subjective Evaluation Result

Data from 116 Taiwanese participants were collected, and all of them were included for analysis. The participants self-reported the gender they identified with (36 female, 80 male) and age (19–33). 50% said they listened to loop-based music more than 5 days a week, 54% had classical musical training, and 42% had experience composing loop-based music. Overall, the responses indicated an acceptable level of reliability (Cronbach's $\alpha = 0.778$).

Figure 4 indicates the mean opinion score (MOS). A Wilcoxon signed-rank test was conducted to statistically evaluate the comparative ability of the 5 methods.

Overall, we observe that AutoMashUpper performs least well in almost all the evaluation metrics. Outside of the *Innovation* (Figure 4A) and *Drum* metrics (Figure 4C), AutoMashUpper is significantly worse than almost all the other methods. This suggests that, in loop selection, considering only loop similarity and spectral balance (as assumed by AutoMashUpper) is not enough—perceptually compatible loops are not necessarily similar in content.

On the other hand, we note that 'CNN with reverse negative sampling' performs the best in almost all of the evaluation metrics. To our surprise, the loop combinations picked by the CNN are preferred even to the original loop combinations extracted from FMA songs. The performance difference between 'CNN + reverse' and 'Original' is significant (p-value < 0.001) in terms of several metrics, including *Matchness* and *Usability*. In contrast, there is no significant performance difference between 'Original' and the two SNN models.

One finding was surprising: The MOS obtained by 'CNN + reverse' for *Matchness* is 4.0, higher even than the MOS of 3.15 obtained by 'Original.' I.e., the loop combinations proposed by the system were found to match better than the original combinations. We are not sure how to ac-

count for this success. One might conjecture that the quality of the 'Original' pair suffers from the imperfect loop extraction algorithm, but since all the loops were extracted the same way, they should be on equal footing. Alternatively, the novelty of the non-original mash-ups could be more interesting to the listeners than the originals; however, there was no clear difference among the systems in terms of 'Innovation.' We can only conjecture that 'CNN + reverse' found better loop combinations than human-made ones because the model-based method could examine all the possible loop combinations, while humans cannot.

We found that SSN methods, despite their performing better in ranking-based objective metrics than the CNN method, did not create perceptually-better loop combinations. This suggests a discrepancy between the objective and subjective metrics. And lastly, while it seems fair to say that CNNs outperform SNNs in the subjective evaluation, it is hard to say whether 'reverse' is the most effective negative sampling, because we were not able to evaluate all the possible methods here. This is left as a future work.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a data generation pipeline and several negative sampling strategies to address the need of ground-truth labels for building a machine learning model for estimating the compatibility of loops. We have also implemented and evaluated two different network architectures (CNN and SNN) to build such models. Our objective evaluation shows that a CNN does well in classifying incompatible loop pairs and an SNN is good at imitating how producers combined loops, and our subjective evaluation suggests the loop combinations created by a CNN are favored over those created by an SNN and even, in some aspects, real data from FMA. Both CNNs and SNNs outperform the rule-based system AutoMashUpper.

We have two plans in place for future work. First, as we see some inconsistency between the results of objective and subjective evaluations, we plan to investigate other objective metrics for performance evaluation. Second, we plan to exploit the loop layouts estimated by the loop extraction algorithm [5] to study further the relationship between loops and their arrangement, which may aid in the automatic creation of loop-based music.

7. ACKNOWLEDGEMENT

8. REFERENCES

[1] T. Kitahara, K. Iijima, M. Okada, Y. Yamashita, and A.Tsuruoka, "A loop sequencer that selects music loops based on the degree of excitement," in *Proc. Int. Sound and Music Computing Conf.*, 2015.

[2] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "AutoMashUpper: Automatic creation of multi-song music mashups," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 22, no. 12, p. 1726–17370, 2014.

[3] J. B. L. Smith, G. Percival, J. Kato, M. Goto, and S. Fukayama, "CrossSong Puzzle: Generating and unscrambling music mashups with real-time interactivity," in *Proc. Int. Sound and Music Computing Conf.*, 2015.

[4] K. Lee and J. Nam, "Learning a joint embedding space of monophonic and mixed music signals for singing voice," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2019.

[5] J. B. L. Smith and M. Goto, "Nonnegativetensor factorization for source separation of loops in audio," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing.*, 2018.

[6] J. Bromley, I. Guyon, Y. Lecun, E. Sckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *In Proc. Annual Conf. Neural Information Processing Systems*, 1994.

[7] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2017.

[8] S. Streich and B. S. Ong, "A music loop explorer system," in *Proc. Int. Computer Music Conf.*, 2008.

[9] Z. Shi and G. J. Mysore, "LoopMaker: Automatic creation of music loops from pre-recorded music," in *Proc. SIGCHI Int. Conf. Human Factors in Computing Systems*, 2018.

[10] P. Seetharaman and B. Pardo, "Simultaneous separation and segmentation in layered music," in *Proc. Int. Soc. Music Information Retrieval Conf.*, New York, NY, USA, 2016, pp. 495–501.

[11] J. B. L. Smith, Y. Kawasaki, and M. Goto, "Unmixer: An interface for extracting and remixing loops," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2019.

[12] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "AutoMashUpper: An automatic multi-song mashup system," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2013.

[13] C.-L. Lee, Y.-T. Lin, Z.-R. Yao, F.-Y. Lee, and J.-L. Wu, "Automatic mashup creation by considering both vertical and horizontal mashabilities," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2015.

[14] G. Bernardes, M. E. P. Davies, and C. Guedes, "Psychoacoustic approaches for harmonic music mixing," *Applied Science*, 2016.

[15] ——, "A perceptually-motivated harmonic compatibility method for music mixing," in *Proc. Int. Symp. Computer Music Multidisciplinary Research*, 2017, pp. 105–115.

[16] B. Xing, X. Zhang, K. Zhang, X. Wu, H. Zhang, J. Zheng, L. Zhang, and S. Sun, "Popmash: an automatic musical-mashup system using computation of musical and lyrical agreement for transitions," *Multimedia Tools and Applications*, 2020.

[17] C. Zauner, "Implementation and benchmarking of perceptual image hash functions," 2010, master Thesis. University of Applied Sciences Upper Austria.

[18] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proc. IEEE Int. Conf. Computer Vision*, 2015.

[19] F. Schroff, D. Kalenichenko, and J. Philbi, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015.

[20] F. Schroff, D. Kalenichenko, and J. Philbin, "Sampling matters in deep embedding learning," in *Proc. IEEE Int. Conf. Computer Vision*, 2017.

[21] R. Riad, C. Dancette, J. Karadayi, T. S. N. Zeghidour, and E. Dupoux, "Sampling strategies in siamese networks for unsupervised speech representation learning," in *Proc. Interspeech.*, 2018.

[22] J. Royo-Letelier, R. Hennequin, V. AnhTran, and M. Moussallam, "Disambiguating music artists at scale with audio metric learning," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2018.

[23] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and state-of the art music source separation tool with pre-trained models," Late-Breaking/Demo ISMIR, 2019.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Machine Learning*, 2015.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," in *J. Machine Learning Research*, 2014.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Computer Vision*, 2015.

[27] G. Koc, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. Int. Conf. Machine Learning, ICML Deep Learning Workshop*, 2015.

[28] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, "Representation learning of music using artist labels," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2017.

[29] Y.-S. Huang, S.-Y. Chou, and Y.-H. Yang, "Generating music medleys via playing music puzzle games," in *Proc. Int. Conf. Artificial Intelligence*, 2018.

[30] U. Sandouk and K. Chen, "Learning contextualized music semantics from tags via a Siamese neural network," *ACM Trans. Intelligent Systems and Technology*, vol. 8, no. 2, 2016.

[31] P. Manocha, R. Badlani, A. Kumar, A. Shah, B. Elizalde, and B. Raj, "Content-based representations of audio using siamese neural networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2018, pp. 3136–3140.

[32] L.-C. Yu, Y.-H. Yang, Y.-N. Hung, and Y.-A. Chen, "Hit song prediction for Pop music by Siamese CNN with ranking loss," 2017, arxiv preprint: 1710.10814.

[33] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

DATA QUALITY MATTERS: ITERATIVE CORRECTIONS ON A CORPUS OF MENDELSSOHN STRING QUARTETS AND IMPLICATIONS FOR MIR ANALYSIS

Jacob deGroot-Maggetti[1,2]   **Timothy de Reuse**[1,2]   **Laurent Feisthauer**[2,3]
Samuel Howes[1,2]   **Yaolong Ju**[1,2]   **Suzuka Kokubu**[1,2]   **Sylvain Margot**[1,2]
Néstor Nápoles López[1,2]   **Finn Upham**[1,2]

[1] Schulich School of Music, McGill University, Canada
[2] Computational Tonal Study Group
[3] Department of Information, University of Lille, France

jacob.degroot-maggetti@mail.mcgill.ca, timothy.dereuse@mail.mcgill.ca,
laurent.feisthauer@univ-lille.fr, samuel.howes@mail.mcgill.ca, yaolong.ju@mail.mcgill.ca,
suzuka.kokubu@mail.mcgill.ca, sylvain.margot@mail.mcgill.ca,
nestor.napoleslopez@mail.mcgill.ca, finn.upham@mail.mcgill.ca

ABSTRACT

In this paper, we describe a workflow of successive corrections on Optical Music Recognition (OMR) generated MusicXML files and their respective outputs under Music Information Retrieval (MIR) tasks. The original OMR-generated files of six Mendelssohn String Quartets were initially corrected by individual members of this interdisciplinary group, then reviewed by others to further standardize the quality and music analysis priorities of the team. Four MIR tasks are applied to each round of corrections on this collection: cadence detection, chord labeling, key finding, and monophonic pattern discovery. We measure changes in the outputs of these four MIR tasks from one round of corrections to the next in order to evaluate the impact of corrections. Results show that expert revision is more beneficial to some MIR tasks than to others. The resulting corpus of curated MusicXML files is available as an open-source repository under a Creative Commons Attribution 4.0 International License for further MIR research.

1. INTRODUCTION

Music Information Retrieval (MIR) algorithms that analyze symbolic music require high-quality data to produce accurate results. When building symbolic music corpora for MIR research, manually transcribing data using music notation software is expensive [1].

A faster option might be to use Optical Music Recognition (OMR) software on existing images of printed scores as an initial step. For example, Condit-Schultz et al. [2] worked on automated harmonic analysis of 571 chorales by Johann Sebastian Bach and Michael Praetorius. OMR was used in the process of creating symbolic encodings, with the results reviewed and manually corrected by a human annotator. Cumming et al. [3] created symbolic corpora of Renaissance music using OMR-generated scores as the first step and followed strict guidelines of manual corrections for the retention, addition, or removal of specific notations such as ties and fermatas. Although the performance of OMR applications has been improving over the years [4], extensive manual revisions are still required to ensure data quality and consistency for MIR analysis. This expensive and time-consuming task is especially relevant for OMR-induced errors since small ambiguities can lead to substantial variation in analytical output [5]. *What are the impacts of expert curation on data for MIR analysis tasks?*

We answer this question using files produced in the process of building a symbolic corpus of Mendelssohn string quarters. The OMR-generated passed through three iterations of increasingly-stringent manual corrections without additional annotations. We measured the impact of each round of corrections on four MIR analysis tasks (key finding, chord labeling, melodic pattern discovery, and cadence detection) through the changes between each iteration. Expert analysis of the scores exposes the types of errors to which these tasks are sensitive, demonstrating the need to tune corpus content to the anticipated analyses.

2. CORPUS CREATION

2.1 Mendelssohn String Quartets

Our corpus consists of the string quartets of Felix Mendelssohn. The Classical string quartet is a particularly relevant genre for computer-assisted analysis of music. In contrast to piano repertoire, where voice leading is often obscured by the limitations of the performer's hands and by what can be notated on the grand staff, a string quartet score preserves the independent parts of four separate instruments, allowing attribution of the role of each part (such as melody, bass, accompaniment, leading and imitative voices, etc.). The Classical aesthetic is characterized by a very clear harmonic, melodic, and formal organization, so it is not surprising that Beethoven's and Mozart's string quartets have already been encoded and annotated [6,7] for music analytical purposes. Mendelssohn's string quartets are a natural next step; his works have Classical characteristics that place them in the tradition of Beethoven's late quartets [8].

Specifically, we encoded quartets Op. 12, Op. 13, and Op. 44, Nos. i, ii, and iii, all composed between 1827 and 1847, and *Four Pieces for String Quartet*, Op. 81. [1] The initial OMR encodings of these 24 movements were generated from scans of the 1875 edition published by *Breitkopf und Härtel*, available as PDF files on IMSLP. [2]) Although not part of the set studied in this paper, an additional quartet, Op. 80, is incorporated in our final published corpus. It had been previously encoded in MusicXML [3] by user *Musemeister*.

2.2 The CTS Team

An interdisciplinary team of nine people collaborated to create this corpus, with members from music technology, music theory, string performance, and music cognition. Each member brought unique viewpoints, skillsets, and objectives to the project. Whether they were interested in a specific MIR task or the applications of MIR to music theory, cognition, or pedagogy, team members refined encoding objectives together to satisfy their varied interests during the iterative correction and cleaning of OMR-generated symbolic music files.

2.3 From PDF to MusicXML

The first step in building the corpus was to transcribe the PDF files into a symbolic, machine-readable format. We used the commercial OMR software PhotoScore to analyze the original score images in PDF. It first detected the position of each staff on the page and we ensured that these detected positions were correct, adjusting as necessary. We also manually corrected the key and time signatures. The OMR results were then exported to MusicXML because the format is widely supported by music notation software. These files formed the Corrections 0 dataset "C0",

---
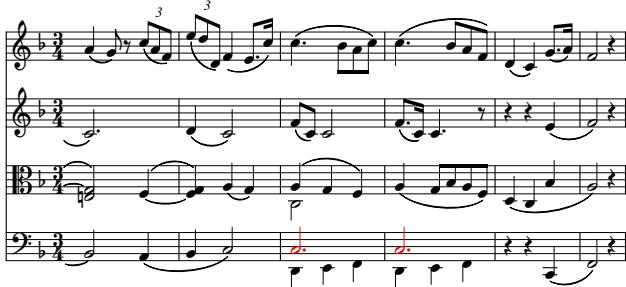
Figure 1. Measures 60-62 of Op. 44, No. iii, Mvt. 4. The upper system is initial OMR output (C0) and the lower system is after three rounds of manual corrections (C3).

with no additional manual corrections of score information. All subsequent corrections were made using MuseScore v2.3.2.

2.4 OMR Corrections

Starting from these initial OMR-generated music files (C0) we applied three successive stages of manual corrections: "C1", "C2", and "C3". The goal of each stage was to improve the accuracy of the previous stage(s) and to ensure that all information necessary for the MIR algorithms was included. The original C0 files included score elements both unlikely to be used by existing symbolic MIR algorithms and deemed by the team's music theorists to be less essential for the specific analytical approaches that we chose. Many of these score elements—for example, hairpin dynamics—were ignored or removed during the rounds of corrections (for a full list, refer to the supplementary materials).

The focus of C1 was accuracy of pitch and rhythm, while elements such as dynamic markings and articulations were largely ignored in the interest of time. As work progressed, it became clear which errors were most common in the OMR output. For example, there were many misaligned or missing notes in passages with higher note density (see Figure 1). The OMR software frequently encoded ties as slurs and *vice versa*. Despite their visual similarity, these curved lines produce different rhythmic values of notes, with consequences for our MIR analysis algorithms. Mendelssohn's scores also included many detailed performance instructions, prompting lengthy discussions about what information should be preserved in the final dataset

---

Figure 2. Measures 14-19 of Op. 13, Mvt. 2.

and what should be left out. With a more rigorous protocol in place, we reviewed each other's work to produce C2. The main purpose of this phase of review was to ensure that no errors had been missed in the previous round of revision. Again, during C2, details of the score came to light that necessitated further discussion, and the varied perspectives of the multidisciplinary team informed decisions about how to proceed. For example, double-stops, where more than one string is played at once, posed a recurring challenge. While chords on a staff line can be encoded in most analysis systems, Mendelssohn wrote many passages with moving lines against held notes (see Figure 2). This texture is easily misinterpreted by algorithms unfamiliar with the particularities of string music. Ultimately, note accuracy and articulation of onsets (namely, ties, slurs, and staccatos) were prioritized, while most indications of dynamics and ornamentation were excluded: a trade-off between comprehensiveness and machine interpretability.

Finally, the last round of corrections (C3) was a review to align all the encodings according to the conclusions of discussions during C2 and to standardize formatting and metadata. A full account of the score elements preserved in the final dataset is included in the supplementary materials. For consistency, a single person reviewed all the movements in preparing C3. Further discussion is provided in Section 4.1.

2.5 Differences between Correction Rounds

Besides a qualitative report on the amount of corrections we made in each round, quantitative measurement of the scale of changes is possible on these digital files. As the ideal MusicXML tool [9] is not publicly available with open source code, we used the more generic `SequenceMatcher.quick_ratio()` from Python's `difflib` library to produce percent differences. The median and range of similarity scores between C0 and C1, C1 and C2, and C2 and C3 are shown in Table 1.

Interpreting these numbers directly is difficult as MusicXML files include a plethora of elements beyond the focus of our corrections. Still, it is reassuring to see the median difference between successive corrections decrease by an order of magnitude each round. If all file modifications had the same impact on the MIR analyses, their outputs would show a similar pattern of decreasing impact.

Comparison Pair	Percent Difference median [min, max]
C0 to C1	10.0% [2.8%, 21.8%]
C1 to C2	1.3% [0.0%, 7.3%]
C2 to C3	0.2% [0%, 1.3%]

Table 1. Percent difference for each comparison pair. The results are medians across all 24 movements, with maximum and minimum values indicated in brackets.

3. MIR ALGORITHMS

Four symbolic MIR algorithms were applied to each version of the Mendelssohn String Quartet Corpus. These algorithms were chosen because they were either designed or extensively used by members of the CTS team. Without ground truth annotations to assess the *accuracy* achieved by each MIR tasks, the corrections were evaluated through their *perceivability* to the algorithms in output *changes* between successive versions. [4] For two of the tasks that produce sequences of annotations, results from different versions of the same movement had to be aligned to one another before comparison; this procedure is detailed in the supplementary materials. In total, 96 evaluations per task were performed as each analysis algorithm was applied to all four versions of the 24 movements in the corpus.

3.1 Key Analysis

A recent key-finding algorithm [10] provided two predictions: global key per movement and local key per onset slice. We ran the algorithm using the default parameters provided in the implementation. Between C0 and C1, predictions of global key changed in 3 of the 24 files tested. There was no change in prediction between C1, C2, and C3. Predictions of local key changed substantially between C0 and C1 across all files, and changed much less between C1 and C2, and between C2 and C3, as shown in Table 2.

Comparison pair	Changes in local key annotations median(%) [min(%), max(%)]
C0 to C1	46.8% [9.9%, 71.1%]
C1 to C2	0.4% [0.0%, 9.8%]
C2 to C3	0.0% [0.0%, 3.0%]

Table 2. Percent differences in local key annotations for each comparison pair. The results are medians across all the 24 movements with minimum and maximum values indicated in brackets.

3.2 Chord Labeling

The automatic chord labeling model [11] was applied to each stage of the dataset, predicting chords for every onset slice of the piece.

---

[4] E.g. comparing the outputs of a key-finding algorithm applied to C0 and C1.

Comparison Pair	Changes in chord labels median(%) [min(%), max(%)]
C0 to C1	69.1% [17.5%, 96.7%]
C1 to C2	0.7% [0.0%, 41.1%]
C2 to C3	0.0% [0.0%, 12.5%]

Table 3. Percent difference in chord annotations for each comparison pair, shown as medians across all the 24 movements with minimum and maximum values indicated in brackets.

Each chord is labelled according to its root (C, F♯, B♭, etc.) and its quality (major, minor, fully diminished seventh, dominant seventh, etc.), with no mention of its inversion or its harmonic function (Roman numeral analysis).

The results are shown in (Table 3). We can see differences between C0 and C1 were substantial (median 69.1%), while the percent change between C1 and C2 was much smaller (median 0.7%). The majority of the movements showed no change in chord labels between C2 and C3 (median 0.0%). Such results indicate that chord labelling is not sensitive to local differences.

3.3 Monophonic Pattern Discovery

The SIARCT-C Algorithm [12] was used on each version of each movement to discover sets of repeating patterns. A "pattern" here refers to a set of excerpts of a piece that are all nearly identical in pitch and rhythm under transposition. While the algorithm is capable of operating on polyphonic music, here we focus on finding monophonic patterns between voices. To this end, each MusicXML file was transformed into point sets of (onset time in quarter notes, morphetic pitch) pairs; for example, the first measure of the first violin's part in Figure 1 is notated as the sequence `(69, 0) (69, 1) (71, 2) (71, 2.75)`. Dynamics, articulatons, and durations are discarded. Some algorithmic pattern discovery methods do use this kind of information, such as the Automatic Time-span Tree Analyzer [13], but the majority use only rhythmic and pitch-related data, partially due to the computational complexity of the problem. The four voices in each file were concatenated into one sequence for the purpose of evaluation.

We searched only for patterns that were at least eight notes long that occurred at least five times within each movement, allowing for a small amount of variation. These parameters were chosen as a compromise in light of the number of movements we had to analyze and the running time of the algorithm; searches for short patterns take significantly longer than searches for long patterns. We illustrate the effect of iterative corrections by their impact on descriptive statistics of these results: the number of unique patterns detected, the coverage of these patterns over all notes in the music, and the median cardinality (i.e., number of instances) of each pattern discovered. [5]

Table 4 shows how these statistics change between versions. Many more patterns were found in C1 than in C0 (median 85%, maximum 2100%), with a small amount of gain and loss from C1 and C2, and no change in total from C2 to C3. Coverage also grew substantially from C0 to C1, including more than twice the number of notes after this first round of corrections for more than half the movements. In contrast, the median cardinality did not change as drastically for those patterns detected in these different versions, and no apparent changes occurred during the last round of corrections.

Comp. Pair	Magnitude Increase, median [min, max] (%)		
	Num. Patterns	Coverage	Cardinality
C0 to C1	85% [16%, 2100%]	110% [21%, 1100%]	8.3% [-22%, 29%]
C1 to C2	0.0% [-5.1%, 7.1%]	0.041% [0.85%, 9.8%]	0.0% [-3.4%, 3.5%]
C2 to C3	0.0% [0.0%, 0.0%]	0.0% [0.0%, 0.0%]	0.0% [0.0%, 0.0%]

Table 4. Median magnitude increase in three statistics taken on the sets of discovered patterns over the course of the corrections. Minimal and maximal values for change are shown in brackets.

Comparison pair	Change in PACs detected	New PACs detected	PACs lost
C0 to C1	154.5% 22 to 56	177.3% 39	22.7% 5
C1 to C2	-3.6% 56 to 54	1.8% 1	5.4% 3
C2 to C3	1.9% 54 to 55	3.7% 2	1.9% 1

Table 5. Change in the number of PACs detected. 'New PACs detected' report the number of PACs detected in the latter that were not in the former. 'Lost PACs' is the number of PACs that were in the former but not the latter. As the number of PACs detected is quite small, both relative changes and exact numbers are given.

3.4 Cadence Detection

Finally, the cadence detection algorithm introduced by Bigo et al. [14] was used to detect perfect authentic cadences (PACs) [6] throughout the corpus. Each beat was evaluated as a potential point of cadential arrival using a Support Vector Machine. As there are only few cadences within single movements, Table 5 reports the results of

---

[5] Comparing sets of discovered patterns is difficult because of their highly heterogeneous structure, with individual patterns spanning a wide

range of cardinalities and number of notes per instance. While it is possible to devise a more direct evaluation based on similarities between individual patterns, we used an approach based on descriptive statistics for the sake of brevity and interpretability.

[6] Too few cadences of other types, such as half cadences, were successfully detected to interpret sensibly in this context.

this evaluation as a count of cadences detected rather than percent change. As expected, the model benefited greatly from the initial round of corrections: twice as many cadences were identified in C1 files as in C0 files. Additional rounds of reviews had little impact on the total number of PACs detected. Close investigation of the differences between detected cadences in C1, C2 and C3 revealed that some changes in the algorithm's output were due to corrections in notated pitch.

4. DISCUSSION

The different rounds of corrections prompted a wide range of considerations for the group, which are discussed below. Proofreaders with different kinds of expertise, whether in MIR, music theory, string musicianship, or the use of the chosen music notation software, communicated various concerns and discoveries relating to their respective tasks. Finally, special situations are discussed, in which music theoretical and analytical considerations collide with MIR objectives in notable ways.

4.1 MIR Significance of Correction Rounds

Using OMR to create datasets of symbolic music is an attractive proposition. Our results suggest that there is significant variation in the quality of the output between files when using software like PhotoScore. No task evaluated here was able to totally overcome the errors introduced by OMR, with all of the results seeing some amount of change after the first round of corrections, and the degree of change in this initial round varied widely between tasks. Global key estimations changed for only 3 of the 24 movements, while the discovered patterns on the raw OMR output bear little resemblance to those discovered after just one round of corrections. However, these findings cannot be extrapolated directly to other algorithms that perform the same tasks; different machine-learning methods may cause models to become more sensitive to some errors and less sensitive to others. For the specific algorithms applied here, we may consider this as evidence that the underlying symbolic-musical structures they use to make judgments are affected by errors in the OMR process to different magnitudes. For most tasks, though, initial correction is necessary when using OMR to create datasets, given the current capabilities of commercially-available OMR software.

For subsequent rounds of corrections, the sizes of changes shrink dramatically but still vary between tasks. In particular, the discovered patterns barely change at all after the first round of corrections; this is likely due to the fact that the algorithm uses only onset times and morphetic pitch, thus ignoring some pitch changes with harmonic consequences.

4.2 Experience of Doing Corrections

For the members of our team with solid experience in copying music, correcting OMR required an average time of 30 to 45 minutes per printed page depending on the variety and amount of errors. To review Mendelssohn's complete string quartets (C0 to C1) thus took approximately 75 to 110 hours. Even though the standardization step (C1 to C2) in itself was much shorter (5 to 15 minutes per printed page, or an approximate total of 25 hours), discussions concerning what should be kept and what should be ignored lasted over a month. Finally, checking the consistency represented 30 additional hours (C2 to C3). While the dataset at C3 was standardized to meet the requirements of our analysis tasks, one might wonder whether investing this amount of time was necessary.

Different movements, and different passages within individual movements, required vastly different degrees of effort to correct. In some sections, only corrections to articulations and accidentals were needed, whereas other sections needed to be completely rewritten. The first round of corrections (C0 to C1) was the most difficult, involving many decisions about which elements of the score to preserve. Some time-consuming corrections had to be rolled back after standardization protocols had been decided on. This round of corrections was particularly difficult for proofreaders who had never used MuseScore. Certain features of the program, such as the addition of key signatures, introduced multiple additional errors when used incorrectly, while some features that might have saved time, such as batch addition of articulation marks, went unused through much of the correction process. There were a few musical situations that tended to produce predictable errors in the OMR encoding. Errors frequently arose when the OMR software missed or misplaced rests, and proofreaders quickly learned that passages with higher note density required much more effort to correct.

4.3 Musical Considerations

Figure 1 gives a general sense of the differences between the initial (C0) and final (C3) stages of the correction process. These differences fall into two broad categories: "pitch-rhythm" differences in the vertical (pitch) or horizontal (rhythm) placement of notes and "notational" differences in articulations, ornaments, and other non-pitch elements of the score. Pitch-rhythm differences had a large effect on the outcomes of analysis tasks. Pitches that were incorrect (Vla., m. 61), misaligned (Vln. 1, m. 62), missing (Vla., m. 62), or extraneous (Vln. 2, m. 61; Vc., m. 60) affected all four analysis tasks. Notational differences in tremolos (Vln. 2 & Vla., mm. 60-62), and slurs (Vln. 1, m. 60) had a smaller effect on the outcomes of analysis tasks, but could be disruptive in MIR tasks that make use of recurring notational cues, especially for pattern finding or cadence detection. For example, Mendelssohn often uses slurs, staccatos, and dynamic markings such as hairpins to highlight recurring motives. When these motives are transposed or altered non-uniformly, as in the tonal answer of a fugue or the development section of a sonata-form movement, they may become undetectable by pattern-finding algorithms that rely exclusively on pitch and rhythm. An analyst relying on these results may be led to misinterpret larger tonal, hypermetric, and formal structures if, for example, the algorithm fails to detect a main theme at the

beginning of a returning section. Algorithms that also consider articulations and dynamic markings might perform better in situations like these. One other complication is that PhotoScore sometimes generates hidden rests (Vla. m. 61), slowing the correction process by making the score less readable to humans.

Extraneous or incorrect clefs were found in four of 24 movements during the final round of corrections (C3). In the fourth movement of Op. 44 No. iii, an incorrect (French violin) clef in the first violin transposed the part up by a major third. This error persisted through several rounds of correction because it was obscured by system breaks. Incorrect clefs had a large effect on key-finding, chord-labeling, and cadence-detection tasks, but not on the transposition-invariant pattern discovery task.

Another recurring issue concerns multiple voices within a single staff (i.e., "double-stops" on a string instrument). While this is possible to encode in MusicXML format, it is often unreadable to the software used for analysis tasks. In these cases, the encoder must decide which voice to keep and which to discard, which can result in the loss of information. In Op. 13, Mvt. 2, mm. 16-17 (see Figure 2), the cello plays both a held C (in red) along with a moving line D-E-F. Without the moving line, the C becomes a pedal at the bottom of the texture, changing the harmonic and cadential sense of the music. In Op. 13, Mvt. 3, mm. 143-53, the cello and violin II have melodic lines below a harmonic pedal resulting in two-note chords for each of them. Choosing one voice or the other is difficult: keeping the pedal tones preserves the harmony, allowing for the detection of cadences and chords; keeping the melodic motives allows for the discovery of more patterns throughout the movement. Deciding what to keep depends on the type of analysis to be carried out.

4.4 Implications for OMR

The initial errors in OMR files disproportionately impacted these tasks: a 10% change in the MusicXML files produced a 47% change in local key judgments and a 69% change in chord labels. This proportion of incorrect results underlines how this commercial software struggled with these scores. One cause was missing notes: runs of sixteenths and eighths typical of this genre of music were often dropped, when the dense or complicated stemmings were incorrectly interpreted. The numerous articulation and ornamentation markings were often misinterpreted as notes, suggesting poor recognition of shapes within staves. There was also confusion between parts in the four staff systems, with notes and text annotations packed more tightly in vertical arrangements than in other genres and publication styles. Software tuned to this era and style of work would hopefully reduce the amount of information lost. However, given the variety in performance quality across these scores, human supervision is highly recommended.

5. CONCLUSION

This project is a case study in how human corrections on OMR can influence MIR analysis results. The range of outcomes across these analyses suggests that the value of human correction time depends on the MIR task. If some noise in the results is permissible and one is only interested in large-scale qualities like global key, the raw OMR files may suffice, but anything closer to the notes would benefit from some review and correction. Without human intervention, half of the outputs for local key detection and chord labeling were corrupted, while monophonic pattern discovery and cadence detection missed substantial portions of the relevant material in most pieces.

The second and third rounds of corrections had progressively smaller effects on the aggregate results of these analyses, as expected, but there are instances when these smaller changes were crucial for the type of analysis at hand. Passing the symbolic music files between multiple reviewers minimized the impact of human error. Some of the changes in these later rounds were motivated by new understanding of the music, music encoding limitations, and what could be used by our MIR algorithms.

This project is not representative of all symbolic music corpus-building with OMR. PhotoScore was not necessarily the best OMR processor for string quartet music printed in 1875. The sensitivity of these MIR analysis tasks on changes in symbolic music information is also specific to their implementations; a study of monophonic patterns that included articulation or dynamics would tell a different story from that above. However, the novel comparison process across iterations of corrections highlights the importance of expert musical care in the developing of symbolic music corpora, as well as the need for explicit acknowledgement of the types of score information preserved therein.

Discussions between team members about the potential relevance of ornamentation and articulation to each analytical objective resulted in a set of files that contained more information than could be used by the algorithms applied here. At the same time, important layers like dynamics were removed because of the difficulty of producing machine-interpretable encodings. We hope to see that the retained layers of performance information are used in future work with this collection of symbolic scores, and that symbolic music encoding and analysis tools continue to progress towards capturing a richer range of musical information. The final version of this Mendelssohn String Quartet Corpus, a pedagogical, scholarly, and artistic resource for musicians, composers, and music researchers alike, can be downloaded from: `https://github.com/DDMAL/felix_quartets_got_annotated`.

While OMR can be a helpful tool for corpus building, such projects still require human expertise in both the music represented and in its intended uses. For MIR-related research, some tasks benefit from manual review more than others.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] M. Gotham, P. Jonas, B. Bower, W. Bosworth, D. Rootham, and L. VanHandel, "Scores of scores: An openscore project to encode and share sheet music," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, ser. DLfM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 87–95. [Online]. Available: https://doi.org/10.1145/3273024.3273026

[2] N. Condit-Schultz, Y. Ju, and I. Fujinaga, "A flexible approach to automated harmonic analysis: Multiple annotations of chorales by Bach and Prætorius," in *Proc. of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 66–73.

[3] J. E. Cumming, C. McKay, J. Stuchbery, and I. Fujinaga, "Methodologies for creating symbolic corpora of Western music before 1600," in *Proc. of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 491–498.

[4] J. Calvo-Zaragoza, J. Hajic Jr, and A. Pacha, "Understanding optical music recognition," *arXiv preprint arXiv:1908.03608*, 2019.

[5] N. Nápoles López, G. Vigliensoni, and I. Fujinaga, "Encoding matters," in *Proc. of the 5th International Conference on Digital Libraries for Musicology*, Paris, France, 2018, pp. 69–73. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3273024.3273027

[6] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, "The Annotated Beethoven Corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets," *Frontiers in Digital Humanities*, vol. 5, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fdigh.2018.00016

[7] N. Zaslaw, "Review: Digital Mozart Edition (DME)," *Journal of the American Musicological Society*, vol. 71, no. 2, pp. 572–586, 08 2018. [Online]. Available: https://doi.org/10.1525/jams.2018.71.2.572

[8] R. L. Todd, "Mendelssohn(-Bartholdy), (Jacob Ludwig) Felix," *Grove Music Online*, 2000, available at https://doi.org/10.1093/gmo/9781561592630.article.51795. Accessed March 20th, 2020.

[9] F. Foscarin, F. Jacquemard, and R. Fournier-S'niehotta, "A diff procedure for music score files," in *6th International Conference on Digital Libraries for Musicology*, 2019, pp. 58–64.

[10] N. Nápoles López, C. Arthur, and I. Fujinaga, "Key-finding based on a hidden markov model and key profiles," in *Proc. of the 6th International Conference on Digital Libraries for Musicology*, New York, NY, 2019, pp. 33–37.

[11] Y. Ju, S. Howes, C. McKay, N. Condit-Schultz, J. Calvo-Zaragoza, and I. Fujinaga, "An interactive workflow for generating chord labels for homorhythmic music in symbolic formats," in *Proc. of the 20th International Society for Music Information Retrieval Conference*, Delft, Netherlands, 2019, pp. 862–869.

[12] T. Collins, A. Arzt, S. Flossmann, and G. Widmer, "SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations," in *Proc. of the 14th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2013, pp. 549–554. [Online]. Available: http://www.cp.jku.at/research/papers/collins_etal_ismir_2013.pdf

[13] M. Hamanaka, K. Hirata, and S. Tojo, "ATTA: Automatic time-span tree analyzer based on extended GTTM," in *Proceedings of the 6th International Society for Music Information Retrieval Conference*, London, UK, 2005, pp. 358–365.

[14] L. Bigo, L. Feisthauer, M. Giraud, and F. Levé, "Relevance of musical features for cadence detection," in *Proc. of the International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 355–361. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01801060

METRIC LEARNING VS CLASSIFICATION
FOR DISENTANGLED MUSIC REPRESENTATION LEARNING

Jongpil Lee[1]   **Nicholas J. Bryan**[2]   **Justin Salamon**[2]   **Zeyu Jin**[2]   **Juhan Nam**[1]

[1] Graduate School of Culture Technology, KAIST, Daejeon, South Korea

[2] Adobe Research, San Francisco, CA, USA
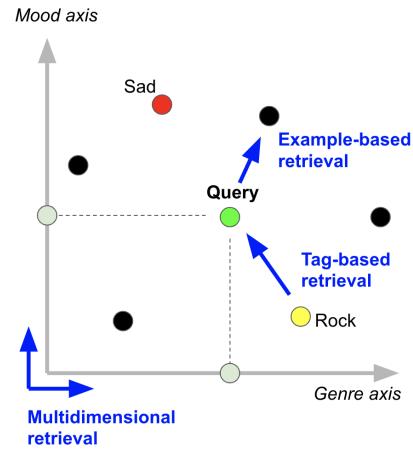
`{richter, juhannam}@kaist.ac.kr, {nibryan, salamon, zejin}@adobe.com`

ABSTRACT

Deep representation learning offers a powerful paradigm for mapping input data onto an organized embedding space and is useful for many music information retrieval tasks. Two central methods for representation learning include deep metric learning and classification, both having the same goal of learning a representation that can generalize well across tasks. Along with generalization, the emerging concept of disentangled representations is also of great interest, where multiple semantic concepts (e.g., genre, mood, instrumentation) are learned jointly but remain separable in the learned representation space. In this paper we present a single representation learning framework that elucidates the relationship between metric learning, classification, and disentanglement in a holistic manner. For this, we (1) outline past work on the relationship between metric learning and classification, (2) extend this relationship to multi-label data by exploring three different learning approaches and their disentangled versions, and (3) evaluate all models on four tasks (training time, similarity retrieval, auto-tagging, and triplet prediction). We find that classification-based models are generally advantageous for training time, similarity retrieval, and auto-tagging, while deep metric learning exhibits better performance for triplet-prediction. Finally, we show that our proposed approach yields state-of-the-art results for music auto-tagging.

1. INTRODUCTION

Learning a good representation, or embedding space, is a key goal in deep learning and is central to music classification and retrieval tasks. An important quality of a good representation is its generalization capability, i.e., its applicability to a diverse set of downstream tasks, including those relying on small datasets in a transfer learning setting [1–3]. While numerous representation learning methods have been explored to date, two learning

Figure 1. A disentangled music representation space. The green dot depicts a query song, the black dots depict retrieval songs, the red and yellow dots depict centroids of musical concepts, the gray arrows depict multidimensional axis, and the blue arrows depict retrieval methods.

paradigms are particularly common: deep metric learning and classification-based representation learning. The former is based on deriving similarity scores (or distances) between examples, while the latter is achieved via a cross-entropy loss over similarity scores between example and class centroids.

While both paradigms share the goal of learning a generalizable representation, the results from each approach are generally different. For example, a learned representation optimized via a classification task may perform poorly on a similarity-search task, and vice versa. While recent studies have elucidated the theoretical relationships between these paradigms and validated them through experimental findings [4], these developments have not been explored in the music domain. Furthermore, the relationship has not been explored for multi-label data, which is central to many music information retrieval tasks.

Beyond seeking a representation that generalizes across tasks, the emerging concept of disentangled representations [5,6] is of great interest for music applications. Music is often labeled with multiple semantic dimensions simultaneously (e.g., genre, mood, and instrumentation) and learning a representation that can capture this structure is advantageous. We often need to search for music that is similar along a particular semantic dimension in one ap-

plication (e.g., a music playlist with lighthearted mood), while requiring music similar along a different semantic dimension for another application (e.g., era for musicological analysis). Disentangled representations allow us to address both problems with a single model, and were recently proposed for audio-based music similarity search [7]. However, this study only explored disentanglement via a single deep metric learning approach, and the applicability and performance of more recent metric- and classification-based learning methods is yet to be explored.

In this paper, we present a unified representation learning framework that elucidates the relationship between metric learning, classification, and disentanglement. First, we outline past work on the relationship between metric learning and classification. We then extend this relationship to multi-label and multi-concept data (common to music applications) by exploring three different learning approaches and their disentangled versions – two of which are novel to this work. Finally, we evaluate all models against four tasks (training time, similarity retrieval, autotagging, and triplet prediction) and compare various aspects of the learned representations.

2. RELATED WORK

2.1 Metric Learning and Classification

The goal of distance metric learning is to obtain an embedding space where similar items are close together and dissimilar items are far apart. A common strategy is to use pairwise [8, 9] or triplet-based samples to train a model [10–13]. An important advantage of deep metric learning is that it can efficiently model an extremely large number of classes (e.g., for face recognition) [12]. However, training models using this strategy are relatively slow as models operate on triplets of input samples [14]. Recently, more efficient sampling techniques have been proposed to speed up convergence, including hard negative mining, semi-hard negative mining [12], distance weighted sampling [15], and proxy-based training [14]. Proxy-based training [14] assigns one or several proxies to each class (given by per-class embedding centroids) and optimizes the learned space by comparing embedded input samples to proxies instead of directly comparing them to positive and negative samples. This reduces training time significantly while improving retrieval performance on images.

Classification models, on the other hand, are typically trained such that classes are linearly separable in the embedding space of the last hidden layer of the deep neural network. Since classification models are not optimized based on distances in the learned embedding space, they may not perform well when directly used for similarity-based retrieval. To overcome this, recent work proposed the application of a normalization layer over the embedding space during training, and showed that this simple technique increases model performance on similarity-based image retrieval [4].

Recent and parallel advances in both paradigms (metric- and classification-based learning) have shown that there is an inherent link between them [4, 16, 17]. The per-class embedding centroids used in proxy-based training are, in fact, equivalent to the per-class vectors obtained from the linear transformation in the last hidden layer of a classification model [16]. Further, a recent comparative study demonstrated that the loss function of a triplet-based model is equivalent to that of a classification model up to a smoothing factor for single-label, multi-class data [16]. These findings suggest that deep metric- and classification-based learning are not as different as initially thought and we could, potentially, use either to learn a representation that generalizes well to both similarity-based retrieval and classification tasks.

2.2 Disentangled Representation Learning

Another important measure of representation learning is *disentanglement* [18]. Recently, Lee et al. adapted Conditional Similarity Networks (CSN) applied to triplet-based deep metric learning to the music domain [7, 19]. The main idea in CSN is to apply a masking function over the embedding space, where each mask corresponds to a different semantic dimension of similarity corresponding to musical notions such as genre, mood, instrument and tempo. They showed that the disentangled music representation not only enables multidimensional music search via its sub-dimensions, but also improves general music retrieval performance when all embedding dimensions are used. However, CSN for disentangled music representation learning was only explored using a deep metric learning strategy, and classification-based approaches were not studied. Considering the close relationship between the two, we propose to study disentanglement under classification, particularly for multi-labeled music data, and compare and contrast it to disentanglement via metric learning.

3. DISENTANGLED LEARNING MODELS
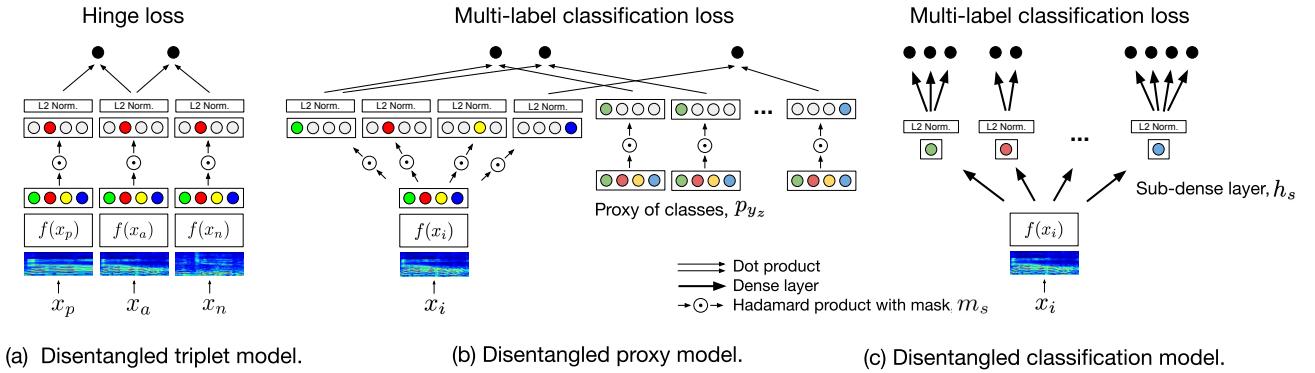
In this section, we introduce three disentangled learning methods, which are triplet-based, proxy-based, and classification-based models. The first model was previously developed [7], and the latter two are novel contributions. The overall architectures are illustrated in Figure 2. In the following descriptions, x denotes a data point, $f(\cdot)$ a nonlinear embedding function, y a multi-hot class label, and s a category (or a similarity notion such as mood, genre or instrumentation) of y. For example, if y_z is *rock*, then s_{y_z} is *genre*.

3.1 Triplet-based Model

Disentangled triplet-based models were recently proposed in [7, 19]. We first define a triplet as $t = (x_a, x_p, x_n; y_z)$, where x_a is the anchor sample, x_p is the positive sample, and x_n is the negative sample. x_a and x_p are sampled to have the same positive label y_z, while x_n is negative for y_z. Then, the basic triplet loss is defined as

$$L(t) = \max\{0, D(f(x_a), f(x_n)) - D(f(x_a), f(x_p)) + \Delta\},$$
$$(1)$$

where $D(f(x_i), f(x_j)) = cos(f(x_i), f(x_j))$ is a distance metric, and Δ is a margin value [11]. To disentangle the

Figure 2. A unified framework for disentangled triplet- and proxy-based metric learning and multi-label classification.

embedding feature of size d, a masking function $m_s \in \mathbb{R}^d$ is applied. The number of masks corresponds to the number of similarity notions s and each mask occupies certain dimensions of the $\mathbb{R}^d$ space evenly as illustrated in Figure 2 (a). Thus, when the $t = (x_a, x_p, x_n; y_z)$ is used, a mask for the similarity notion s_{y_z} is applied to the embedding feature space. The loss for training the model is given by:

$$L(t) = \max\{0, D(f(x_a) \circ m_s, f(x_n) \circ m_s) \\ - D(f(x_a) \circ m_s, f(x_p) \circ m_s) + \Delta\}, \quad (2)$$

where $\circ$ denotes the Hadamard product.

3.2 Proxy-based Model

The core idea of proxy-based metric learning is that proxy embeddings are learned and assigned to each class and used to measure the distance to an anchor data point instead of directly measuring distances to pairs or triplet data samples [14]. This can be interpreted as a supervised clustering algorithm, where proxies play a role of class centroids. In this approach, the distance metric becomes

$$D(f(x_i), p_{y_z}) = cos(f(x_i), p_{y_z}) = \frac{f(x_i)}{||f(x_i)||} \cdot \frac{p_{y_z}}{||p_{y_z}||}, \quad (3)$$

where x_i is a data point, p_{y_z} is a proxy for class y_z, and $\cdot$ is the dot product. If the data is single-labeled (multiclass), one can apply triplet loss, Neighborhood Component Analysis (NCA) loss [20], or Softmax loss over the above distance metric [14, 16], but with our multi-labeled data, it is not directly applicable. To address this, we replace these losses with a multi-label classification loss, i.e., binary cross entropy. The prediction score for each class becomes

$$\hat{y_z} = sigmoid(D(f(x_i), p_{y_z})), \quad (4)$$

and the loss is

$$L(x_i) = \sum_z [-y_z log(\hat{y_z}) - (1 - y_z) log(1 - \hat{y_z})]. \quad (5)$$

However, from our preliminary experiments, we found that the sigmoid function with cosine similarity score causes numerical problem in optimization. We speculate that the reason for this is that the cosine similarity score (bounded between -1 to +1) only activates the linear regions of the

downstream sigmoid activation, reducing model capacity. [1] Therefore, we modify the distance metric to be

$$D(f(x_i), p_{y_z}) = \frac{f(x_i)}{||f(x_i)||} \cdot p_{y_z}, \quad (6)$$

to ensure that both the learned embedding space is normalized and the sigmoid activations can have nonlinear properties.

From this basic multi-label proxy-based model, we expand the model by applying the masking function as used in the disentangled triplet-based model. Then, the prediction score for each class is updated to

$$\hat{y_z} = sigmoid(D(f(x_i) \circ m_s, p_{y_z} \circ m_s)), \quad (7)$$

as illustrated in Figure 2 (b).

3.3 Classification-based Model

Classification-based metric learning has recently been explored [4, 16]. The core idea is to apply a normalization layer on the embedding feature space. This simple technique ensures that the learned representation has unit length and makes similarity-based retrieval more effective compared to the vanilla classification model. Therefore, the prediction score of classification-based metric learning model for each class is

$$\hat{y_z} = sigmoid(\frac{f(x_i)}{||f(x_i)||} \cdot c_{y_z}), \quad (8)$$

where c_{y_z} is a centroid for each class (parameters of the last hidden layer). [2] At this stage, we observe that the distance metric inside the sigmoid function of Equation 8 is equivalent to that of our modified distance metric in Equation 6 of the proxy-based model.

As for triplet-based metric learning, we extend classification-based metric learning to learn a disentangled embedding space. We begin from the disentangled distance metric, which is

---

[1] In proxy-triplet loss, this type of numerical problem does not occur because they are relative comparison based losses. In proxy-NCA or proxy-Softmax loss, some of the previous works encountered similar problem, and solved the problem by applying a smoothing factor over the similarity score [4, 16, 21]. We also tested applying a smoothing factor, but for our multi-label classification problem, it turns out that the proposed modified distance metric is more effective.

[2] In our preliminary experiments, we found that removing the bias term does not decrease the model performance, so we did not include it in the Equation 8.

$$D(f(x_i) \circ m_s, c_{y_z} \circ m_s) = \frac{f(x_i) \circ m_s}{||f(x_i) \circ m_s||} \cdot (c_{y_z} \circ m_s)$$
$$= \frac{1}{||f(x_i) \circ m_s||} \cdot (f(x_i) \circ m_s) \cdot (m_s \circ c_{y_z}). \quad (9)$$

From the above equation, if we split $f(x_i)$ into the nonlinear function $f_{n-1}(x_i)$ and the embedding feature layer h (here, h layer includes nonlinear activation), then the equation becomes

$$= \frac{1}{||f(x_i) \circ m_s||} \cdot (f_{n-1}(x_i) \cdot h \circ m_s) \cdot (m_s \circ c_{y_z})$$
$$= \frac{1}{||f(x_i) \circ m_s||} \cdot f_{n-1}(x_i) \cdot h \circ m_s \cdot m_s \circ c_{y_z}. \quad (10)$$

In this equation, $(h \circ m_s \cdot m_s \circ)$ is actually a sub-dense layer that has the same dimensionality as the disjoint mask m_s, which is applied when $y_z \in s$. Henceforth, we denote the sub-dense layer h_s. Now, $||f(x_i) \circ m_s||$ can be replaced to $||f_{n-1}(x_i) \cdot h_s||$. Finally, the disentangled distance metric becomes

$$= \frac{1}{||f_{n-1}(x_i) \cdot h_s||} \cdot (f_{n-1}(x_i) \cdot h_s) \cdot c_{y_z}. \quad (11)$$

This is the same formula for multi-task learning in the multi-label classification problem formulation, surprisingly, proving a previously unknown link between the two concepts. We illustrate this disentangled classification-based model in Figure 2 (c). Through experimental evaluation, we further verify that this multi-task learning-based classification model is equivalent to the disentangled proxy-based model while being much simpler to implement and benchmark.

4. EXPERIMENTS

4.1 Dataset and Input Features

For our experiments, we use the Million Song Dataset (MSD) [22] and Last.FM tag annotations associated with MSD tracks, which have been previously grouped into different categories [23], resulting in 28 genre tags, 12 mood tags, 5 instrument tags, and 5 era tags. We treat each category as a similarity notion s. We use these tags for evaluating similarity-based retrieval, auto-tagging, and triplet prediction tasks. The data are split into 201680, 11774, and 28435 samples for the train, validation, and test sets, respectively, following a previous auto-tagging benchmark [24]. For triplet prediction evaluation, we follow the same procedure as in [7], albeit switch one similarity notion (era replaces tempo) to match auto-tagging benchmarks. We sample 40,000 triplets per each similarity notion (genre, mood, instruments, era, track) and use a cleaned version of the *dim-sim* dataset to evaluate the models on human-annotated triplets.

The input to the embedding function $f(\cdot)$ is 3-second excerpts represented as a log-scaled mel-spectrogram S, extracted with librosa [25]. We use a window size of 23 ms with 50% overlap and 128 mel-bands, resulting in input dimensions of 129×128 as in [7]. The input features

are z-scored standardized using fixed mean and standard deviation values of 0.2 and 0.25, respectively.

4.2 Backbone Model and Training Parameters

For the embedding function or backbone model $f(\cdot)$, we use the same architecture as described in [7], which is an Inception-based model [26]. The model is comprised of a convolution layer with 5×5 sized 64 filters followed by 2×2 strided max-pooling, followed by six Inception blocks. Each Inception block consist of two Inception modules, a *naïve* module and *dimension reduction* module, which are applied in sequence. Both of the modules include filters of mixed size, but the *naïve* module has 2×2 strides in the last convolution layers of the module, so that the spatial feature map is reduced, and the *dimension reduction* module has a fixed number of filters in the last convolution layers of the module, so that the feature map is fixed to 256 in the intermediate layers. At the end, one fully connected layer with 256 units is added, except for the disentangled (multi-task learning) classification-based model, which uses sub-dense layers instead of a single fully connected layer. We use ReLU nonlinearities for all layers.

Since our embedding dimensionality is 256 and we consider four music similarity notions (genre, mood, instruments, era), each has a disjoint subspace of size 64. For the disentangled (multi-task learning) classification-based model, the sub-dense layers are also 64 units each. We use the Adam optimizer [27] for training. We initialize the learning rate to 0.005 and reduce it by a factor of 5 when the validation loss does not decrease for 10 epochs, up to 5 times, after which we apply early stopping. The margin for the triplet-based models is set to 0.1.

4.3 Evaluation Tasks

Our learned representations can be utilized for many applications, so there are many aspects to consider when evaluating representation learning models. Therefore, as a unified evaluation framework, we evaluate the models on four tasks: training time, similarity-based retrieval, auto-tagging, and triplet prediction.

4.3.1 Training Time

We first measure the overall training time to see the efficiency of the representation learning model. The training time is calculated as the total number of epochs multiplied by the time consumption of 1 epoch. Then, we report the value as a ratio to the shortest training time.

4.3.2 Similarity-based Retrieval

For the similarity-based retrieval evaluation, we use the recall@K (R@K) metric to measure retrieval quality following the standard evaluation setting in image retrieval [4, 14–16, 30]. This metric is useful for evaluating a search system because it measures the quality of the top K retrieved results, which are more important than long-tail retrieved results. The definition of the standard recall@K that is used for single-label problems is as follows. A query song is used to search a test set of recordings and retrieve

Models	Normalization	Disentanglement	Training time ratio	Similarity-based retrieval				Auto-tagging AUC
				R@1	R@2	R@4	R@8	
Triplet	✓	✗	1.87	31.8	45.2	59.9	73.0	0.815
Triplet	✓	✓	2.37	36.5	50.5	64.1	76.0	0.825
Triplet + track reg.	✓	✓	3.05	33.9	47.5	61.9	74.3	0.813
Proxy	✓	✗	1.11	**45.0**	**58.5**	**71.0**	80.9	**0.890**
Proxy	✓	✓	1.29	44.7	58.2	70.7	80.6	**0.890**
Classification	✗	✗	**1.00**	6.1	11.5	21.1	35.9	0.887
Classification	✓	✗	**1.00**	43.8	57.8	70.3	80.3	0.887
Classification	✓	✓	1.27	44.7	58.4	70.7	**80.9**	**0.890**

Table 1. Results for training time, similarity search, and auto-tagging.

Model	AUC
CRNN [23]	0.850
Self-attention [28]	0.881
Sample-level ReSE-2 [29]	0.885
Multi-level & multi-scale [24]	0.888
Proposed Model	**0.890**

Table 2. Auto-tagging SOTA comparison.

similar sounding results. If one of the top K retrieved results has the same class label as the query song, the recall@K is set to 1, otherwise it is set to 0. This process is repeated for all samples in the test set and then averaged.

Our data is multi-labeled, however, so we adapt the standard single-label (multi-class) R@K metric to create a multi-label variant. Our definition is

$$R@K = \frac{1}{N} \sum_{q=1}^{N} \frac{n(y^q \cap (\cup_{i=1}^{K} y^i))}{n(y^q)}, \qquad (12)$$

where N is the number of test samples, y^q is the ground truth labels of a query, and y^i is the ground truth labels of the top K retrieved results. And, $n(\cdot)$ denotes the number of the elements of a set. In this setup, if the set of labels of the top K retrieved results contains all the multiple labels of the query song, the recall@K is set to 1, otherwise it is set to the correct answer ratio. We report R@K when K is 1, 2, 4, and 8.

4.3.3 Auto-tagging

Music auto-tagging has been extensively studied in the literature with diverse model architectures [3]. As such, we follow standard benchmarking and evaluation criteria, and report area under the receiver-operator curve (AUC) to measure tag-based retrieval performance.

Unlike the proxy-based and classification-based approaches, the triplet-based model doesn't directly predict a class (or several classes) for a given input. Thus, we use the concept of prototypes to obtain classification result from the triplet-based models [31]. We first average all the embedding features of the training samples that are assigned to each class label to construct prototype (or centroid) of each class label. Then, we measure a distance between these prototypes and embedding feature of each sample and regard it as a prediction score for classification, which itself is directly used for AUC evaluation.

4.3.4 Triplet Prediction

Triplet prediction score is simply measured by counting the number of correct predictions among all test triplets. Here,

it is regarded as correct if the distance between the embedding features of the anchor and the positive is smaller than that of the distance between the anchor and the negative.

5. RESULTS

In Table 1, we present the results for training time, similarity-based retrieval, and auto-tagging. We compare a total of eight models, which are categorized into three learning methods: triplet-based, proxy-based, and classification-based models. "Disentanglement" indicates whether a CSN masking function is applied to each learning method, and "Normalization" indicates whether a normalization layer is applied to the model's embedding layer. "Track regularization" (track reg.) indicates whether, in addition to tag-based triplets, we also sample triplets by taking the anchor and positive from the same track and the negative from a different track, as proposed in [7].

First, we see that the training time, represented as the ratio between each model's training time and the training time of the fastest approach, is significantly reduced for the proxy-based and classification-based models compared to the triplet-based models. This is because each training sample for the triplet model is actually composed of 3 inputs (anchor, positive and negative) or even 5 when track regularization is also applied, whereas the proxy-based and classification-based approaches only require one input per training sample.

Second, for similarity-based retrieval, we see that the vanilla classification model without a normalization layer exhibits poor performance. This confirms our conjecture that using the representation learned by the classification model without normalization layer directly is not optimal for similarity-based retrieval, as the model is not optimized based on distances in the learned embedding space. We also see that the proxy- and classification-based models are superior to the triplet-based models across the board. We hypothesize that this is due to the latter strategy using only a single label per training sample, whereas the former two use all (multi-)labels for each training sample, thus exploiting a richer signal during training.

Third, for auto-tagging, we see that the proxy-based and classification-based models outperform the triplet-based model by a large margin. As expected, the vanilla classification-based model performs well on this task. In Table 2, we compare our proposed classification-based disentangled model to the state of the art (SOTA) for music auto-tagging. Our model outperforms all baselines, setting

Embedding space	Models	Normalization	Disentanglement	Genre	Mood	Instruments	Era	Overall
Complete space	Triplet	✓	✗	0.771	0.725	0.653	0.701	0.712
	Triplet	✓	✓	0.762	0.744	0.696	0.733	0.733
	Triplet + track reg.	✓	✓	0.757	0.733	0.673	0.715	0.720
	Proxy	✓	✗	0.774	0.742	0.645	0.693	0.714
	Proxy	✓	✓	0.762	0.742	0.660	0.716	0.720
	Classification	✗	✗	0.783	0.745	0.659	0.723	0.728
	Classification	✓	✗	0.776	0.747	0.647	0.704	0.719
	Classification	✓	✓	0.758	0.742	0.659	0.715	0.719
Sub-space	Triplet	✓	✓	**0.790**	**0.785**	**0.798**	**0.797**	**0.792**
	Triplet track reg.	✓	✓	0.775	0.748	0.743	0.742	0.752
	Proxy	✓	✓	0.777	0.740	0.734	0.700	0.738
	Classification	✓	✓	0.775	0.739	0.732	0.701	0.737

Table 3. Results on tag-based triplets.

Models	Normalization	Disentanglement	Track	Human-labeled
Triplet	✓	✗	0.957	0.820
Triplet	✓	✓	0.964	0.820
Triplet + track reg.	✓	✓	0.961	**0.852**
Proxy	✓	✗	0.978	0.784
Proxy	✓	✓	0.978	0.791
Classification	✗	✗	0.978	0.780
Classification	✓	✗	0.978	0.795
Classification	✓	✓	**0.984**	0.801

Table 4. Results on track-based & human-labeled triplets.
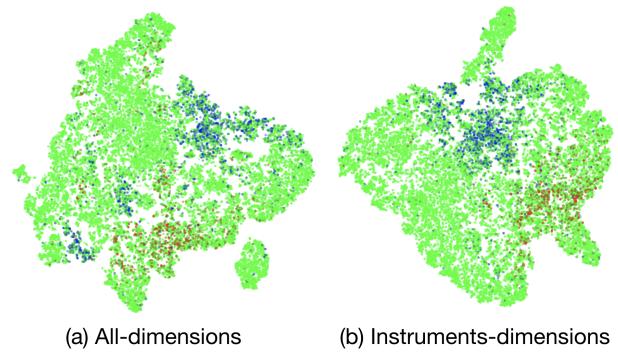
the new state-of-the-art for music auto-tagging.

Fourth, for triplet prediction, we report tag-based triplet results in Table 3 using different similarity dimensions (genre, mood, instruments, era), and in Table 4 the results for track-based and human-labeled triplets. The "Embedding space" column indicates whether we use the complete embedding space to measure the similarity between pairs of examples, or whether we only use the disjoint sub-space $(f(x_i) \cdot m_s$ or $h_s)$ corresponding to the similarity notion s used to sample the test triplets (genre, mood, instruments or era). In Table 4 we use the complete space.

Fifth, in Table 3 we see that while proxy- and classification-based embeddings are superior for music retrieval and tagging, triplet-based embeddings perform better (unsurprisingly) on the triplet-prediction task. It is noteworthy that while the triplet task is often used as a proxy for evaluating music similarity modelling, models that do best on this task are not necessarily the best at downstream retrieval tasks as evidenced by Table 1. In Table 4, we also see that while classification-based embeddings perform better at predicting track-based triplet similarity, triplet-based embeddings perform better when it comes to matching human judgements of triplet similarity. This is particularly true when we apply triplet learning with track regularization, in accordance with previous work [7].

6. VISUALIZATION OF DISENTANGLED SPACE

To qualitatively evaluate the disentangled representation space learned by our model, we visualize the embeddings of the test set as a t-SNE plot [32] in Figure 3. We take embeddings from the disentangled triplet model and highlight samples with the *female vocalists* and *instrumental* tags as an example. While the highlighted samples are relatively dispersed when considering all dimensions, we see that they are nicely clustered together when only con-

(a) All-dimensions   (b) Instruments-dimensions

Figure 3. t-SNE plot of test set embedding features. The blue dots are labeled positive for the female vocalists tag, the red dots are labeled positive for the instrumental tag, and the green dots are negative.

sidering the instrument sub-space of the embedding. This illustrates the benefits of a disentangled space, which supports both global similarity and specialized similarity over specific music dimensions.

7. CONCLUSION

In this paper, we presented a detailed study of metric-based and classification-based learning approaches for music representation learning. We extended both strategies to learn disentangled spaces from multi-label data, and showed both analytically and empirically that under certain conditions, proxy-based learning is equivalent to classification-based learning. We benchmark multiple variants of each strategy in terms of training efficiency and performance on music retrieval, auto-tagging, and triplet prediction tasks. Our results show that, when coupled with disentanglement and normalization, classification-based representation learning produces superior benchmark results on all tasks, except for triplet prediction where triplet models are (predictably) strong performers, indicating that triplet prediction is not necessarily a reliable proxy for real-world retrieval performance. Our best performing disentangled model obtains state-of-the-art results for music auto-tagging, outperforming all previous baselines. Finally, we complement our quantitative analysis with qualitative results that further illustrate the benefits of learning a disentangled music embedding space.

8. REFERENCES

[1] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, "Representation learning of music using artist labels," in *ISMIR*, 2018.

[2] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks." ISMIR, 2017.

[3] J. Nam, K. Choi, J. Lee, S.-Y. Chou, and Y.-H. Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *Signal Processing Magazine*, vol. 36, no. 1, 2018.

[4] A. Zhai and H.-Y. Wu, "Classification is a strong baseline for deep metric learning," *BMVC*, 2019.

[5] S. Reed, K. Sohn, Y. Zhang, and H. Lee, "Learning to disentangle factors of variation with manifold interaction," in *ICML*, 2014.

[6] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *NeurIPS*, 2016.

[7] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled multidimensional metric learning for music similarity," in *ICASSP*. IEEE, 2020.

[8] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, vol. 1. IEEE, 2005.

[9] R. Hadsell, s. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *CVPR*, vol. 2. IEEE, 2006.

[10] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *NeurIPS*, 2004.

[11] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, 2009.

[12] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*. IEEE, 2015.

[13] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *Int. Workshop on Similarity-Based Pattern Rec.* Springer, 2015.

[14] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *ICCV*. IEEE, 2017.

[15] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *ICCV*. IEEE, 2017.

[16] Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin, "Softtriple loss: Deep metric learning without triplet sampling," in *ICCV*. IEEE, 2019.

[17] K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," *arXiv preprint arXiv:2003.08505*, 2020.

[18] K. Ridgeway and M. C. Mozer, "Learning deep disentangled embeddings with the f-statistic loss," in *NeurIPS*, 2018.

[19] A. Veit, S. Belongie, and T. Karaletsos, "Conditional similarity networks," in *CVPR*.

[20] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *NeurIPS*, 2005.

[21] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *CVPR*. IEEE, 2018.

[22] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *ISMIR*, 2011.

[23] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *ICASSP*. IEEE, 2017.

[24] J. Lee and J. Nam, "Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging," *SPL*, vol. 24, no. 8, 2017.

[25] B. McFee, C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *14th Python in Science Conf.*, 2015.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[28] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.

[29] T. Kim, J. Lee, and J. Nam, "Comparison and analysis of samplecnn architectures for audio classification," *JSTSP*, vol. 13, no. 2, 2019.

[30] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *CVPR*. IEEE, 2016.

[31] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NeurIPS*, 2017.

[32] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, 2008.

USER INSIGHTS ON DIVERSITY IN MUSIC RECOMMENDATION LISTS

Kyle Robinson[1] **Dan Brown**[1] **Markus Schedl**[2]

[1] David R. Cheriton School of Computer Science, University of Waterloo, Canada
[2] Institute of Computational Perception, Johannes Kepler University Linz, Austria

`kyle.robinson@uwaterloo.ca, dan.brown@uwaterloo.ca, markus.schedl@jku.at`

ABSTRACT

While many researchers have proposed various ways of quantifying recommendation list diversity, these approaches have had little input from users on their own perceptions and preferences in seeking diversity. Through an exploratory user study, we provide a better understanding of how users view the concept of diversity in music recommendations, and how they might optimise levels of intra-list diversity themselves. In our study, 17 participants interacted with and rated the suggestions from two different recommendation systems. One provided static top-7 collaborative filtering recommendations, and the other provided an interactive slider to re-rank these recommendations based on a continuous diversity scale. We also asked participants a series of free-form questions on music discovery and diversity in semi-structured interviews. User-preferred levels of diversity varied widely both within and between subjects. Although most users agreed that diversity is beneficial in music discovery, they also noted a risk of dissatisfaction from too much diversity. A key finding is that preference for diversification was often linked to user mood. Participants also expressed a clear distinction between diversity within existing preferences, and outside of existing preferences. These ideas of inner and outer diversity are not well defined within the bounds of current diversity metrics, and we discuss their implications.

1. INTRODUCTION

As music consumption has moved from physical media to digital collections to streaming, people have changed the way they discover new music. As with other forms of consumption which have made the shift to digital media and marketplaces such as movies, television, and consumer products, data on music listening habits is more prevalent than ever. Accordingly, systems which use this data to market or recommend new content to users have become ubiquitous. These *music recommender systems* aim to provide satisfying music recommendations to users across a wide variety of contexts [23].

One common way of recommending music is to create a ranked list where the items are formed by the top-*n* recommendations, as produced by the used recommendation algorithm, sorted by recommendation relevance. To judge the quality of recommendations, various forms of accuracy metrics have been proposed. Typically borrowed from the field of *information retrieval*, these accuracy measurements aim to quantify how well a recommendation (or set of recommendations) aligns with a user's known preferences, or in some cases how satisfied a user will be with those recommendations [10].

In addition to accuracy, various other metrics have been proposed [10, 12]. These aptly named *beyond-accuracy* metrics include novelty, coverage, serendipity, and diversity [10]. Novelty relates to items which are unknown to the user, coverage relates to the proportion of items that can be recommended (item coverage) or to the proportion of users for which at least one recommendation can be made (user coverage), serendipity relates to the unexpectedness of a recommendation, and diversity relates to the dissimilarity of recommended items [10]. We focus our attention to diversity as it is well researched, and easily understood for music [10, 12, 25].

Diversity in *music recommender systems* is well researched, but we are unaware of any research which specifically explores user provided perceptions of diversity. Additionally, most implementations of diversity treat the metric as a static variable between or within users. We argue that desire for diversity in recommendations may instead be situationally dependent so we present users with an interactive system which allows them to select diversity as a continuous trade off against accuracy across numerous personalized recommendation lists. Here, alongside a live user study using this prototype system, we present the results of semi-structured interview questions in order to address the following research questions:

- RQ1: How do users feel about diversity in personalized music recommendation lists?

- RQ2: How might users optimise their own level of diversity in personalized recommendation lists?

We found that users presented a range of definitions for diversity, linked ideal diversity levels to their mood, and distinguished between what we call inner and outer diversity. When asked to optimise their own level of diversity using our system selections differed greatly within and between subjects.

2. BACKGROUND & RELATED WORK

2.1 Diversity in Recommender Systems

Alongside novelty, coverage, and serendipity, diversity has long been identified as an important metric in providing satisfying automated recommendations to users across varying domains [12]. Diversity in this context traces back to information retrieval tasks, where it was used to resolve ambiguity in search queries [3]. Within recommender systems, diversity prevents over-personalization of recommendations to users, thereby increasing user satisfaction with recommendations [12, 13]. Research on diversity in recommender systems is extensive, and numerous different definitions have been proposed [13]. More generally, recommender system diversity has been described as the opposite of *similarity* [1, 10]. Among the most commonly researched and implemented definitions of diversity in *music recommender systems* is intra-list diversity (ILD) which measures the average pairwise dissimilarity of items using some chosen similarity metric; typically calculated using content features [1, 32].

2.2 Optimising for Diversity

Research on selecting optimal levels of diversity for recommender systems is extensive. In their original paper defining diversity as the opposite of similarity, Bradley and Smyth show that traditional recommender system outputs are not diverse, and diversity, in one metric, can be increased with minimal negative impact on accuracy [1]. Ziegler et al. further showed that user satisfaction with recommendation lists relies on more than accuracy by computing precision, recall, and satisfaction curves in a large user study [32]. Studies following this theme of incorporating existing diversity metrics with minimal negative impact on accuracy and/or satisfaction are plentiful [21, 31]. Whereas these works applied a global level of diversity to recommendations, recent work has focused on selecting levels of diversity on a per-user basis through user modeling [5, 8, 17, 18]. Interactive systems which allow users to explore recommendations through diversity have been explored outside of the music domain, but these systems aim to abstract diversity into a user interface rather than allow for user selection of existing diversity metrics [22, 27, 30].

Differences in user perceived diversity levels have been identified across varying recommendation algorithms [6], and varying levels of intra-list diversification [29]. Finally, user listening habits on diversity have been extracted from social networks [7] and playlists [20].

We are not aware of any research which explores user provided perceptions of diversity in personalized music recommendations, or allows them to directly modify existing diversity metrics on the fly. We begin to fill in this gap by providing knowledge on how well formalizations of diversity align with user perceptions of diversity.

3. METHODOLOGY

To control all aspects of recommendation and diversity inclusion, and to minimise restricting participants' consumption method, we implemented a collaborative filter recommender. We used Last.fm as a source of raw listening data, and presented song previews in the form of standardised 30 second track previews from Spotify.

3.1 Interactive Recommendation Lists

3.1.1 Data

We collected a total of 341,764,569 unique listening events (LEs) from 51,669 unique users whose region was set to North America using the Last.fm API. Users were found by crawling the Last.fm social graph using the *user.getFriends* endpoint. We had a limit of 10,000 LEs accepted per user, and only accepted LEs between January 12, 2019 and when we collected them in February 2020. The median number of LEs per user is 7744, 25th percentile is 3502, and 75thth percentile is 9842.

We used a simple key consisting of artist and track name tuples in order to identify individual tracks. The final user-track-interaction matrix, used to generate recommendations (see Section 3.1.2), contains 141,205,668 non-zero entries (play counts) across 12,300,857 unique artist-track tuples, resulting in a 51,669x12,300,857-sparse matrix. This system does not account for potentially inaccurate metadata obtained from Last.fm, but does account for the same track across different releases. Entries in this matrix are integers which correspond to the number of unique times a user (row) played the track (column). An anonymized version of this data is available upon request.

3.1.2 Collaborative Filtering & Diversity

For generating recommendations we used an Alternating Least Squares (ALS) matrix factorization algorithm which is designed specifically for implicit feedback data sets [8, 11]. This algorithm results in one vector for each user consisting of a non-negative real number (recommendation relevance) for each track in the database; higher numbers are considered more relevant recommendations. The ALS collaborative filter recommender was implemented using the *Implicit* python library [9], and was trained using the dataset described in Section 3.1.1. Hyperparameters were optimised using 5-fold cross-validation and Mean Average Precision for top-10 recommendations (MAP@10) over 60 iterations of randomized search resulting in 160 factors, 28 iterations, a scaling factor of $\alpha = 774$, and regularization term of $\lambda = 1$.

The trained collaborative filter recommender was used to generate top-400 track recommendation lists for a single Last.fm username (see Section 3.2). To facilitate multiple recommendation lists per-user we split this list evenly into four smaller lists of 100 tracks each. Each track within each of the four lists was assigned a rank from 1-100 with one being the most relevant. In order to measure diversity we used the latent vectors generated for each track during matrix factorization as descriptors. Similar to previous work [8, 29], we calculated a form of ILD (d_i) by summing the Euclidean distance of one track's descriptors (v_i) from all other descriptors (v_j) in each top-100 list.

$$d_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} ||v_i - v_j|| \tag{1}$$

This calculation differs from previous work in that diversity is only calculated once and not as part of a greedy diversification algorithm. Higher values of d_i correspond to more diverse tracks in relation to others in the list. Tracks are assigned additional ranks from 1-100 where rank one is the most diverse. We are left with four unique top-100 recommendation lists for a given user where each track is assigned a rank for relevance (R_i), and diversity(D_i).

The final ranking (F_i) is calculated as a trade off between relevance and diversity controlled by a convex combination of both ranks, with a diversity parameter β.

$$F_i = (1 - \beta) * R_i + \beta * D_i \qquad (2)$$

The user interface, shown in Figure 1, displays the top-7 tracks of each top-100 recommendation list based on F_i in the form of 30 second previews using Spotify Play Button widgets. [1] We chose to use top-7 recommendation lists to ensure user study session times under 70 minutes (see Section 3.2). An interactive slider that controls the value of β is situated above the song previews. The left of this slider corresponds with $\beta = 0$ and the right side corresponds with $\beta = 1$ with a step size of 0.001. A *Well Known* button appears to the left of each song preview allowing users to remove songs which are not new to them.
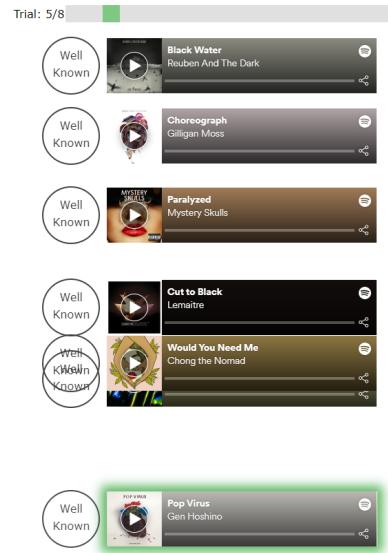
Due to differences in the music collection available on Spotify and our own music database, as well as to avoid false-positives in retrieving song previews, we omitted all songs which did not match exact artist and song string queries to Spotify. This typically resulted in final recommendation lists of 95-100 tracks each.

3.2 User Study

Participants were recruited on the University of Waterloo campus through internal email lists and posters. After completing a digital information consent form participants were asked to complete a brief survey. As part of this survey they were asked to provide their Last.fm usernames, or alternatively were provided instructions on how to set up a Last.fm account and record their listening events to it. We required that participants had a minimum of 5 hours of LEs recorded before continuing to the interactive portion.

The interactive portion of the study involved a pre-interaction interview, two conditions of 4 trials using four unique recommendation lists, and a post-interaction interview. Interviews were semi-structured. Pre-interaction interview questions focused on the importance of music discovery to the participant, how the participant finds new music, and what a diverse list of personalized recommendations means to them. Post-interaction interview questions focused on the perceived effect of the slider on recommendations, the static or variable nature of their selections across trials, and positives and negatives of diversity in music recommendations.

Trials 1-4 consisted of static top-7/100 recommendation lists each corresponding with one evenly split quarter of

[1] https://developer.spotify.com/documentation/widgets/generate/play-button/

Figure 1. The mid-motion user interface state directly after moving the diversity slider seen at the top. The top 7/100 songs as ranked by Equation (2) are displayed as 30 second song previews. As the slider is moved the songs shift from the old order to the new order over a period of 2 seconds. Songs which leave the top 7 move off the bottom and new songs appear from the bottom highlighted in green for 5 seconds. The circular *Well Known* buttons on the left remove songs from the list entirely.

their top-400 recommendations as ranked only by recommender output (relevance) (see Section 3.1.2). The user-interface was similar to Figure 1 but without the slider. Participants were asked to listen to each preview, remove well known tracks, mark if they were familiar with the artist, and rate the recommendation on a four-point Likert scale of 'Strongly Dislike', 'Dislike', 'Like', or 'Strongly Like'. Only once every track was rated could the participant move to the next trial.

Trials 5-8 consisted of the same ranked lists as trials 1-4 (minus tracks marked as well-known) with the addition of the interactive slider to re-rank the larger hidden list based on the participants' selected level of diversity (see Section 3.1.2). The user-interface can be seen in Figure 1. Participants were not told what the slider did and were instructed to find the position on the slider that resulted in the most satisfying recommendation list as a whole while removing tracks that were well known to them. Once the participant locked in this position they were again asked to mark if they were familiar with each song's artist, and rate each individual recommendation on the same four-point Likert scale before moving to the next trial.

Between each trial participants completed a survey with questions on their satisfaction with the final recommendation list, the level of diversity in the recommendation list, and how well the recommendation list portrayed the definition of diversity they provided in their pre-interview survey. Participants were paid $10 CAD upon completing the interactive portion of the study.

Pre- and post-interaction interviews were transcribed, and comments were then sorted into three categories: in-

teraction, music discovery, and diversity. Similar to other qualitative music consumption studies we extracted individual ideas as statements from transcriptions and proceeded to build connections and groupings through affinity diagramming [2, 19]. Main ideas were highlighted and categorized into groupings of similar themes, and finally counts of each theme were collected. We specifically focused on responses regarding diversity.

4. RESULTS

We recruited 18 participants, and removed one participant for marking all recommendations as *Well Known*, leaving 17 total participants. The median participant age was 23; the oldest was 29 and the youngest 19. Each user session took 50-70 minutes inclusive of interviews. Some sessions were completed face to face, and others involved the users connecting remotely to the interactive system.

4.1 Music Discovery
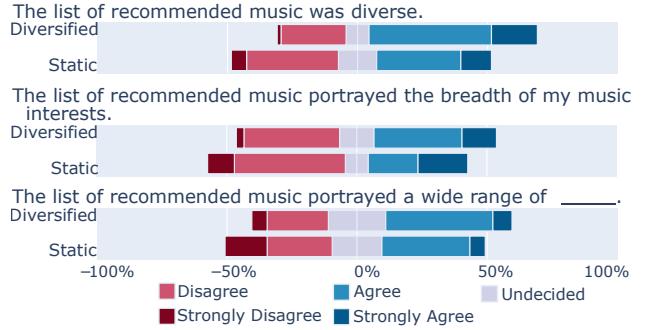
When asked how they discovered new music, 9 said they used Spotify, 9 used YouTube, 5 used movies and/or television, 4 relied on friends, 3 used radio, and 4 used some other online service such as Amazon or Soundcloud. The importance and frequency of finding new music varied significantly from user to user, and no clear patterns were observed. Some users noted that the primary reason they use music services such as Spotify is to enable easier music discovery. When asked how important finding new music is to them, one user reported previously spending 5 hours per week looking for new music, but added:

> "While it's still very important to me, I basically don't do it very often on my own anymore; I rely on Spotify to do almost all of it for me."

4.2 Recommendations

None of the participants had an existing Last.fm account, and the length of time during which users recorded their listening histories to Last.fm varied from one to three weeks. The median percentage of user LEs which existed in our CF database was 95%, with a max of 100% and a min of 65%. Median LE counts per-user used for recommendation generation were 256, with a max of 1156 and min of 86. All users marked and removed fewer than 100 tracks as well known across all trials, with the exception of one user who marked and removed 208.

When asked to rate individual recommendations on a 4-point Likert scale (Strongly Dislike, Dislike, Like, Strongly Like) 72.69% of songs were rated as 'Like' or 'Strongly Like' after locking in the diversity slider, and 74.79% in static lists. In addition to rating individual songs, participants were asked if they were satisfied with the list of recommended music for every trial. On a 5-point Likert scale (Strongly Disagree, Disagree, Undecided, Agree, Strongly Agree) 75% of diversified recommendation lists resulted in a positive response, with 50% for static recommendation lists.

Figure 2. Responses to Likert questions completed after every recommendation list, split between static lists and lists which were selected using the diversity slider. The final question was customized for each individual using their own definition of diversity obtained during the pre-interaction interview (see Section 4.4).

4.3 Interactive Diversity

In addition to the task of selecting an optimal position for the diversity slider, participants were asked a series of questions on how diverse they felt each recommendation list was. Responses to these questions can be seen in Figure 2. In order to visualise how participant responses on diversity align with their diversity selections, Figure 3 shows all 17 user's diversity selections coded with their Likert response on diversity. User selections varied greatly between their own recommendation lists and between other users'. Likert responses for perceived diversity did not fall in line with levels of β.
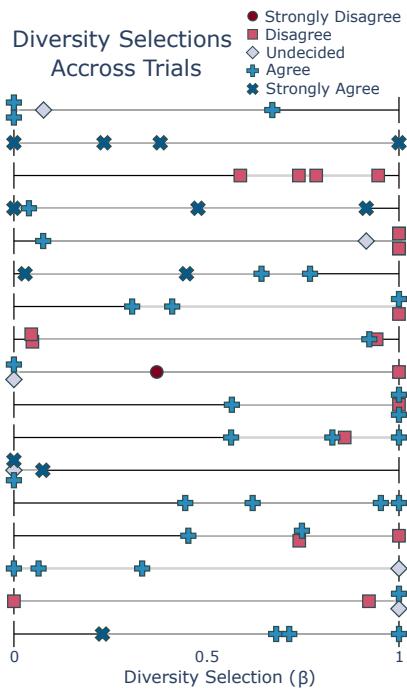
As a part of the post-interaction interview participants were asked to identify what they thought the slider was changing within their recommendation lists. Of the 17 participants, 5 identified it to increase diversity directly, 3 identified some change in genres, and 4 had no explanation. The remaining participants identified the slider to change the perceived gender of vocalists, increase 'newness', increase distaste, increase quality, and decrease quality. In one case where a participant identified the slider to effect genre they stated:

> "I noticed initially that the first side of the slider was giving me a bunch of songs from different genres. The more I was sliding it the more it was giving me the songs. . . from the genre which I like."

In another case where a participant was unable to identify the effect of the slider and was asked what they would like the slider to do they answered:

> "The way I imagined it was. . . less diverse on the one side and more and the other side. That's something I could definitely use."

When asked about their experience using the system some users expressed difficulty in remembering which locations of the slider they preferred most, and frustration over which songs remained on the list and which were moved off. In total, 10 users preferred interacting with the static list, and 7 preferred using the interactive slider.

Figure 3. All user selections for diversity using the slider found in Figure 1. The legend corresponds to Likert responses to: "The list of recommended music was diverse."

4.4 User Perceptions of Diversity

During the pre-interaction interview participants were asked what they would mean if they were looking for diverse recommendations. In addition to their open ended responses, they were challenged to come up with a single word or idea that could be used in place of diversity. Of the responses to this question, 13 answered a difference in **genres**, 2 answered **cultural differences**, and the remaining participants responded with **originality**, **variety**, and differences in **artists**. [2] These definitions were used to complete the third question in Figure 2.

Coding of participants' comments on diversity in their own personalized music recommendations resulted in two primary themes which we labeled **diversity meaning**, and **listener mood**. Comments which we classified under **diversity meaning** are deeply intertwined with personal definitions of diversity, and can be more specifically categorized into what we identify as **inner and outer diversity**; that is music within the bounds of existing preferences, and music outside of these bounds. In answering the interview question on the meaning of diverse recommendations, 8 participants made reference to a preference for this idea of inner or outer diversity. Participant comments expressing a preference for inner diversity include:

> "Diverse in the–within the boundaries of the things that I like."

> "I like a playlist which recommends me songs on the genre I like. . . the important thing is to get diversified music in my genre only. . . to stay in the same genre but diversity in artists."

[2] One participant was unable to choose between genre and culture.

> "A diverse music recommendation I think should still be within the category of music that I usually listen to, but it should be different artists or different albums that I haven't listened to so far."

Comments expressing a preference for outer diversity include:

> "[Diverse recommendations are] something new, something exciting. Something that I'm not used to, like I've never heard before."

> "[Diverse recommendations] would be music from other genres that maybe I haven't listened to very much, but still somewhat akin to the ones that I have listened to."

Secondary in frequency of occurrence to diversity meaning, **mood** was explicitly mentioned by 7 participants. Only 2 participants mentioned context. Participants referenced mood as a primary factor in how much diversity they want in their music recommendations at any given time. Notable comments on mood include:

> ". . . depending on my mood–whether I'm looking for more of the same things that I already like–I could set that slider to show me less diverse music–if I'm in the mood."

> "[I] like a piece of music right now because of the mood that I am in, but I might not like it while I'm listening to a very different kind of music. So diversity is good but I think in a weird way the recommender system should know when to recommend it."

> "Sometimes you're in the mood of listening to one specific–like you don't want [a] diverse playlist. You just want to listen to sad songs. You just want a playlist that has a sad song. You don't want diversity"

> "If you're in a melancholic mood and then you don't have a very diverse playlist of melancholic music then you'd be happy about your music because that's your mood."

Participants also provided their thoughts on the positives and negatives of diversity in personalized recommendations, and a summary of these thoughts can be found in Table 1. Participants generally felt that while diversity could enable music discovery, it also increased the risk of disliking some recommendations.

5. DISCUSSION

In this study we provided a primary analysis of user perceptions on diversity in personalized music recommendations. We also provided users an opportunity to directly optimise a diversity metric which until now had been algorithmically optimised for them. Although our results do not hold statistical power due to the small sample size, our semi-structured interviews facilitated valuable insights and answers to our posed research questions. These insights add to the growing number of other qualitative works in Music Information Retrieval research [2, 14–16, 19].

	More Diversity	Less Diversity
Pos.	Music Discovery (N=11) Preference Discovery (N=4) Interesting (N=4)	Likely to Like (N=2)
Neg.	Likely to Dislike (N=8) Dissatisfaction/Annoyance (N=2)	Restrictive (N=4) Repetitive (N=2) High Risk/Reward (N=2) Unremarkable (N=2)

Table 1. Positives and negatives of more and less diversity in recommendation lists expressed by participants.

5.1 RQ1: How do users feel about diversity in personalized music recommendation lists?

Despite a large variance in user's feelings towards diversity in music recommendations, their ideas on its positives and negatives (Table 1) mostly align with the metric's purpose of reducing over-personalization. Beyond this, however, users attached more complex ideas such as personal preference discovery and interestingness to more diversity. Ideas such as this may in part explain the higher levels of satisfaction reported by users given more diverse recommendations.

The prevalence of mood in participants descriptions of diversity is especially notable when compared to the lack of references to their context. As more focus is directed towards context-aware recommender systems [24], careful attention should be paid to not assume that ideal diversity levels can be determined by context alone. Diversity optimisation may also serve as an ideal jumping off point for mood-based recommendation [4,26]. In designing systems which incorporate diversity, it is also important to note that preferred diversity levels may not remain static on an individual user basis.

Although most participants described diversity as a difference in genres, genre was not the exclusive answer. To some participants, a recommendation list which spans genre may not be considered diverse unless those genres span a range of cultures, and to other users a recommendation list which spans artists in just one genre may be considered diverse.

The occurrence of inner and outer diversity–that is diversity within the bounds of existing preference, and outside of those bounds–was an unexpectedly binary result, and neither of these ideas are well defined by existing beyond-accuracy metrics. Inner diversity is not well described as novelty, nor is outer diversity well described by serendipity. The idea of inner diversity does however align with idea of user genre coverage [28]. More research on the universality of inner and outer diversity preference is clearly required.

In their foundational paper on diversity in *information retrieval*, Clarke et al. use a query for 'jaguar' as an example to show the usefulness of diversity; a diverse response might include the cars, the cats, and the classic Fender guitar [3]. In the case of music recommendations, all diverse responses may be simultaneously correct to one user, and incorrect to another.

5.2 RQ2: How might users optimise their own level of diversity in personalized recommendation lists?

The interactive system we implemented (Figure 1) represents a first attempt in allowing users to optimise diversity metrics in line with how they are optimised in existing studies. As such, all variables other than the level of diversity (Equation (2)) were fixed. We note that in allowing users to remove well-known songs the system represents a specific use for diversity in discovering novel music.

Diversity selections across the interactive trials varied widely within and between users. Ideally in Figure 1, users' Likert ratings would be distributed with positive responses on the right ($0.5 \leq \beta \leq 1$), and negative responses on the left ($0 \leq \beta \leq 0.5$). While results do not follow this distribution, the responses in Figure 2 show that users generally found the slider system to enable more diversity. We hypothesise a combination of three reasons for these results. First, the Likert survey provided no frame of reference for diversity and participants used their own idiosyncratic definitions. Second, the users' responses were heavily impacted by music previewed before locking in a diversity value. Third, the diversity metric did not match users' models of diversity. All three of these hypotheses should be considered for future implementations.

We also note that while our selection of CF recommender and diversity metric have a basis in previous work, there are countless combinations of them which may be used to comprise of a system such as ours. Also, more recent music recommendation algorithms based on deep neural networks could be investigated [24].

6. CONCLUSION & FUTURE WORK

The work we present here provides a much needed connection between quantitative diversity metrics and user perceptions of diversity in music recommendation lists. Through analysis of semi-structured interviews with 17 participants we identified two primary themes on user selections for diversity: listener mood, and diversity meaning. More specifically many users expressed a clear distinction between diversity within the bounds of their existing preferences, and diversity outside of these preferences. This inner and outer diversity was often expressed as a binary preference. Additionally, we found that when given the ability to select their own level of diversity in recommendation lists, user selections varied widely within and between subjects.

Much future work is required in order to generalize our qualitative findings to a larger population, and further inform *music recommender systems* from the user's perspective. Additionally, we plan to explore the connection between listener mood, and their preference for inner and outer diversity. Diversity in music recommendations should have at least as solid a foundation in user perception as in *information retrieval*.

7. REFERENCES

[1] K. Bradley and B. Smyth. Improving recommendation diversity. In *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*, pages 85–94. Citeseer, 2001.

[2] L. Chen, W. Wu, and L. He. How Personality Influences Users' Needs for Recommendation Diversity? In *Conference on Human Factors in Computing Systems - Proceedings*, volume 2013-April, pages 829–834. Association for Computing Machinery, apr 2013.

[3] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *ACM SIGIR 2008 - 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Proceedings*, pages 659–666, 2008.

[4] S. Deng, D. Wang, X. Li, and G. Xu. Exploring User Emotion in Microblogs for Music Recommendation. *Expert Systems with Applications*, 42(23):9284–9293, 2015.

[5] T. Di Noia, J. Rosati, P. Tomeo, and E. D. Sciascio. Adaptive multi-attribute diversity for recommender systems. *Information Sciences*, 382-383:234–253, mar 2017.

[6] M. D. Ekstrand, F. M. Harper, M. C. Willemsen, and J. A. Konstan. User perception of differences in recommender algorithms. In *RecSys 2014 - Proceedings of the 8th ACM Conference on Recommender Systems*, pages 161–168, New York, New York, USA, oct 2014. Association for Computing Machinery, Inc.

[7] K. Farrahi, M. Schedl, A. Vall, D. Hauger, and M. Tkalčič. Impact of listening behavior on music recommendation. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014*, pages 483–488, 2014.

[8] B. Ferwerda, M. Graus, A. Vall, M. Tkalčič, and M. Schedl. The Influence of Users' Personality Traits on Satisfaction and Attractiveness of Diversified Recommendation Lists. In *Proceedings of the 4th workshop on emotions and personality in personalized services (EMPIRE 2016)*, pages 43–47, Boston, USA, 2016.

[9] B. Frederickson. Fast Python Collaborative Filtering for Implicit Datasets. https://github.com/benfred/implicit, 2019.

[10] A. Gunawardana and G. Shani. Evaluating recommender systems. In *Recommender Systems Handbook, Second Edition*, pages 265–308. 2015.

[11] Y. Hu, C. Volinsky, and Y. Koren. Collaborative filtering for implicit feedback datasets. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 263–272, 2008.

[12] M. Kaminskas and D. Bridge. Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7(1):1–42, 2016.

[13] M. Kunaver and T. Požrl. Diversity in recommender systems – A survey. *Knowledge-Based Systems*, 123:154–162, 2017.

[14] J. H. Lee. How similar is too similar?: Exploring users perceptions of similarity in playlist evaluation. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*, pages 109–114, 2011.

[15] J. H. Lee and R. Price. Understanding users of commercial music services through personas: Design implications. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, pages 476–482, 2015.

[16] J. H. Lee, L. Pritchard, and C. Hubbles. Can we listen to it together?: Factors influencing reception of music recommendations and post-recommendation behavior. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pages 663–669, nov 2019.

[17] A. L'Huillier, S. Castagnos, and A. Boyer. Understanding usages by modeling diversity over time. In *CEUR Workshop Proceedings*, volume 1181, pages 81–86, 2014.

[18] F. Lu and N. Tintarev. A diversity adjusting strategy with personality for music recommendation. In *CEUR Workshop Proceedings*, volume 2225, pages 7–14, 2018.

[19] S. Y. Park, A. Laplante, J. H. Lee, and B. Kaneshiro. Tunes together: Perception and experience of collaborative playlists. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pages 723–730, 2019.

[20] L. Porcaro and E. Gomez. 20 Years of Playlists : a Statistical Analysis on Popularity and Diversity. *Proceedings of the 20th International Symposium on Music Information Retrieval, ISMIR 2019*, (July):4–11, 2019.

[21] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *RecSys'12 - Proceedings of the 6th ACM Conference on Recommender Systems*, pages 19–26, 2012.

[22] J. B. Schafer, J. A. Konstan, and J. Riedl. Meta-recommendation systems: User-controlled integration of diverse recommendations. In *International Conference on Information and Knowledge Management, Proceedings*, pages 43–51, 2002.

[23] M. Schedl, P. Knees, B. McFee, D. Bogdanov, and M. Kaminskas. Music recommender systems. In *Recommender systems handbook*, pages 453–492. Springer, 2015.

[24] M. Schedl, H. Zamani, C. W. Chen, Y. Deldjoo, and M. Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, jun 2018.

[25] M. Slaney and W. White. Measuring playlist diversity for recommendation systems. In *Proceedings of the ACM International Multimedia Conference and Exhibition*, pages 77–82, New York, New York, USA, 2006. ACM Press.

[26] M. Tkalčič, A. Košir, and J. Tasič. Affective recommender systems: The role of emotions in recommender systems. In *CEUR Workshop Proceedings*, volume 811, pages 9–13, 2011.

[27] C. H. Tsai and P. Brusilovsky. Leveraging interfaces to improve recommendation diversity. *UMAP 2017 - Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 65–70, 2017.

[28] S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *RecSys 2014 - Proceedings of the 8th ACM Conference on Recommender Systems*, pages 209–216, 2014.

[29] M. C. Willemsen, B. P. Knijnenburg, M. P. Graus, L. C. Velter-Bremmers, and K. F. Eindhoven. Using latent features diversification to reduce choice difficulty in recommendation lists. In *CEUR Workshop Proceedings*, volume 811, pages 14–20, 2011.

[30] D. Wong, S. Faridani, E. Bitton, B. Hartmann, and K. Goldberg. The Diversity Donut: Enabling participant control over the diversity of recommended responses. In *Conference on Human Factors in Computing Systems - Proceedings*, pages 1471–1476, 2011.

[31] Y. C. Zhang, D. Ó. Séaghdha, D. Quercia, and T. Jambor. Auralist: Introducing serendipity into music recommendation. In *WSDM 2012 - Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, pages 13–22, 2012.

[32] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web - WWW '05*, page 22, New York, New York, USA, 2005. Association for Computing Machinery (ACM).

POLYPHONIC PIANO TRANSCRIPTION USING AUTOREGRESSIVE MULTI-STATE NOTE MODEL

Taegyun Kwon[1]   **Dasaem Jeong**[1*]   **Juhan Nam**[1]

[1] Graduate School of Culture Technology, KAIST, South Korea

{ilcobo2, jdasam, juhan.nam}@kaist.ac.kr

ABSTRACT

Recent advances in polyphonic piano transcription have been made primarily by a deliberate design of neural network architectures that detect different note states such as onset or sustain and model the temporal evolution of the states. The majority of them, however, use separate neural networks for each note state, thereby optimizing multiple loss functions, and also they handle the temporal evolution of note states by abstract connections between the state-wise neural networks or using a post-processing module. In this paper, we propose a unified neural network architecture where multiple note states are predicted as a soft-max output with a single loss function and the temporal order is learned by an auto-regressive connection within the single neural network. This compact model allows to increase note states without architectural complexity. Using the MAESTRO dataset, we examine various combinations of multiple note states including on, onset, sustain, re-onset, offset, and off. We also show that the autoregressive module effectively learns inter-state dependency of notes. Finally, we show that our proposed model achieves performance comparable to state-of-the-arts with fewer parameters.

1. INTRODUCTION

Automatic music transcription (AMT) refers to an automated process that converts musical signals into a piano roll or a musical score. Polyphonic piano transcription is a specific AMT task for piano music. Due to the complex nature of piano sound such as overlapping spectra and interference among notes and inharmonic overtones, most of recent approaches are based on learning algorithms such as non-negative matrix factorization (NMF) and deep neural networks (DNN) [1]. In particular, the transcription performance has been significantly improved by virtue of the representation power of DNN [2–5] and large-scale piano music data such as the MAESTRO dataset [6].

A key element in designing state-of-the-art DNN architectures is detecting multiple states of a note beyond the conventional binary states (i.e., on/off) and modeling the temporal evolution of the note states. For example, the *Onsets and Frames* model incorporated a note onset detection network into the frame-level pitch detection network [4]. This onset-aware model significantly reduced note-level false positive errors, which is critical in perceptual evaluation of the transcription. Similar multi-state note modeling approaches are found in [4, 7–10] and some detect even more phases of note envelope including onset, sustain and offset [11, 12]. As such, various versions of note state representations have been suggested so far and showed improved performances. However, the majority of them use separate neural networks for each note state. This requires to optimize multiple loss functions, progressively increasing the model complexity for more note states. Also, they handle the temporal evolution of note states by an abstract connection between the hidden layers of the state-wise neural networks [4] or using a separate neural network to model piano-roll data [2, 13, 14].

In this paper, we propose a unified neural network architecture where individual neural networks for each state and the temporal order modeling are integrated within a single neural network. We implement the all-in-one architecture by predicting multiple states of a note as a softmax output and modeling the temporal order in an auto-regressive connection in the output layer. Specifically, the architecture is composed of convolution neural network (CNN) and recurrent neural network (RNN). For each time step, the CNN module summarizes local acoustic features into a frame-level latent vector. The RNN modules predicts the note states from the softmax outputs conditioned on the latent vector and the previous outputs via an auto-regressive connection. The multi-class classification approach for note states allows the model to increase the number of note states without architectural complexity. Taking the advantage of this property, we examine various combinations of multiple states including *on*, *onset*, *sustain*, *re-onset*, *offset*, and *off* by comparing the performances and visualizing an example of the note state activations. In addition, we show that the autoregressive module effectively learns inter-state dependency by comparing it to a non-auto-regressive version. Finally, we show that the auto-regressive multi-state note model can achieve transcription performance comparable to the state-of-the-art *Onsets and Frames* model on the MAESTRO dataset.

2. RELATED WORKS
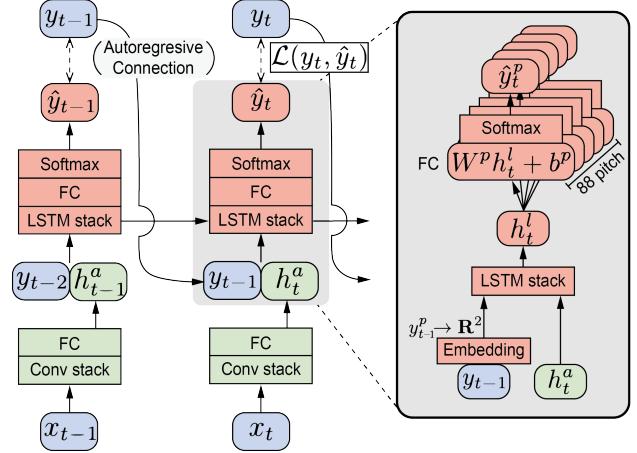
2.1 Multi-State Note Modeling

Most of recent approaches in polyphonic piano transcription are based on deep learning. The model architectures are diverse, including CNN [2, 3, 10, 12], RNN [9, 15], CRNN [4, 5, 16], and U-Net [17]. The loss function is typically the cross-entropy between predicted and ground truth labels but also includes the adversarial loss [5]. An important direction in designing a neural network architecture is detecting note onset explicitly apart from the binary on/off states [4, 9, 10, 12], considering that piano sound starts with a percussive tone but, after the attack park, it slowly decays with a harmonic tone [18]. This multi-state note modeling even including note offset was already explored before the DNN approaches become dominant [7, 8, 11]. While this multi-state note modeling has significantly improved the transcription performance, to the best of our knowledge, there has been no study that systemically compares various combinations of multi-state note representations. In this paper, we define five note states even considering the sustain pedal that globally changes the note states, and examine the different representations of temporal evolution.

2.2 Temporal Modeling of Multi-State Notes

Modeling the temporal order of note states is an essential step to improve the transcription performance [1]. A popular choice is hidden Markov model (HMM), which learns the temporal dependency of note states typically for each pitch. The note states can range from binary (on/off) [19–21] to more complete note phases (attack, decay, sustain, and release) [12]. Another approach is the autoregressive modeling which has been implemented with an RNN or its variants [2, 13, 14]. The autoregressive models can learn much wider musical context than HMM, covering inter-note and long-term dependency at the cost of sophisticated decoding algorithm [13]. Since this is analogous to the language model in speech recognition, it is also called musical language model (MLM). The MLMs are trained only with frame labels without paired audio data. This enables them to leverage large-scale symbolic data such as MIDI files. However, this decoupling from audio data may not take advantage of the synergy when the MLM is conditioned with the acoustic information, for example, using a transduction model [22, 23]. The *Onsets and Frames* model learns the temporal order of note states without an MLM by having a directed connection between different columns of neural networks that account for onsets and frames states, respectively [4]. However, this hidden-layer connection implicitly learns the temporal orders and the design choice is heuristic.

Our proposed architecture integrates the acoustic model with the MLM by conditioning the autoregressive multi-state note modeling on the acoustic latent features. Unlike the transduction models that use the pre-trained features or posterior as input [22, 23], we train the acoustic model and MLM jointly within a single neural network in an end-to-end manner. This unified approach was recently at-

Figure 1. A diagram of the proposed CRNN architecture at time step t.

tempted by the image-to-image translation model between mel-spectrogram and piano-roll [5]. However, our model has multiple states for each note and the state transitions are learned via the autoregressive connection.
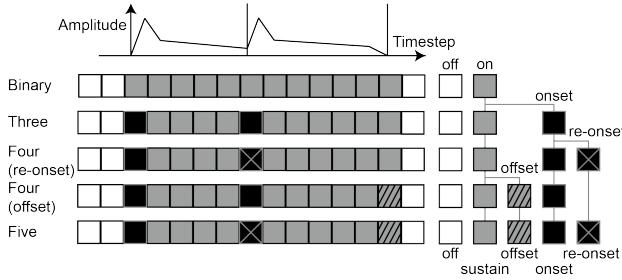
3. PROPOSED METHOD

3.1 Term Definitions

In this paper, we will use frame as an unit of time. t indicates the frame index. For a given audio recording, we express its audio feature as $\mathbb{X} = \{x_t\}$, where x_t is a vector that represents local audio feature at t, and $\mathbb{N} = \{n_i\}$, where n_i denotes a musical note with index i over the frames. Each note consists of onset, offset time and pitch. We also represent notes with a piano-roll-like form $\mathbb{Y} = \{y_t^p\}$, where y_t^p denotes a frame-level state of a note with pitch p at time step t. The pitch p corresponds to each key of piano. The actual form of y_t^p are determined by the chosen note state representation and network architecture. For example, the *Onsets and Frames* model uses three parallel binary rolls $\mathbb{Y}_{onset}$, $\mathbb{Y}_{frame}$, and $\mathbb{Y}_{offset}$, as they have three columns of networks for separate detection of onset, offset and frame. In our proposed model that uses a single-column network, y_t^p is a one-hot vector of multiple states. Without the pitch superscript, y_t denotes concatenated one-hot vectors (y_t^p) for all pitches (88 keys in piano) at time step t. In Section 3.3, we explain various combinations of multiple states of a note.

3.2 Model Overview

Our proposed network architecture consists of two modules as shown in Figure 1. The first module is a CNN-based acoustic model to extract local feature h_t^a from the input x_t. The second module is an RNN-based autoregressive MLM to estimate the output y^t from the previous output y^{t-1} but it is conditioned on the extracted audio feature h_t^a. The output layer have 88 independent softmax functions, each of which corresponds to pitch p. The following equations summarize the input and output in each module.

Figure 2. Visualization of five note state representations. The idealized ADSR curve of two consecutive notes and corresponding annotations are displayed.

$$h_t^a = CNN(x_t)$$
$$h_t^l = RNN(y_{t-1}, h_t^a) \qquad (1)$$
$$\hat{y}_t^p = softmax(W^p h_t^l + b^p)$$

For the acoustic model, we borrow the CNN architecture (*convnet*) proposed in [3], which is also used in [4,5]. The acoustic model consists of three convolutional layers followed by a fully-connected layer. For the autoregressive MLM, we employ two layers of uni-directional long short-term memory (LSTM). When the one-hot vector y_{t-1}^p at the previous time step $t-1$ is used as the input of the LSTM stack, it is embedded into a two-dimensional vector with continuous values to be matched with the audio feature h_t^a, another input of the LSTM stack. Since all of note states are predicted from the single CRNN architecture, our model has fewer parameters compare to the *Onsets and Frames* model. In addition, the causal uni-directional RNN in our model enables real-time applications.

3.3 Multiple Note States

We illustrate five different note state representations in Figure 2. Our main idea is to extend the conventional binary state (*onset* and *off*) to multiple states using the states transition of note activations such as note *onset* and note *offset*. Considering that sustain pedal affects the note transition, we also add *re-onset*, the moment that a new note is played while the previously played note on the same key is being sustained with the pedal. In addition to the transition states, we distinguished *sustain* from *on*. Using the multiple note states, we examine five types of note state representations: binary, three states (*off, onset, sustain*), four states which have additional {*re-onset* or *offset*} state, and five states which utilize all local states.

We define the note state representation over the softmax output in the LSTM in contrast to other multi-note-state models which have a separate binary classifier for each state [4, 12].

$$\mathcal{L}(y, \hat{y}) = -\sum_{t=0}^{T} \sum_{p=p_{min}}^{p_{max}} \sum_{i=0}^{\#states} y_t^p(i) log(\hat{y}_t^p(i)) \quad (2)$$

We expect two advantages from this multi-class approach. When the states are explicitly defined by a single variable,

the relation among states becomes concise. This may help the autoregressive model to learn note transition patterns more easily. Also, it prevents unrealistic combination of states which can be occur in inference. (i.e. $P(onset)=1$ but $P(on)=0$). Adding *re-onset* class independently, out of onset or sustain class, is also expected to be helpful to train the neural network due to the distinct percussive timbre of piano at note attack.

3.4 Autoregressive Model

For given a audio recording, our goal is estimate the notes $\mathbb{N}$ from the acoustic features $\mathbb{X}$. In practice, it is common to estimate the frame-level note states first $\mathbb{Y}$ by computing the maximum-likelihood of $P(\mathbb{Y}|\mathbb{X})$ and then to decode the estimated $\mathbb{Y}$ into $\mathbb{N}$. In the majority of AMT algorithms, the condition probability is factorized as follows (see [2] for details):

$$P(\mathbb{Y}|\mathbb{X}) \propto P(y_0|x_0) \prod_{t=1}^{T} P(y_t|y_0...y_{t-1})P(y_t|x_t) \quad (3)$$

where $P(y_t|x_t)$ corresponds to an acoustic model and $P(y_t|y_0...y_{t-1})$ accounts for a (musical) language model. The factorization allows the MLM to be trained with large-scale symbolic data such as MIDI without paired audio recordings. However, this approach has two issues. First, the factorization may not take advantage of the synergy when the MLM is conditioned with the acoustic information as input, for example, using a transduction model [22, 23]. Second, the frame-level MLM in Equation 3 is usually set up to learn the dependency of binary on or off states over piano-rolls [2, 13]. The recurring nature of the binary representation may lead the model to play a role of smoothing, rather than learning any kind of musical structure [2, 24]. While a note-level MLM (i.e, $P(n_i|n_{\leq i-1})$) can solve this problem [14,24] (and also this learns the distribution of notes more meaningfully as notes in AMT are analogous to words in speech recognition), it requires a separate beat and meter detection algorithm to convert the time unit [25].

Our proposed method addresses the two issues by 1) jointly training the acoustic model and MLM and 2) using multiple note states. Unlike the transduction models that use the pre-trained features or posterior as input [22, 23], we train the acoustic model and MLM in an end-to-end manner through the auto-regressive CRNN architecture. More precisely, we express the condition probability by conditioning the autogressive model on the acoustic input.

$$P(\mathbb{Y}|\mathbb{X}) \propto P(y_0|x_0) \prod_{t=1}^{T} P(y_t|y_{\leq t-1}, x_{\leq t}) \quad (4)$$

Although our model maintains the frame-level MLM, the note-aware multiple state representation may mitigate the repeated patterns of the simple binary representation.

3.5 Note Decoding

Once we estimates frame-level note states, we decode them into musical notes in the inference phase. We examine two ways of note decoding. One is a simple greedy approach and the other is a global optimization strategy using a modified beam search. The simple greedy decoding samples from the most probable estimated state of $\hat{y}_t$ every frame. However, it does not guarantee global optimum over multiple time steps. On the other hand, it is intractable to examine all possible sequences especially in the high-dimensional sequence. To overcome this problem, a beam search is usually used to obtain a global optimum. For MLM, a high-dimensional beam search [22] or a hashed beam search [2] was proposed to reduce the complexity in high dimensional situation. However, their methods mainly aim to capture dependency across pitches on the binary piano-roll representation. To focus on dependency over the multiple states, we simplified the beam search to find the optimum for every pitch independently. We achieve this by examining the sub-sequences in the beam search tree only when the second-best state of a pitch has a higher possibility than a certain threshold. When such pitches and frames are detected, we perform beam-search for five more steps only with that pitch while fixing other pitches with greedy sampling. We also consider only two states with the highest probability at each frame because we expect the probability becomes negligible from the third in most cases. After the best path for the pitch is found, we proceeded to other pitches and next frame recursively. Most of the target frames were close to onset and offset. Therefore, the pitch-wise path search can be regarded as locating the onset or offset to an optimal position. This method operates like the greedy decoding if all of the estimated frames have a high confidence.

After note states are inferred by one of the decoding algorithms, we apply a simple rule to determine notes. Along with frame axis, a note is initiated if a {*onset* or *on*} state is detected after *off*. The offset of the note is determined if {*offset* or *off* } is detected after the note initialization.

4. EXPERIMENTS

4.1 Dataset

We trained our model with the MAESTRO dataset v1.0.0 [6] with the published training and test split. The dataset provides 1184 performances played in the *International Piano-e-Competition*. Both audio recordings and the corresponding aligned MIDI files captured through Disklavier are given. To compensate the effect of sustain pedal on note offsets in labeling, we elongated the offset of notes according to the sustain pedal followed by methodology in [4].

4.2 Metrics

We employed the standard frame-based and note-based metric using *mir_eval* [26] package. We report precision, recall and F1 score. We used a threshold of $50msec$ to de-

tect note onset. We counted note offset as hit when the difference is within $50msec$ or $\pm20\%$ of note duration.

4.3 Hyperparameters

We used log-compressed mel-spectrograms as input of the acoustic model. We computed the mel-spectrograms with 229 logarithmically-spaced frequency bins, a hop length of 512, an FFT window of 2048, and a sample rate of 16kHz following [4, 5]. The CNN consists of $48/48/96$ nodes from the bottom to the top layer, and the following fully-connected (FC) layer has 768 nodes. The RNN consists of two layers of LSTM with 768 nodes. The output FC layer has $(88 \times N)$ nodes where N is the number of note states. The receptive field size of the top hidden layer in the CNN is 7 frames of mel-spectrograms centered at time step t. This yields 176 msec latency when the model runs in real-time.

We used the categorical cross entropy loss and Adam optimizer for training, applying a batch size of 32 and a learning rate of $6e-4$ with a decay of 0.02 in every 10000 steps. We trained our model with the teacher-forcing method, which provides ground-truth data of the previous time step for the AR layer. We tried scheduled sampling [27] but our preliminary result showed a significant degradation in performance. We randomly segmented audio into $10sec$ while training, but the whole sequence is transcribed at once during inference. We evaluated the models after 200k steps in training. In addition to the aforementioned hyperparameters, we report the experimental result of a smaller model where the number of nodes in the LSTM is set to 256 (three times smaller than the above) to reduce the number of model parameters.

4.4 Comparative Experiments

We conducted a comparative experiment with the proposed method with the five note state representations. Also, we evaluated the same set of models without the autoregressive connection to verify the effectiveness of the autoregressive model. For the autoregressive models, we evaluated the note decoding results as well. We compared our model mainly with the *Onsets and Frames* model and its GAN-based extension [5], which is also trained on the same dataset. In addition to the reported performance, we evaluate *Onsets and Frames* model with the publicly available pre-trained model [1] and our own re-implementation.

5. RESULTS

5.1 Effect of Multi-State Note Representations

We report the averaged metrics over the test set in Table 1. Overall, the use of *onset* makes a significant improvement as seen in comparison between the 'Binary' model and other multiple models. This result is in accordance with previous studies [4]. The note-with-offset score also increases along but this might be seen as affected by the increased number of matched notes. The 'Binary' model

---

[1] https://github.com/tensorflow/magenta, accessed on Sep 14, 2019

Models	AR	Used states			#Parameters	Frame			Note Onset			Note with Offset		
		onset	re-onset	offset		precision	recall	F-score	precision	recall	F-score	precision	recall	F-score
Binary	○				14.4M	0.7768	0.8881	0.8161	0.9874	0.6281	0.7560	0.6399	0.4278	0.5064
Binary					13.9M	0.9128	0.8815	0.8961	0.8184	0.6646	0.7279	0.5655	0.4696	0.5095
Three	○	○			14.5M	0.7643	0.8828	0.8047	0.9878	0.9042	0.9433	0.7949	0.7249	0.7593
Three		○			13.9M	0.9502	0.8104	0.8740	0.9907	0.8247	0.8985	0.8259	0.6904	0.7508
Four(offset)	○	○		○	14.5M	0.7957	0.8774	0.8258	0.9874	0.9015	0.9416	0.8105	0.7409	0.7735
Four (offset)		○		○	14.0M	0.9549	0.8080	0.8745	0.9890	0.8392	0.9065	0.8264	0.7039	0.7590
Four (re-onset)	○	○	○		14.5M	0.7834	0.8864	0.8207	0.9871	0.9103	0.9465	0.8074	0.7450	0.7744
Four (re-onset)		○	○		14.0M	0.9529	0.8049	0.8717	0.9928	0.8258	0.8997	0.8313	0.6946	0.7553
Five	○	○	○	○	14.6M	0.8191	0.8732	0.8382	0.9856	0.9121	0.9467	0.8259	0.7648	0.7936
Five		○	○	○	14.1M	0.9391	0.8170	0.8730	0.9923	0.8278	0.9009	0.8213	0.6878	0.7472
Five (small)	○	○	○	○	6.1M	0.7264	0.8933	0.7889	0.9889	0.9043	0.9438	0.7755	0.7093	0.7403
Onsets and Frames (paper) [6]					18.3M*	0.9211	0.8841	0.9015	0.9827	0.9261	0.9532	0.8295	0.7824	0.8050
Onsets and Frames (pretrained)[1]					23.5M	0.8737	0.8768	0.8733	0.9792	0.9182	0.9473	0.8114	0.7615	0.7853
Onsets and Frames (reimplemented)					18.3M	0.9350	0.8771	0.9045	0.9939	0.8993	0.9436	0.8135	0.7371	0.7730
Onsets and Frames Uni-LSTM (reimplemented)					15.6M	0.9356	0.8599	0.8954	0.9929	0.8917	0.9388	0.8028	0.7218	0.7595
Non-Saturating GAN (paper) [5]					26.9M*†	0.931	0.898	**0.914**	0.981	0.932	0.956	0.835	0.793	0.813

Table 1. Frame and note metrics for the five note state representations. All measures are based on decoded sequence with greedy decoding. Precision, Recall and F1 score are averaged over piece-wise results. AR stands for 'autoregressive'. Refer 4.3 for detail. * This number was estimated based on hyperparameters in the paper. †This includes a neural network to estimate note velocity.

achieves a high frame-level F1-score but at the same time it has the lowest note onset F1-score. We suspect that overlapped notes without offsets (notes with *re-onset*) were not distinguishable in the binary note state representation, thereby leading low recall in the note onset score.
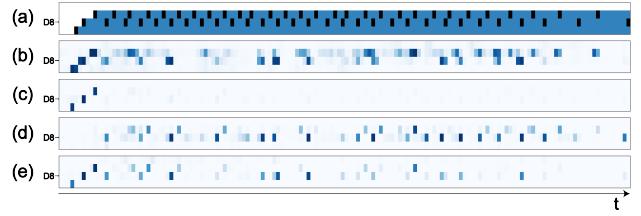
The influence of *re-onset* and *offset* is observed from the note-with-offset score. The 'Four' models that have either one of the states achieve higher scores than the 'Three' model. The 'Five' model that has both states achieves a higher score than the 'Four' models, particularly with the AR connection. However, *re-onset* and *offset* do not help improving the note onset score much.

Among the five note state representations, the 'Five' model achieves slightly higher frame-level and note-level F-scores than others. We investigated the model further by downsizing the LSTM units (small). The small model has a lower F-score in the note-with-offset score but it achieves comparable F-scores to the original 'Five' model in the note onset score.

We also observed that *re-onset* in the 'Five' model estimates sharper activations compared to the common *onset* when it estimates repeated notes with extremely short intervals such as trills, as shown in Figure 3. The number of such note patterns is too small to affect overall performance but it would be helpful when detailed analysis on articulation is necessary.

5.2 Effect of Autoregressive Connection

All AR models with *onset* show similar high performances in the note onset score. While the AR connection improves the F-score in both onset and offset of notes, it significantly decreases the frame-level scores. This might be because some extremely elongated note offsets are predicted by the AR model. This issue is discussed in the following section. The improvement in the note-with-offset score with *re-onset* and *offset* should be carefully understood because
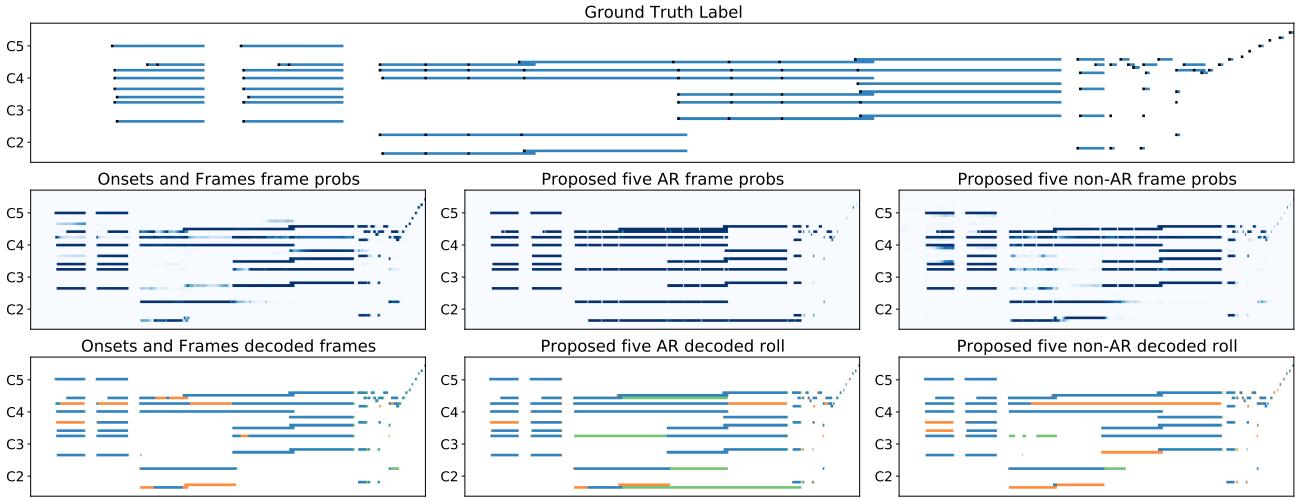
Figure 3. Comparison between *onset* and *re-onset* activation on trill notes. (a) ground truth (b) onset activation of *Onsets and Frames* (c) *onset* activation of 'Five' AR model (d) *re-onset* activation of 'Five' AR model (e) *onset* activation of 'Four (offset)' AR model

our model cannot learn state dependency backward (for example, considering that there will be an offset few frames later, the current frame is more likely to be *on*). Therefore, it is not clear why it is effective only on the AR model. Since our model was trained with a teacher-forcing scenario, we suspect that part of the benefits might be related to resilience on the noisy output, which can occur in the inference phase.

5.3 Learned State Transition

Figure 4 shows an example of frame activations and decoded piano rolls of the selected models. In the proposed 'Five' model with the AR connection, a note always starts with its onset prediction and the activation is clearly maintained with the following sustain predictions. This contrasts with the blurry activation in the *Onsets and Frames* model and the non-AR model, where some notes were estimated too short. This is identified by the short blue frames followed by the orange frames in the decoded roll in Figure 4. Estimating continuous sustain frames also has an negative effect. The AR model often fails to detect note offsets as the sustain frames are estimated too long. This is

Figure 4. Frame probabilities (or activations) and decoded piano rolls with the ground truth. The excerpt was selected from the test set. *Sustain* frames and *onset* frames are displayed in blue and black, respectively, in the ground truth. In the decoded piano rolls below, the estimated frames with {true positive, false negative, false positives} are annotated in blue, orange, and green, respectively. Best viewed in color.
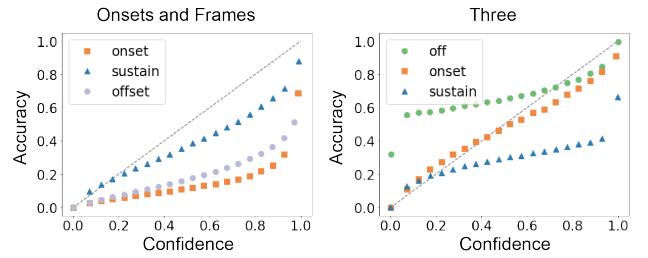
Decoding	F1 score	Models				
		Five	Four (offset)	Four (re-onset)	Three	Binary
Greedy	Note Onset	0.9467	0.9416	0.9465	0.9433	0.7560
	Note w. Offset	0.7936	0.7735	0.7744	0.7593	0.5064
Beam Search	Note Onset	0.9380	0.9305	0.9361	0.9310	0.7480
	Note w. Offset	0.7886	0.7648	0.7657	0.7484	0.5035

Table 2. Summary of note decoding results. All models have autoregressive model.

identified by the green frames followed by the blue frames in the decoded roll in Figure 4.

5.4 Beam Search and Error Calibration

We summarized the comparison between the two decoding algorithms in Table 2. Counter-intuitively, the proposed beam search performed slightly worse than the simple greedy decoding method. To analyze the reason, we investigated the confidence errors of class prediction. With the greedy decoding, we regarded their softmax predictions as a confidence measure and classified each estimation according to the value. We equally divided the confidence range [0 1] into 20 bins and recorded the averaged accuracy of each bin. The empirical discrepancy between accuracy and confidence indicates the model calibration error [28]. The confidence diagram Figure 5 shows that the model is overconfident on *sustain* prediction but it is underconfident on *off* frames. We suspect this large gap between estimated probabilities leads to sub-optimal paths in the beam search. Label smoothing [29] or temperature scaling [28] may be helpful for relaxation but we leave this for future work.

Figure 5. Confidence diagrams of the *Onsets and frames* model and the 'Three' model. The dashed diagonal line in black indicates perfect calibration.

6. CONCLUSIONS

We proposed a neural network architecture for polyphonic piano transcription where the acoustic and language models are integrated in a unified manner. The architecture is designed to predict multiple note states as a softmax output and learn the dependency among note states through the auto-regressive MLM. Our comparative study shows that the *onset* state is critical to improving note onset scores and the *offset* and *re-onset* states help improving the note-with-offset score. The auto-regressive MLM provides significantly higher accuracy on both note onset and offset estimation compared to its non-auto-regressive version. The visualization of decoded piano roll shows that our models with the auto-regressive connection generates a realistic sequence of note states. We also examined a pitch-wise beam search to decode the frame-level activation but the result showed that it was not as effective as a simple greedy decoding. Finally, the evaluation on the MAESTRO dataset shows that our proposed model achieves transcription performance comparable to the state-of-the-art models even with the unidirectional RNN and fewer parameters.

7. ACKNOWLEDGEMENT

8. REFERENCES

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[2] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, August 2016.

[3] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer, "On the potential of simple framewise approaches to piano transcription," in *Proc. of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, August 2016, pp. 475–481.

[4] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, October 2018, pp. 50–57.

[5] J. Kim and J. Bello, "Adversarial learning for improved onsets and frames music transcription," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 670–677.

[6] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proc. of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[7] Z. Duan and D. Temperley, "Note-level music transcription by maximum likelihood sampling," in *Proc. of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 181–186.

[8] A. Cogliati and Z. Duan, "Piano music transcription modeling note temporal evolution," in *Proc. of 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 429–433.

[9] T. Kwon, D. Jeong, and J. Nam, "Audio-to-score alignment of piano music using RNN-based automatic music transcription," in *Proc. of the 14th Sound and Music Computing Conference (SMC)*, 2017, pp. 380–385.

[10] Q. Wang, R. Zhou, and Y. Yan, "A two-stage approach to note-level transcription of a specific piano," *Applied Sciences*, vol. 7, September 2017.

[11] G. Costantini, R. Perfetti, and M. Todisco, "Event based transcription system for polyphonic piano music," *Signal Processing*, vol. 89, no. 9, p. 1798–1811, 2009.

[12] R. Kelz, S. Böck, and G. Widmer, "Deep polyphonic ADSR piano note transcription," in *Proc. of the 44th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2019, pp. 246–250.

[13] N. Boulanger-Lewandowski, Y. Bengio, P. Vincent, P. Gray, and C. Naguri, "Modeling temporal dependencies in high-dimensional sequences," in *Proc. of the 29th International Conference on Machine Learning (ICML)*, 2012, pp. 1881—1888.

[14] Q. Wang, R. Zhou, and Y. Yan, "Polyphonic piano transcription with a note-based music language model," *Applied Sciences*, vol. 8, no. 3, p. 470, March 2018.

[15] S. Böck and M. Schedl, "Polyphonic piano note transcription with recurrent neural networks," in *Proc. of the 37th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, March 2012, pp. 121–124.

[16] R. G. C. Carvalho and P. Smaragdis, "Towards end-to-end polyphonic music transcription: Transforming music audio directly to a score," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 151–155.

[17] R. Kelz and G. Widmer, "An experimental analysis of the entanglement problem in neural-network-based music transcription systems," in *Proc. of AES International Conference on Semantic Audio*. Audio Engineering Society, 2017.

[18] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, 2013.

[19] G. E. Poliner and D. P. Ellis, "A discriminative model for polyphonic piano transcription," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–9, 2006.

[20] J. Nam, J. Ngiam, H. Lee, and M. Slaney, "A classification-based polyphonic piano transcription approach using learned feature representations," in *Proc. of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 175–180.

[21] D. Cazau, Y. Wang, O. Adam, Q. Wang, and G. Nuel, "Improving note segmentation in automatic piano music transcription systems with a two-state pitch-wise hmm method," in *Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 523–530.

[22] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "High-dimensional sequence transduction," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3178–3182.

[23] A. Ycart and E. Benetos, "Polyphonic music sequence transduction with meter-constrained LSTM networks," in *Proc. of 43th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 2018, pp. 386–390.

[24] ——, "A study on LSTM networks for polyphonic music sequence modelling," in *Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 421–427.

[25] A. Ycart, A. McLeod, E. Benetos, and K. Yoshii, "Blending acoustic and language model predictions for automatic music transcription," in *Proc. of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 454–461.

[26] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "mir_eval: A transparent implementation of common MIR metrics," in *Proc. of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 367–372.

[27] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[28] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. of the 34th International Conference on Machine Learning (ICML)*. JMLR. org, 2017, pp. 1321–1330.

[29] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" in *Advances in Neural Information Processing Systems*, 2019, pp. 4696–4705.

EXACT, PARALLELIZABLE DYNAMIC TIME WARPING ALIGNMENT WITH LINEAR MEMORY

Christopher J. Tralie
Ursinus College
ctralie@alumni.princeton.edu

Elizabeth Dempsey
Ursinus College
eldempsey@ursinus.edu

ABSTRACT

Audio alignment is a fundamental preprocessing step in many MIR pipelines. For two audio clips with M and N frames, respectively, the most popular approach, dynamic time warping (DTW), has O(MN) requirements in both memory and computation, which is prohibitive for frame-level alignments at reasonable rates. To address this, a variety of memory efficient algorithms exist to approximate the optimal alignment under the DTW cost. To our knowledge, however, no exact algorithms exist that are guaranteed to break the quadratic memory barrier. In this work, we present a divide and conquer algorithm that computes the exact globally optimal DTW alignment using O(M+N) memory. Its runtime is still O(MN), trading off memory for a 2x increase in computation. However, the algorithm can be parallelized up to a factor of min(M, N) with the same memory constraints, so it can still run more efficiently than the textbook version with an adequate GPU. We use our algorithm to compute exact alignments on a collection of orchestral music, which we use as ground truth to benchmark the alignment accuracy of several popular approximate alignment schemes at scales that were not previously possible.

1. INTRODUCTION

The go-to algorithm for computing alignments between two audio clips is Dynamic Time Warping (DTW) [1, 2], and DTW and its variants have seen wide application in music processing applications [3]. However, the textbook version of exact DTW has quadratic memory constraints. While some MIR applications, such as cover song identification, can get away with coarse, beat-synchronous features [4] to remain in a low memory regime, other applications may require finer scale features and can quickly explode in memory requirements. For instance, in orchestral music, onsets are weak, so one must often revert to frame-level features for satisfactory alignments. Singing voices also present unique challenges in this regard [5], and both stringed instruments and singing voices have precise, expressive attacks at sub-beat scales.

In this work, we present a simple divide and conquer variant of DTW to compute a globally optimal alignment between two audio sequences with linear memory. Our contributions are as theoretical as they are practical; though there are many approximate algorithms that work well in practice (Section 2), we are not aware of any other linear algorithms for DTW with this guarantee. A related advantage is that there are no approximation parameters to tune; there is only one exact cost (with some caveats on numerical precision in Section 4.2).

Once we establish the algorithm, we present an experiment on a hand-curated collection of classical music (Section 4). Since our memory only scales linearly with a small factor (Section 4.3), we are able to run it on longer pieces, enabling us to evaluate the precision of approximation algorithms at scales not previously possible.

2. BACKGROUND

2.1 The Textbook DTW Algorithm

We now briefly review the standard DTW algorithm for context. Given a (possibly multivariate) time series X with M points and a time series Y with N points, there is a notion of allowable matchings that preserve the time order, known as a time-ordered correspondence, or "warping path." A warping path $\mathcal{W}$, is a correspondence between X and Y [1] with K ordered tuples of indices of X and Y so that (assuming 0-indexing) $\mathcal{W}_1 = (0, 0)$, $\mathcal{W}_K = (M-1, N-1)$, and $\mathcal{W}_i - \mathcal{W}_{i-1} \in \{(0, 1), (1, 0), (1, 1)\}$. In other words, matched points between time series must always stay still or advance by at most one along each, and at least one must move forward.

Given a cost measure between the ith element X_i in the first time series and the jth element Y_j in the second time series, $C_{X,Y}(i, j)$, then an "optimal" or "exact" solution to the *Dynamic Time Warping* problem is a warping path $\mathcal{W}^*$ that minimizes the sum

$$\sum_k C_{X,Y}(\mathcal{W}_k^*(1), \mathcal{W}_k^*(2)) \qquad (1)$$

We'll refer to an optimal path as $\mathcal{W}^*$ and the optimal cost as $D_{X,Y}(M, N)$. It is possible to compute the $\mathcal{W}^*$ and $D_{X,Y}(M, N)$ using a well-established dynamic programming approach, which is also shared among edit distance

---

[1] A *correspondence* $\mathcal{C}$ between two indexing sets I and J is a subset of the cartesian product $I x J$ so that every element of I is contained in at least one element of $\mathcal{C}$ and every element of J is contained in at least one element of $\mathcal{C}$

algorithms such as Smith Waterman [6]. In particular, if $D_{X,Y}(i,j)$ refers to the optimal cost of aligning the first i points of X to the first j points of Y, then the following recurrence holds for $i, j \geq 2$

$$D_{X,Y}(i,j) = \min \left\{ \begin{array}{ll} D_{X,Y}(i,j-1) & \text{LEFT} \\ D_{X,Y}(i-1,j) & \text{UP} \\ D_{X,Y}(i-1,j-1) & \text{DIAG} \end{array} \right\} + C_{X,Y}(i,j)$$

(2)

the boundary conditions are set as

$$D_{X,Y}(0,j) = \sum_{k=0}^{j} C_{X,Y}(0,k), D_{X,Y}(i,0) = \sum_{k=0}^{i} C_{X,Y}(k,0)$$

(3)

After filling in the first row and column by Equation 3, it is possible to compute all values of $D_{X,Y}(i,j)$ by applying Equation 2 from left to right, row by row. After processing all MN pairs of subsets in this fashion, $D_{X,Y}(M,N)$ contains the optimal cost.
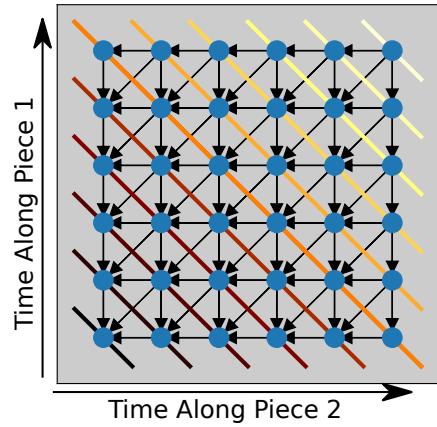
At this point in the algorithm, we merely have a cost, not an optimal warping path that realizes this cost. But if we store a second matrix $P(i,j)$ which remembers one of the three "backpointers" LEFT, RIGHT, and UP that realized the minimum cost at that cell, then we can "backtrace" by following these arrows back from (M,N) to $(1,1)$ to figure out the elements of an optimal $\mathcal{W}$ in between.

2.2 Variants And DTW Algorithms in MIR

There are countless works that incorporate and expand on DTW, so we constrain our focus to approaches and conventions that apply to music processing [3], with a particular focus on techniques that accelerate the algorithm.

There is theory to suggest that in general, $O(N^2)$ computation will always be the worst-case for optimal DTW [7], so many settle for approximate solutions. The so-called "Itakura Parallelogram" [8] and "Sakoe-Chiba Band" [2] were early fixed global alignment restrictions proposed to reduce memory and computation. More adaptive algorithms have also been used to approximate the DTW alignment on audio streams. One popular such example is a recursive multiresolution algorithm known as "FastDTW" [9], which has been used to synchronize orchestral music at large scales [10, 11]. The algorithm computes the warping path of lower resolutions versions of the time series, and then it recursively constrains alignments at finer scales to lie within some band of the lower resolution path. It is guaranteed to have worst case $O(M + N)$ run-time and memory consumption. A similar algorithm, known as "Memory-Restricted MultiScale DTW" (MrMs-DTW) [11] was devised to have constant memory usage, where performance degrades gracefully with a decreasing constant memory, and this technique has proved useful in MIR synchronization applications to pedagogy [12]. We compare to both of these algorithms in Section 4.

Beyond this, researchers have cut down on memory with approximate algorithms that use overlapping blocks [13] and which greedily expand cells to evaluate [14], though, like all of the approximations we've mentioned

Figure 1. A "linear systolic" array for computing the DTW cost. Arrows show dependencies. All elements along a diagonal can be computed in parallel if the diagonals are processed in order from the lower left (dark) to the upper right (light).
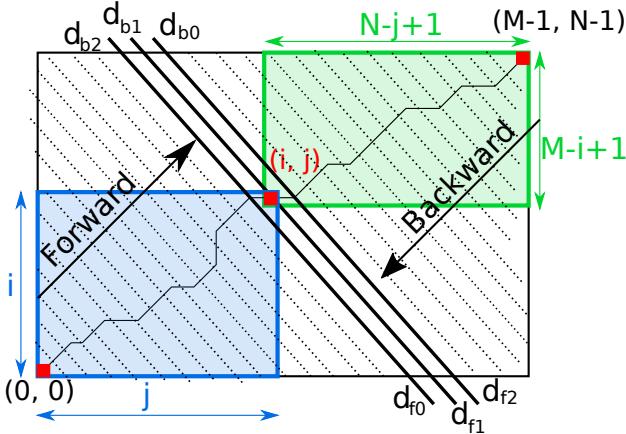
so far, they have no worst-case approximation guarantees. The authors of [15] and [16], on the other hand, present some of the only algorithms with worst case guarantees for DTW cost in Euclidean spaces. They achieve linearithmic runtime complexity, with a runtime proportional to the geometric complexity of the time series and inversely proportional to the approximation ratio. There are also several exact algorithms in the literature that use parallel architectures both for DTW [17] and for the related problem Smith-Waterman scoring between gene sequences [18] to speed up computation. We draw on these algorithms in our design in Section 3.1. However, they were designed simply to compute costs/scores, not to extract alignments, so we must build on this work to extract alignments (Section 3.2).

The closest work in spirit to ours is an algorithm for finding the longest common subsequence between strings [19], which also uses a divide and conquer scheme for subproblems that overlap on $O(M)$ cells, yielding an algorithm of $O(MN)$ time complexity and $O(M + N)$ space complexity like ours. However, it does not guarantee a globally optimal solution in the context of DTW [20].

3. OUR ALGORITHM

3.1 Computing the Cost (DiagDTW)

The backbone of our algorithm relies on a different order of filling in sub-problems of the alignment, the hardware implementation of which is an instance of a "linear systolic array" in computer architecture [17, 18]. This part is not yet novel, but it is crucial to our approach, so we review it here. Rather than processing the cells of $D_{X,Y}$ matrix row by row, as in the textbook version, it is also possible to satisfy dependencies needed to complete the recurrences in Equation 2 while filling in $D_{X,Y}$ along diagonals. Figure 1 shows the idea. If the diagonals are processed in order from lower left to upper right, then it follows that all elements on a single diagonal d_k can be computed in parallel from two previous adjacent diagonals d_{k-1} and d_{k-2}. Then, to

Figure 2. Choosing a pivot for the divide and conquer algorithm. At least one element (i, j) on the optimal warping path $\mathcal{W}^*$ resides on one of three central diagonals that overlap from forward and backward computation done halfway, and this element is used to recursively split computation of other warping path points into two halves.

move to the next diagonal, the three diagonal buffers circularly shift; d_k becomes d_{k-1}, d_{k-1} becomes d_{k-2}, and the previous d_{k-2} can be reused as the new as d_k. Since each diagonal has a max length of $\min(M, N)$, this means that one only needs a memory of $3 \min(M, N)$ to maintain such a system of circularly shifting buffers [2]; one never needs to store all of D in memory to compute the optimal cost. Once the algorithm has completed, the optimal cost can be read off as the only element in the last buffer.

Algorithm 1 Diagonal DTW

1: **procedure** DIAGDTW($X, Y, C_{X,Y}$, kstop)   ▷ Time series X and Y with M and N points, a cost $C_{X,Y}$ between them, and the diagonal on which to stop
2:     $d_0 \leftarrow C_{X,Y}(0, 0)$
3:     $d_1[\leftarrow [C_{X,Y}(1, 0) + d_0[0], C_{X,Y}(0, 1) + d_0[0]]$
4:     $d_2 \leftarrow []$
5:     **for** $k = 2$:kstop **do**
6:         Update all elements in d_2 based on d_0 and d_1
7:         $d_0, d_1, d_2 \leftarrow d_1, d_2, d_0$ ▷ Circularly shift
8:     **end for**
9: **return** d_0, d_1, d_2
10: **end procedure**

Algorithm 1 shows a sketch of the process. We refer to this algorithm as *DiagDTW*, and we have implemented it on the GPU using CUDA. For reasons that will become clear in Section 3.2, we take as a parameter $2 \leq$ kstop $\leq M + N - 1$ on which to stop the computation, and we return the states of all three diagonals at that point.

3.2 Extracting Alignment

The linear systolic array provides a way to compute cost, but if we insist on only remembering the most three recent diagonals of backpointers, then there is no obvious way to

---

[2] Ignoring the cost of storing features for the moment

Algorithm 2 Divide And Conquer Linear Memory Dynamic Time Warping (linmdtw)

1: **procedure** LINMDTW($X, Y, C_{X,Y}, m$)   ▷ Time series X and Y with M and N points, cost $C_{X,Y}$ between them, and a minimum dimension m
2:     **if** $M < m$ or $N < m$ **then**
        return DTW($X, Y, C_{X,Y}$)   ▷ Bruteforce path
3:     **end if**
4:     $K \leftarrow M + N - 1$ ▷ Number of diagonals
5:     kstop $\leftarrow \lceil K/2 \rceil$ ▷ Halfway point
6:     $d_{f0}, d_{f1}, d_{f2} \leftarrow$ DiagDTW($X, Y, C_{X,Y}$, kstop)
7:     **if** K is even **then**
8:         kstop $\leftarrow$ kstop $+1$
9:     **end if**
10:     $X_R \leftarrow$ reverse(X), $Y_R \leftarrow$ reverse(Y)
11:     $d_{b0}, d_{b1}, d_{b2} \leftarrow$ DiagDTW($X_R, Y_R, C_{X,Y}$, kstop)
       ▷ $C_{X,Y}(d_{fk})$ is all costs along the kth forward diag
12:     $d_0 \leftarrow d_{f0} +$ reverse(d_{b2}) $- C_{X,Y}(d_{f0})$
13:     $d_1 \leftarrow d_{f1} +$ reverse(d_{b1}) $- C_{X,Y}(d_{f1})$
14:     $d_2 \leftarrow d_{f2} +$ reverse(d_{b0}) $- C_{X,Y}(d_{f2})$
15:     $(i, j) \leftarrow$ argmin index in d_0, d_1, d_2
16:     Now recursively compute other points on $\mathcal{W}^*$
17:     LPath $\leftarrow$ linmdtw($X(1, 2, ..i), Y(1, 2, ..j), C, m$)
18:     RPath $\leftarrow$ linmdtw($X(i, i + 1, ..M), Y(j, j + 1, ..N), C, m$)
19: **return** LPath $+$ RPath($2, 3, ...$) ▷ Don't double count common point on overlapping sub-paths
20: **end procedure**

---

recover all of the backpointers to reconstruct an optimal warping path under $O(M + N)$ memory constraints. Instead, we make the following two observations, which we use to build a different algorithm from standard backtracing which works in a memory-restricted setting:

Lemma 1. *For any warping path $\mathcal{W}$ and any adjacent set of 3 diagonals, at least one element of $\mathcal{W}$ is incident on one of the three diagonals.*

This follows directly from the definition of a warping path in Section 2.1. We also have the following observation

Lemma 2. *Let $\mathcal{W}^*$ be an optimal warping path and $(i, j) \in \mathcal{W}^*$, and let X_R and Y_R be the time series X and Y in reverse order, respectively. Then the cost of the warping path $\mathcal{W}^*$ can be broken into three parts as follows*

$$C_{X,Y}(i, j) + C_{X_R, Y_R}(M - i + 1, N - j + 1) - D_{X,Y}(i, j) \tag{4}$$

This is depicted by the overlapping boxes in Figure 2. In other words, the total cost is the optimal cost of aligning the first half of the path from $(0, 0)$ up to and including (i, j), plus the optimal cost of aligning second half of the path from (i, j) up to and including $(M - 1, N - 1)$, minus the distance from X_i to Y_j (so we don't double count that distance where they overlap). This follows from the fact that warping paths must start and end at the beginning and

end of each time series (so each sub-path is forced to touch (i, j)), and the fact that reversing both time series has no effect on the optimal cost. This is similar to the observation used in MrMsDTW to break up computation into smaller parts [11].

Now we are ready to present the divide and conquer algorithm to compute an optimal warping path $\mathcal{W}^*$. Lemma 1 and Lemma 2 together imply that if we trace the first half of the diagonals in a forward direction and the second half of the diagonals in the reverse direction and add them up pointwise where they meet at the center, subtracting off the distance at those points, then at least one element (i, j) on the three diagonals will contain the optimal cost $C(M, N)$. Furthermore, since this cost occurs on the optimal path, it will by definition be the *minimum cost* over all pointwise sums. Therefore, to find a point towards the center of $\mathcal{W}^*$, we simply do the following

1. Run Algorithm 1 halfway in the forward direction, starting at the beginning

2. Run Algorithm 1 halfway in the reverse direction, starting at the end

3. Perform the sums in Equation 4 where they overlap

4. Take the indices (i, j) of the value that achieves the minimum over all three diagonals (breaking ties arbitrarily, the result of which we explore in Section 4.2)

.
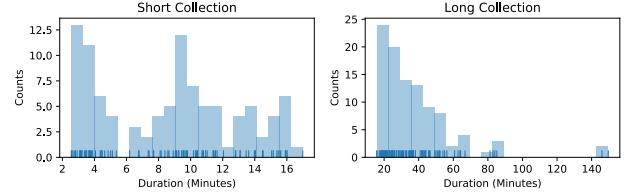
We refer to (i, j) as the "pivot" at this step, and we are guaranteed that (i, j) resides on $\mathcal{W}^*$. At this point, we divide the problem in half at the pivot and find two more points on the warping path to the left and right, which is the recursive step. Algorithm 2 summarizes this process.

Because Algorithm 2 calls DiagDTW as a subroutine and DiagDTW uses $3 \min(M, N)$ memory, Algorithm 2 also uses at most $3 \min(M, N)$ memory. What is slightly less obvious, but still fairly straightforward to show, is that a serial version of the algorithm takes $O(MN)$ time. To see this, parameterize the diagonal by a variable x, where $x = 0$ at the center of the central diagonal, then the total area of the sub-block to the left of the chosen pivot and to the right of the chosen pivot is bounded from above by the following sum of two products

$$A(x) = \left(\frac{M}{2} + x + 1 \right) \left(\frac{N}{2} - x + 1 \right) \quad (5)$$

$$+ \left(\frac{N}{2} + x + 1 \right) \left(\frac{M}{2} - x + 1 \right) \quad (6)$$

Then, $A'(x) = -4x$, and $A''(x) = -4$, so a global maximum occurs at $x = 0$, for an area of $A(0) = M^2 N^2 / 2 + M + N$. In other words, ignoring the edge effects $M + N$ due to the overlap, at most half of the total cells are processed across the two halves of each recursive split. This leads to the recurrence $MN(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ...)$, which is bounded from above by $2MN$. To understand the

Figure 3. The distribution of lengths of our orchestral pieces in the short collection and long collection.

edge effects, we note the following: since the number of diagonals, $M + N - 1$, can be subdivided $\log_2(M + N)$ times, this leads to a bound of $(M + N) \log_2(M + N)$, for a total worst-case cost of

$$2MN + (M + N) \log_2(M + N) \quad (7)$$

However, the $2MN$ term will usually swamp the $(M + N) \log_2(M + N)$, unless one of M or N is very small (e.g $M = 1$, in which case it's simply subdividing an interval of length N repeatedly $\log_2(N)$ times). In practice, we parallelize the DiagDTW step on a GPU, so the algorithm runs faster than this. We also keep track of the number of cells processed, and we assume $2MN$ to indicate progress of the the algorithm to the user.

Finally, to save the overhead of initiating too many small GPU alignments, we break off the recursion when the sub-blocks get small enough (Line 2, Algorithm 2). In practice, if either length of of the subdivided time series goes below 500, we use the textbook DTW algorithm to complete the alignment, which uses an insignificant amount of computation and memory at that scale.

4. EXPERIMENTS

Now that the theory for our algorithm has been established, we apply it to align real audio data. We created a dataset with 100 pairs of mostly orchestral pieces, where each pair is performed under different conductors. All of the performances can be found on Youtube, and we provide code to automatically download them for reproducibility [3]. We do not have access to human annotated alignments, but since we can compute exact costs with linear memory, we can use our exact paths to assess the precision of approximate algorithms at very large scales to get an idea of how they perform in that regime. To that end, we split our dataset into two parts. The first 50 pairs are "shorter" pieces that can be handled (albeit sometimes slowly) by the textbook CPU DTW algorithm. The second set are pieces that would quickly use up all memory with the textbook algorithm on a personal computer, including many pieces around an hour or longer. Figure 3 shows the distribution of the lengths of pieces in both sets.

4.1 Features

So far, our discussion has assumed that we had access to some distance function $D_{X,Y}$ between time series X and

---

[3] If any links go down, the code is robust to that and will simply skip those examples

Table 1. Memory requirements of the dynamic programming accumulated cost cells for different algorithms on some of the pieces in our dataset. DTW refers to the naive algorithm, while FastDTW refers to the algorithm in [9] using a band width of $\delta = 30$. The memory requirements for MrMsDTW for 10^5 and 10^7 constant cells is 391KB and 38Mb, respectively

Piece	Version 1	Version 2	DTW	FastDTW	Ours
Vivaldi Spring	Abbado (188 sec)	Gunzenhauser (209 sec)	277 MB	3.86 MB	194 KB
Candide Overture	Bernstein (268 sec)	Dudamel (279 sec)	527 MB	5.5 MB	270 KB
Beethoven. Symph. No.5	Thielemann (445 sec)	Bernstein (514 sec)	1.58 GB	9.12 MB	448 KB
Schumann - Symph. No. 3	Bernstein (2124 sec)	Muti (2199 sec)	23.2 GB	36.9 MB	1.77 MB
Stravinsky The Rite of Spring	Rattle (2053 sec)	Bernstein (2082 sec)	29.4 GB	42.1 MB	2.02 MB
Tchaikovsky Symph. No. 4	Bernstein (2645 sec)	Rozhdestve.. (2530 sec)	46.1 GB	51.9 MB	2.48 MB
Shostakovich: Symph. No. 11	Søndergård (3647 sec)	Nelsons (3765 sec)	94.6 GB	74.8 MB	3.6 MB
Verdi Requiem	Bychkov (4983 sec)	Solti (5042 sec)	173 GB	102 MB	4.9 MB
Wagner - Das Rheingold	Kuhn (8799 sec)	Solti (8759 sec)	542 GB	180 MB	8.6 MB

Y. We now finally describe two different features sets that allow us to compute distances for synchronization, which we use in our experiments. The first set of features are the so-called "decaying locally adaptive normalized C0" (DLNC0) features [21], which are popular for fine scale alignments [4]. The second set of features are referred to as "mfcc-mod" features, which consist of a large number of MFCC coefficients, throwing away the lower order ones to control for loudness. These features were shown to work well at precisely capturing human annotations [22].

For both feature sets, we sampled audio at 22050hz, and we used a hop size of 512 between feature frames. This corresponds to about 43 frames per second of resolution. For the DLNC0 features, we used librosa's implementation of the CQT with default parameters as a starting point [23]. The DLNC0 features were concatenated to a 0.1 factor of CENS features to improve stability in steady-state regions, as suggested in [21]. For the mfcc-mod coefficients, we used an fft-length of 2048 and computed 120 "HTK" coefficients, leaving the first 20 out. Although [21] recommends using cosine distance for the DNLC0 component, we found lower discrepancies using the Euclidean distances as our measure across the board on all of our features.

4.2 Numerical Precision / Tie Breaking

Since our algorithm is on the GPU, we revert to 32-bit computations, and there is a worry that numerical precision could cause discrepancies, especially since the numbers along warping paths are summed together in a different order in our algorithm, and ties are broken at a different stage. To rule this out as a source of error when comparing approximation precisions, we compare our GPU answer to the brute force 32-bit CPU answer on the textbook algorithm. We also compare two different tie breaking rules on 64-bit CPU brute force implementations; one where diagonal takes precedence over left, and one the other way around. Ultimately, though there are discrepancies, they are negligible compared to errors in approximation, as shown in Figure 4. And the 32-bit versus 64-bit appears to make little difference at these scales.

---
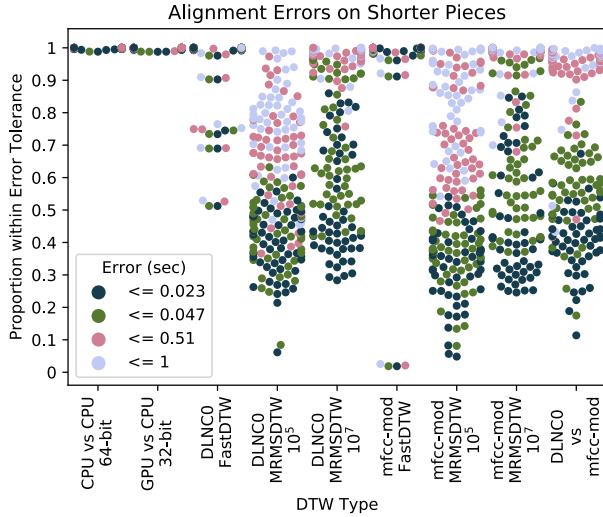
[4] Unlike [10], we do not use a multiscale version of DLNC0, since we are assessing approximations of exact alignments at a single scale

4.3 Memory Requirements
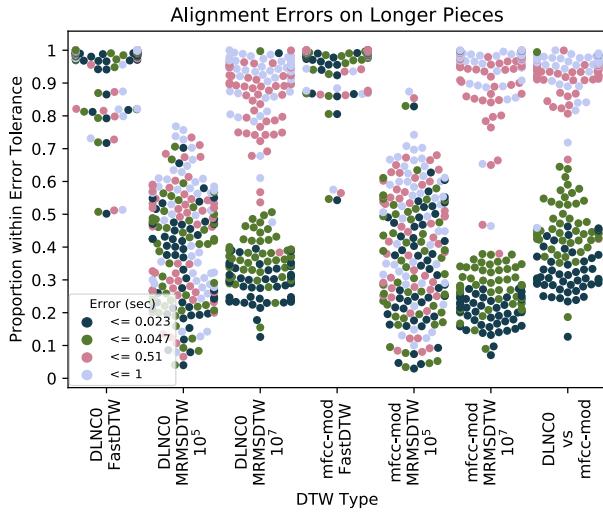
We compare our alignments to both FastDTW [9] with a band $\delta = 30$ and to MrMsDTW using a constant amount of 10^5 and 10^7 cells. To make Equation 4 more convenient to compute in our implementation, we store the distances between corresponding points on the three diagonals in addition to the cumulative sums, so we end up using $6 \min(M, N)$ storage instead of $3 \min(M, N)$ storage. Still, we note that 10^7 cells is an order of magnitude beyond this requirement, while 10^5 cells is on the shorter end of what our algorithm needs on the short dataset, so these are two good reference points for MrMsDTW. To compare memory with FastDTW, we use the equation from [9] which states that the total worst-case space complexity for storing the cells is $N(4\delta + 5)$ values. Hence, though FastDTW also has linear memory requirements, it has a larger constant factor, particularly for reasonable band sizes ($\delta = 30$ is less than a second of wiggle room). Table 1 shows the memory requirements for storing the cells for different algorithms with variable memory, assuming 32-bit precision (4 bytes per cell). This neglects the memory for storing the warping path, which is negligible compared to the cost of storing the accumulated cost cells, and it also neglects the memory requirements of storing features, which is a separate issue mostly independent of the algorithms, since these are all run offline. Still, the memory differences are striking.

4.4 Results

We now examine the results closely. We computed the alignment discrepancies between two warping paths $\mathcal{W}_1$ and $\mathcal{W}_2$ as follows. For every element $(i, j) \in \mathcal{W}_1$, we report the error as $\min |j - k|$ for $(i, k) \in \mathcal{W}_2$. To maintain symmetry, we also add an analogously defined column error to our distributions for every element. Figure 4 shows the approximation error distributions for different algorithms on all of the shorter pieces, including tie breaking discrepancies on the exact algorithm (Section 4.2), while Figure 5 shows approximation errors on all of the longer pieces. In each figure for each pairwise comparison, there are four different color dots per piece that indicate the proportion of correspondences (i, j) that fall below the alignment discrepancies (23 ms, 47 ms, 510ms, and 1 second).

Figure 4. Shorter pieces alignment errors.
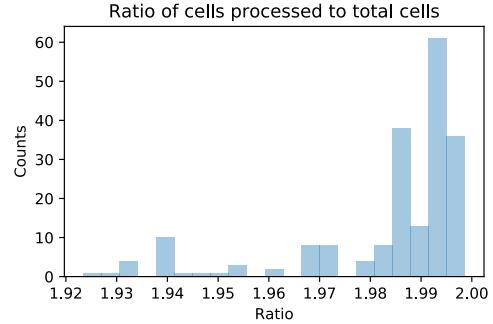
Figure 5. Longer pieces alignment errors.

Overall, we find that the approximation algorithms often fail to agree at very fine scales, but they usually agree to within a second of audio, which is particularly impressive on the long pieces. And unsurprisingly, MrMsDTW performs better with more memory.

In addition to approximation errors, we also show the discrepancy between the mfcc-mod and DLNC0 feature sets for reference. Interestingly, their discrepancy is similar to that of approximation with MrMsDTW, suggesting that feature design is at least as important as a good approximation. However, under a good feature choice at a fine scale, it is likely that our exact algorithm will give the most desirable alignment.

Finally, to empirically validate the correctness of our computational complexity bound in Section 3.2, we report the ratio of cells processed to total cells in the full accumulated cost matrix in Figure 6 across all pieces, and we find that the ratio is very close to 2 in all cases, as predicted. Only under very extreme warps away from the center of the matrix would one expect this to be much smaller.

Figure 6. In most cases, our algorithm uses close to the factor of 2 bound we established for computation in Section 3.2

5. SOFTWARE

Since some of the details of linmdtw (Algorithm 2) are tricky to implement correctly, and in the spirit of reproducibility [24], we have provided our CPU and GPU (pycuda) implementations of linmdtw, FastDTW, and MrMsDtw in an open source package at `https://github.com/ctralie/linmdtw`, which can be installed simply with "pip install linmdtw". We have documentation and Jupyter notebooks on the repo for example usage. The software will try run CUDA by default, but if it fails, it will fall back to the CPU implementation. There is also code to replicate the experiments in Section 4 by downloading URLs from Youtube, robustly skipping those no longer available. Finally, we used the Rubberband Library [25] and implemented the refinement technique of Ewert (Section 4 of [26]) to stretch audio to conform to warping paths.

6. DISCUSSION

In this paper, we presented a novel exact memory efficient algorithm for DTW. In addition establishing this new algorithm and proving its correctness, we empirically benchmarked a couple of popular approximation algorithms for DTW alignment in MIR at larger scales than had ever been shown. We found that these algorithms still have fairly good performance with reference to an exact alignment even on longer pieces. MrMsDTW is particularly fast computationally, so this suggests that it's good as a first stop in many cases, though there are outliers, and there are always quality gains to be had for an exact algorithm.

Furthermore, though the focus of this paper was on memory constraints, our vanilla GPU implementation also led to speed increases over the textbook CPU version and had similar but slightly slower runtimes than FastDTW. However, a better GPU implementation would treat global and local memory with more care, along with addressing myriad other issues [27], so we do not believe this algorithm has yet reached its full computational potential.

There are also other computational problems with very similar dynamic programming design DTW, such as edit distance and Smith Waterman [6], which could benefit from the ability to align large sequences under memory restrictions. Even approximate DTW algorithms may benefit from tricks in this paper.

7. REFERENCES

[1] H. Sakoe and S. Chiba, "A similarity evaluation of speech patterns by dynamic programming," in *Nat. Meeting of Institute of Electronic Communications Engineers of Japan*, 1970, p. 136.

[2] ——, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[3] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[4] D. P. Ellis, "Identifying 'cover songs' with beat-synchronous chroma features," *MIREX 2006*, p. 32.

[5] S. Waloschek and A. Hadjakos, "Driftin'down the scale: Dynamic time warping in the presence of pitch drift and transpositions." in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR), in print*, 2018, pp. 630–636.

[6] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[7] K. Bringmann and M. Künnemann, "Quadratic conditional lower bounds for string problems and dynamic time warping," in *IEEE 56th Annual Symp. on Foundations of Computer Science*. IEEE, 2015, pp. 79–97.

[8] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 67–72, 1975.

[9] S. Salvador and P. Chan, "Fastdtw: Toward accurate dynamic time warping in linear time and space." *Proc. of ACM Knowledge Data And Discovery (KDD), 3rd Wkshp. on Mining Temporal and Sequential Data*, 2004.

[10] M. Müller, H. Mattes, and F. Kurth, "An efficient multiscale approach to audio synchronization." in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR), in print*, vol. 546. Citeseer, 2006, pp. 192–197.

[11] T. Prätzlich, J. Driedger, and M. Müller, "Memory-restricted multiscale dynamic time warping," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 569–573.

[12] T. Tsai, S. K. Tjoa, and M. Müller, "Make your own accompaniment: Adapting full-mix recordings to match solo-only user recordings." in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR), in print*, 2017, pp. 79–86.

[13] R. Macrae and S. Dixon, "Accurate real-time windowed time warping." in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR), in print*, 2010, pp. 423–428.

[14] S. Dixon, "Live tracking of musical performances using on-line time warping," in *Proc. of the 8th Intl. Conf. on Digital Audio Effects (DAFX)*, vol. 92. Citeseer, 2005, p. 97.

[15] R. Ying, J. Pan, K. Fox, and P. K. Agarwal, "A simple efficient approximation algorithm for dynamic time warping," in *Proc. of the 24th ACM SIGSPATIAL Intl. Conf. on Advances in GIS*, ser. GIS '16. New York, NY, USA: ACM, 2016, pp. 21:1–21:10. [Online]. Available: http://doi.acm.org/10.1145/2996913.2996954

[16] P. K. Agarwal, K. Fox, J. Pan, and R. Ying, "Approximating Dynamic Time Warping and Edit Distance for a Pair of Point Sequences," in *32nd Intl. Symp. on Computational Geometry (SoCG), in print*, ser. Leibniz Intl Proc. in Informatics (LIPIcs), S. Fekete and A. Lubiw, Eds., vol. 51. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 6:1–6:16. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2016/5898

[17] E. Dijkstra and C. Piguet, "On minimizing memory in systolic arrays for the dynamic time warping algorithm," *Integration*, vol. 4, no. 2, pp. 155–173, 1986.

[18] C. W. Yu, K. Kwong, K.-H. Lee, and P. H. W. Leong, "A smith-waterman systolic cell," in *New Algorithms, Architectures and Applications for Reconfigurable Computing*. Springer, 2005, pp. 291–300.

[19] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Communications of the ACM*, vol. 18, no. 6, pp. 341–343, 1975.

[20] G. Al-Naymat, S. Chawla, and J. Taheri, "Sparsedtw: a novel approach to speed up dynamic time warping," in *Proc. Australasian Data Mining Conf.*, vol. 101, 2012, pp. 117–127.

[21] S. Ewert, M. Müller, and P. Grosche, "High resolution audio synchronization using chroma onset features," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2009, pp. 1869–1872.

[22] T. Gadermaier and G. Widmer, "A study of annotation and alignment accuracy for performance comparison in complex orchestral music," in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR), in print*, 2019.

[23] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. of the 14th python in science conf.*, vol. 8, 2015.

[24] B. McFee, J. W. Kim, M. Cartwright, J. Salamon, R. Bittner, and J. P. Bello, "Open-source practices for music signal processing research," *IEEE Signal Processing Magazine*, vol. 1053, no. 5888/19, 2019.

[25] C. Cannam, "Rubber band library," *Software released under GNU General Public License (version 1.8. 1)*, 2012.

[26] S. Ewert and M. Müller, "Refinement strategies for music synchronization," in *Intl, Symp. on Computer Music Modeling and Retrieval*. Springer, 2008, pp. 147–165.

[27] L. Xiao, Y. Zheng, W. Tang, G. Yao, and L. Ruan, "Parallelizing dynamic time warping algorithm using prefix computations on gpu," in *2013 IEEE 10th Intl. Conf. on High Performance Computing and Communications & 2013 IEEE Intl Conf. on Embedded and Ubiquitous Computing*. IEEE, 2013, pp. 294–299.

Paper Session 4

CLASSIFYING LEITMOTIFS IN RECORDINGS OF OPERAS BY RICHARD WAGNER

Michael Krause, Frank Zalkow, Julia Zalkow, Christof Weiß, Meinard Müller

International Audio Laboratories Erlangen, Germany

{michael.krause,meinard.mueller}@audiolabs-erlangen.de
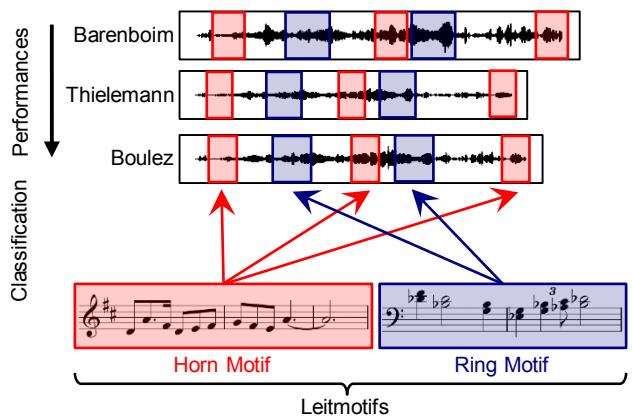
ABSTRACT

From the 19th century on, several composers of Western opera made use of *leitmotifs* (short musical ideas referring to semantic entities such as characters, places, items, or feelings) for guiding the audience through the plot and illustrating the events on stage. A prime example of this compositional technique is Richard Wagner's four-opera cycle *Der Ring des Nibelungen*. Across its different occurrences in the score, a leitmotif may undergo considerable musical variations. Additionally, the concrete leitmotif instances in an audio recording are subject to acoustic variability. Our paper approaches the task of classifying such leitmotif instances in audio recordings. As our main contribution, we conduct a case study on a dataset covering 16 recorded performances of the *Ring* with annotations of ten central leitmotifs, leading to 2403 occurrences and 38448 instances in total. We build a neural network classification model and evaluate its ability to generalize across different performances and leitmotif occurrences. Our findings demonstrate the possibilities and limitations of leitmotif classification in audio recordings and pave the way towards the fully automated detection of leitmotifs in music recordings.

1. INTRODUCTION

Music has long been used to accompany storytelling, from Renaissance madrigals to contemporary movie soundtracks. A central compositional method is the association of a certain character, place, item, or feeling with its own musical idea. This technique culminated in 19th century opera where these ideas are denoted as *leitmotifs* [1, 2]. A major example for the use of leitmotifs is Richard Wagner's tetralogy *Der Ring des Nibelungen*, a cycle of four operas [1] with exceptional duration (a performance lasts up to 15 hours) and a continuous plot spanning all four operas. As many characters or concepts recur throughout the

---

[1] While Wagner referred to his works as *music dramas* instead of operas, we choose the more commonly used latter term.

Figure 1. Illustration of example leitmotifs (red for the Horn motif, blue for the Ring motif) occurring several times in the Ring cycle and across different performances.

cycle, so do their corresponding leitmotifs. This allows the audience to identify these concepts not only through text or visuals, but also in a musical way. While all these different *occurrences* of a leitmotif in the score share a characteristic musical idea, they can appear in different musical contexts and may vary substantially in compositional aspects such as melody, harmony, key, tempo, rhythm, or instrumentation. When considering recorded *performances* of the *Ring*, another level of variability is introduced due to acoustic conditions and aspects of interpretation such as tempo, timbre, or intonation. In the following, we denote the concrete realization of a leitmotif in an audio recording as an *instance* of the motif. This paper approaches the problem of classifying such leitmotif instances in audio recordings, as illustrated in Figure 1. In particular, we study generalization across occurrences and performances.

Cross-version studies on multiple performances have been conducted regarding the harmonic analysis of Beethoven sonatas [3] or Schubert songs [4], but also for the *Ring* [5, 6]. Beyond harmonic aspects, the *Ring* scenario was considered for capturing audience experience using body sensors and a live annotation procedure [7] or for studying the reliability of measure annotations [8, 9]. Regarding leitmotifs, several works have focused on the human ability to identify motifs [10–12]. In particular, [13] found that distance of chroma features correlates with difficulty for listeners in identifying leitmotifs. In [6], Zalkow et al. presented a framework for exploring relationships between leitmotif usage and tonal characteristics of the *Ring*.

Name (English translation)	ID	Score	# Occurrences	Length Measures	Seconds
Nibelungen (Nibelungs)	L-Ni		536	0.96 ± 0.23	1.72 ± 0.50
Ring (Ring)	L-Ri		286	1.49 ± 0.65	3.64 ± 2.30
Mime (Mime)	L-Mi		242	0.83 ± 0.25	0.87 ± 0.24
Nibelungenhass (Nibelungs' hate)	L-NH		237	0.96 ± 0.17	3.10 ± 1.11
Ritt (Ride)	L-RT		228	0.66 ± 0.17	1.24 ± 0.37
Waldweben (Forest murmurs)	L-Wa		223	1.10 ± 0.30	2.70 ± 0.76
Waberlohe (Swirling blaze)	L-WL		190	1.21 ± 0.39	4.39 ± 1.60
Horn (Horn)	L-Ho		172	1.38 ± 1.05	2.44 ± 1.57
Geschwisterliebe (Siblings' love)	L-Ge		155	1.31 ± 0.83	3.03 ± 2.55
Schwert (Sword)	L-Sc		134	1.89 ± 0.55	3.68 ± 1.88

Table 1. Overview of the leitmotifs used in this study. Lengths are given as mean and standard deviations over all annotated occurrences (in measures) or instances (in seconds) from all performances given in Table 2.

From a technical perspective, our scenario entails the task of automatically detecting leitmotifs within an audio recording. This paper represents a first step towards this goal by considering a simplified classification scenario with pre-segmented instances (see Figure 1).

Due to the multiple sources of variability described above, we opt for a data-driven approach. Neural networks have emerged as the dominant classification models. In particular, recurrent neural networks (RNNs) are able to handle input sequences of varying length. Our study shows that despite the difficulties of the scenario, an RNN classifier is surprisingly effective in dealing with the variability across occurrences and performances.

The main contributions of our work are as follows: We conduct a case study on classifying leitmotif instances in audio recordings of the *Ring*. For this, we describe the task of leitmotif classification and provide a dataset of more than 38000 annotated instances within 16 performances of the *Ring* (Section 2). We further build an RNN model for classifying leitmotifs in audio recordings (Section 3). We carefully evaluate our model with respect to variabilities across performances and leitmotif occurrences over the course of the *Ring*. Moreover, we investigate the effect of adding temporal context and critically discuss the potential limitations and generalization capabilities of our classifier (Section 4). Finally, we suggest new research directions that may continue our work (Section 5).

2. SCENARIO

We now discuss the dataset and leitmotif classification scenario underlying our experiments.

2.1 Leitmotifs in Wagner's Ring

While Wagner mentioned the importance of motifs for his compositional process [14], he did not explicitly specify the concrete leitmotifs appearing in the *Ring*. Whether a recurring musical idea constitutes a leitmotif—and how to name it—is a topic of debate even among musicologists, see, e. g., [15] where differences in leitmotif reception are discussed. In line with [6], we follow Julius Burghold's specification of more than 130 leitmotifs in the *Ring* [16].

For our experiments, we selected ten central motifs frequently occurring throughout the *Ring* (see Table 1 for an overview including the number of occurrences per motif). These motifs constitute the classes of our classification task. The selection comprises motifs associated with an item such as the sword (L-Sc), with characters such as the dwarf Mime (L-Mi), or with emotions such as love (L-Ge). All occurrences of these motifs were annotated by a musicologist using a vocal score of the *Ring* as a reference, resulting in 2403 occurrences.

As discussed in Section 1, a leitmotif may occur in different shapes over the course of a drama. These musical variations may be necessary to fit the musical context in which the occurrences appear and, thus, be adjusted to the

ID	Conductor	Year	hh:mm:ss
P-Ba	Barenboim	1991–92	14:54:55
P-Ha	Haitink	1988–91	14:27:10
P-Ka	Karajan	1967–70	14:58:08
P-Sa	Sawallisch	1989	14:06:50
P-So	Solti	1958–65	14:36:58
P-We	Weigle	2010–12	14:48:46
P-Bo	Boulez	1980–81	13:44:38
P-Bö	Böhm	1967–71	13:39:28
P-Fu	Furtwängler	1953	15:04:22
P-Ja	Janowski	1980–83	14:08:34
P-Ke	Keilberth/Furtwängler	1952–54	14:19:56
P-Kr	Krauss	1953	14:12:27
P-Le	Levine	1987–89	15:21:52
P-Ne	Neuhold	1993–95	14:04:35
P-Sw	Swarowsky	1968	14:56:34
P-Th	Thielemann	2011	14:31:13

Table 2. Recorded performances of the *Ring* used in this study (see also [6]). Measure positions have been annotated manually for the topmost three performances (P-Ba, P-Ha and P-Ka), which also constitute the test set in our performance split. The three middle performances (P-Sa, P-So and P-We) constitute the validation set.
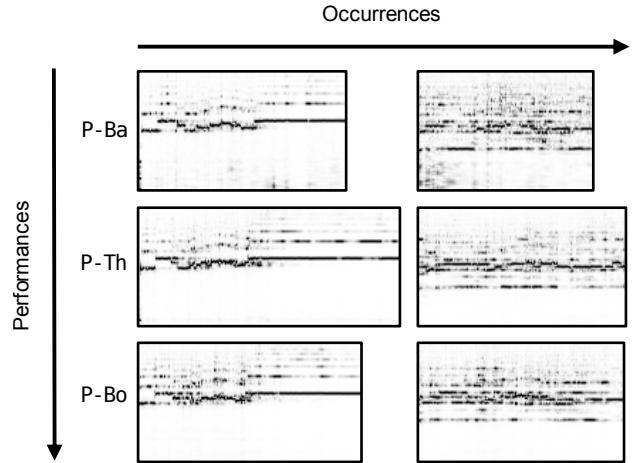
current key, meter, or tempo. Moreover, occurrences of leitmotifs may appear in different registers, musical voices, or instruments. In addition to this, motifs can also occur in abridged or extended shape, with parts of the motif being repeated, altered, or left out. Despite these diverse musical variations across occurrences, listeners can often identify motifs easily when listening to a performance. This is in line with Wagner's intention of using the motifs as a guideline, thus forming the musical surface of the *Ring* [17].

2.2 Recorded Performances

As mentioned in the introduction, we do not attempt to classify leitmotifs within a score representation but on the basis of a performance given as an audio recording. To be more concrete, our work relies on 16 recorded performances of the *Ring* that have been used before in [6]. For three of these performances, the positions of measures from the score were manually annotated in the audio [8]. For the remaining 13 performances, the measure positions were transferred from the manually annotated performances using automatic audio-to-audio synchronization [9]. Table 2 specifies the performances. We automatically located the 2403 leitmotif occurrence regions from the score in each of the 16 recorded performances using linear interpolation between measure positions. This way, we obtained the 38448 instances used for our experiments. The occurrence and instance positions are made publicly available as a dataset for further research. [2]

2.3 Leitmotif Classification Task

In this paper, we consider the task of leitmotif classification. We define this as the problem of assigning a given audio excerpt to a class according to the occurring leitmotif. Here, we consider ten classes corresponding to the mo-

[2] https://www.audiolabs-erlangen.de/resources/MIR/2020-ISMIR-LeitmotifClassification

Figure 2. Variability of L-Ho across occurrences and performances. Six instances (two occurrences for three performances) are shown in a CQT representation, which is also used as input to our classification model.

tifs in Table 1. We further make the simplifying assumption that only a single leitmotif is played at a time. Thus, we omit excerpts where multiple motifs occur simultaneously. Our classification task therefore becomes a multi-class, single-label problem.

Our dataset allows us to approach the leitmotif classification task from two perspectives, each of which incorporates its own types of variabilities. First, the *performance* perspective concerns variabilities across different performances, resulting from different instrumental timbres, tempi, or other decisions made by the artists. Furthermore, this perspective encompasses technical properties such as acoustic, recording, and mastering conditions, which can lead to the so-called "album effect" [18]. Second, the compositional or *occurrence* perspective concerns diverse musical variabilities of leitmotif occurrences in the score (as discussed in Section 2.1). Figure 2 shows the Horn motif L-Ho for different performances and occurrences. The variability is evident in different durations of the instances as well as different energy distributions due to other musical events sounding simultaneously. These variabilties make our classification task a challenging problem. In our experiments, we investigate the generalization across these two perspectives, similar to the study in [4].

3. RECURRENT NEURAL NETWORK FOR LEITMOTIF CLASSIFICATION

Neural networks have previously proven to be useful for classification tasks in the music domain, see, e. g., [19–21]. As we are dealing with variable length inputs (leitmotif instances may last from less than one to over ten seconds in a performance), recurrent neural networks (RNNs) are a natural choice for our scenario.

As input to our system, we take audio excerpts containing leitmotif instances from our 16 performances of the *Ring*, sampled at 22050 Hz. These excerpts are processed by a constant-Q-transform (CQT) [22, 23] with semitone resolution over six octaves and a hop length of 512 sam-

Layer	Output Shape	Parameters
Input	(V, 84)	
LSTM	(V, 84)	109056
LSTM	(V, 128)	131584
LSTM	(V, 128)	131584
Take last	(128)	
Batch normalization	(128)	512
Dense	(10)	1290
Output: Softmax	(10)	

Table 3. Architecture of our RNN for leitmotif classification. V indicates variable length.

Context	Strict			Variable			Fixed (10 sec.)		
	P	R	F	P	R	F	P	R	F
L-Ni	0.94	0.95	0.94	0.90	0.95	0.92	0.93	0.93	0.93
L-Ri	0.93	0.92	0.93	0.84	0.93	0.88	0.86	0.89	0.87
L-Mi	0.96	0.95	0.96	0.95	0.93	0.94	0.92	0.98	0.95
L-NH	0.94	0.92	0.93	0.96	0.88	0.92	0.97	0.87	0.92
L-RT	0.95	0.94	0.95	0.94	0.90	0.92	0.96	0.95	0.96
L-Wa	0.94	0.98	0.96	0.98	0.96	0.97	0.96	0.99	0.98
L-WL	0.98	0.93	0.96	0.93	0.93	0.93	0.95	0.94	0.94
L-Ho	0.90	0.89	0.89	0.93	0.85	0.89	0.92	0.91	0.91
L-Ge	0.94	0.94	0.94	0.93	0.91	0.92	0.97	0.94	0.96
L-Sc	0.91	0.96	0.93	0.94	0.89	0.92	0.84	0.86	0.85
Mean	0.94	0.94	0.94	0.93	0.91	0.92	0.93	0.92	0.93

Table 4. Main results of our method on the test set of the performance split for different strategies of using temporal context.

ples, where we adjust for tuning deviations (estimated automatically per performance and opera act). These steps are implemented using librosa [24]. Finally, all CQT frames are normalized using the max-norm and the resulting representations serve as inputs to our network.

Table 3 gives an overview of the network structure. We use an RNN-variant, the long short-term memory (LSTM) proposed in [25]. We stack multiple LSTM layers and, after the final LSTM output, append batch normalization [26] as well as a single fully connected classification layer to obtain leitmotif predictions. We set the number of LSTM layers and the size of their internal representation to 3 and 128, respectively. We train this network for 50 epochs by minimizing the cross-entropy loss between predictions and correct classes using the Adam optimizer [27] with a learning rate of 0.001 on mini-batches of 32 excerpts. Since the excerpts in a batch may have different lengths, we need to zero-pad them to the maximum number of frames among excerpts in that batch. During computation, we then use masking to ignore zeros added to shorter inputs. We further avoid overfitting by selecting the weights of the epoch that yields the highest mean F-measure on the validation set (as described in Section 4.2). The network is implemented in Python using Tensorflow.

4. EXPERIMENTS

4.1 Setup and Splits

We follow the common machine learning approach of partitioning our dataset into training, validation, and test subsets to train, tune hyperparameters, and estimate the results on unseen samples, respectively. In contrast to standard procedures, we partition the data according to musical aspects as motivated in Section 2.3. We will consider two splits: the performance and occurrence splits.

For the performance split, we select the three recordings with manually annotated measure positions (P-Ba, P-Ha and P-Ka, see Table 2) for the test set and three performances with automatically transferred measure positions for the validation set (P-Sa, P-So and P-We). The remaining ten performances are used for training. In this split, all subsets comprise all occurrences of all motifs. Results on the performance split are given in Section 4.3.

In contrast, for the occurrence split, we randomly choose 80% of the occurrences for training and 10% each

for the validation and test set. [3] We further ensure that the proportions of occurrences for each motif is the same in all subsets. In this split, each subset contains all instances of the occurrences in that subset. Results on the occurrence split are given in Section 4.4.

4.2 Evaluation Measures

We adopt standard measures from information retrieval for evaluating our models. For a given class (i. e., motif), we treat the classification problem as a retrieval problem, yielding class-dependent precision (P), recall (R), and F-measure (F) as usual, see, e. g., [28].
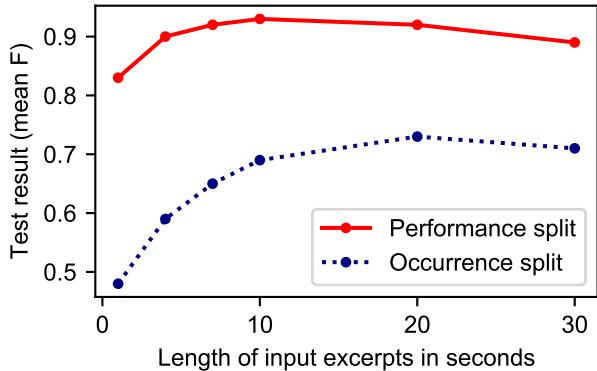
We also report the mean precision, recall, and F-measure over all classes. This gives a general impression of the classification quality. Note that these averages are not affected by class imbalance. Therefore, low results on an infrequent class will influence the mean results as much as low results on a frequent class.

4.3 Results on the Performance Split

Basic Experiment. The left block in Table 4 (*Strict*) summarizes results for our model on the test subset of the performance split. We obtain high classification results with a mean F-measure of 0.94. Results are similar across motifs. Highest precision (P = 0.98) is obtained for L-WL, while highest recall (R = 0.98) is reached for L-Wa. Recall and precision per motif are often similar. We conclude that it is indeed possible to classify leitmotif instances in previously unseen performances, provided that all occurrences were seen before in other performances. In the following, we expand on this result by considering other classification and split scenarios.

Temporal Context. In our basic experiment, we considered isolated leitmotif instances as input to our classification model, i. e., the audio excerpts to be classified start and end strictly at instance boundaries. We therefore call this the *Strict* scenario. Identifying leitmotifs when instance boundaries are not known in advance could pose an additional challenge. However, the temporal context before and

---

[3] The same occurrences are chosen in all experiments for comparability.

Figure 3. Mean F-measures for our model when using different input lengths in the *Fixed* scenario.

Context	*Strict*			*Variable*			*Fixed* (10 sec.)		
	P	R	F	P	R	F	P	R	F
L-Ni	0.67	0.80	0.73	0.67	0.86	0.75	0.80	0.91	0.85
L-Ri	0.36	0.41	0.38	0.44	0.43	0.43	0.49	0.67	0.56
L-Mi	0.79	0.87	0.83	0.82	0.80	0.81	0.97	0.96	0.97
L-NH	0.72	0.20	0.31	0.62	0.25	0.36	0.92	0.32	0.47
L-RT	0.57	0.65	0.61	0.60	0.77	0.68	0.71	0.91	0.80
L-Wa	0.87	0.80	0.84	0.81	0.88	0.84	0.95	0.95	0.95
L-WL	0.25	0.21	0.23	0.23	0.17	0.20	0.52	0.20	0.28
L-Ho	0.46	0.57	0.51	0.52	0.57	0.54	0.61	0.91	0.73
L-Ge	0.28	0.30	0.29	0.38	0.43	0.40	0.58	0.68	0.63
L-Sc	0.52	0.50	0.51	0.64	0.53	0.58	0.76	0.58	0.66
Mean	0.55	0.53	0.52	0.57	0.57	0.56	0.73	0.71	0.69

Table 5. Main results of our method on the test set of the occurrence split for different strategies of using temporal context.

after the instance boundaries might also be helpful in identifying the class of an excerpt. Next, we analyze the effect of temporal context on the leitmotif classification results.

To this end, we compare the *Strict* scenario with an alternative, called *Variable*, where we add a randomly chosen amount of temporal context to the input excerpts. Context may be added before and after the motif instance. More specifically, the excerpt length is at most doubled and the instance in question is not constrained to be in the excerpt center. Such use of context also prevents our model from relying on length and boundary properties of the leitmotif instances. The center block in Table 4 gives results for this scenario. Compared to the *Strict* case, the mean F-measure decreases slightly to 0.92.

We also perform experiments on fixed input lengths, which we call the *Fixed* scenario. Here, we randomly take subsections of an instance if it is longer than the fixed input length or add context before and after in case it is shorter. Mean F-measure values for different fixed input lengths are shown in Figure 3 (solid red line). The plot indicates that results decrease for lengths that are shorter than most instances,[4] e. g., one second. When a fixed length of ten seconds is chosen, which encompasses almost all instances in the dataset, results are comparable to the *Strict* case (see also the right block in Table 4). Longer inputs again yield lower results, which may be attributed to the difficulty posed by additional context. However, one should note that for such large durations, input excerpts are no longer guaranteed to contain instances of a single motif only and thus, our initial assumption on a single label per input may be violated.

In Section 5, we discuss how the results for different amounts of temporal context may be interpreted in the context of a leitmotif detection scenario.

Potential for Overfitting. Deep learning models often rely on features of the input that would be deemed task-irrelevant by human experts, see, e. g., [29, 30]. In our case, the correct class for each input may be inferred not only from musically relevant aspects of leitmotifs such as melody or rhythm (as given in Table 1), but also from confounding features of the excerpts such as instrument activ-

ity or volume. This is especially true for the performance split, where a classification model may predict correct outputs on the test set by merely memorizing all occurrences during training instead of distinguishing musically relevant features of the leitmotifs (we will revisit this possibility in Section 4.6). In contrast, for the occurrence split, the model needs to generalize to previously unseen realizations of the leitmotif classes and therefore needs to rely on their common musical characteristics.

4.4 Results on the Occurrence Split

Table 5 presents results for the occurrence split with different strategies for adding temporal context. Overall results are lower than for the performance split. In the *Strict* scenario, the obtained mean F-measure of 0.52 is substantially lower than for the performance split, but still well above chance (which corresponds to 0.1 mean F-measure). Results vary considerably among motifs, with F-measures ranging from 0.23 for L-WL to 0.84 for L-Wa. In addition, the differences between precision and recall per motif can be large as in the case of L-NH (P = 0.72 and R = 0.20). We conclude that classifying leitmotif instances for unknown occurrences is challenging but possible.

We further observe that—in contrast to the performance split—context is beneficial in the occurrence split. Mean F-measures of the *Variable* and *Fixed* scenarios increase to 0.56 and 0.69, respectively. Figure 3 shows F-measures for different amounts of context in the occurrence split (dotted blue line). Results increase for excerpt lengths up to ten seconds and then stabilize. We see two potential reasons for this. Firstly, by training with temporal context, the classifier may learn to identify features that indicate instance starts and ends, which could be helpful for identifying instances in the test set. Secondly, however, longer temporal context also means that instances from the training set may occur in the context added to validation and test instances. Indeed, we observed that for a context length of 10 seconds, 67% of test excerpts overlap with a training instance of the same class, while 8% overlap with a training instance of another class. Predicting the class of known training occurrences would therefore yield good results on

---

[4] Statistics on instance lengths are given in Table 1 (rightmost column).

Split	Performance			Occurrence		
	P	**R**	**F**	**P**	**R**	**F**
Noise	0.90	0.87	0.89	0.32	0.36	0.34
L-Ni	0.90	0.95	0.93	0.63	0.74	0.68
L-Ri	0.89	0.89	0.89	0.28	0.32	0.30
L-Mi	0.94	0.93	0.94	0.78	0.75	0.76
L-NH	0.95	0.88	0.91	0.52	0.28	0.37
L-RT	0.93	0.93	0.93	0.54	0.73	0.63
L-Wa	0.93	0.96	0.94	0.79	0.79	0.79
L-WL	0.94	0.93	0.94	0.17	0.12	0.14
L-Ho	0.89	0.87	0.88	0.40	0.45	0.42
L-Ge	0.91	0.91	0.91	0.20	0.18	0.19
L-Sc	0.90	0.95	0.93	0.68	0.38	0.49
Mean	0.92	0.92	0.92	0.48	0.46	0.46

Table 6. Results of our method when incorporating a noise class in the performance and the occurrence split. No temporal context is added (*Strict* scenario).

the test set. The results for adding temporal context may thus partly be explained by overfitting to the training set.

4.5 Noise Class

So far, we only considered excerpts that contain one of ten leitmotifs. However, the *Ring* also contains regions with other or with no leitmotifs at all. Because of this, we also perform experiments with an additional Noise class, denoting excerpts where none of the leitmotifs in our selection are being played. We evaluate whether our model is able to correctly classify our selection of leitmotifs in the presence of this noise class, both for the performance and the occurrence split. To do so, we randomly select 400 Noise occurrences from the *Ring*, leading to 6400 Noise instances. The model described in Section 3 remains unchanged except for the final classification layer, which now has eleven outputs.

Results are given in Table 6. For the performance split, the additional noise class does not change results by much. Leitmotif classes obtain somewhat lower results (e. g., P = 0.90 for L-Ni compared to P = 0.94 in Table 4) while the noise class yields an F-measure lower than most leitmotif classes (F = 0.89). For the occurrence split, results for the leitmotif classes again decrease slightly (e. g. P = 0.63 for L-Ni compared to P = 0.67 in Table 5), while the noise class itself is especially hard to distinguish (F = 0.34). In both splits, the noise class does not lead to a complete deterioration of results. Section 5 discusses the implications of this for the task of leitmotif detection.

4.6 Random Labels

In all experiments, our model has consistently obtained higher results on the performance than on the occurrence split. As discussed at the end of Section 4.3, the latter split requires generalizing to new musical realizations of a motif. In contrast, the performance split could be tackled by memorizing all leitmotif occurrences, which is not possible on the occurrence split.

To further investigate the gap in results between performance and occurrence split, we now evaluate our model's capability to memorize input features on the performance

split. To do so, we create a variant of the performance split where we assign a random class label from one to ten to each occurrence. Thus, while occurrences are labeled consistently across performances, their classes no longer correspond to leitmotifs. In this variant of the performance split, the class of a test excerpt can only be obtained by memorizing classes for occurrences during training and not by learning common properties of all occurrences for a motif. This random-labeling experiment is inspired by [31].

When training our model on this variant, we obtain a mean F-measure of 0.54 on the test set after 50 epochs, which is much lower than the 0.94 obtained for the original labels (see Table 4). We observed that training for this experiment had not converged after 50 epochs and trained for an additional 75 epochs, leading to an F-measure of 0.57. The faster convergence and higher results on the original labels suggest that our model does learn some relevant characteristics of leitmotifs. Our experiment shows, however, that memorizing excerpts may also contribute to the results.

5. SUMMARY AND FUTURE WORK

In this work, we evaluated the capability of a neural network classification model for identifying leitmotifs in audio excerpts. Despite the complex musical variabilities in this scenario, our RNN-based classification model is able to differentiate between a fixed set of motifs and to distinguish them from non-motif excerpts. Generalization is strong across performances and—to a lesser extent—across occurrences. Using temporal context is helpful in the latter case, although the improvement may partly be the result of overfitting.

Our results encourage the development of a system for automated detection of motif instances in full performances. Unlike the classification task, no pre-segmented instance boundaries would be available for detection. We therefore expect this to be a more challenging scenario.

In our experiments, we have already explored the use of fixed input lengths. Using these, our model may be applied to all positions in an audio recording in a sliding window fashion [32]. This way, we can obtain leitmotif predictions for an entire performance of the *Ring* and not just individual excerpts. Additionally, a model used for automated leitmotif detection from audio would also need to deal with input excerpts that do not contain any leitmotifs at all. Our experiments with a noise class suggest that this may lead to somewhat lower but still useful results.

Furthermore, a detection system would need to handle a much larger number of motifs (around 130 for the complete *Ring*) as well as excerpts containing multiple motifs played simultaneously. Multi-label extensions of our model on fixed input lengths may be suitable for this.

As an even more advanced scenario, one may imagine an informed detection setting in which instances of a previously unseen motif must be identified given only a few exemplary instances of that motif.

6. REFERENCES

[1] H. M. Brown, E. Rosand, R. Strohm, R. Parker, A. Whittall, R. Savage, and B. Millington, "Opera," in *The New Grove Dictionary of Music and Musicians*, 2nd ed., S. Sadie, Ed. London: Macmillian Publishers, 2001, pp. 416–471.

[2] M. Bribitzer-Stull, *Understanding the Leitmotif.* Cambridge University Press, 2015.

[3] V. Konz, M. Müller, and R. Kleinertz, "A cross-version chord labelling approach for exploring harmonic structures—a case study on Beethoven's Appassionata," *Journal of New Music Research*, vol. 42, no. 1, pp. 61–77, 2013.

[4] H. Schreiber, C. Weiß, and M. Müller, "Local key estimation in classical music recordings: A cross-version study on Schubert's Winterreise," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain, 2020.

[5] C. Weiß, F. Zalkow, M. Müller, S. Klauk, and R. Kleinertz, "Computergestützte Visualisierung harmonischer Verläufe: Eine Fallstudie zu Wagners Ring," in *Proceedings of the Jahrestagung der Gesellschaft für Informatik (GI)*, Chemnitz, Germany, 2017, pp. 205–217.

[6] F. Zalkow, C. Weiß, and M. Müller, "Exploring tonal-dramatic relationships in Richard Wagner's Ring cycle," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Suzhou, China, 2017, pp. 642–648.

[7] K. R. Page, T. Nurmikko-Fuller, C. Rindfleisch, D. M. Weigl, R. Lewis, L. Dreyfus, and D. D. Roure, "A toolkit for live annotation of opera performance: Experiences capturing Wagner's Ring cycle," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Málaga, Spain, 2015, pp. 211–217.

[8] C. Weiß, V. Arifi-Müller, T. Prätzlich, R. Kleinertz, and M. Müller, "Analyzing measure annotations for Western classical music recordings," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, New York, USA, 2016, pp. 517–523.

[9] F. Zalkow, C. Weiß, T. Prätzlich, V. Arifi-Müller, and M. Müller, "A multi-version approach for transferring measure annotations between music recordings," in *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, 2017, pp. 148–155.

[10] D. J. Baker and D. Müllensiefen, "Perception of leitmotives in Richard Wagner's Der Ring des Nibelungen," *Frontiers in Psychology*, vol. 8, p. 662, 2017.

[11] Y. Morimoto, T. Kamekawa, and A. Marui, "Verbal effect on memorisation and recognition of Wagner's leitmotifs," in *Triennial Conference of the European Society for the Cognitive Sciences of Music (ESCOM)*, 2009.

[12] H. Albrecht and K. Frieler, "The perception and recognition of Wagnerian leitmotifs in multimodal conditions," in *Proceedings of the International Conference of Students of Systematic Musicology (SysMus)*, London, UK, 2014.

[13] D. Müllensiefen, D. Baker, C. Rhodes, T. Crawford, and L. Dreyfus, "Recognition of leitmotives in Richard Wagner's music: An item response theory approach," in *Analysis of Large and Complex Data*. Cham, Switzerland: Springer, 2016, pp. 473–483.

[14] R. Wagner, *Opera and Drama.* University of Nebraska Press, 1995, translation of the original edition from 1851.

[15] L. Dreyfus and C. Rindfleisch, "Using digital libraries in the research of the reception and interpretation of Richard Wagner's leitmotifs," in *Proceedings of the International Workshop on Digital Libraries for Musicology*, London, UK, 2014, p. 1–3.

[16] R. Wagner, *Der Ring des Nibelungen. Vollständiger Text mit Notentafeln der Leitmotive*, J. Burghold, Ed. Mainz: Schott Music, 2013, reprint of the original edition from 1913 (Ed. Julius Burghold).

[17] ——, "On the application of music to the drama," in *Prose Works.* Broude Brothers, New York, 1966, pp. 175–191, translation of the original edition from 1879.

[18] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of audio-based music similarity and genre classification," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, London, UK, 2005, pp. 628–633.

[19] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 637–644.

[20] F. Korzeniowski and G. Widmer, "Genre-agnostic key classification with convolutional neural networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 264–270.

[21] I. Jeong and K. Lee, "Learning temporal features using a deep neural network and its application to music genre classification," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016, pp. 434–440.

[22] J. C. Brown, "Calculation of a constant Q spectral transform," *Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[23] C. Schörkhuber and A. P. Klapuri, "Constant-Q transform toolbox for music processing," in *Proceedings of the Sound and Music Computing Conference (SMC)*, Barcelona, Spain, 2010, pp. 3–64.

[24] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "Librosa: Audio and music signal analysis in python," in *Proceedings the Python Science Conference*, Austin, Texas, USA, 2015, pp. 18–25.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, November 1997.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 448–456.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference for Learning Representations (ICLR)*, San Diego, California, USA, 2015.

[28] M. Müller, *Fundamentals of Music Processing*. Springer Verlag, 2015.

[29] B. L. Sturm, "The "horse" inside: Seeking causes behind the behaviors of music content analysis systems," *Computers in Entertainment*, vol. 14, no. 2, pp. 3:1–3:32, 2016.

[30] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, BC, Canada, 2019, pp. 125–136.

[31] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[32] P. Grosche and M. Müller, "Toward characteristic audio shingles for efficient cross-version music retrieval," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, 2012, pp. 473–476.

CAMERA-BASED PIANO SHEET MUSIC IDENTIFICATION

Daniel Yang
Harvey Mudd College
dhyang@g.hmc.edu

TJ Tsai
Harvey Mudd College
ttsai@g.hmc.edu

ABSTRACT

This paper presents a method for large-scale retrieval of piano sheet music images. Our work differs from previous studies on sheet music retrieval in two ways. First, we investigate the problem at a much larger scale than previous studies, using all solo piano sheet music images in the entire IMSLP dataset as a searchable database. Second, we use cell phone images of sheet music as our input queries, which lends itself to a practical, user-facing application. We show that a previously proposed fingerprinting method for sheet music retrieval is far too slow for a real-time application, and we diagnose its shortcomings. We propose a novel hashing scheme called dynamic n-gram fingerprinting that significantly reduces runtime while simultaneously boosting retrieval accuracy. In experiments on IMSLP data, our proposed method achieves a mean reciprocal rank of 0.85 and an average runtime of 0.98 seconds per query.

1. INTRODUCTION

Imagine the following scenario. A musician is sitting down in front of a piano learning a new piece of music. She pulls out her cell phone, takes a picture of the physical page of sheet music sitting in front of her, and is immediately able to access Youtube videos of performances of that piece and alternate editions of the sheet music. In this paper, we present a method to solve the main technical challenge of identifying the page of music. This is the camera-based piano sheet music identification task.

Most previous works on sheet music retrieval come from the literature on finding correspondences between audio and sheet music. There are three general approaches to the cross-modal retrieval problem. The first approach is to convert the sheet music into MIDI using optical music recognition (OMR), to compute chroma-like features on the MIDI, and then to compare the result to chroma features extracted from audio. This approach has been applied to audio–sheet music synchronization [1] [2] [3] [4], and it translates very naturally to retrieval applications like using a segment of sheet music to identify its corresponding audio recording [5] or to retrieve the corresponding temporal

passage from a specific audio recording [6]. The second approach is similar to the first, except that it replaces the full OMR with a mid-level feature representation based on the location of noteheads relative to the staff lines [7] [8]. The third approach is to train a multimodal convolutional neural network to learn a latent feature space that directly encodes similarity between chunks of audio and sheet music snippets. This approach has been applied to audio–sheet music alignment [9] [10] and retrieval applications like using a snippet of audio to retrieve its corresponding sheet music snippet and vice versa [10] [11] [12] [13]. See [14] for an overview of work on cross-modal retrieval of music data. Also, we note that a recent work [15] has proposed a neural network-based approach for finding corresponding measures between two different sheet music versions of a piece.

This current study differs from previous work in two ways. First, we study the sheet music retrieval problem at a much larger scale. Previous works have studied sheet music retrieval using searchable databases containing hundreds of sheet music scores or a few thousand short snippets of sheet music. In contrast, we perform experiments using all solo piano sheet music scores in the entire International Library Music Score Project (IMSLP) [1] as a searchable database. We believe that this is several orders of magnitude larger than any previous study on sheet music retrieval. Second, we focus on queries that are cell phone images of sheet music. Previous works have primarily focused on synthetic sheet music, scanned sheet music, and audio recordings as input queries. While there have been a handful of works that study OMR on cell phone pictures of sheet music [16] [17] [18] [19] [20], this area of study is still in its infancy. Even though using cell phone pictures arguably makes the task much more challenging due to the additional sources of noise and distortion, we believe that this change leads to a much more practical, user-facing application.

Our approach to the piano sheet music identification task is to combine a recently proposed bootleg score feature representation with a novel hashing scheme. The bootleg score feature was originally proposed for a MIDI–sheet music alignment task [8]. A recent work has explored using the bootleg score features in a hashing framework for de-anonymizing files in the Lakh MIDI dataset by finding matches in sheet music data [21]. We will show that this previously proposed fingerprinting approach is far too slow for our current scenario. Because our task is a real-

---

[1] https://imslp.org

time application, latency is an extremely important factor (unlike [21], which is an offline task). In this work, we impose a hard constraint that our system must have an average runtime of 1 second or less. We diagnose the reason why this previously proposed fingerprinting scheme is slow, and we develop a novel fingerprinting scheme that is able to achieve our stringent runtime constraint.

This paper has three main contributions. First, we propose a novel hashing scheme called dynamic n-gram fingerprinting. This approach dynamically constructs n-gram fingerprints of variable length in order to ensure that each fingerprint is discriminative enough to warrant a table lookup. Second, we present empirical validation of our proposed method on a very large-scale retrieval task. We perform experiments using all solo piano scores in IMSLP as a searchable database. We show that dynamic n-gram fingerprinting achieves both higher retrieval accuracy and significantly lower runtimes than a previously proposed approach. Our best system achieves a mean reciprocal rank of 0.85 and has an average runtime of 0.98 seconds per query. Third, as a byproduct of this project, we release the precomputed bootleg score features on all piano scores in IMSLP. [2] Because this task required a tremendous amount of time and computation involving the use of a supercomputing infrastructure, we release the features as a standalone repository in the hopes that it will be useful in a variety of other MIR-related tasks.

2. SYSTEM DESCRIPTION

Figure 1 shows the architecture of our proposed system. We will describe the system in two parts: constructing the database and performing a search at runtime.
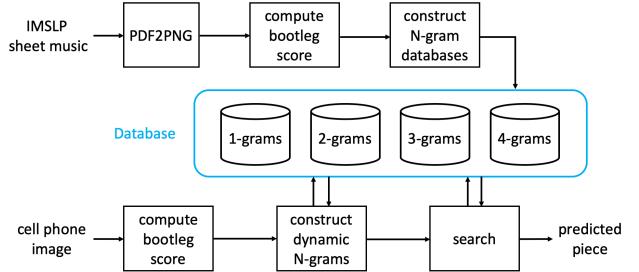
2.1 Database Construction

Our first goal is to construct a database which will enable us to perform searches very efficiently. The process of constructing this database consists of three steps, as shown in the upper half of Figure 1. These three steps will be described in the next three paragraphs.

The first step is to convert each sheet music PDF into a sequence of PNG images. We decode the PDF into PNG images at 300 dpi, and then resize each image to have a width of 2550 pixels while preserving the aspect ratio. Because there is an extremely large range of image sizes in the IMSLP dataset, we resize the images to ensure that they are within a range that the bootleg score feature computation was designed for.

The second step is to compute a bootleg score for each page. The bootleg score is a recently proposed feature representation of piano sheet music that encodes the position of filled noteheads relative to staff lines [8]. The bootleg score representation itself is a $62 \times N$ binary matrix, where 62 indicates the total number of possible staff line positions in both the left and right hands, and where

Figure 1. Overview of proposed system.

N indicates the total estimated number of simultaneous note events. Figure 2 shows a short section of sheet music and its corresponding bootleg score representation. Note that this representation discards a significant amount of information like duration, key signature, accidentals, octave markings, and clef changes, and it simply ignores non-filled noteheads (e.g. half or whole notes). Nonetheless, it has been shown to be effective in aligning sheet music and MIDI, and we hypothesize that it may also be used effectively for large-scale retrieval. The main benefit of using the bootleg score representation over a full OMR pipeline is processing time. Because OMR is typically cast as an offline task, the best-performing systems require a significant amount of computation. [3] In contrast, the bootleg score can be computed on a high-resolution image in less than 1 second using a CPU. By focusing exclusively on simple geometrical shapes like circles (filled noteheads) and lines (staff lines and bar lines), it can detect objects robustly and efficiently using classical computer vision techniques.

The third step is to construct the n-gram databases. The concept of an n-gram is adopted from the language modeling literature, where the likelihood of a sequence of N consecutive words is estimated based on the frequency of its occurrence in a large set of data. Here, we treat each bootleg score column as a word and consider N consecutive words as a single fingerprint. We generate four separate n-gram databases for $N = 1, 2, 3, 4$. Each n-gram database is constructed in the following manner. First, we concatenate the bootleg score features from all pages into a single, global bootleg score for each PDF. Second, we represent each bootleg score column as a single 64-bit integer. This allows us to represent the bootleg score very compactly as a sequence of integers. Third, we consider every n-gram in the sequence as a fingerprint. For example, if a bootleg score is given by a sequence of 64-bit integers $x_1, x_2, x_3, \cdots$, then the set of 3-gram fingerprints for this bootleg score is given by $(x_1, x_2, x_3), (x_2, x_3, x_4), (x_3, x_4, x_5), \cdots$. Fourth, we store the location information for each fingerprint in a reverse index. For each n-gram database, the hash key is a $(64 \cdot N)$-bit fingerprint, and the reverse index stores a list of $(id, offset)$ tuples for all occurrences of that fingerprint in the database, where id is a unique identifier for the PDF

---

[3] For example, in a recent survey on state-of-the-art music object detectors [22], the best performing system required 40-80 seconds to process each image using a GPU.

Figure 2. A short section of sheet music and its corresponding bootleg score.

and *offset* specifies the offset in the bootleg score.

2.2 Search

At runtime, our goal is to identify the piece of music showing in a cell phone image query. This process consists of three steps, as shown in the bottom half of Figure 1.

The first step is to compute a bootleg score on the cell phone image. This is done using the same feature extraction as in the database construction phase. Note that the inputs in the offline and online phases are very different: whereas the IMSLP data is primarily digital scans of physical sheet music, the queries are cell phone images of physical sheet music. This introduces a lot of noise due to variable lighting conditions, zoom, camera angle, cropping, blur, unwanted objects outside the boundaries of the page, etc. The only reason we can get away with using the same feature extractor in these two very different scenarios is that the bootleg score feature extraction has no trainable weights and only a small set of hyperparameters. This makes it less likely to highly overfit to a set of data. It is also worth pointing out that the bootleg score feature extraction was originally designed to handle the challenging case of cell phone images of sheet music, so we surmise that it will handle the easier case of scanned sheet music reasonably well.

The second step is to construct a sequence of dynamic n-gram fingerprints. We will explain and motivate the use of dynamic n-grams by describing our initial attempts to solve the problem, the issues with these earlier approaches, and how the dynamic n-gram addresses these issues.

Our initial attempt was to consider each column of the bootleg score as a fingerprint. This is equivalent to a 1-gram in the terminology used in this paper. This approach was proposed in a recent work that attempts to de-anonymize files in the Lakh MIDI dataset by finding matches in a set of known sheet music data [21]. When we implemented this approach, we found that the retrieval accuracy was good, but that the system was far too slow. This is an acceptable solution in [21] because the task is offline, but it is an unacceptable solution in our current application because we have a very stringent runtime constraint. Upon further analysis, we found that the frequency distribution of fingerprints was highly peaked and thus ill-suited for hashing. In other words, there was a small set of finger-

prints that occurred very frequently in the database. These fingerprints tended to be bootleg score columns containing a single note event. Because this occurs so frequently in piano sheet music, it forces the system to process an extremely large number of spurious fingerprints at runtime, which significantly slows down the system.

Our second attempt was to use an n-gram fingerprint to address this issue. This introduces a tradeoff. On the one hand, as we increase N the fingerprint becomes more discriminative, which leads to fewer matches in the database and faster runtime. On the other hand, increasing N increases the likelihood that the fingerprint is erroneous, since the entire fingerprint is wrong if even one of its elements has an error. If we roughly model each n-gram as N, independent Bernoulli random variables, then the probability that the entire n-gram is correct decreases exponentially in N. Given this tradeoff, one very reasonable approach is to try different values of N and to select the value that yields the best performance.

The dynamic n-gram gets the best of both worlds. If a single bootleg word x_i (i.e. a bootleg score column converted to a 64-bit integer) is very distinctive, then we simply do a table lookup in the 1-gram database. In this case, it would not benefit us to do a 5-gram lookup if the first element only occurs a few times in the whole database. If, however, the bootleg word is very common, then we prefer not to do a table lookup on the 1-gram database because this would require us to process a large number of spurious fingerprints. In this case, we bump the 1-gram up to a 2-gram and repeat the process. If the 2-gram fingerprint (x_i, x_{i+1}) is distinctive, then we do a table lookup on the 2-gram database. If (x_i, x_{i+1}) is not distinctive, then we bump the 2-gram up to a 3-gram (x_i, x_{i+1}, x_{i+2}). We repeat this process until the fingerprint is distinctive enough to warrant doing a table lookup (up to $N = 4$). As an example, given a sequence of bootleg words $x_1, x_2, x_3, \cdots$, one possible dynamic n-gram sequence would be $(x_1), (x_2, x_3), (x_3), (x_4, x_5, x_6), \cdots$. Note that there is only one hyperparameter γ that specifies the maximum number of fingerprint matches we are willing to process for every table lookup.

The third step is to search the database using the histogram of offsets method. The histogram of offsets was proposed in [23] as a way to efficiently search a very large database. It is based on the observation that the true match in the database will yield a sequence of matching fingerprints at an approximately constant relative offset. For example, if a query bootleg word sequence $x_1, x_2, x_3, \cdots$ matches a reference sequence $\tilde{x}_i, \tilde{x}_{i+1}, \tilde{x}_{i+2}, \cdots$, then the matching fingerprints would all have a relative offset of $i - 1$. If we compute a histogram of relative offsets for all matching fingerprints with that item in the database, we would see a large spike in the histogram at the bin corresponding to an offset of $i - 1$. We can therefore compute a similarity score by constructing a histogram of offsets for matching fingerprints, and then calculate the maximum bin count in the histogram. Once we have calculated a match score for every PDF in the database in this way, we group

the PDFs by piece. The *piece* match score is calculated as the maximum score of any of its constituent PDFs. Finally, we sort the pieces in the database by their match scores. This yields our final predicted ranked list of pieces.

3. EXPERIMENTAL SETUP

In this section, we describe the data and metrics used to evaluate our proposed system.

The cell phone image queries were taken from the Sheet MIDI Retrieval dataset [24]. We include a description of the dataset here for completeness. This data was originally created to study the task of aligning a cell phone picture of piano sheet music and its corresponding MIDI file. It contains 2000 cell phone images of 200 different piano pieces across 25 well-known composers. For each of the 200 pieces, a PDF from IMSLP was downloaded and printed onto physical paper. Ten cell phone pictures were taken across the length of each piece in a variety of locations, lighting conditions, perspectives, and levels of zoom. The pictures contain between 1 and 5 lines of music and were all taken in landscape orientation. We use the same train-test split as the original paper: 400 cell phone images (corresponding to 40 pieces) were used for training, and 1600 cell phone images (corresponding to 160 pieces) were used for testing. By using the same train-test split as the original paper, we ensure that the bootleg score feature extraction has not been tuned to the test data.

The database comes from IMSLP. We first scraped the website and downloaded all PDF scores and associated metadata.[4] We then filtered the data by instrumentation tag label in order to identify a list of solo piano pieces. The resulting dataset contains 29,310 pieces and 31,384 PDFs and 374,758 individual images. This is the dataset that we used to construct the database described in Section 2.1.

We release the precomputed bootleg score features for all piano scores in the IMSLP dataset in a separate standalone repository.[5] We believe this is in itself a significant contribution to the MIR research community, given the amount of time, memory, and computation required to generate it. For example, it took us over a month to scrape the IMSLP website and download all the scores in PDF format. This set of PDFs was approximately 1.2 terabytes in size. If the PDFs had been decoded into high-resolution images, the dataset would be in the tens of terabytes. Because this was too large to store on disk, we decompressed each PDF to a set of high-resolution images, computed the bootleg score features, and then deleted the high-resolution images to conserve disk space. We performed all feature computation on the NSF XSEDE supercomputing infrastructure [25].

We evaluate our system along two dimensions: retrieval accuracy and runtime. Because our goal is to identify the matching *piece* rather than just the exact same PDF,[6] there

is always exactly one correct item in the database. Accordingly, we use mean reciprocal rank (MRR) as our measure of retrieval accuracy. The MRR is calculated as

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{R_i} \qquad (1)$$

where $N = 1600$ indicates the number of test queries and R_i indicates the rank of the true matching item for the ith query. In our task, R_i can range between 1 and 29310, the total number of pieces in the database. MRR ranges between 0 and 1, where 1 indicates perfect performance. We also measure the runtime required to process each test query. The runtime includes all data pre-processing such as converting the JPG image to PNG format. Note, however, that the runtimes do *not* include the network latency that would be present in a real cell phone application. All experiments are done on a single core of a 2.1 GHz Intel Xeon CPU.

4. RESULTS

In this section, we present our experimental results on the piano sheet music identification task.

We are not aware of previous work that directly studies sheet music identification based on cell phone images. As mentioned in Section 1, there are works in the audio–sheet music alignment literature that have studied cross-modal sheet music retrieval. These works could in principle serve as baseline comparisons. However, all of the works we are aware of would not have been practical to evaluate on our task for one of two reasons. First, some approaches do not scale to a large database. For example, approaches that use subsequence DTW [5] [6] [2] [8] would have exorbitantly high runtimes on the IMSLP database. Second, some approaches might have acceptable runtimes at test time, but would have required too much computation to construct the database. For example, the sheet–audio alignment system in [11] is 20 times slower than our proposed system and would have exceeded our computational budget on the XSEDE supercomputing infrastructure. Any approaches that use OMR to convert the sheet music into MIDI format would likewise be too computationally expensive.

Nonetheless, we compare our proposed approach to nine other baseline systems. The first four baselines are representative of the state-of-the-art in image retrieval in the computer vision community. These systems were developed for the Oxford 5k [26] and Paris 6k [27] benchmarks, where the goal is to identify a famous landmark in a query image given a database of known images. All four systems are built on top of pretrained ImageNet classifiers like VGG [28] and ResNet [29], but they differ in the method by which they convert model activations into a final feature representation. The first baseline (MAC [30]) takes the $K \times W \times H$ tensor of activations at the last convolutional layer and computes the maximum activation within each feature map. This yields a fixed-size K-dimensional feature representation regardless of the image size. The second baseline (SPoC [31]) adopts a sim-

---

[4] We downloaded the data over the span of several weeks in May 2018.
[5] https://github.com/HMC-MIR/piano_bootleg_scores
[6] In Section 5.2, we will test how well our system can identify a piece when only an alternate edition of the sheet music exists in the database.

System	MRR		Runtime	
	cond 1	cond 2	avg	std
MAC [30]	.037	.043	1.17s	.12s
SPoC [31]	.003	.004	1.14s	.10s
GeM [32]	.025	.029	1.18s	.11s
R-MAC [33]	.036	.039	.96s	.11s
1-gram [21]	.709	.659	21.5s	12.5s
2-gram	.845	.784	2.76s	1.11s
3-gram	.808	.767	1.99s	.36s
4-gram	.755	.722	1.12s	.25s
5-gram	.688	.668	1.07s	.13s
dynamic n-gram	.853	.812	.98s	.12s

Table 1. System performance on the piano sheet music identification task. Condition 1 is when the exact same PDF exists in the database. Condition 2 is when only an alternate version of the sheet music is in the database.

ilar approach, but uses average pooling rather than max pooling. The third baseline (GeM [32]) uses generalized mean pooling, which is a generalization of both average and max pooling where the type of pooling is specified by a single, trainable parameter. The fourth baseline (R-MAC [33]) applies max pooling over different regions of the image at various scales and combines the results through another pooling stage. All four baselines also apply various forms of post-processing, such as dimensionality reduction through principal component analysis, whitening, and L2 normalization. Given a query feature representation, similarity with database images is computed with a simple inner product. In our experiments, we compute piece similarity as the maximum similarity with any page in any of the piece's constituent PDFs. We evaluate the baseline systems with their provided pretrained models.[7] The last five baselines are equivalent to our proposed system but using a fixed n-gram fingerprint for $N = 1, 2, 3, 4, 5$. The 1-gram system corresponds to the approach proposed in [21].

Table 1 compares the performance of all models. The systems are presented in three groups: the image retrieval baselines (top), the fixed n-gram systems (middle), and the proposed dynamic n-gram system (bottom). The second column (labeled "cond 1") indicates the MRR on the test set, and the last two columns indicate the average runtime per query and corresponding standard deviation. The column labeled "cond 2" will be discussed in Section 5.2.

There are three things to notice about these results. First, the image retrieval baselines all perform very poorly. The best-performing image retrieval system is MAC, which achieves an MRR of .037. This is not a surprise, since these systems were not designed for working with sheet music images, but it does confirm that existing image retrieval systems do not work out-of-the-box on the sheet music identification task. We do observe, however, that the

systems achieve results significantly better than random guessing (approximately .001 MRR). Second, the fixed n-gram systems show a tradeoff between retrieval accuracy and runtime. As N increases from 1 to 5, we see the average runtime decrease from 21.5 seconds to 1.07 seconds. This reflects the fact that the fingerprint is becoming more and more discriminative, which leads to fewer and fewer matches in the database. At the same time, we observe that the retrieval accuracy decreases from .845 to .688 as N increases from 2 to 5. This reflects the fact that more and more fingerprints are erroneous as fingerprint size increases. The increase in MRR from $N = 1$ to $N = 2$ indicates that the 1-gram fingerprints are not sufficiently distinctive. Third, the dynamic n-gram system achieves both the highest retrieval accuracy (.853) and the lowest average runtime (0.98 seconds). This indicates that the design has achieved its intended goal: to avoid the tradeoff between retrieval accuracy and runtime, and to instead get the best of both worlds.

5. ANALYSIS

In this section, we conduct four different analyses to answer key questions of interest.

5.1 Failure Modes

The first question of interest is, "What are the failure modes of the system?" To answer this question, we identified the queries with poor reciprocal rank values and investigated the reasons for failure. By far the biggest reason for poor performance was failure in the bootleg score feature computation. Common mistakes included missed detection of non-filled noteheads or noteheads occurring in block chords, notehead detection false alarms arising from text and other musical symbols on the page, and staff line estimation errors. Fixing these issues would require re-designing the bootleg feature computation. Another (minor) reason for poor performance came from non-distinctive sections of music. For example, when there are repetitive octaves or long sequences of alternating between two notes in only one hand, this can have a strong match with unrelated pieces of music.

5.2 Effect of Sheet Music Version

The second question of interest is, "How well does the system handle different sheet music versions?" To answer this question, we ran a separate set of experiments in which we remove the exact same PDF from the database. This means that the system can only match against alternate versions of the sheet music. Because some queries only had 1 sheet music version in IMSLP, this additional benchmark was run on a reduced subset of 930 test queries.

The results of this alternate benchmark are indicated in Table 1 as "Condition 2." We see that the MRR of the fixed n-gram systems has been reduced somewhere between .02 and .06, and the MRR of the dynamic n-gram system is reduced by .04. This performance gap between condition

---

[7] Training the baseline systems from scratch would require a large amount of labeled data (to retrain the ImageNet classifier) and would constitute a significant research project on its own. In this work, we simply evaluate the baseline systems out-of-the-box using the provided pretrained models.

System	MRR		Runtime	
	cond 1	cond 2	avg	std
dyn n-gram (20k)	.864	.802	1.67s	.18s
dyn n-gram (10k)	.865	.802	1.53s	.16s
dyn n-gram (5k)	.860	.803	1.21s	.15s
dyn n-gram (1k)	.853	.812	.98s	.12s

Table 2. Comparison of dynamic n-gram systems with various values of γ, which specifies the maximum number of fingerprint matches the system will process on a table lookup before bumping an N-gram to an $(N+1)$-gram.

1 and condition 2 can be interpreted as the additional performance loss that is caused by variations in different sheet music editions. While this is a nontrivial decrease in performance, the proposed system still has a robust overall retrieval accuracy (.812 MRR) when the exact same version is not in the database.

5.3 Effect of γ
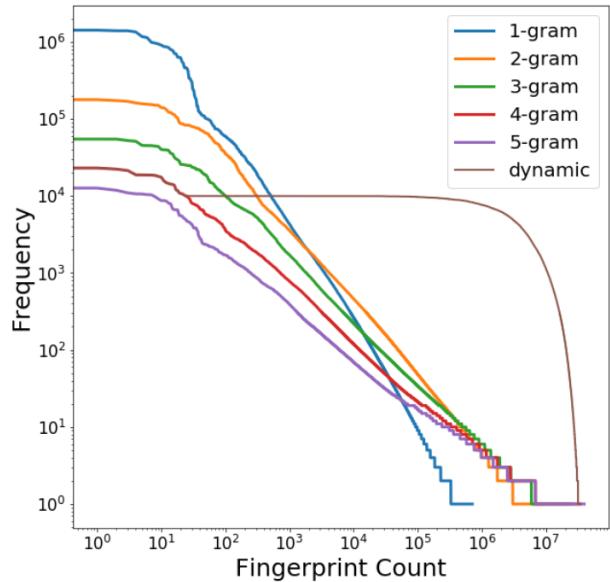
The third question of interest is, "How does system performance vary with γ?" Recall that the dynamic n-gram approach has one hyperparameter γ that specifies the maximum number of fingerprints we are willing to process for each table lookup. We ran experiments with several values of γ to determine its effect on system performance.

Table 2 shows system performance for γ ranging from 1000 to $20,000$. As γ decreases, we see a very slight decrease in retrieval accuracy (.864 to .853) and significant improvement in average runtime (1.67s to .98s). In this case, we have a very nice tradeoff: for only a small sacrifice in retrieval accuracy, we can significantly speed up the system. The dynamic n-gram results in Table 1 correspond to $\gamma = 1000$, which is the best system that meets our constraint of 1 second average runtime per query.

5.4 Fingerprint Distribution

The fourth question of interest is, "How well suited for hashing is the dynamic n-gram fingerprint distribution?" As described in Section 2.2, we found that the 1-gram fingerprint proposed in [21] had a frequency distribution that was very peaked and thus ill-suited for hashing. [8] To see how well the dynamic n-gram approach addresses this issue, we compared its frequency distribution to the fixed n-gram approaches.

Figure 3 shows this comparison. Each curve shows the frequency of different fingerprint values, where the fingerprints have been sorted from most frequent (left) to least frequent (right). Both axes are shown on a log scale in order to better visualize the wide dynamic range. Note that all of the fixed n-gram distributions have approximately the same total number of fingerprints in the database, so their only difference is how many unique fingerprint values there are and how the fingerprints are distributed across

Figure 3. Comparing the fingerprint frequency distributions of the fixed n-gram systems and dynamic n-gram with $\gamma = 10,000$. For each curve, the fingerprints have been ordered from most frequent (left) to least frequent (right).

these different values. The curve for the dynamic n-gram system corresponds to $\gamma = 10,000$.

Figure 3 shows the same trends that we see in Table 1. The difference, however, is that Figure 3 explains *why* the results in Table 1 are the way they are. For example, we observed earlier that the fixed n-gram systems exhibit a tradeoff between retrieval accuracy and runtime. Figure 3 explains this tradeoff from a hashing perspective. As N increases, the fixed n-gram distributions become less and less peaked, which translates to fewer fingerprint matches in the database and smaller runtimes. At the same time, this manner of reducing the peak comes with an exponential explosion in the number of unique fingerprint values. This means that the fingerprints will be less generalizable and more error-prone. We also notice that the only fixed n-gram curves that intersect early on are the 1-gram and 2-gram curves. This explains why the 2-gram system is unilaterally better than the 1-gram system: it has a less peaked distribution (smaller runtime) and it has a higher frequency of fingerprints than the 1-gram system for a large fraction of fingerprint values (better retrieval accuracy). Finally, we observe that the dynamic n-gram system has a flatter distribution than any of the fixed n-gram systems across a wide range of its distribution. This confirms that the dynamic n-gram approach is able to transform the distribution into one that is more well-suited for hashing. [9]

---

[8] Note that the ideal distribution for hashing is a uniform distribution.

[9] Note that the curve for the dynamic n-gram system has about 30 fingerprints that occur more than $\gamma = 10,000$ times. This is because we do not include n-grams higher than $N = 4$ in our database.

7. REFERENCES

[1] D. Damm, C. Fremerey, F. Kurth, M. Müller, and M. Clausen, "Multimodal presentation and browsing of music," in *Proc. of the International Conference on Multimodal Interfaces (ICMI)*, 2008, pp. 205–208.

[2] C. Fremerey, M. Müller, and M. Clausen, "Handling repeats and jumps in score-performance synchronization," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2010, pp. 243–248.

[3] F. Kurth, M. Müller, C. Fremerey, Y. Chang, and M. Clausen, "Automated synchronization of scanned sheet music with audio recordings," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2007, pp. 261–266.

[4] V. Thomas, C. Fremerey, M. Müller, and M. Clausen, "Linking sheet music and audio – challenges and new approaches," in *Multimodal Music Processing*, ser. Dagstuhl Follow-Ups, M. Müller, M. Goto, and M. Schedl, Eds. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2012, vol. 3, pp. 1–22.

[5] C. Fremerey, M. Müller, F. Kurth, and M. Clausen, "Automatic mapping of scanned sheet music to audio recordings," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2008, pp. 413–418.

[6] C. Fremerey, M. Clausen, S. Ewert, and M. Müller, "Sheet music-audio identification," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2009, pp. 645–650.

[7] Ö. İzmirli and G. Sharma, "Bridging printed music and audio through alignment using a mid-level score representation," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2012, pp. 61–66.

[8] D. Yang, T. Tanprasert, T. Jenrungrot, M. Shan, and T. Tsai, "MIDI passage retrieval using cell phone pictures of sheet music," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 916–923.

[9] M. Dorfer, A. Arzt, S. Böck, A. Durand, and G. Widmer, "Live score following on sheet music images," in *Late Breaking Demo at the International Conference on Music Information Retrieval (ISMIR)*, 2016.

[10] M. Dorfer, A. Arzt, and G. Widmer, "Learning audio-sheet music correspondences for score identification and offline alignment," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2017, pp. 115–122.

[11] M. Dorfer, J. Hajič, A. Arzt, H. Frostel, and G. Widmer, "Learning audio-sheet music correspondences for cross-modal retrieval and piece identification," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 22–33, 2018.

[12] M. Dorfer, J. Schlüter, A. Vall, F. Korzeniowski, and G. Widmer, "End-to-end cross-modality retrieval with CCA projections and pairwise ranking loss," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 117–128, 2018.

[13] S. Balke, M. Dorfer, L. Carvalho, A. Arzt, and G. Widmer, "Learning soft-attention models for tempo-invariant audio-sheet music retrieval," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2019, pp. 216–222.

[14] M. Müller, A. Arzt, S. Balke, M. Dorfer, and G. Widmer, "Cross-modal music retrieval and applications: An overview of key methodologies," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 52–62, 2019.

[15] S. Waloschek, A. Hadjakos, and A. Pacha, "Identification and cross-document alignment of measures in music score images," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 137–143.

[16] J. Calvo-Zaragoza and D. Rizo, "Camera-PrIMuS: Neural end-to-end optical music recognition on realistic monophonic scores," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2018, pp. 23–27.

[17] H.-N. Bui, I.-S. Na, and S.-H. Kim, "Staff line removal using line adjacency graph and staff line skeleton for camera-based printed music scores," in *IEEE International Conference on Pattern Recognition*, 2014, pp. 2787–2789.

[18] Q. N. Vo, S. H. Kim, H. J. Yang, and G. Lee, "An MRF model for binarization of music scores with complex background," *Pattern Recognition Letters*, vol. 69, pp. 88–95, 2016.

[19] A. R. Blanes and A. F. Bisquerra, "Camera-based optical music recognition using a convolutional neural network," in *IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 2, 2017, pp. 27–28.

[20] Q. N. Vo, T. Nguyen, S.-H. Kim, H.-J. Yang, and G.-S. Lee, "Distorted music score recognition without staffline removal," in *IEEE International Conference on Pattern Recognition*, 2014, pp. 2956–2960.

[21] T. Tsai, "Towards linking the lakh and IMSLP datasets," in *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020, pp. 546–550.

[22] A. Pacha, J. Hajič, and J. Calvo-Zaragoza, "A baseline for general music object detection with deep learning," *Applied Sciences*, vol. 8, no. 9, p. 1488, 2018.

[23] A. Wang, "An industrial strength audio search algorithm," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, vol. 2003, 2003, pp. 7–13.

[24] T. Tsai, D. Yang, M. Shan, T. Tanprasert, and T. Jenrungrot, "Using cell phone pictures of sheet music to retrieve MIDI passages," *IEEE Transactions on Multimedia*, 2020, to appear.

[25] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr, "XSEDE: Accelerating scientific discovery," *Computing in Science & Engineering*, vol. 16, no. 5, pp. 62–74, Sept.-Oct. 2014. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MCSE.2014.80

[26] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[27] ——, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[30] F. Radenović, G. Tolias, and O. Chum, "CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples," in *Proc. of the European Conference on Computer Vision*, 2016, pp. 3–20.

[31] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proc. of the IEEE International Conference on Computer Vision*, 2015, pp. 1269–1277.

[32] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning CNN image retrieval with no human annotation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018.

[33] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of cnn activations," in *Proc. of the International Conference on Learning Representations*, 2016.

INVESTIGATING USER PERCEPTIONS UNDERLYING SOCIAL MUSIC BEHAVIOR USING Q METHODOLOGY

Louis Spinelli
University of Washington
spinelli@uw.edu

Josephine Lau
University of Washington
jolau@uw.edu

Jin Ha Lee
University of Washington
jinhalee@uw.edu

ABSTRACT

While prior studies investigating the social aspects of music provide a landscape of users' various social behaviors around commercial music services (CMS), there remains a lack in understanding of users' perceptions and value judgments underlying these behaviors. Specifically, there is more to learn about what influences and behaviors individual music users perceive as meaningful in social contexts. We used the Q methodology to explore which behaviors and influences are important to CMS users and why. We extracted two factors that explain the two different viewpoints shared by groups of music users, focusing on how they perceive the meaning and value of different social music behavior and interactions. From these findings, we then revised an existing social music coding dictionary and interaction model and offer new CMS design insights.

1. INTRODUCTION

Music is both personal [1], [2] and part of our social experience [1], [2], [3], [4]. Technologies in use affect what activities occur around music [5, 6] as does the physical and social context of how and where people interact with technology [7]. Furthermore, information technologies have become entangled with individuals' sense of self and social experiences [8]. Commercial music services (CMS), a type of information technology, are part of this ecosystem where individual and social experiences come together with current technologies.

Since music and technology are both so personal, improving our understanding of why CMS users find certain interactions with music and technology to be personally meaningful can help derive design decisions for new CMS technology to better meet these specific needs. Furthermore, in recognizing the influence of technology on music-related activities and the pace of CMS technology change [6], understanding what is meaningful to individuals within socially complex ecosystems can support forward-looking and contextually relevant design decisions.

Although there has been periodic research on music-related social behaviors and technology [4], [6], [9], [10], [11], [12], [13], [14], [15], [16], these studies focused on describing the different behaviors surrounding CMS and less on understanding how and why some behaviors and behavioral influences may be more significant to certain users than to others in social contexts. Previous research methods used include surveys [11], [12], [13], [15], [16],

interviews [4], [12], [14], [15], contextual inquiry [10], ethnographic inquiry [9], ethnographic observation of prototypes [15], field trial of prototype technologies [15], and focus groups [6]. While these methods were appropriate for their studies' goals, we still have a limited understanding of the personal significance of social behaviors surrounding music and the significance of influences on those social behaviors. In other words, we have a better understanding of how users behave in certain ways when it comes to using various music services, but less on what they perceive as meaningful or valuable from their own individual perspectives.

To address this gap, we conducted a study using the Q methodology, a method that better captures the personal significance of social behaviors. First developed in psychology in 1935 [17], [18], the Q methodology has since entered the Human-Computer Interaction (HCI) community [19], [20]. The Q methodology "asks its participants to decide what is 'meaningful' and hence what does (and what does not) have value and significance from their perspective" [18]. Furthermore, it is "used to explore (and to make sense of) highly complex and socially contested concepts and subject matters from the point of view of the group of participants involved" [18].

Using the Q methodology, we identified two distinct segments of CMS users within our participant group. Within each segment, users share similar perspectives about social behaviors surrounding CMS and associated influences on those behaviors. Additional contributions of this research include revisions for an existing coding dictionary and interaction model, and new design insights. Furthermore, the user segmentation could potentially contribute to existing personas identified in previous Music Information Retrieval (MIR) user studies [14], [21].

2. RELATED WORK

2.1 Social Practices Related to Music

Prior research laid the foundation for understanding social practices—and the influences on those social practices—where music intersects with technology. O'Hara and Brown's work captured social practices surrounding music and technology such as sharing, exploring, and peeping at a time when MP3 sharing platforms like Gnutella, Kazzaa, and Soulseek were still in use [15]. The social contexts for these practices included cars, public locations, workspaces, and dance clubs [15].

In 2013, as more users started to stream music and use the Bluetooth features on their mobile phones, Leong and Wright [4] observed the following social practices in

shared settings: exploration, discovery, selection, listening, and sharing. In the same year, Belcher and Haridakis [22] identified social motivators as influences to music listening and selection behaviors. In 2015, Yang et al. [23] identified peers as an influence on the practices of unauthorized music downloading and sharing. Later in 2017, Hagen and Lüders [1] studied how users listen, discover, share, and follow given social features on music streaming services. More recently, Spinelli et al. [6] identified nine social practices and twenty-four influences on these practices at the intersection of music and technology. Park et al. [16] also derived a collaborative playlist framework identifying sharing, recommending, and bonding as social purposes for how collaborative playlists served study participants. Lee et al. [24] explored music recommendations and also identified possible "disparities in how people wish to receive music recommendations and what will influence them to listen to recommendations, versus how they would like to offer recommendations to others."

This body of work builds a compelling story around the social practices surrounding CMS. Our work specifically aims to build upon the comprehensive model of practices and influences identified in prior research [6] and provide insight into how individual CMS users or segments of users perceive the different practices and influences around CMS. The Q methodology was selected to provide a holistic understanding [18] of how these practices and influences were perceived by a group of CMS users.

2.2 Methodology

Meloche introduced an established form of the Q methodology to the field of HCI in 1999 [19]. He believed the field would benefit from the method's ability to reveal the subjective views of individuals [19]. While the Q methodology has not previously been used to study CMS, forms of the method have been used to study other information technologies such as: studying a communication system for children [20], exploring user segmentation of technology services by information seeking preferences [25], and studying the health and technology attitudes of patients to inform the design of self-management interventions [26].

The Q methodology has also been used to study subjective views around music. Wacholtz [27] applied the method to investigate musical preferences and identify different listener types for country music. McKenzie and Brown [28] also studied the musical preferences of students and teachers related to popular music, identifying and describing three factors. While not directly focused on music and CMS, Davis and Michelle conducted research using the Q methodology focused on relevant media audiences and included a comprehensive bibliography of Q methodology research that studied media audiences and media users [29].

Both critiques and criticisms of the Q methodology and its implementation have been made and addressed over the years [30], [31]. A commonly noted challenge of the Q methodology is the potentially ambiguous nature and process of building the Q set [18], [32], [33]. To address this, our application of the Q methodology incorporated results from focus groups where constant comparative analysis was used to develop a Q set after the initial phase of a multi-method study design. Although interviews are often included as a possible method to support the development of a Q set [33], focus groups can enable richer open-ended discussion and interaction between participants which then can help form a meaningful Q set [25], [34]. Researchers also take mixed method approaches incorporating Q methodology—for instance, combining the Q methodology with R-method surveys, a quantitative method [35], [36].

3. STUDY DESIGN AND METHOD

We employed a multi-method approach that harnesses the strengths of both focus groups and the Q methodology. In Phase I, we selected exploratory focus groups to capture a wide breadth of statements from participants that described their social practices and associated influences they have experienced surrounding CMS. In total, the focus group study identified twenty-four possible influences on nine different social practices (both social practices and their influences are subsequently referred to as "themes"). The findings are reported in the Codebook of Social Practices and Influences [6]. In Phase II, the Q methodology was used to study the personal significance of the themes that were uncovered in the focus groups. Focus group statements from Phase I formed the basis for a set of items (the Q set) used in the Q methodology. This paper reports our findings from Phase II.

Twenty-four participants took part in the Q methodology component of this study, of which seventeen participated in Phase I. Each participant completed an in-person sorting activity, followed by brief interviews at the University of Washington, Seattle. These sorting activities and interviews, followed by a factor analysis method and factor interpretation to uncover participant viewpoints, comprise the Q methodology. Each of the 24 individual sorting activities and their follow-up interviews took between 30 minutes and an hour. A facilitator and a note-taker were present at each session. Sessions were also recorded and transcribed to ensure accurate analysis.

3.1 Selection of Participants

Recruitment activities for study participation consisted of displaying flyers, posting to listservs, and posting on social media as well as physical flyers placed on boards around the university campus and in nearby businesses. Participants were compensated with $15 Amazon gift cards for being part of this study. All recruiting avenues directed potential participants to a screener survey.

A screener survey was used to ensure all participants were between the ages of 18 and 34, currently lived with at least one other person, and used at least one CMS. The same screening criteria was used for identifying focus group participants. In total, 24 participants from the screener took part in an individual, in-person card sorting activity and subsequent interview. Of the 23 participants who chose to report a gender identity, 15 were female and 8 were male. Twenty-one participants were between the ages 18 to 24, and 3 were ages 25 to 34. Participants used a diverse array of CMS currently on the market including Spotify, Pandora, Google Play Music, YouTube, Soundcloud, etc.

A Level of Group Intimacy Researchers believed, a priori, that items of this theme pertained to the level of familiarity between group members in a social situation.

A.1 I am comfortable recommending or picking songs to listen to when hanging out with close friends.

A.2 I am comfortable picking music that my close friends will like in social gatherings.

A.3 I am comfortable with my roommates hearing everything I play.

A.4 I would not hesitate to tell my friends to change the music that is playing.

A.5 I would put my headphones on if I did not want to listen to the music that is playing in a shared space.

A.6 I do not feel comfortable making music recommendations in a large group setting.

A.7 I feel comfortable asking to connect my phone/laptop to a speaker at someone's house that I do not know well.

A.8 I do not mind sharing my music taste with people I do not know well at social gatherings.

A.9 I change the music in a large group setting if I do not like it.

A.10 I trust people to use my phone or laptop to play music at a large gathering (e.g., party).

B Effort/Engagement Researchers believed, a priori, that items of this theme pertained to the level of effort or engagement an individual is willing to put forth or the responsibility an individual is willing to take on when engaging with music in a social situation.

B.1 I like being the DJ if others give recommendations.

B.2 I like being the DJ and playing only my music at social gatherings.

B.3 I match the music playing to the mood of the group.

B.4 I would like for everyone to take turns adding music to a playlist.

B.5 I am comfortable forcing my friends to listen to my music recommendations in a social gathering.

B.6 I do not mind being the DJ if I can easily pick a playlist for the mood or activity.

B.7 I like not having to think about what music to play in social gatherings.

B.8 I do not mind changing the song if others do not like the music that is playing.

B.9 I like letting other people choose the music in social gatherings.

B.10 I like being able to add music to a queue in social gatherings.

C Privacy and Security Considerations Researchers believed, a priori, that items of this theme pertained to considerations relating to privacy and/or security that influence an individual's actions in a social music practice.

C.1 I would not let other people use my phone/laptop to listen to music.

C.2 I trust people to not snoop around on my phone/laptop if they are using it to pick music to play.

C.3 There are certain types of music I only listen to when I am alone.

C.4 I chaperone my phone/laptop if it is being used by others to play music in social gatherings.

C.5 I do not mind others playing music from my music accounts if I am already logged in and we are listening to music.

C.6 I am not concerned about other people knowing what I listen to.

C.7 I do not mind sharing my login for a service with my roommates.

C.8 I keep an eye on my phone/laptop if it is being used to play music at a party at my house.

Table 1. The Q set consists of items that represent themes of intimacy, effort, and privacy and security. Analysis and interpretation of study results are based on the participant's interpretations of items as expressed in post-sort interviews, not necessarily the themes they were intended to represent (shown here).

3.2 Q Set Generation from Focus Group Data

A Q set is a diverse collection of items curated to broadly represent the subject matter at hand, with each item "making a different (but nonetheless recognizable) assertion" [18]. For this study, the themes and items were elicited from focus groups where social practices and their related influences were discussed [6]. Researchers chose the themes to investigate using two criteria. First, the personal significance of each theme had to appear to vary within the participant group. Second, the themes had to appear interrelated. Both the themes and relationships between themes were identified on an affinity diagram created using a process of constant comparative analysis. Three influences on social music behaviors met these criteria and were selected as themes of focus for this study: 1) level of group intimacy, 2) level of effort and engagement, and 3) privacy and security considerations.

Researchers reviewed the focus group transcripts that had previously been coded with at least one of the three themes to gather items for the Q set. Statements were selected as items for the Q set based on three criteria: focus, coverage, and balance [33]. Focus refers to including items that can be sorted by a "single, face-valid assumption" [33]. Coverage refers to the set being "broadly representative" of the domain at hand [33]. Balance refers to including all the opinions and perspectives in the Q set [33]. After items were selected, the Q set was piloted to ensure participants understood both the items and sorting activities for the study.

3.3 Q Sorting Activity and Interviews

During a Q sort, individual participants organize items (Table 1) into a forced distribution known as a Q pyramid (Figure 1) and place items they agree with most to the right (+3) and most disagree with to the left (-3). During the Q

sort and subsequent interviews, participants express their interpretations of items along with their reasoning for placement into the Q pyramid.

We laid out three pieces of paper for Agree, Disagree, and Neutral along with the Q set items printed out on cards. We asked participants to sort the Q set cards into the three groups and afterwards, arrange the cards in accordance with a provided image of a Q pyramid. After participants had finished sorting the cards, we asked them to tell us about the items they felt most and least strongly about as well as the items in the center of their distribution (See supplemental material for the complete protocol). Individual Q sorting activities with post-sort interviews were held privately for confidentiality.

Exemplary Q-Sort for Factor 1

-3	-2	-1	0	+1	+2	+3
A.10	A.7*	A.3	B.4*	B.6*	A.2	A.1
C.7	A.9*	A.5	A.6	B.10*	B.1	C.3
	C.2	A.8	B.5	B.9	B.3	
	B.2	C.1	A.4	C.4	C.8	
		C.6	B.7	B.8		
			C.5			

Exemplary Q-Sort for Factor 2

-3	-2	-1	0	+1	+2	+3
A.9*	A.7*	A.4	B.4*	B.10*	A.1	B.9
B.2	B.5	A.6	B.6*	B.7	A.3	B.8
	C.1	A.5	B.3	A.2	C.6	
	C.4	B.1	A.8	C.2	C.5	
		C.7	A.10	C.3		
			C.8			

Figure 1. Factor exemplifying sorts for Factor 1 and 2 displayed in the Q pyramid used in this study. Consensus items that do not distinguish one factor from another are flagged (*).

3.4 Analysis

Analysis of the Q sorts is completed using "a by-person factor analysis in order to identify groups of participants who make sense of (and who hence Q 'sort') a pool of items in comparable ways" [18]. We took an inductive approach using a process in line with exploratory factor analysis [33] as we observed that individuals in the population valued the three themes in the Q set differently, but we did not have a hypothesis about the differences. Factor arrays—representative viewpoints of the perspectives expressed by participants in their Q sorts—were extracted from a correlation matrix, built from the intercorrelation of all the Q sorts [33]. Factor extraction was done at the same time as the coding of post-sort interview transcripts.

Factor extraction was conducted using PQMethod; a free software dedicated to the Q methodology [33]. To determine the appropriate number of factors to extract, we first used the Kaiser-Gutman Criterion followed by a Scree Test. We then performed a varimax rotation on the factors and extracted our final factors. We determined PQMethod's pre-flagging was both appropriate and more than sufficient for our exploratory purposes. Including every sort in the creation of factor estimates increased the reliability of our factor estimates and arrays while reducing error [33]. Thirteen Q sorts were flagged for Factor 1 and eleven Q sorts for Factor 2. To enable cross-comparison between factors, total factor estimate scores are converted to Z scores [33]. Exemplary factor arrays for Factor 1 and Factor 2 were then created from Z scores (Figure 1).

In preparation of Factor interpretation, post-sort interview transcripts were coded. The transcripts were coded with the relevant Q sort item being discussed, the exemplary Factor number that represents the participant making the statement, and the applicable themes from the Codebook of Social Practices and Influences [6]. By coding transcripts with the Q sort item and Factor number representative of the participant, we were able to quickly filter relevant statements to interpret each factor array. Themes from the Codebook of Social Practices and Influences captured the meaning of the Q set item as expressed by each participant in post-sort interviews rather than by the researchers' a priori beliefs. The two exemplary factor arrays identified during factor extraction are representative viewpoints of the perspectives expressed by participants in their Q sorts [33].

Factor extraction and coding of transcripts provided the foundation for factor interpretation, which was conducted applying Stenner and Watts' Crib Sheet method to each factor array [33]. Items were separated into four categories: items ranked at +3, items that ranked higher than other arrays, items that ranked lower than other arrays, and items that ranked -3 (Figure 1). Following an abductive process, each item was interpreted individually and then in the context of the entire viewpoint. Item by item, the viewpoint grew into the holistic viewpoint for the factor array. During this process, participant statements and themes from the prior work provided insight into how participants interpreted each item in context of the larger Q set. Our interpretation at this stage reflected this understanding of items, not our a priori understanding of each item. Using this lens, we found all items supported a holistic viewpoint

for each factor, including consensus items that did not rank differently across factors.

4. RESULTS

For this study, we reached a two factor outcome that explains 45% of the total study variance in the correlation matrix. Common factor solutions that capture 35-40% or more of the total study variance are considered sound [33], [37]. Eigenvalues (EV) provide another way to compare factors within a study, and a higher EV is viewed as positive [33].

A cross-factor analysis identifies 23 items as being ranked significantly differently at the p < 0.01 level. Five consensus items were identified as non-significant at p > 0.01. This means the two identified factors (participant groups) had statistically significantly different views about 23 items but generally agreed on 5 items. Factor 1 is factor-exemplifying for 13 Q sorts, or put another way, Factor 1 is representative of the Q sorts of 13 participants. Factor 2 is factor-exemplifying for 11 Q sorts.

In the following subsections, we provide the viewpoints developed from interpreting the factors in the context of post-sort interviews. The supporting Q sort item identifier and item rank used for factor interpretation are included for each statement in the viewpoints (i.e. identifier: rank).

4.1 Factor 1 Interpretation: Viewpoint 1

Users with Impression Management and Security Concerns, but also Confident Music Selectors

Factor 1 explains 23% of the total study variance and has an EV of 7.26. Thirteen participants are significantly associated with this factor.

These users did not want others to know the type of music they were listening to due to impression management concerns or appropriateness for the social situation. They had impression management concerns; they did not want their roommates to know what they listen to (C.3: +3; A.3: -1) and considered some of the music they listen to as guilty pleasures (C.3: +3; A.3: -1; C.6: -1). Another reason they listen to music alone is they believed some music types are not appropriate for social situations (C.3: +3).

In social situations, these CMS users are confident in their ability to pick music that their social group will enjoy (A.1: +3; A.2: +2)**.** They believe they can match music selections with the mood of a gathering, and this becomes easier when the gathering is intimate (B.3: +2). They are so confident in their understanding of their close friends' music tastes that they may force them to listen to a song they know their friend will like; this would not be the case for friends they know less intimately (B.5: 0). Similarly, in large group settings they recommend generic, safe choices —they choose popular songs due to impression management concerns that an untested song will not be appreciated by the group (A.6: 0; A.8: -1).

Although they would not hesitate to tell a close friend to change the music currently playing, they feel it is often unnecessary or inappropriate (A.4: 0). **They believe they can tolerate any music in a social situation** (A.5: -1) **and think it is especially rude to change other people's music**

in large groups (A.9: -2). In fact, they do not expect to be able to provide input in many situations—like at someone else's wedding (B.10: 1). When they are playing music, they do not mind changing the song if there is group consensus that it should be changed (B.8: 1). They also do not mind situations where everyone takes turns adding songs to a playlist as long as the flow is consistent (B.4: 0).

When they are with their friends, these CMS users prefer to focus on interacting with them, and not on choosing the music (B.7: 0; B.6: +1). It is a lot of effort to pick all the music for an event, and it is not considerate to ask guests to pick songs (B.2: -2). *Anything that reduces the amount of effort needed is seen as beneficial, like selecting a playlist* (B.6: +1). They want to reduce their impression management concerns and their effort in selecting music for social situations (A.9:0). When others give recommendations, it ensures these users do not have to take responsibility for what they play—something they really like (B.1: +2; B.9: +1). An added benefit is that letting others choose the music reduces the effort they need to put into the activity (B.1: +2).

These CMS users generally do not trust that others will not snoop on their phone or laptop (C.2: -2). This is especially true in large groups where they would never share their phone (A.10: -3). In more intimate settings, they will likely let friends use their account to play music if they are already logged into an account (C.5: 0). Regardless of how intimate the situation is, they will monitor their device to make sure snooping does not occur and device use is limited to the music app (C.8: +2; C.4: +1).

4.2 Factor 2 Interpretation: Viewpoint 2

Very Considerate CMS Users with Almost No Impression Management or Security Concerns

Factor 2 explains 22% of the total study variance and has an EV of 3.44. Eleven participants are significantly associated with this factor.

These CMS users are not concerned about other people knowing what they listen to at all (C.6: +2). *They do not have any privacy concerns about their music tastes and do not mind if that information becomes known to the group* (A.3: +2). They would share their tastes if people at a gathering were interested but are also okay if people are not interested (A.8: 0). They listen to certain types of music alone if it is not popular with their friends or if the music does not fit with the social event (C.3: +1).

They are comfortable making individual song recommendations to friends because they are familiar with their tastes (A.1: +2; A.2: +1). It would be very unlikely for them to force a friend to listen to a music recommendation (B.5: -1).

When confronted with music they do not like, they want to be considerate, likely tolerating a song they do not like or leaving the physical space (A.9: -3; A.5: -1). Some in this group worry that putting on headphones to block out music is antisocial (A.5: -1). They really want to be considerate of others and not critical (A.9: -3). Rather than putting on headphones they might instead comment that they liked music that was played earlier, and in that way, gently nudge music selection back in that direction

(A.9: -3). Unless they are in an intimate situation, like a small group in a car, they probably would not tell a friend to change the music (A.4: -1).

These CMS users are very considerate of others' experience with music and will participate in music activities that support everyone's enjoyment (B.8: +3). However, they do not want to make the decisions and do not want to be super engaged (B.9: +3). They prefer not to think about what to play at a social gathering (B.7: +1). They would hate being the DJ and sole decision maker at a social gathering because of how much effort it would involve (B.2: -3). To reduce the effort of selecting music, they might select a playlist, but they would still rather not have to select anything at all (B.6: 0). When selecting music they would try to match the mood of the group, but they are not confident they would be able to (B.3: 0) and are unsure that they could select the best music for the group and situation (B.2: -3). They also do not believe it would be considerate to others if they were the only ones selecting music (B.2: -3). Thus, if needed, they would take recommendations to ensure everyone is happy (B.1: -1). They like the idea of everyone being able to give input even in a large group setting, but they do not think it is always necessary or that people should feel compelled to do so (A.6: -1; B.4: 0; B.10: +1). They definitely want to make sure everybody is happy (B.3: 0).

Although they would prefer to use someone else's device, they would let others use their phone/laptop to listen to music (C.1: -2; C.4: -2). While they would not want people to snoop, they trust that people will not do so (C.2: +1). They also do not think they have anything embarrassing on their phones and laptops (C.2: +1). They envision that they might be concerned about leaving their phone or laptop out to play music in situations with a lot of strangers, such as a large party, but they have done so in the past without issues (C.8: 0; A.10: 0). However, when they are at someone else's house they do not know well, they would not feel comfortable asking to connect their phone/laptop to a speaker (A.7: -2). They would prefer others to share their logins, but situations have come up where they would share their own logins (C.7: -1); e.g., they do not mind sharing an account if they are already logged in (C.5: +2).

5. DISCUSSION

This study provided two clear viewpoints in the form of exemplifying Q sorts for two different segments of participants. The viewpoints captured the inter-relatedness of themes pertaining to social music behaviors and associated influences surrounding CMS. We confirmed that the viewpoints are an excellent way to evaluate design insights especially with a deeper understanding of influences that could drive or inhibit the adoption of a new design [25]. While a focus group study inspired many ideas, the Q methodology left us with a clearer vision of what some segments of participants would love, hate, or not care about. Based on the results of this study, researchers will also be able to investigate how viewpoints identified here can contribute to personas already existing in the field [21].

5.1 Updating an Existing Model

The Q methodology uncovered shortcomings in our interpretation and understanding of participants' statements made in focus groups. For example, Q set item A.5 *I would put my headphones on if I did not want to listen to the music that's playing in a shared space* was interpreted by researchers as it relates to intimacy. This was our a priori belief when building the Q set. After interpreting the post-sort interviews however, it became clear many participants interpreted item A.5 as it relates to considerateness and social norms rather than intimacy. Thus, analyzing participants' post-sort interview statements not only gave insight into why they sorted items the way they did, but also into how they interpreted those items.

Insight gleaned from post-sort interviews led to a better understanding of participants' viewpoints and to the identification of issues with the coding dictionary and the model from the focus group study [6]. For example, *Privacy and Security Considerations* is a theme that emerged from focus groups describing an internal influence on social practices surrounding music. Items capturing the breadth of *Privacy and Security Considerations* discussed by participants in focus groups were selected for the Q set. This Q methodology study uncovers that privacy was an inclination, almost a behavior, driven by impression management, security, or both. An updated codebook and model reflects this finding by keeping Impression Management as a theme and separating Privacy from Security considerations.

5.2 Applied Design Insights for CMS

5.2.1 Social Playlist for Gatherings

As participants in both viewpoints appreciate any features that reduce the effort needed to select songs or playlists for a group due to impression management concerns, we suggest CMS to include auto-suggested playlists that are based on the listening history of group members who have opted into this function. At a group gathering, hosts can invite their guests to add their listening history and music preferences into the mix so that the CMS can add or suggest songs for the queue. The host's invitation to guests validates and follows a previous recommendation to maintain social norms [6], such as the host having ultimate say in who chooses music for a co-located gathering. After group members have opted in, the CMS would (1) automatically queue up "safe" songs that have been previously played by a majority of individuals in the group and (2) suggest additional songs for each user to add to the queue. Suggested songs would either have been played/liked by someone else in the group or have a strong match for other criteria that the group could also select, such as a mood or social situation. Suggested songs would not be as "safe" as songs that are automatically queued, but would still reflect the interests and character of the group. This feature could alleviate viewpoint 2's lack of confidence in suggesting songs for the group since the CMS would only suggest songs where evidence of it being liked by others exists. For viewpoint 1, this feature would decrease the effort needed to think of songs that the group would like, providing them with more time to connect with others, which they value.

5.2.2 Jukebox Mode: Public-friendly Mode of CMS

For situations where a device, such as a phone or laptop, is passed around in social gatherings for guests to add songs to the queue, we recommend designers include a "Jukebox mode" on the CMS. This would lock away all of the device's other applications and private communications so that they are hidden, and only the owner can unlock the phone again to its full capabilities. Additionally, Jukebox mode would switch the CMS interface to a public-friendly version of the app, hiding the owner's private playlists so that other guests cannot view the owner's music listening history. Essentially the device becomes a jukebox, where guests can only use the device to access the CMS and its library of music. Also, a guest's music selection would not affect future music recommendations for the owner, which was another concern that CMS users expressed with sharing their phones in group settings. While akin to the Guided Access feature currently available on iPhones and Androids [38], [39], where users can lock the device to a single app on the phone through a phone setting, this feature would be part of the CMS. This mode would accommodate viewpoint 1's hesitation to have their device be used for music selection purposes, assuaging their fear of others snooping and mitigating their need to chaperone their phone. While viewpoint 2 was slightly more comfortable with others using their devices than viewpoint 1, they were less comfortable sharing in larger, less intimate groups. The "Jukebox mode" could thus address both of these viewpoints' concerns.

6. CONCLUSIONS AND FUTURE WORK

In this work, we identified two viewpoints shared by different segments of our participant group, updated a codebook and interaction model, and generated design insights. We learned how our participant group perceives social practices and associated influences surrounding CMS.

While this research is a step forward in addressing the gap in understanding social practices and associated influences surrounding CMS, the identified segments are not generalizable to the general population. A survey, informed by the segments identified in this study, would provide insight into the generalizability of these findings. Future research could also explore CMS users' perspectives in relation to culture and geography.

We believe that focus groups and other qualitative methods could support systematic Q set development, but this process has yet to be fully explored. Specifically, from this case study on social music practices, we found that exploratory focus groups and the Q methodology are excellent complementary methods. Analyzing data through constant comparative analysis, affinity diagramming, and coding of transcripts were effective in identifying the scope of the Q methodology research and generating a Q set. In future research, we plan to investigate other elicitation methods and analysis techniques to form Q sets, and explore the use of the Q methodology as a complement to methods such as narrative analysis and ethnographies. Additional research into this multi-method approach is notably important for researchers studying topics that are especially personal and private, where focus groups would not be an appropriate complementary method.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] N. Hagen and M. Lüders, "Social streaming? Navigating music as personal and social," *Convergence: The International Journal of Research into New Media Technologies*, vol. 23, no. 6, pp. 643–659, Dec. 2017.

[2] S. Jones, "Music and the Internet," in *The Handbook of Internet Studies*, M. Consalvo and C. Ess, Eds. Wiley-Blackwell, 2011, pp. 440–451.

[3] D. Boer and A. Abubakar, "Music listening in families and peer groups: benefits for young people's social cohesion and emotional well-being across four cultures," *Frontiers in Psychology*, vol. 5, May 2014.

[4] T. W. Leong and P. C. Wright, "Revisiting Social Practices Surrounding Music," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2013, pp. 951–960.

[5] B. Brown and A. Sellen, "Sharing and Listening to Music," in *Consuming Music Together*, K. O'Hara and B. Brown, Eds. Springer Netherlands, 2006, pp. 37–56.

[6] L. Spinelli, J. Lau, L. Pritchard, and J. H. Lee, "Influences on the Social Practices Surrounding Commercial Music Services: A Model for Rich Interactions," in *Proc. of the 19th Int. Society for Music Information Retrieval Conf., Paris, France*, 2018, pp. 671–677.

[7] A. J. B. Brush and K. M. Inkpen, "Yours, Mine and Ours? Sharing and Use of Technology in Domestic Environments," in *UbiComp 2007: Ubiquitous Computing*, Sep. 2007, pp. 109–126.

[8] M. Carter and V. Grover, "Me, My Self, and I(T): Conceptualizing Information Technology Identity and its Implications," *MIS Quarterly*, vol. 39, no. 4, pp. 931–957, Apr. 2015.

[9] S. J. Cunningham and D. M. Nichols, "Exploring social music behaviour: An investigation of music selection at parties," in *Proc. of the 10th Int. Society for Music Information Retrieval Conf.*, 2009, pp. 747–752.

[10] S. J. Cunningham, D. M. Nichols, D. Bainbridge, and H. Ali, "Social music in cars," in *Proc. of the 15th Int. Society for Music Information Retrieval Conf.*, 2014, pp. 457–462.

[11] J. Fuller, L. Hubener, Y.-S. Kim, and J. H. Lee, "Elucidating user behavior in music services through persona and gender," in *Proc. of the 17th Int. Society for Music Information Retrieval Conf.*, New York, N.Y., United States, 2016, pp. 626–632.

[12] S. Komulainen, M. Karukka, and J. Häkkilä, "Social Music Services in Teenage Life: A Case Study," in *Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction*, New York, NY, USA, 2010, pp. 364–367.

[13] J. H. Lee, Y.-S. Kim, and C. Hubbles, "A look at the cloud from both sides now: An analysis of cloud music service usage," in *Proc. of the 15th Int. Society for Music Information Retrieval Conf.*, New York, N.Y., United States, 2016, pp. 299–305.

[14] J. H. Lee, R. Wishkoski, L. Aase, P. Meas, and C. Hubbles, "Understanding users of cloud music services: Selection factors, management and access behavior, and perceptions," *Journal of the Association for Information Science and Technology*, vol. 68, no. 5, pp. 1186–1200, 2017.

[15] K. O'Hara and B. Brown, *Consuming music together: social and collaborative aspects of music consumption technologies*. Dordrecht: Springer, 2006.

[16] S. Y. Park, A. Laplante, J. H. Lee, and B. Kaneshiro, "Tunes Together: Perception and experience of collaborative playlists," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, Delft, Netherlands, 2019, pp. 723–730.

[17] W. Stephenson, "Correlating persons instead of tests," *Journal of Personality*, vol. 4, no. 1, pp. 17–24, 1935.

[18] S. Watts and P. Stenner, "Doing Q methodology: theory, method and interpretation," *Qualitative Research in Psychology*, vol. 2, no. 1, pp. 67–91, Jan. 2005.

[19] J. Meloche, "Q Methodology as a research methodology for human computer interaction," *Faculty of Commerce - Papers (Archive)*, pp. 149–152, Jan. 1999.

[20] K. O'Leary, J. O. Wobbrock, and E. A. Riskin, "Q-methodology as a Research and Design Tool for HCI," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2013, pp. 1941–1950.

[21] J. H. Lee and R. Price, "Understanding users of commercial music services through personas: design implications," *Proc. of the 16th Int. Society for*

Music Information Retrieval Conf., 2015, pp. 476-482.

[22] J. D. Belcher and P. Haridakis, "The Role of Background Characteristics, Music-Listening Motives, and Music Selection on Music Discussion," *Communication Quarterly*, vol. 61, no. 4, pp. 375–396, Sep. 2013.

[23] Z. Yang, J. Wang, and M. Mourali, "Effect of peer influence on unauthorized music downloading and sharing: The moderating role of self-construal," *Journal of Business Research*, vol. 68, no. 3, pp. 516–525, Mar. 2015.

[24] J. H. Lee, L. Pritchard, and C. Hubbles, "Can we listen to it together?: Factors influencing reception of music recommendations and post-recommendation behavior," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, Delft, Netherlands, 2019, pp. 663–669.

[25] A. Morton and M. A. Sasse, "Desperately seeking assurances: Segmenting users by their information-seeking preferences," in *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, Jul. 2014, pp. 102–111.

[26] K. O'Leary, L. Vizer, J. Eschler, J. Ralston, and W. Pratt, "Understanding patients' health and technology attitudes for tailoring self-management interventions," *AMIA Annual Symposium Proceedings*, vol. 2015, pp. 991–1000, Nov. 2015.

[27] L. E. Wacholtz, "The country music audience: A Q-technique portrait of seven listener types," In *The meeting of the International Society for the Scientific Study of Subjectivity*, Columbia, MO, Oct. 1992.

[28] T. P. McKenzie and S. R. Brown, "Musical Preferences and Forms of Life," *Operant Subjectivity*, vol. 37, no. 1/2, pp. 97–105, Jun. 2014.

[29] C. H. Davis and C. Michelle, "Q Methodology in Audience Research: Bridging the Qualitative/Quantitative 'Divide'?," vol. 8, no. 2, p. 35, 2011.

[30] S. R. Brown, S. Danielson, and J. van Exel, "Overly ambitious critics and the Medici Effect: a reply to Kampen and Tamás," *Qual Quant*, vol. 49, no. 2, pp. 523–537, Mar. 2015.

[31] S. Ramlo, "Mixed Method Lessons Learned From 80 Years of Q Methodology," *Journal of Mixed Methods Research*, vol. 10, no. 1, pp. 28–45, Jan. 2016.

[32] B. C. Curt, *Textuality and tectonics: Troubling social and psychological science*. Maidenhead, BRK, England: Open University Press, 1994.

[33] S. Watts and P. Stenner, *Doing Q Methodological Research: Theory, Method and Interpretation*. 1 Oliver's Yard, 55 City Road, London EC1Y 1SP United Kingdom: SAGE Publications Ltd, 2012.

[34] A. Morton, "'All my mates have got it, so it must be okay': Constructing a Richer Understanding of Privacy Concerns—An Exploratory Focus Group Study," in *Reloading Data Protection*, S. Gutwirth, R. Leenes, and P. D. Hert, Eds. Springer Netherlands, 2014, pp. 259–298.

[35] S. Danielson, "Q Method and Surveys: Three Ways to Combine Q and R," *Field Methods*, vol. 21, no. 3, pp. 219–237, Aug. 2009.

[36] K. Y. Kim and B. G. Lee, "Marketing insights for mobile advertising and consumer segmentation in the cloud era: A Q–R hybrid methodology and practices," *Technological Forecasting and Social Change*, vol. 91, pp. 78–92, Feb. 2015.

[37] P. Kline, *An Easy Guide to Factor Analysis*. Routledge, 2014.

[38] "Pin & unpin screens," *Android Help*. https://support.google.com/android/answer/9455138?hl=en (accessed Apr. 05, 2020).

[39] "Use Guided Access with iPhone, iPad, and iPod touch," *Apple Support*, Sep. 19, 2019. https://support.apple.com/en-us/HT202612 (accessed Apr. 05, 2020).

THE RHYTHMIC DICTATOR: DOES GAMIFICATION OF RHYTHM DICTATION EXERCISES HELP?

Matevž Pesek[1]   **Lovro Suhadolnik**[1]   **Peter Šavli**[2]

Matija Marolt[1]

[1] Faculty of computer and information science, University of Ljubljana, Slovenia

[2] Conservatory of Music and Ballet Ljubljana, Slovenia

`matevz.pesek@fri.uni-lj.si`

ABSTRACT

We present the development and evaluation of a gamified rhythmic dictation application for music theory learning. The application's focus is on mobile accessibility and user experience, so it includes intuitive controls for input of rhythmic exercises, a responsive user interface, several gamification elements and a flexible exercise generator. We evaluated the rhythmic dictation application with conservatory-level music theory students through A/B testing, to assess their engagement and performance. The results show a significant impact of the application on the students' exam scores.

1. INTRODUCTION

Music theory learning and ear training is not very popular among students in music education and informal music learning, although knowing about music theory stimulates knowledge about music and enhances music appreciation. Considering the expansion of e-learning, an overwhelming part of the music theory learning still takes place in the traditional paper-and-pen form. Opportunities therefore exist for increasing student engagement with appropriate information and communications technology (ICT) tools that would support the learning process while motivating the students to use them through the use of gamification elements.

Games and gamified applications have gained traction in recent years and have become important tools in the ICT and e-learning communities. The evaluation of gamification [1, 2] and student engagement [3] has received significant attention, and the development of specialised platforms and apps for e-learning has flourished [4, 5]. Gamification has often been a medium for information retrieval and collaborative data gathering [6]. In music information retrieval, several approaches for gamification of music annotation and meta-data gathering have been proposed.

Kim et al. [7] proposed the Moodswings game for mood labelling, where the users were asked to plot the mood on the valence-arousal graph. They collected over 50.000 valence-arousal point-labels on more than 1000 songs. The authors identified gamification as the key component of user engagement. In a similar manner, Mandel and Ellis [8] proposed a web-based game for collecting song meta-data, such as genre and instrumentation. Law et al. [9] created the TagATune game for music and sound annotation. The game collects comparative information about sounds and music, where users play the game in pairs. The authors collected responses from 54 test users. They also focused on the user engagement through three aspects: sense of competence for the user, pleasantness and sensory user experience, and the opportunity to connect with a partner. Burgoyne et al. [8] presented a game named Hooked to explore the "catchiness" of songs on the responses provided by 26 users. The dataset consisted of 32 songs. Aljanaki et al. [10] developed a 'game with a purpose' to gather emotion responses to music. They collected more than 15,000 responses from 1,595 participants. Overall, in the MIR community, the developed applications mainly served as a medium to gather data.

On the other side, many web and mobile platforms for music learning exist, that also incorporate gamification elements, from instrument-related applications (e.g. My Piano Assistant [1], Yousician [2]) and music accompaniment software (e.g. iReal Pro [3]) to music-theory platforms (e.g. theoria.com, musictheory.net, Musition [4]). These platforms, however are commercial and closed-source, they are not extensible to new topics (by teachers) or adjustable to individual learning groups and curricula. While there is no doubt that they can help the user to improve their knowledge and performance, the lack of adjustment to in-class use within existing curricula is difficult without code-level access. Access to such commercial platforms may also not be affordable for all parties (e.g. public music schools).

In the paper, we present the Rhythmic dictator: a rhythmic dictation application, which is part of our larger effort to gamify various aspects of music theory learning into a common open-source platform. It is implemented as a

---

[1] Available on Google Play and Apple App store

[2] `https://yousician.com`

[3] `https://irealpro.com/`

[4] `https://www.risingsoftware.com/musition`

web-based application with a responsive user interface that is specifically designed for mobile devices, since these are the most commonly used by students. The application automatically generates exercises according to the student's level of knowledge and in-app progress. To increase student engagement, gamification elements, including badges and leaderboards are implemented.

We analyse two aspects of the application's in-class use with first and second year conservatory-level students: the students' engagement, and the application's impact on the students' performance.

2. THE RHYTHMIC DICTATOR

Three exercise types are commonly performed by music theory students: melodic (interval) dictation exercises, rhythmic dictation exercises, and harmony exercises. Conventional practice usually consists of listening to a pre-recorded or teacher-performed dictation and solving it on paper. Evaluation and grading is done by the teacher.

Our rhythmic dictation application (the Rhythmic dictator) offers an easy to use and automated way for students to solve rhythmic dictation exercises in-class and out-of-class with immediate feedback on their performance and a customizable exercise generator which adapts the difficulty of generated exercises to the student's level of knowledge. The application was developed as a responsive web application, which adapts well to mobile devices. In this way, the development and maintenance of the platform is simplified, as the platform is browser-accessible on all major platforms—Windows, Linux, OS X for desktop environments, as well as Android and iOS for mobile devices.
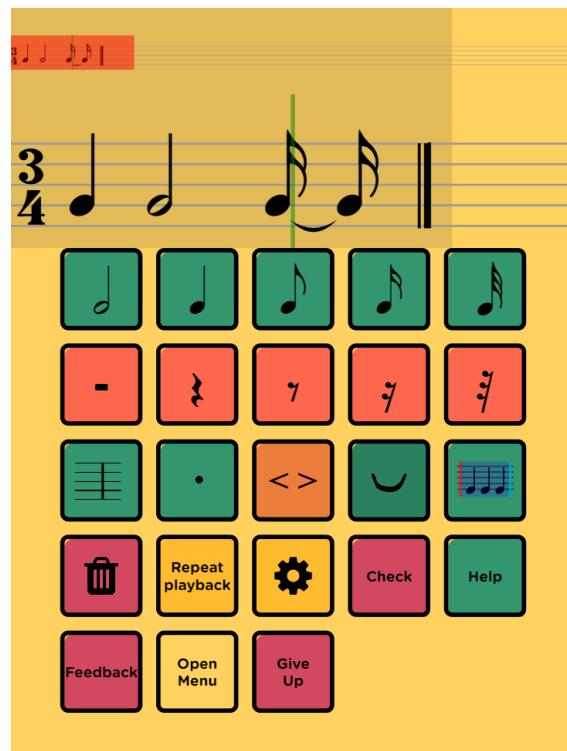
The application is incorporated into the Troubadour platform [5], which is a framework for music theory learning with support for gamification elements including badges, points and leaderboards. The application and the platform are easily deployable with the use of package management tools, and the code is available as open source software and publicly accessible on GitHub [6].

2.1 The user interface

In rhythmic dictation, the students listen to a rhythmic sequence, which they have to write down in music notation. The main part of the Rhythmic dictator's user interface (Figure 1) therefore includes two staves displaying the input rhythmic sequence and a rhythm input interface. The upper (smaller) stave shows the entire sequence with a red rectangle indicating the area shown in the lower larger stave, where the user inputs their response to the dictation. The dictation can be played-back repeatedly and paused while playing.

The rhythm input keyboard supports a variety of rhythmic inputs: note and pause lengths, subdivisions and syncopation. To accommodate for the small screens of mobile

---

[5] https://trubadur.si
[6] https://bitbucket.org/ul-fri-lgm/troubadour_production

devices, the inputs are split into two layouts: on the primary layout, the most common note and pause lengths are displayed. With keys for subdivision and syncopation, the layout changes to show a set of additional input options, as shown in Figure 2.

Figure 1: The main screen of the rhythm dictation application on a mobile device. The primary rhythm input interface is shown below the staves.
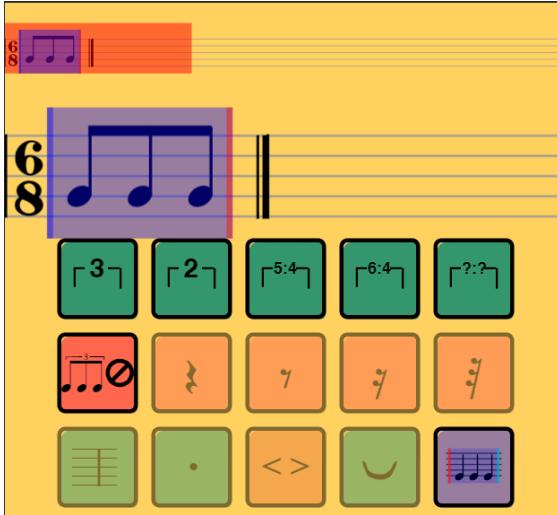
Each exercise begins with a metronome indicating the meter and is followed by the rhythmic dictation playback. The student can pause and replay the dictation, and adjust the playback speed and volume. The dictation is played using an organ sound. The sound was chosen in discussion with music theory teachers due to its fast onset, steady sustain and a clear offset. While the sound of piano is commonly used for melodic dictation, its unclear offset can cause ambiguities in determining the event length (vs. pause). Our choice of the sound was also evaluated with the users during the evaluation period.

2.2 Automatic generation of exercises

The Rhythmic dictator includes an exercise generator that can generate exercises of different difficulty levels. The difficulty of a rhythmic exercise is governed by several parameters: subdivision complexity (from quarter notes, to 32nd notes), subdivision types (dual vs. ternary), subdivision distributions, and the number of events (length of the sequence). Randomly generating the exercises with uniform distributions of these parameters yields meaningless and unrealistic sequences that are non-intuitive and difficult to solve and which lower the student's motivation. We therefore analyzed the existing materials that teachers used

Figure 2: The secondary keyboard layout with options for adding and modifying subdivisions.

in their classes and created parameter distributions for various difficulty levels. The distributions take into account the frequency of event occurrences, as well as their in-bar position, to reflect the rhythmic patterns, which are common in music. In this way, the randomly generated sequences become more musically meaningful and engage the student individually with sufficient difficulty, while not overwhelming them with either too difficult or meaningless examples.
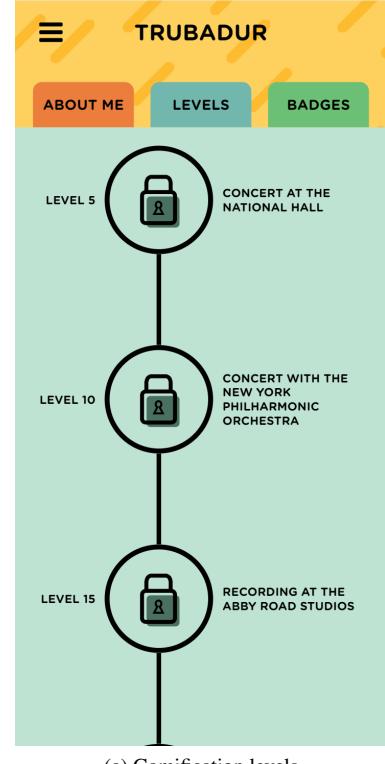
We arranged the distributions into 16 difficulty levels, ranging from elementary music school to academia. The levels are split into four major levels, and each major level is split into additional four minor levels. We marked the levels with numbers {11-14, 21-24, 31-34 and 41-44}, with the first digit corresponding to the major and the second to the minor difficulty level. The parameter distributions for each level were set as the default values for exercise generation in the rhythmic dictation application, however teachers are able to modify the distributions according to their didactic expertise and needs.

2.3 Gamification elements

To increase the motivation for using the application among the students, we enriched it with elements of gamification. We use three gamified elements related to students' performance: progression between multiple levels of proficiency, a leaderboard and achievement badges. The gamification elements are visible on the home screen, where students can browse through their achievements, as seen in Figures 3a and 3b.

While using the application, students earn points by solving the exercises, which directly affects their achievements. Each exercise consists of two sequences that can be answered multiple times. After each completed exercise, the student's points are calculated, measured as a function of several factors: exercise difficulty (across the 16 difficulty levels), time taken (in minutes), number of corrections (additions and deletions of notes), the use of the

metronome (yes/no), the number of submission attempts (checks for correctness), and whether the final sequence was correct. The sum of points can be either positive or negative.

(a) Gamification levels

(b) Achieved badges

Figure 3: Gamification elements. The left screen shows progression between multiple badge levels of proficiency, while the right screen shows the badges obtained during the practice.

By solving more excercises and progressing through levels of difficulty, students increase their level of proficiency (Figure 3a). The levels were defined by the teachers, and vary from local orchestra, to different competitions and international institutions.

The badges, shown in Figure 3b, reflect three different aspects of student progress. The first aspect is accuracy: completing an exercise with(out) a certain amount of mistakes (from 50% up to 100% correct answers). The second aspect is the continuity of the student's engagement with the platform: playing an exercise for a certain amount of days in a row—3 days, 5 days, a week, two weeks, a month. The third aspect is the student's speed: the amount of time needed to complete an exercise in 5 minute intervals, ranging from 25 minutes to 5 minutes.

As an additional element of gamification, we implemented a leaderboard. The leaderboard shows the cumulative points collected by an individual. More points can be achieved by both the number of solved exercises and the exercise difficulty. The students can observe their performance and compare it to the other players. By clicking on one of the platform's users the selected user's profile page is displayed, with their achieved levels and badges.

3. EXPERIMENT

The primary goal of the developed application was to provide an open platform, which would engage students and increase their performance in rhythmic dictation tasks. The application was tailored to increase student engagement through gamification elements and an intuitive interface on mobile devices. In our experiment, we wanted to assess whether these goals were achieved.

We evaluated several hypotheses. First, we assumed the mobile-friendly interface will enable the students to engage with the application. While the students might spend more time with an individual exercise at the beginning to get used to the interface, the time spent for solving an exercise should in time decrease due to familiarity with the interface and the student's increased proficiency. Second we hypothesized that student engagement will have an impact on their exam performance.

During the experiment, we collaborated with first and second year students at the Conservatory of Music and Ballet Ljubljana, Slovenia. First, we developed the application through continuous evaluation with four conservatory students (two first and two second year), who represented a sample of our target audience. We continually evaluated the students' interaction with the application: whether they understood the user interface, whether the exercises were appropriately demanding and whether the exercises were sufficiently interesting and engaging.

We then evaluated the application with the first and second year students at the conservatory. The students were randomly divided into one test and one control group in each year. The evaluation lasted for five weeks, during which we held four in-class meetings with the students of the test groups. Students were asked to use the application during the meetings through a group student challenge, during which the students competed to achieve points in the application.

After the five week period, students of both test and control groups participated in a standard curriculum exam. We compared the exam results and observed the application's impact on exam performance. The test groups consisted of 11 first and 12 second year students, while the control groups consisted of 11 first and 13 second year students.

In this section, we first describe the evaluation, followed by an analysis of the collected data.

3.1 Application evaluation - student challenge

During the five-week application evaluation period, four meetings with the test group students were organised. The meetings were held during the music theory classes. Our goal was to observe the student engagement with the application. To gain the interest of students, we proposed a student challenge, where the students competed to gain points and rank high on the leaderboard.

3.1.1 Initial questionnaire

At the first meeting, the students of both control and test groups were given a questionnaire that contained general questions about the use of tools for practicing music theory on mobile devices. The first part of the questionnaire involved questions about which applications (including music theory apps) the students use on their mobile devices. The second part of the questionnaire consisted of questions about the students' rhythm practicing at home.The questionnaire was answered by 47 students.

All students were using mobile applications, such as social, messaging and music apps (SnapChat, Instagram, FB Messenger, YouTube). 79% of the students reported on using mobile apps for learning new skills, such as foreign languages and instruments. However, applications for practicing music theory, such as Teoria.com, TonedEar and MyEarTrainer, were rarely mentioned. Only a few students (17%) used various rhythmic dictation exercises. In the second part of the questionnaire, most students reported practicing rhythmic exercises at home (67%), however they showed mixed opinions on whether they wanted additional ways to exercise rhythmic dictation, as 55% of students did not want additional rhythmic dictation exercises. The shift in their opinion therefore posed a key challenge for the success of the proposed application.

3.1.2 Weeks 2 and 3

During weeks 2 and 3, we enabled access to the application to the test groups. We conducted a live challenge during a music theory class. The goal of the challenge was to increase student engagement with the application, by enticing them to gather points and rank high on the leaderboard. During the challenge, we motivated the students further by presenting intermediate results live on a classroom display. Symbolic rewards were given to the first three students.

During the week 3 meeting, we also distributed questionnaires to the test group students. We asked the students about their experience with the application. The results of

the questionnaire were mostly positive. When asked if the application was difficult to use, all students answered no (100%). Most of them answered that the exercises were not difficult (16 students, 69%) and that they got used to the application's use over time (16 students, 69%). The answers were consistent with the goal that the application should be easy to use. Most students responded that the rhythm input keyboard worked as intended (13 students, 72%). The majority did not use additional paper and pencil (17 students, 94%) exercises. However, the sound of the organ used by the application was perceived as disturbing (16 students, 89%). Many answered that they would rather listen to the piano because they are more accustomed to it. Two thirds of the students had enough time to complete the exercises (12 students, 67%), and did not adjust the speed of the dictation playback (67%).

3.1.3 Week 5

Five weeks after the beginning of the application's evaluation period, we organised the fourth meeting. We presented the final results of the participating students and handed out plaques to the winners of the challenge. All students received symbolic rewards in gratitude.

We also asked the test group students to respond again to the questionnaire which was handed out during the third meeting. We investigated the changes in their opinions after one month of application use. Again, we received positive responses. The students replied that the exercises were not difficult (91%). All students got used to the application during this time. To most students, the rhythm input keyboard functionality seemed logical and worked as intended (82%). All students began using the application to practice and stopped using the conventional paper and pencil practice. The students also grew accustomed to to the sound of the organ used for playback (73 percent).

Most students had enough time to complete the exercises (91%), and did not adjust the playback speed (91%). When asked whether they showed the application to their friends, the majority responded positively (73%). In their final remarks, the students highlighted the following features:

- the students liked the ability of using the application on a personal computer in addition to the mobile device,

- the scoring and achievements (badges) were motivating,

- the ability to pause/stop the playback was helpful.

3.2 Analyzing the application data

Twenty-three students, 11 from the first and 12 from the second year, completed 496 exercises in total. Each exercise consisted of two rhythmic sequences that could be answered multiple times. In total, the students answered 837 sequences correctly. The first-year students averaged 24.5 sequences and the second-year 26 sequences.

The rhythm dictation application is organised into 16 difficulty levels. In order to advance to a higher level, the student had to complete at least 12 exercises at the current level. When starting the application, the student could choose which level to start at from the subset of levels they already achieved. 39% of the students remained at the first level (level 11), because they did not complete enough exercises to pass on to the next level, while others moved to higher levels. During the evaluation, only one student reached all the rhythmic levels available. We also observed the time needed to complete the individual exercises. In their first exercises, the students needed more time than in later repetitions - the average time gradually decreased with the number of exercises played. With increasing difficulty levels, we noticed that the number of event deletions increased. The number of dictation plays remained steady across all levels of difficulty. The number of attempts to solve also remained steady with the exception of level 11 (1/16 difficulty level), where sequences were trivial for the conservatory-level students to solve.
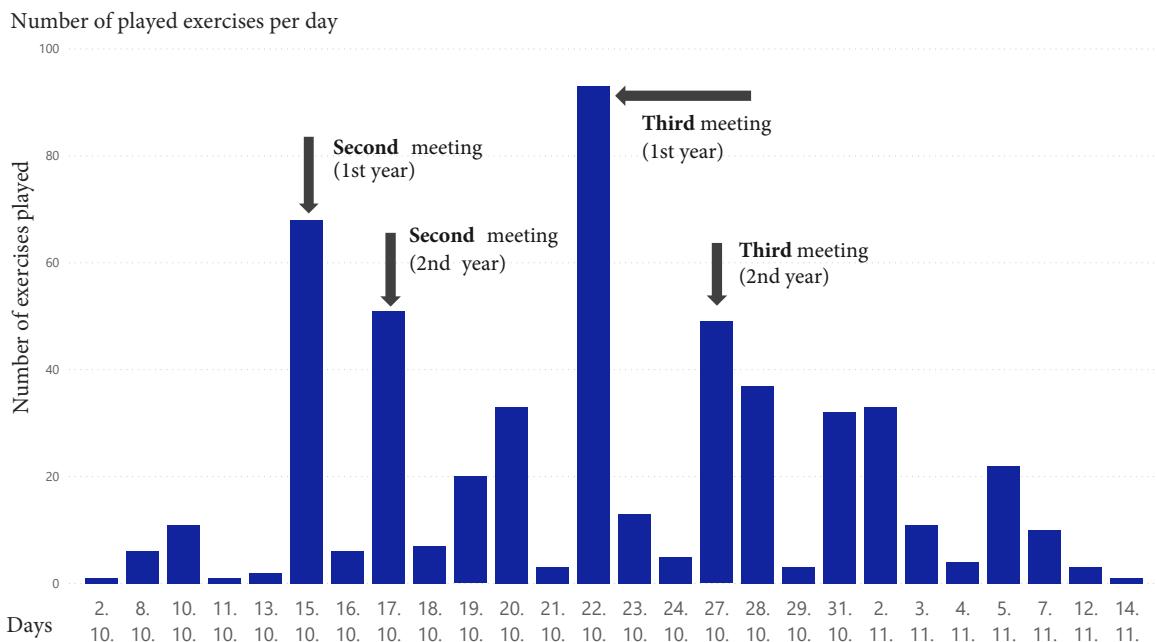
The gathered data confirmed our assumption that some of the observed values, such as time spent, decreased over time, while others remained steady due to the increasing difficulty of exercises. Student engagement in out-of-class use gradually increased, which we consider a success in terms of user experience - student liked the interface and found it easy to use - as well as gamification elements, which, through the student challenge, brought competitiveness into interaction between students.

3.3 Exam performance

At the end of the evaluation period, the students completed an exam as part of their standard curriculum. The exam was taken in the traditional form, with the teacher dictating the rhythmic sequences and students writing their responses on paper. We analyzed the exam results and compared the grades within and between the groups. The exam was evaluated with grades 1 (worst grade) to 5 (best grade). The first-year control group students achieved an average grade of 4.3, while the test group students achieved an average of 4.5 (4% increase). The results were statistically tested using the MannWhitney U-test and the difference was not statistically significant (U=16, p >0.05). A larger difference was observed for second-year students, where the control group achieved an average score of 3.58 and the test group 4.44 (19% better, significant difference, U=24, p<0.001). As better results were achieved by students using the rhythmic dictation application, we can conclude that the use of the application had a positive effect on their performance in the exam. As both groups were relatively small, we also used a resampling method to compare the group averages. At 1000 replicates, the method estimated a 69.2% probability that the average test group score was greater than the average control group score for the first-year student groups. For second-year results, the algorithm estimated this probability at 99.6% at 1,000 iterations. These estimates confirmed the Mann-Whitney U test, therefore showing that students who used the application performed better in the exam.

The difference between the first (no significant impact)

Number of played exercises per day

Figure 4: Graph of the application's usage frequency per day. The second and third meetings are marked individually for each test group. The engagement (# of exercises) was initially higher for the first year students, however, it decreased over time, while the engagement for the second year student group remained steady.

and second year (significant impact) students could be attributed to the fact that the second year students were on average more steadily engaged during the application's evaluation period (Figure 4). The students who ranked highest in the competition rankings were from the second year test group. However, both groups were small and a larger longitudinal study is needed to further confirm the results of this evaluation, and to fully evaluate the application's impact on the learning process and performance.

4. CONCLUSION AND FUTURE WORK

In this paper, we presented the Rhythmic dictator—a rhythmic dictation application. The application features a mobile-friendly user interface supported by gamification elements for attaining student engagement, while offering a flexible environment for the teachers. We investigated two aspects of the application—student engagement and exam performance. To engage students, we created a five week challenge during which the students were asked to use the application through a gamified experience of collecting points and badges, which were visible to other students. Their performance was later tested in a conventional exam, where we compared the results of the students who used and who did not use the application.

The evaluation showed that students support the use of a gamified application. Overall, the students reported a very positive user experience, which was further substantiated by the claim that they would recommend the application to their friends.

The comparison of exam results between the control and the test groups showed a positive impact of the application's use on exam results, which was statistically sig-

nificant for second year students. Although the test and control groups were small and the results should not be too quickly generalised, the study was carried out at the Conservatory of music and ballet, Ljubljana, Slovenia, which represents roughly 50% of the state-wide student population enrolled in a music programme at this level. Based on the evaluation presented in this paper, we can corroborate the gamified Rhythmic dictator application aids the students' performance, which we attribute to gamification and automatization of rhythmic dictation exercises.

To further confirm the application's impact on music theory learning, our current work includes a longitudinal study with new exercise types and an evaluation of the use of the multi-player mode for real-time remote interaction. The application is currently also used in class, where we are collecting new student engagement data.

5. REFERENCES

[1] B. E. Wiggins and B. E, "An Overview and Study on the Use of Games, Simulations, and Gamification in Higher Education," *International Journal of Game-Based Learning*, vol. 6, no. 1, pp. 18–29, jan 2016.

[2] S. de Sousa Borges, V. H. S. Durelli, H. M. Reis, and S. Isotani, "A systematic mapping on gamification applied to education," in *Proc. of the 29th Annual ACM Symposium on Applied Computing - SAC '14*. New York, New York, USA: ACM Press, 2014, pp. 216–222.

[3] C. E. Morton, S. N. Saleh, S. F. Smith, A. Hemani, A. Ameen, T. D. Bennie, and M. Toro-Troconis,

"Blended learning: how can we optimise undergraduate student engagement?" *BMC Medical Education*, vol. 16, no. 1, p. 195, dec 2016.

[4] C. Wagner, *Antistasis an open educational journal*. University of New Brunswick, Faculty of Education, mar 2010, vol. 7.

[5] C. Muntean, "Raising engagement in e-learning through gamification," in *The 6th International Conference on Virtual Learning ICVL 2012*, 2012, pp. 323–329.

[6] R. A. Bartle, "Information reconstruction," in *Proc. of the First International Workshop on Gamification for Information Retrieval - GamifIR '14*. New York, New York, USA: ACM Press, 2014, pp. 1–1.

[7] Y. E. Kim, E. M. Schmidt, and L. Emelle, "MoodSwings: {A} Collaborative Game for Music Mood Label Collection," in *Procedings of the International Conference on Music Information Retrieval*, J. P. Bello, E. Chew, and D. Turnbull, Eds., 2008, pp. 231–236.

[8] M. I. Mandel and D. P. W. Ellis, "A web-based game for collecting music metadata," in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, 2007, pp. 1–6.

[9] E. Law, L. von Ahn, R. B. Dannenberg, and M. J. Crawford, "TagATune: A Game for Music and Sound Annotation," *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2007.

[10] A. Aljanaki, F. Wiering, and R. Veltkamp, "Collecting annotations for induced musical emotion via online game with a purpose emotify," *Technical Report Series*, vol. 2014, no. UU-CS-2014-015, 2014.

LEARNING TO DENOISE HISTORICAL MUSIC

Yunpeng Li   **Beat Gfeller**   **Marco Tagliasacchi**   **Dominik Roblek**

Google

`{yunpeng,beatg,mtagliasacchi,droblek}@google.com`

ABSTRACT

We propose an audio-to-audio neural network model that learns to denoise old music recordings. Our model internally converts its input into a time-frequency representation by means of a short-time Fourier transform (STFT), and processes the resulting complex spectrogram using a convolutional neural network. The network is trained with both reconstruction and adversarial objectives on a synthetic noisy music dataset, which is created by mixing clean music with real noise samples extracted from quiet segments of old recordings. We evaluate our method quantitatively on held-out test examples of the synthetic dataset, and qualitatively by human rating on samples of actual historical recordings. Our results show that the proposed method is effective in removing noise, while preserving the quality and details of the original music.

1. INTRODUCTION

Archives of historical music recordings are an important means for preserving cultural heritage. Most such records, however, were created with outdated equipment, and stored on analog media such as phonograph records and wax cylinders. The technological limitation of the recording process and the subsequent deterioration of the storage media inevitably left their marks, manifested by the characteristic crackling, clicking, and hissing noises that are typical in old records. While "remastering" employed by the recording industry can substantially improve the sound quality, it is a time-consuming process of manual labor. The focus of this paper is an automated method that learns from data to remove noise and restore music.

Audio denoising has a long history in signal processing [1]. Traditional methods typically use a simplified statistical model of the noise, whose parameters are estimated from the noisy audio. Examples of these techniques are spectral noise subtraction [2, 3], spectral masking [4, 5], statistical methods based on Wiener filtering [6] and Bayesian estimators [7, 8]. Many of these approaches, however, focus on speech. Moreover, they often make simplifying assumptions about the structure of the noise, which makes them less effective on non-stationary real-world noise.

Recent advances in deep learning saw the emergence of data-driven methods that do not make such *a priori* assumptions about noise. Instead they learn an implicit noise model from training examples, which typically consist of pairs of clean and noisy versions of the same audio in a supervised setup. Crucial challenges facing the adoption of the deep learning paradigm for our task are: i) can we design a model powerful enough for the complexity of music, yet simple and fast enough to be practical, and ii) how can we train such a model, given that we have no clean ground truth for historical recordings? In this paper, we address these issues and show that it is indeed feasible to build an effective and efficient model for music denoising.

1.1 Related Work

Sparse linear regression with structured priors is used in [9] to denoise music from synthetically added white Gaussian noise, obtaining large SNR improvements on a "glockenspiel" excerpt, and on an Indian polyphonic song. [10] considers the problem of removing artifacts of perceptual coding audio compression with low bit-rates. That work, which uses LSTMs, is the first successful application of deep learning for this type of music audio restoration. Note that in contrast to our work, aligned pairs of original and compressed audio samples are readily available. Statistical methods are applied in [11] to denoise Greek Folk music recorded in outdoor festivities. In [12], the author applies structured sparsity models to two specific audio recordings that were digitized from wax cylinders, and describes the results qualitatively. In [13], the authors describe how to fill in gaps (at known positions) of several seconds in music audio, using self-similar parts from the recording itself.

Our method is also related to audio super-resolution, also known as bandwidth extension. This is the process of extending audio from low to higher sample rates, which requires restoring the high frequency content. In [14, 15] two approaches which work for music are described. On piano music, for example, [15] obtains an SNR of 19.3 when upsampling a low-pass filtered audio from 4kHz to 16kHz.

Many existing denoising approaches focus on speech instead of music [16–19]. Given that these two domains have very different properties, it is not clear a priori how well such methods transfer to the music domain. Nevertheless, our work is inspired by recent approaches that use generative adversarial networks (GANs) to improve the quality of audio [18, 20, 21]. For example, [21] obtains

significant improvements denoising speech and applause sounds that have been decoded at a low bit-rate, using a wave-to-wave convolutional architecture.

In this paper, we present a method to remove noise from historical music recordings, using two sources of audio: i) a collection of historical music recordings to be restored, for which no clean reference is available, and ii) a separate collection of music of the same genre that contains high-quality recordings. We focus on classical music, for which both public domain historical recordings as well as modern digital recordings are available. This paper makes the following contributions:

- We provide a fully automated approach that succeeds in removing noise from historical recordings, while preserving the musical content in high quality. Quality is measured in terms of SNR and subjective scores inspired by MUSHRA [22], and examples on real historical recordings are provided [1].

- Our approach employs a new architecture that transforms audio in the time domain, using a multi-scale approach, combined with STFT and inverse STFT. As this architecture is able to output high-quality music, it may be a useful architecture for other tasks that involve the transformation of music audio.

- We provide an efficient and fully automated method to extract noise segments (without music) from a collection of historical music recordings. This is a key ingredient of our approach, as it allows us to create synthetic pairs of <clean, noisy> audio samples.

The rest of this paper is organized as follows. Our approach is described in detail in Section 2, and experimental results are given in Section 3. We conclude in Section 4.

2. METHOD

Our model is an audio-to-audio generator learned from paired examples with both reconstruction and adversarial objectives.

2.1 Creating paired training examples

For training, we use time-aligned pairs of <clean, noisy> examples, where clean music is used as targets, and noisy music as inputs to the generator. We take a data-driven approach to generate noisy audio from clean references. We synthesize noisy samples by simulating the degradation process affecting the historical recordings, namely applying band-pass filtering, followed by additive mixing with noise samples extracted from "quasi-silence" segments of historical recordings.

Specifically, we scan the noisy historical recordings looking for low-energy segments in the time domain, which corresponds to pauses in the musical scores. To this end, we compute the rolling standard deviation from the raw audio samples with a window size equal to 100ms.

Then, we estimate an adaptive threshold τ based on the q-th quantile of the standard deviations and keep the segments that satisfy the following two conditions: i) the local standard deviation is below τ, and ii) the segment has a minimum duration of T. Intuitively, the value of q is selected based on a trade-off between the number of extracted segments and the need of extracting noise-only segments. In our experiments, we set $q = 0.5\%$ and $T = 100$ms. In this way, from 801 different recordings, we are able to extract around 8900 noise samples.
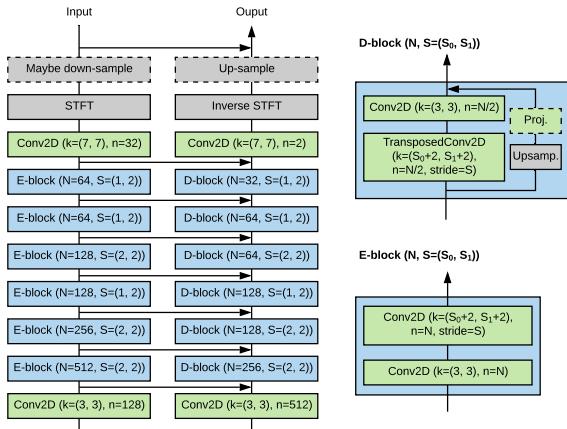
From each of these short noise segments, we need to generate noise samples having the same length as the clean audio references. We do this by replicating the noise segment in time, using overlap-and-add (OLA) with an overlap equal to 20% of the segment length. Given the short duration of most noise segments, this operation alone would lead to periodic noise patterns which differ from the noise characteristics found in historical recordings. Therefore, we alter each noise segment replica before the OLA synthesis step in two ways: i) applying a random perturbation to the phase of the noise segment (adding Gaussian noise $\sim \mathcal{N}(0, 0.1)$ to the phase of the STFT); ii) applying a random shift in time (with wraparound). We found that these simple operations produce longer noise samples with auditory characteristics similar to the ones encountered in the historical recordings, avoiding artificial periodic patterns.

Finally, we create time-aligned pairs of <clean, noise> examples by: i) applying band-pass filtering with cut-off frequencies randomly sampled in [50Hz, 150Hz] and [5kHz, 10kHz], respectively; ii) mixing a randomly selected noise sample with a gain in the range [10dB, 30dB].

2.2 Model architecture

The generator processes the audio in the time-frequency domain. It first computes the STFT of the input, the real and imaginary components of which are then fed as a 2-channel image to a 2D convolutional U-Net [23] followed by an inverse STFT back to the time domain. Finally the output is added back to the input, making the model a residual generator.

The U-Net in our generator is a symmetric encoder-decoder network with skip-connections, where the architecture of the decoder layers mirrors that of the encoder and the skip-connections run between each encoder block and its mirrored decoder block. Each encoder block is a 3×3 convolution followed by either a 3×4 convolution with stride of 1×2 (if down-sampling in the frequency dimension), or a 4×4 convolution with stride of 2×2 (if down-sampling in both time and frequency dimensions). We choose kernel sizes to be multiples of strides to ensure even contribution from all locations of the input feature map, which prevents the formation of checkerboard-like patterns in resampling layers [24]. The decoder blocks mirror the encoder blocks, and each consists of a transposed convolution for up-sampling followed by a 3×3 convolution. Each decoder block additionally includes a shortcut connection between its input and output. The shortcut consists of a nearest-neighbor up-sampling layer, which

---

[1] https://www.youtube.com/playlist?list=PLa5CkN3odpnxi3WqMH4MgVk7XUjCP99d3

Figure 1. Generator architecture. Dashed-line components are included on a need-to-have basis: Up/down-sampling of the input/output audio is needed for processing at coarser resolutions in a multi-scale setup; The linear projection (by 1x1 convolution) in the decoder block is present only when the output of the block has a different number of channels from its input.

is followed by a linear projection using 1x1 convolution when the output has a different number of channels from the input. We do not include a shortcut in the encoder block, since it already shares the same input with a U-Net skip connection and therefore only needs to produce the residual complementary to the skip path. The architecture of the generator is shown in Figure 1.

We use two discriminators for the adversarial objective, one in the waveform domain and one in the STFT domain. The STFT discriminator has the same architecture as the encoder module of the generator. For the waveform discriminator, we use the same architecture as MelGAN [25] except that we only double (instead of quadruple) the number of channels in the down-sampling layers. We found this light-weight version to be sufficient in our setup, and that using the full version had no additional benefit. Both discriminators are fully convolutional. Hence the waveform discriminator produces a 1D output spanning the time domain, and the STFT discriminator has a 2D output spanning the time-frequency domain.

We use weight normalization [26] and ELU activation [27] in the generator, while layer normalization [28] and Leaky ReLU activation [29] with $\alpha = 0.3$ are used in the discriminator.

2.2.1 STFT Representation

In the generator, the STFT is represented by a 2-channel image, where the channels are the real and imaginary components. We also explored a polar representation, where the channels are the modulus and the phase; additionally we experimented with processing only the modulus channel and reusing the original phase, as is done in [30]. Nevertheless, we found the real/imaginary representation to perform better in our experiments.

Furthermore, we tried aligning the phase so that the phase in each frame is coherent with a global reference (e.g., the first frame) rather than its local STFT window. Again, we observed no advantage in doing so, which suggests that the neural network is capable of internally handling the phase offsets. Unlike [30], we do not convert STFT to logarithmic scale as we found it be detrimental to performance (even with various smoothing and normalization schemes).

2.2.2 Multi-scale Generator

We can further stack multiple copies of the generator described above, each with its own separate parameters, in a coarse-to-fine fashion: The generators at earlier stages process the audio at reduced temporal resolutions, whereas the later-stage generators focus on restoring finer details. This is equivalent to halving the sampling rate in each scale. This type of multi-scale generation scheme is routinely used in computer vision and graphics to produce high-resolution images (e.g., [31]).

Let K be the total number of scales, then generator G_k at scale k ($k \in \{0, \ldots, K-1\}$) down-samples its input by a factor of 2^k before computing the STFT and up-samples the output residual (after computing the inverse STFT) by the same factor to match the resolution of the input. The overall generator G is the composite of $G_0 \circ \cdots \circ G_{K-1}$.

Compared with simply stacking U-Nets all at the original input resolution, as done in [32], the benefit of the multi-scale approach is two-fold: i) the asymptotic computational complexity is constant with respect to the number of scales, as opposed to linear in [32], due to exponentially decreasing input sizes at coarser levels; ii) the intermediate outputs of the generator correspond to the input audio processed at lower resolutions, which allows us to meaningfully impose multi-scale losses on the intermediate outputs in addition to the final output. We will describe how this can be accomplished in the next section.

2.3 Training

The generator can be trained using the reconstruction loss between the denoised output and the clean target. This can be further complemented with an adversarial loss, given by discriminators trained simultaneously with the generator, a practice often used in audio enhancement (e.g., [18,20,30], among others). In the case of our multi-scale generator, we use the same number of waveform and STFT discriminators as generator scales. This way, there is one discriminator of both types for each of the (down-sampled) intermediate outputs and final output in each domain. For the adversarial loss, we use the hinge loss averaged over multiple scales. Since the discriminators are convolutional, this loss is further averaged over time for the waveform discriminator and over time-frequency bins for the STFT discriminator. Similarly, the reconstruction loss is also imposed on the outputs at each scale.

More formally, let (x, y) denote a training example, where x is the noisy input and y is the clean target, and $k \in \{0, \ldots, K-1\}$ denote the scale index. Hence y_k is the

clean audio down-sampled to scale k, and $\hat{y}_k$ represents the intermediate output of the generator $G_k \circ \cdots \circ G_{K-1}(x)$ down-sampled to the same scale. Note that for the finest scale $k = 0$ at full resolution, $y_0 = y$ is simply the original clean audio and $\hat{y} \triangleq \hat{y}_0 = G(x)$ is the final output of the generator. Thus the L^1 reconstruction loss in the STFT domain can be written as

$$\mathcal{L}_G^{\text{rec}} = \mathbb{E}_{(x,y)} \left[\sum_k \frac{\|\omega_k - \hat{\omega}_k\|_1}{S_k^{\text{STFT}}} \right], \qquad (1)$$

where 2D complex tensors ω_k and $\hat{\omega}_k$ denote the STFT of down-sampled clean audio y_k and generator output $\hat{y}_k$ for scale k, respectively, and S_k^{STFT} is the total number of time-frequency bins in ω_k and $\hat{\omega}_k$. We find this STFT-based reconstruction loss to perform better than either imposing per-sample losses directly in the waveform domain or using losses computed from the internal "feature" layers of discriminators (e.g. [25]).

For the adversarial loss, let t denote the temporal index over all T_k logits of the waveform discriminator at scale k (recalling that the discriminators are fully convolutional) and let s denote the index over all S_k logits of the STFT discriminator. Then discriminator losses in the wave and STFT domains can be written as, respectively,

$$\mathcal{L}_D^{\text{wave}} = \mathbb{E}_y \left[\sum_{k,t} \frac{1}{T_k} \max(0, 1 - D_{k,t}^{\text{wave}}(y_k)) \right] +$$

$$\mathbb{E}_x \left[\sum_{k,t} \frac{1}{T_k} \max(0, 1 + D_{k,t}^{\text{wave}}(\hat{y}_k)) \right] \qquad (2)$$

$$\mathcal{L}_D^{\text{STFT}} = \mathbb{E}_y \left[\sum_{k,s} \frac{1}{S_k} \max(0, 1 - D_{k,s}^{\text{STFT}}(y_k)) \right] +$$

$$\mathbb{E}_x \left[\sum_{k,s} \frac{1}{S_k} \max(0, 1 + D_{k,s}^{\text{STFT}}(\hat{y}_k)) \right], \quad (3)$$

and the corresponding adversarial loss for the generator is given by

$$\mathcal{L}_G^{\text{adv}} = \mathcal{L}_G^{\text{adv, wave}} + \mathcal{L}_G^{\text{adv, STFT}}$$

$$= \mathbb{E}_x \left[\sum_{k,t} \frac{1}{T_k} \max(0, 1 - D_{k,t}^{\text{wave}}(\hat{y}_k)) + \right.$$

$$\left. \sum_{k,s} \frac{1}{S_k} \max(0, 1 - D_{k,s}^{\text{STFT}}(\hat{y}_k)) \right]. \qquad (4)$$

The overall generator loss is a weighted sum of the adversarial loss and the reconstruction loss, i.e.,

$$\mathcal{L}_G = \mathcal{L}_G^{\text{rec}} + \lambda \cdot \mathcal{L}_G^{\text{adv}}. \qquad (5)$$

We set the weight of the adversarial loss λ to 0.01 in all our experiments, except those where we do not use discriminators (which corresponds $\lambda=0$). We train the model with TensorFlow for 400,000 steps using the ADAM [33]

optimizer, with a batch size of 16 and a constant learning rate of 0.0001 with $\beta_1 = 0.5$ and $\beta_2 = 0.9$. For the STFT, we use a window size of 2048 and a hop size of 512 when there is only a single scale. For each added scale we halve the STFT window size and hop size *everywhere*. This way the STFT window at the coarsest scale has a receptive field of 2048 samples at the original resolution, whereas finer levels have smaller receptive fields and hence focus more on higher frequencies.

Our model has around 9 million parameters per scale in the generator. At inference-time, it takes less than half a second for every second of input audio on a modern CPU and more than an order of magnitude faster on GPUs.

3. EXPERIMENTS

We evaluate our model on a dataset of synthetically generated noisy-clean pairs, using both objective and subjective metrics. In addition, we also provide a subjective evaluation on samples from real historical recordings, for which the clean references are not available.

3.1 Datasets

Our data is derived from two sources: i) digitized historical music recordings from the Public Domain Project [34], and ii) a collection of classical music recordings of CD-quality. The historical recordings are used in two ways: i) to extract realistic noise from relatively silent portions of the audio, as described in Section 2.1; and ii) to evaluate different methods based on the human-perceived subjective quality of their outputs. The modern recordings are used for mixing with the extracted noise samples to create synthetic noisy music, as well as serving as the clean ground truth. We additionally filter our data to retain only classical music, as it is by far the most represented genre in historical recordings. The resulting dataset consists of pairs of clean and noisy audio clips, both monophonic and 5 seconds long, sampled at 44.1kHz. The total duration of the clean clips is 460h.

3.2 Quantitative Evaluation

We quantitatively evaluate the performance of different methods on a held-out test set of 1296 examples from the synthetic noisy music dataset. For the neural network models, whose training is stochastic, we repeat the training process 10 times for each model and report the mean for each metric and its standard error.

Evaluation metrics: Objective metrics such as the signal-to-noise ratio (SNR) faithfully measure the difference between two waveforms on a per-sample basis, but they often do not correlate well with human-perceived reconstruction quality. Therefore, we additionally measure the *VGG distance* between the ground truth and the denoised output, which is defined as the L^2 distance between their respective embeddings computed by a VGGish network [35]. The embedding network is pre-trained for multi-label classification tasks on the YouTube-100M dataset, in which labels are assigned automatically based

	ΔSNR (dB)	-ΔVGG
1 scale	**3.4±0.0**	0.68±0.01
2 scales	**3.4±0.0**	**0.78±0.01**
3 scales	3.2±0.0	0.73±0.01

Table 1. Performance of our model with different numbers of scales K in terms of SNR gain (ΔSNR) and VGG distance reduction (-ΔVGG). Higher is better.

on a combination of metadata (title, description, comments, etc.), context, and image content for each video. Hence we expect the VGG distance to focus more on higher-level features of the audio and less on per-sample alignment. Note that the same embedding used by Frechét audio distance (FAD) [36], which measures the distance between two *distributions*. However, FAD does not compare the content of individual audio samples, and is hence not applicable to denoising.

We report the SNR gain (ΔSNR) and VGG distance reduction (-ΔVGG) of the denoised output relative to the noisy input, averaged over the test set. For reference, the noisy input has an average SNR of 14.4dB and VGG distance of 2.09. Table 1 shows the performance of our model with different numbers of scales. We use $K = 2$ scales for the rest of our experiments. We evaluate variants of our proposed model in an ablation study and compare with alternative approaches and well-established signal processing baselines:

- **Ours, λ=0**: Our model trained with only reconstruction loss.

- **Ours, λ=0.01**: Our model trained with both adversarial and reconstruction losses.

- **Ours, bypass phase**: Same as above, except that the phase of the noisy input is reused and only the modulus of the STFT is processed by the U-Net (as a single-channel image). This is similar to the approach of [30], but trained and evaluated for music denoising instead of speech.

- **MelGAN-UNet**: A 1D-convolutional waveform-domain generator inspired by MelGAN [25], where the decoder is the same as the generator of MelGAN and the encoder mirrors the decoder.

- **DeepFeature generator**: The 1D-convolutional waveform-domain generator of [17], which does not use U-Net but rather a series of 1D convolutions with exponentially increasing dilation sizes. Unlike U-Net, the temporal resolution and number of channels remain unchanged in all layers of this network.

- **log-MMSE**: A short-time spectral amplitude estimator for speech signals which minimizes the mean-square error of the log-spectra [37]. In our implementation, the estimation of the noise spectrum is based on low-energy frames across the whole clip, rather than considering the frames at the start of the audio clip. We use this deviation from the standard implementation as it gives better SNR results.

- **Wiener**: A linear time-invariant filter that minimizes the mean-square error. We adopted the SciPy [38] implementation and used default parameters, as different parameter settings did not improve the results.

For waveform-domain generators, we tried waveform-domain losses – including reconstruction losses in the "feature space" of discriminator internal layers [17, 25] – as well as STFT-domain losses, and found the former to work better with the DeepFeature generator while the latter gave better results for the MelGAN-UNet generator. The results shown for these generators are those obtained with the better loss variant. We also divide the test set into three subsets, each containing the same number of examples, with low noise (avg. 19.8dB SNR), medium noise (avg. 14.2dB SNR), and high noise (avg. 9.4dB SNR), and compute the same metrics on each subset as well as on the full test set.

The results in Table 2 show that, for all noise levels, our model consistently outperforms the signal processing baselines and the waveform-domain neural network models, which have proven highly successful in speech enhancement but are not adequate for the complexity of music signals. The signal-processing baselines (log-MMSE and Wiener filtering) are hardly able to improve upon the noisy input at all. This is not too surprising given the non-Gaussian, non-white nature of the real-world noise in the evaluation data. Comparing the results among the variants of our model, we further make the following observations:
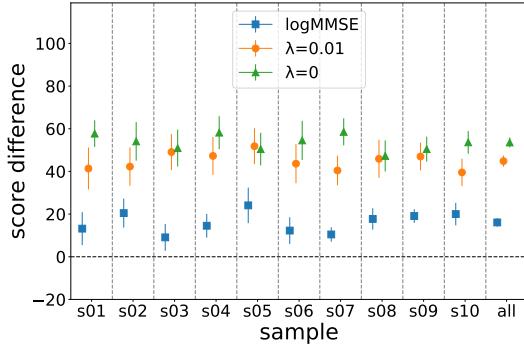
- Using adversarial losses does not help in terms of SNR, as is evident from the top two rows of Table 2. The SNR decrease is small but significant. The adversarially trained variant, however, scores better on the high-level feature oriented VGG distance metric, which is in line with past observations [18, 25]

- It is advantageous to take both the modulus and the phase into account when processing the STFT spectrogram, as the "bypass-phase" variant which reuses the input phase produces consistently worse results across all noise levels. This shows that the proposed model is able to reconstruct the fine-grained phase component of the original clean music.

3.3 Subjective Evaluation
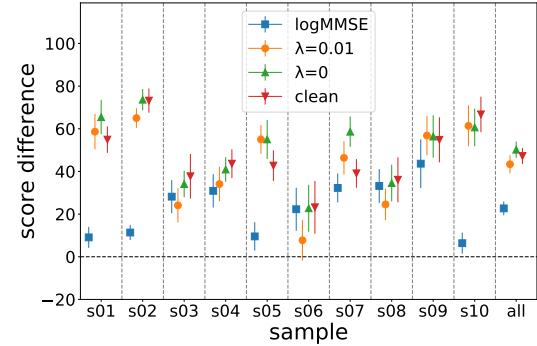
In the previous section we compared results by means of objective quality metrics, which can be quantitatively computed from pairs of noisy-clean examples. These metrics can be conveniently used to systematically run an evaluation over a large number of samples. However, it is difficult to come up with an objective metric that correlates with quality as perceived by human listeners. Indeed, the SNR and VGG distance metrics do not agree in our quantitative evaluation – the proposed model is better in terms of VGG distance, but worse in terms of SNR compared to its counterpart without discriminator. We now describe our

	ΔSNR (dB)				-ΔVGG			
	noise level				noise level			
	low	medium	high	all	low	medium	high	all
Ours, λ=0	**2.5±0.0**	**4.1±0.0**	**4.3±0.0**	**3.7±0.0**	0.30±0.01	0.47±0.01	0.58±0.01	0.45±0.01
Ours, λ=0.01	2.2±0.0	3.9±0.0	4.1±0.0	3.4±0.0	**0.66±0.01**	**0.81±0.01**	**0.87±0.01**	**0.78±0.01**
Ours, bypass phase	2.1±0.0	3.5±0.0	3.7±0.0	3.1±0.0	0.62±0.01	0.77±0.01	0.83±0.01	0.74±0.01
MelGAN-UNet	1.7±0.0	2.9±0.0	3.1±0.0	2.6±0.0	0.16±0.02	0.15±0.03	0.18±0.02	0.16±0.02
DeepFeature generator	-0.7±0.4	1.3±0.1	1.7±0.1	0.8±0.2	0.00±0.02	0.03±0.02	0.00±0.01	0.01±0.02
log-MMSE	-1.4	-0.2	0.1	-0.5	-0.15	-0.04	0.01	-0.07
Wiener	0.1	0.1	0.1	0.1	0.01	0.02	0.01	0.01

Table 2. Performance of different variants of our model and alternative approaches, evaluated on subsets of examples with different noise levels as well as on the full test set.

Figure 2. Average score differences for the historical recordings dataset, relative to the original noisy sample.

Figure 3. Average score differences for the synthetic dataset, relative to the noisy sample.

subjective evaluation which we ran in order to identify the method that performs best when judged by humans.

Following recent work on low-bitrate audio improvement [21], we use a score inspired by MUSHRA [22] for our subjective evaluation. Each rater assigned a score between 0 and 100 to each sample. The main difference to actual MUSHRA scores is that since no clean reference exists for historical recordings, we do not include an explicit reference in the rated samples (although we do include the clean sample in the synthetic dataset evaluation).

We perform our evaluation on 10 samples of historical recordings, and separately on 10 samples from the synthetic dataset, using 11 human raters. As in the objective evaluation, each sample is 5 seconds long. We evaluate the following four versions for each sample: (i) Original historic audio example, (ii) denoised example using our model with λ=0.01, (iii) denoised example using our model with λ=0, (iv) denoised example using log-MMSE.

For the synthetic dataset, we use the four versions above, but instead of the historic audio we use the synthetically noisified one. We do not include Wiener filtering as a competing baseline here since we noticed that it produces outputs that are consistently near-identical to the noisy input, and hence including it in the subjective evaluation would provide little value. We use the original noisy audio as the reference from which to compute score differences for the historical recordings, and the synthet-

ically noisified sample as the reference for the synthetic data. The results are shown in Figure 2 for the historical recordings, and in Figure 3 for the synthetic dataset. Error bars are 95% confidence intervals, assuming a Gaussian distribution of the mean. Both of our methods significantly improve the historical recordings, by around 50 points on average. In comparison, the logMMSE baseline only improves by an average of 16 points. We also performed a Wilcoxon signed-rank test between our λ=0.01 and λ=0 models, to find that the difference is statistically significant (p-value $< 1.19 \times 10^{-11}$). On the synthetic data, again the λ=0 model outperforms the λ=0.01 variant, with a p-value $< 2.13 \times 10^{-8}$. On the other hand, there is no significant difference between the mean score differences of the λ=0 model and the clean sample (p-value = 0.097).

4. CONCLUSION

We presented a learning-based method for automated denoising and applied it to restoration of noisy historical music recordings, matching a high quality bar: Judged by human listeners on actual historical records, our method improves audio quality by a large margin and strongly outperforms existing approaches on a MUSHRA-like quality metric. On artificially noisified music, it even attains a quality level that listeners found to be statistically indistinguishable from the ground truth.

5. REFERENCES

[1] S. H. Godsill and P. J. Rayner, *Digital Audio Restoration: A Statistical Model Based Approach*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 1998.

[2] M. Berouti, R. Schwartz, and J. Makhoul, "Enhancement of speech corrupted by acoustic noise," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, 1979, pp. 208–211.

[3] S. Kamath and P. Loizou, "A multi-band spectral subtraction method for enhancing speech corrupted by colored noise," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, 05 2002.

[4] A. M. Reddy and B. Raj, "Soft mask methods for single-channel speaker separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 6, pp. 1766–1776, 2007.

[5] E. M. Grais and H. Erdogan, "Single channel speech music separation using nonnegative matrix factorization and spectral masks," in *International Conference on Digital Signal Processing (DSP)*, 2011, pp. 1–6.

[6] P. Scalart and J. V. Filho, "Speech enhancement based on a priori signal to noise estimation," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 2, 1996, pp. 629–632.

[7] H. Attias, J. C. Platt, A. Acero, and L. Deng, "Speech denoising and dereverberation using probabilistic models," in *Advances in Neural Information Processing Systems 13*, 2001, pp. 758–764.

[8] P. C. Loizou, "Speech enhancement based on perceptually motivated bayesian estimators of the magnitude spectrum," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 857–869, 2005.

[9] C. Fevotte, B. Torresani, L. Daudet, and S. J. Godsill, "Sparse linear regression with structured priors and application to denoising of musical audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 174–185, 2008.

[10] J. Deng, B. W. Schuller, F. Eyben, D. Schuller, Z. Zhang, H. Francois, and E. Oh, "Exploiting time-frequency patterns with LSTM-RNNs for low-bitrate audio restoration," *Neural Computing and Applications*, vol. 32, no. 4, pp. 1095–1107, 2020. [Online]. Available: https://doi.org/10.1007/s00521-019-04158-0

[11] N. Bassiou, C. Kotropoulos, and I. Pitas, "Greek folk music denoising under a symmetric α-stable noise assumption," in *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, 2014, pp. 18–23.

[12] V. Mach, "Denoising phonogram cylinders recordings using structured sparsity," in *2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2015, pp. 314–319.

[13] N. Perraudin, N. Holighaus, P. Majdak, and P. Balazs, "Inpainting of long audio segments with similarity graphs," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. PP, pp. 1–1, 02 2018.

[14] V. Kuleshov, S. Z. Enam, and S. Ermon, "Audio super resolution using neural networks," in *5th International Conference on Learning Representations (ICLR) 2017, Workshop Track, Toulon, France*, 2017.

[15] S. Birnbaum, V. Kuleshov, Z. Enam, P. Koh, and S. Ermon, "Temporal film: Capturing long-range sequence dependencies with feature-wise modulations," in *Proc. 33rd Annual Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

[16] M. Michelashvili and L. Wolf, "Audio denoising with deep network priors," *CoRR*, vol. abs/1904.07612, 2019. [Online]. Available: http://arxiv.org/abs/1904.07612

[17] F. G. Germain, Q. Chen, and V. Koltun, "Speech denoising with deep feature losses," 2018.

[18] S. Pascual, A. Bonafonte, and J. Serrà, "SEGAN: Speech enhancement generative adversarial network," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, 08 2017, pp. 3642–3646.

[19] D. Rethage, J. Pons, and X. Serra, "A wavenet for speech denoising," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5069–5073.

[20] C. Donahue, B. Li, and R. Prabhavalkar, "Exploring speech enhancement with generative adversarial networks for robust speech recognition," *CoRR*, vol. abs/1711.05747, 2017. [Online]. Available: http://arxiv.org/abs/1711.05747

[21] A. Biswas and D. Jia, "Audio codec enhancement with generative adversarial networks," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[22] "Method for the subjective assessment of intermediate quality levels of coding systems ITU-Recommendation BS.1534-3," 2015. [Online]. Available: www.itu.int/rec/R-REC-BS.1534

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[24] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[25] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, and A. Courville, "MelGAN: Generative adversarial networks for conditional waveform synthesis," 2019.

[26] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 901–909.

[27] S. H. Djork-Arné Clevert, Thomas Unterthiner, "Fast and accurate deep network learning by exponential linear units (elus)," in *ICLR*, 2016.

[28] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[29] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[30] S. Abdulatif, K. Armanious, K. Guirguis, J. T. Sajeev, and B. Yang, "Aegan: Time-frequency speech denoising via generative adversarial networks," 2019.

[31] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=Hk99zCeAb

[32] K. Armanious, C. Yang, M. Fischer, T. Küstner, K. Nikolaou, S. Gatidis, and B. Yang, "Medgan: Medical image translation using GANs," *CoRR*, vol. abs/1806.06397, 2018. [Online]. Available: http://arxiv.org/abs/1806.06397

[33] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[34] "Public domain project," http://pool.publicdomainproject.org, [Online; accessed February-2020]. [Online]. Available: http://pool.publicdomainproject.org

[35] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017. [Online]. Available: https://arxiv.org/abs/1609.09430

[36] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms," in *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, G. Kubin and Z. Kacic, Eds. ISCA, 2019, pp. 2350–2354. [Online]. Available: https://doi.org/10.21437/Interspeech.2019-2219

[37] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 2, pp. 443–445, 1985.

[38] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

MULTILINGUAL LYRICS-TO-AUDIO ALIGNMENT

Andrea Vaglio[12]   **Romain Hennequin**[1]   **Manuel Moussallam**[2]
Gaël Richard[2]   **Florence d'Alché-Buc**[2]

[1] Deezer R&D

[2] LTCI, Télécom Paris, Institut Polytechnique de Paris

research@deezer.com

ABSTRACT

Lyrics-to-audio alignment methods have recently reported impressive results, opening the door to practical applications such as karaoke and within song navigation. However, most studies focus on a single language - usually English - for which annotated data are abundant. The question of their ability to generalize to other languages, especially in low (or even zero) training resource scenarios has been so far left unexplored. In this paper, we address the lyrics-to-audio alignment task in a generalized multilingual setup. More precisely, this investigation presents the first (to the best of our knowledge) attempt to create a language-independent lyrics-to-audio alignment system. Building on a *Recurrent Neural Network* (RNN) model trained with a *Connectionist Temporal Classification* (CTC) algorithm, we study the relevance of different intermediate representations, either character or phoneme, along with several strategies to design a training set. The evaluation is conducted on multiple languages with a varying amount of data available, from plenty to zero. Results show that learning from diverse data and using a universal phoneme set as an intermediate representation yield the best generalization performances.

1. INTRODUCTION

Lyrics-to-audio alignment aims at synchronizing lyrics text units such as paragraphs, lines or words to the timed position of their appearance in the audio signal. Tools dedicated to this task have many practical applications: they can be applied to generate new annotated data to train more robust singing voice recognizers [1]; or be used as building blocks in specific applications such as karaoke [2], navigation within songs [3] or explicit lyrics removal [4]. Lyrics alignment methods are typically inspired from text-to-speech methods. Although text-to-speech alignment is a mature [5] and widely studied task [6], lyrics-to-audio alignment remains a challenging problem with specific limitations. First, the musical accompaniment acts as

a loud background "noise", potentially highly correlated with the signal of interest since vocalists usually sing in harmony and rhythm with instruments. A singing voice separation algorithm pre-processing step is often used to partially overcome this problem [7]. Second, singing voice exhibits more variety than speech with potentially large phonemes pronunciation variations between songs and extended tessitura. Recent studies have proposed efficient alignment methods for singing voice [8,9], but only for the English language, for which annotated data is abundant. The question of their ability to generalize to other languages, especially in low (or even zero) training resource scenarios, has not been properly addressed.

Arguably a monolingual evaluation is unrepresentative of the variety of music recordings available in large scale collections. Commercial streaming services commonly serve content in hundreds of languages and a non-negligible number of popular songs even have multilingual lyrics [10]. However, annotated data on this type of content are scarce. A source of inspiration comes from the related field of multilingual speech recognition [11]. Transfer learning methods [12] have been shown to improve performance on language with few to zero training data. However, this improvement on low-resource languages can sometimes be detrimental to performances on languages with more resources [11].

The goal of this paper is to evaluate and extend state of the art lyrics-to-audio alignment methods to a language-independent setup. First, we review the fitness of these systems to the multilingual framework. Then, we focus on one architecture and study two key features likely to allow generalization to several languages: 1) the intermediate representation space (character versus phoneme) and 2) the design of the training dataset. Evaluation is performed on multiple languages with various amounts of data available, from plenty to zero. The paper is organised as follows: related works are presented in Section 2. We then describe the proposed method in Section 3. The experimental setup and results are described respectively in Section 4 and Section 5. Finally, conclusions are drawn in Section 6 and future works are discussed.

2. RELATED WORKS

Singing voice alignment methods are typically inspired from text-to-speech alignment systems. Classically, an

acoustic model is trained and used to force text to audio alignment using a Viterbi algorithm [5]. These models are usually trained using alignment annotations, at the frame level, between audio and text. However, the availability of such annotations is very limited for polyphonic music where they are traditionally generated by employing an intermediate model [1], leading to suboptimal performances [8]. More generally, the development of such approaches for singing voice was slowed down by the lack of publicly available annotated dataset at word or even line level. Some models were trained on speech and adapted to singing using speaker adaptation technique and a small singing dataset. For instance, in [13], a monophone *Hidden Markov Model* (HMM) is trained on speech and adapted on a small corpus of manually annotated *a cappella* songs with *Maximum Likelihood Linear Regression* (MLLR). The alignment is then performed on polyphonic songs after extracting the singing voice with a melody transcription and a sinusoidal modeling technique. Other models were trained with "low quality" automatic annotations generated with forced alignment using an *Automatic Speech Recognition* (ASR) system. In [1], a speech recognizer is used to generate a large amount of singing annotations by aligning a large corpus of *a cappella* singing to their corresponding lyrics. Annotations are then used to train a new acoustic model. This new model is used to align 19 vocal tracks from English language pop songs: the phoneme sequence is estimated for each track and its Levenshtein distance to the ground truth sequence from the lyrics is computed to find the alignment path. To help alignment, multiple approaches tried extending speech recognizers with external information such as chords [14], score [15] or note onsets [16].

The recent release of the DALI dataset [17] has led to significant progress in lyrics-to-audio alignment. This dataset is the first publicly annotated singing voice dataset available. It contains 5358 audio tracks with time-aligned lyrics at paragraph, line and word levels. These annotations are created from manual annotations and are considered to be very good. It is composed of varied western genres (*e.g.* rock, rap and electronic) in several languages. Novel singing voice separation algorithms displayed impressive results [18] and were also shown to improve significantly lyrics-to-audio alignment systems performances [7]. State-of-the-art approaches for lyrics alignment were compared in the MIREX 2019 challenge [1] . Two submitted systems showed particularly strong performances. The first one was SDE2, described in [8]. It is based on an end-to-end audio-to-character architecture, more precisely a wave-U-net. A preprocessing step of singing voice separation is performed, during training and inference, using a U-net convolutional network. The acoustic model is trained on a private English dataset of 40000 songs using a CTC algorithm. The second one was GYL1, described in [9], which obtained the best results on the challenge. It is based on a *Time Delay Neural Network* (TDNN) which

is trained on the English subpart of the DALI dataset. It uses an extended lexicon to cope with long vowels duration in singing and genre labelling information (phoneme units are annotated with genres) but does not rely on a preprocessing step of singing voice separation.

Although it achieved the best performances in the MIREX challenge, GYL1 can not be straightforwardly used in a multilingual setup: it is composed of multiple parts, some of them, such as the pronunciation dictionary and the language model, being specific to English. To be able to use it on a new language, it would require to modify, or retrain, these parts. In comparison, SDE2 is based on an end-to-end acoustic model, trained with CTC algorithm, that directly outputs characters. It is more suitable to perform multilingual lyrics-to-audio alignment as it can be theoretically applied to any languages being based on the same script (writing system) as the training language.

Employing characters may not be optimal for multilingual lyrics-to-audio alignment: [8] suggest that using phoneme as an intermediate representation could be more relevant for aligning song in other languages. They argue that, for phoneme based systems, only the pronunciation dictionary has to be replaced for a new language, while a character based system is limited by the set of characters that the acoustic model outputs. For instance, SDE2 can only be used to align songs in Latin script languages. The output of the acoustic model could be extended with characters from scripts of new languages, as in [19], but it would require retraining the acoustic model each time a new script is added in the language pool. Using phoneme as an intermediate representation, any language can be theoretically aligned for any trained model if a pronunciation dictionary is available. In this work, we study a system inspired from [8] using either a character or a phoneme intermediate representation.
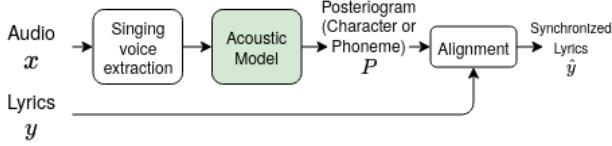
3. PROPOSED METHOD

A general overview of the proposed system is described in Figure 1. It is composed of three parts: a singing voice separation model, an acoustic model and a lyrics-to-audio alignment procedure. It takes as input a song x, its corresponding lyrics y and output the synchronized lyrics $\hat{y}$. Vocals are extracted from the song using a singing voice separation module. The acoustic model processes features extracted from the isolated vocal signal. The acoustic model consists in an RNN trained with a CTC algorithm. The set of outputs of the acoustic model is either characters of the Latin alphabet or phonemes of an universal phoneme set. Lyrics-to-audio alignment is performed on outputs of the acoustic model by a CTC-based alignment decoding function.

3.1 Acoustic model

The acoustic model of our system is a RNN trained with a CTC algorithm. CTC-based acoustic models were successfully used for multilingual speech recognition [19,20]. The RNN part is composed of bi-directional *Long Short-*

Figure 1. Overview of the lyrics-to-audio alignment system. Our study focuses on the training of the acoustic model (section 3.1) and the design of intermediate posteriogram representation spaces (section 3.2). The alignment block is described in section 3.3

Term Memory (LSTM) layers. Authors in [21] argue that such models can give reliable alignments given that outputs at each frame depend on the entire input sequence. In contrast, uni-directional CTC acoustic model suffers from alignment delay [22].

CTC makes it possible to directly train RNN models using weakly aligned annotations, e.g. at word or line level. To do that, the CTC algorithm introduces a new symbol called "blank" (noted ϵ) which represent a non-emission token. The total probability of the output label sequence is then marginalized over all possible alignment for a given input. In our case, the output label sequence is a sequence of character or phoneme. Since the objective function is differentiable, the network can be trained with standard back-propagation through time. The CTC algorithm is more extensively described in [23].

3.2 Character vs Phoneme

We consider two different intermediate representations for our architecture. The first one is a character set, here the Latin alphabet. This representation does not need any kind of expert linguistic knowledge as the acoustic model directly outputs characters probability. However, such a representation is not suitable to perform alignments of songs in a language with a different script. To process those, the acoustic model would need to be retrained with new data on the given script. Moreover, even for languages sharing the same script, a character-based representation is suboptimal for transferring knowledge between languages, as characters pronunciation can significantly differ from one language to another. Our approach relies on the following remarks: all languages share some common phonemes and phonemes are considered to be language independent [24], i.e. to be pronounced the same way across languages. Therefore, using a universal phoneme set as an intermediate representation makes it possible to leverage similarity between sounds across languages. The idea is to use consistent phonemes across languages used for training and that most phonemes from an unseen language appear in the languages used for training.

It can be achieved using *international phonetic alphabet* (IPA) symbols. The IPA is a set of phonetic notations which is a standardized representation of sounds of all spoken language. IPA Pronunciations of words from all languages can be obtained using *Grapheme-To-Phoneme* (GTP) tools. Such tools are available for most common

languages. The universal phoneme set is created by concatenating and merging the union of phoneme sets of all languages based on their IPA symbols.

3.3 Lyrics to audio alignment

In order to align a song to its corresponding lyrics y, the audio is sliced into segments of 5 seconds with a step size of 2.5 seconds. A posteriogram is generated by the trained acoustic model for each segment. To obtain the final posteriogram, all segments posteriograms are concatenated, cropped to half of their duration centered in their middle. We obtain a posteriogram $P \in [0,1]^{|C| \times T}$, C being the set of symbols supported by our acoustic model, either characters of phonemes, and T the number of temporal frames of the song. This matrix provides an estimation of the posterior probabilities of each symbol through time. Alignment annotations are then predicted, using the generated posteriogram P and lyrics y, with a CTC-based alignment function inspired from the CTC-based decoding function presented in [25] and is akin to a Viterbi forced alignment [26]. Viterbi forced alignment is a simpler version of Viterbi decoding where the possible paths are limited to the lyrics symbol sequence. To allow the use of ϵ during decoding, y is extended to z by adding a ϵ at the beginning, end, and between every unit. A decoding network of size $|z| \times T$ is then constructed from z. The goal of the decoding function is to find the path in the network that give the most probable alignment $\hat{y}$ of y given the posteriogram P. More precisely:

$$\hat{y} = \underset{B(\hat{y})=y}{\arg\max} \prod_{j=1}^{T} P(\hat{y}_j, j) \qquad (1)$$

where B is an operator that removes blanks and repetitions from a sequence $\hat{y}$. To do that, network's node $\alpha_{s,j}$ is defined as the probability of the best alignment of the subsequence $z_{1:s}$ after j frames. $\alpha_{s,j}$ scores can be calculated efficiently using a forward-backward algorithm, by merging together paths that reach the same node. $\alpha_{s,j}$ is then computed recursively from the values of α in the previous frame. Only transitions between blank and non-blank characters, and between pairs of distinct non-blank characters are allowed. ϵ at the beginning and end of the sequence being optional, there are two valid starting nodes and two final nodes. The coefficients α are initialized such as:

$$\alpha_{s,1} = P(z_s, 1) \text{ for } s \in \{1,2\} \text{ and } \alpha_{s,1} = 0, \forall s > 2 \quad (2)$$

Recursion is given by:

$$\alpha_{s,j} = \max_{\tau \in \{0,1\}} (\alpha_{s-\tau,j-1}) P(z_s, j), \text{if } z_s \in \{\epsilon, z_{s-2}\}$$

$$\zeta_{s,j} = \underset{\tau \in \{0,1\}}{\arg\max}(\alpha_{s-\tau,j-1})$$

$$\alpha_{s,j} = \max_{\tau \in \{0,1,2\}} (\alpha_{s-\tau,j-1}) P(z_s, j), \quad \text{otherwise}$$

$$\zeta_{s,j} = \underset{\tau \in \{0,1,2\}}{\arg\max}(\alpha_{s-\tau,j-1})$$

$$(3)$$

Then, the probability of the best alignment is given by:

$$P(\hat{y}) = \max_{\tau \in \{0,1\}} (\alpha_{|z|-\tau,T}) \qquad (4)$$

Alignment $\hat{y}$ can finally be computed with an inverse recursion. The initial unit is initialized such as:

$$\hat{y}_T = |z| - \arg\max_{\tau \in \{0,1\}} (\alpha_{|z|-\tau,T}) \qquad (5)$$

Inverse recursion is given by:

$$\hat{y}_{j-1} = \hat{y}_j - \zeta_{\hat{y}_j,j} \qquad (6)$$

Calculations are performed in log-space using the log-sum-exp trick [27] to avoid numerical instabilities. As some phonemes from target languages can be unseen in the training languages, the acoustic model will be unable to predict them, resulting in all alignment having a probability of zero. To get rid of this problem, a small amount of uniformly distributed noise is added to all entries of the posteriogram, as suggested in [8].

4. EXPERIMENTAL SETUP

4.1 Dataset

For this study, we consider several language subsets of the DALI dataset. They are described in Table 1. Experiments are conducted using 5 source languages for the initial multilingual system development. These source languages are: English, German, French, Spanish and Italian. English is considered as a high-resource language. The 4 others languages are considered as low-resource languages in this study. The split between train, validation and test datasets for the first five languages is an artist aware split [28]. We also consider 4 additional target zero-resource languages: Portuguese, Polish, Finnish and Dutch. Data from these languages are only used for evaluation. The split of the different language data, i.e. dali ids belonging to each dataset, are made publicly available at https://github.com/deezer/MultilingualLyricsToAudioAlignment. One dataset, that we named *5lang*, is created for multilingual training. The training and validation sets of this dataset are generated by simply concatenating respectively the training and validation sets of the 5 source languages. This dataset is largely unbalanced, English data dominating the corpus. Balancing the dataset with oversampling was tested without modification on performances of the multilingual model on low-resource and zero-resource languages. Similar results were also found for speech [29]. Worse, it significantly degrades results for the English language. These results were expected as the quantity of English data being far superior in comparison to other languages in the multilingual dataset, diminishing their importance could only degrade results for the multilingual model when tested on English dataset. Results of multilingual models trained with balanced dataset are displayed in supplementary materials.

Language	# Phonemes	Train (h)	Test (h)
English (en)	44 (5)	192.7	31.5
German (de)	44 (1)	17.4	2.3
French (fr)	42 (0)	8.9	0.9
Spanish (es)	35 (3)	8.4	1.1
Italian (it)	33 (0)	8.5	1.2
Portuguese (pt)	37 (0)	X	1.8
Polish (pl)	31 (2)	X	4.2
Finnish (fi)	25 (0)	X	3.1
Dutch (nl)	41 (2)	X	3.1

Table 1. Description of DALI language subset datasets and corresponding phoneme dictionary sizes. In parenthesis are displayed the number of phonemes only occurring in the given language and its equivalent ISO 639.1 code

The procedure to generate training samples and corresponding labels for the acoustic model is similar to the one described in [25]. To recall, Spleeter [18] is used to isolate vocals from each song. Training samples are then computed by segmenting extracted vocals. The character sequence associated with a segment is created from word level annotations of DALI by concatenating all words whose start position is within the segment. An instrumental token is generated if no words are present in the segment. For phoneme models, the phoneme sequence associated with a segment is generated from his corresponding character sequence using Phonemizer[2]. Phonemizer includes GPT tools for most common languages. It decomposes each word into a sequence of IPA symbol. To create the phoneme dictionary of one given language, we collect all IPA phonemes present in the corresponding dataset. For simplicity, we did not consider IPA symbols others than vowels and consonants. Sizes of dictionaries of phoneme of each language are given in Table 1. After concatenating and merging all dictionaries, we obtain a universal phoneme set of 62 phonemes. The language sharing factor [24] for the nine languages we used in this study is 5.35. It means that, on average, one unit of the universal phoneme set is shared by 5 to 6 languages of the language pool which supports the fact that IPA phonemes are rather consistent across languages that we consider in this study.

4.2 Parameters of acoustic models

We use the same architecture for all acoustic models. Several sets of regularisation and architecture's size parameters were tested without a clear impact on performances. Parameters of architecture are similar to those used in [25]. The model has 3 layers of bidirectional LSTM and a dense layer. It takes as input mel-scale log filterbanks coefficients and energy plus deltas and double-deltas. The acoustic model output is the probabilities of characters or IPA phonemes. In the first case, the set of outputs is the concatenation of the Latin alphabet, the apostrophe, the instrumental token, the space token and the CTC blank symbol

---

[2] https://github.com/bootphon/phonemizer

ϵ. A set of size 30 is obtained. In the second case, it is constituted of the universal phoneme set, plus the instrumental token, the space token and the CTC blank symbol ϵ. A set of size 65 is obtained. Parameters of training are the same as those used in [25].

4.3 Evaluation

To evaluate our system, we use the *Average Absolute Error* (AAE) [13]. For its calculation, the absolute difference between the actual start of the word timestamp and its estimation for each word is calculated. The final error score for a song is obtained by averaging over all word-level errors. A known issue of this metric is its perceptive dependence on tempo. In fact, one absolute error will not be perceived the same if the tempo is fast or slow. The *Percentage of correct onsets* (PCO) [14] was proposed to mitigate this effect. It is computed as the percentage of start of the word timestamps whose estimation are below a certain distance from the ground truth. This metric considers that errors bellow a certain threshold fall within human listeners perceptive tolerance. We use 0.3 seconds as the tolerance window. Both metrics are classic metrics of MIREX lyrics-to-audio alignment challenge. They are computed using the same evaluation script as the one used for the challenge [30] [3].

5. RESULTS AND DISCUSSION

5.1 State of the art comparison

To validate our implementation, We first compare our system with two state-of-the-art ones. Results are collected from the 2019 MIREX lyrics-to-audio alignment challenge. For this comparison, we use characters as intermediate representation space and only English for training. We use three standard evaluation datasets for lyrics-to-audio task. Hansen [31] and Mauch [14] are constituted of respectively 9 and 20 English pop music songs. Jamendo [8] is made of 20 English music songs of several western genres. All three datasets are annotated with start-of-word timestamps. Results are summarized in Table 2.

Our system performances are close to those of GYL1, with no significant differences for PCO metric on the three evaluation datasets. Although we use an architecture somewhat similar to SDE2 (i.e. a CTC based approach with a pre-step of singing voice separation), we report significantly better performances. It is worth noting that GYL1 and our system both use the English part of DALI as training dataset, while SDE2 uses a private dataset of unknown quality. We can postulate that the DALI dataset annotation quality is higher, which would explain the better performances reached by our implementation despite using a much smaller train set than SDE2.

[3] https://github.com/georgid/AlignmentEvaluation

Dataset	System	Mean AAE (s)	Mean PCO (%)
Hansen	SDE2 [8]	0.39 (0.12)	88 (3)
	GYL1 [9]	**0.10 (0.03)**	**97 (1)**
	Ours	0.18 (0.05)	95 (2)
Mauch	SDE2 [8]	0.26 (0.04)	87 (2)
	GYL1 [9]	**0.19 (0.03)**	**91 (2)**
	Ours	0.22 (0.03)	**91 (1)**
Jamendo	SDE2 [8]	0.38 (0.11)	87 (3)
	GYL1 [9]	**0.22 (0.06)**	**94 (2)**
	Ours	0.37 (0.05)	92 (2)

Table 2. Comparison between our character based architecture trained with the English part of DALI and state-of-the-art systems on standard lyrics-to-audio alignment evaluation datasets. Mean AAE is better if smaller, mean PCO is better if larger. Standard errors over tested songs are given in parenthesis
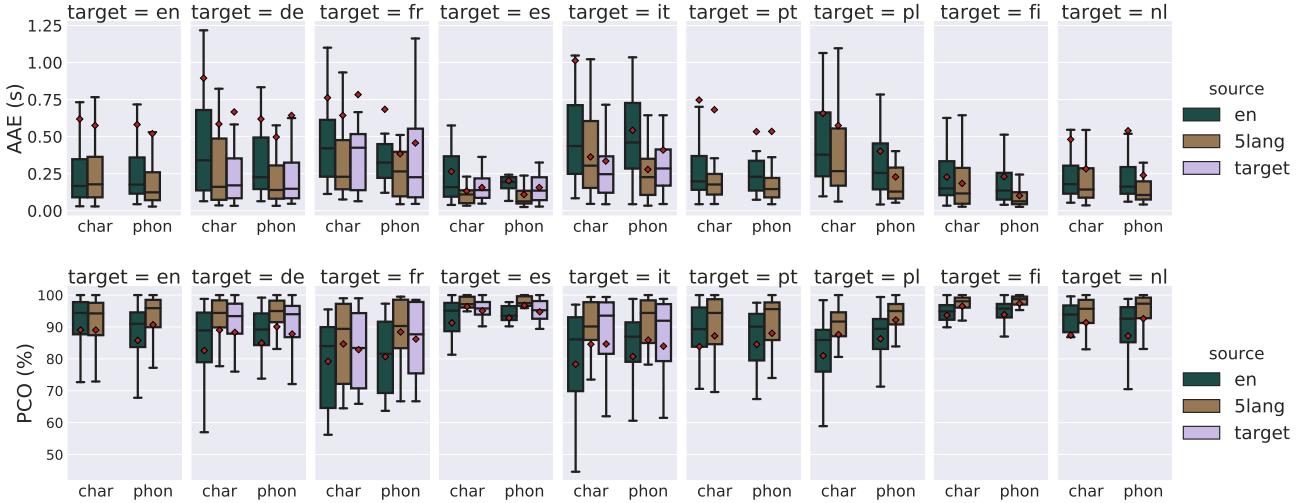
5.2 Multilingual generalization

Results of multilingual generalization experiments are displayed in Figure 2. Precise numerical values are reported in supplementary materials. Several conclusions can be drawn:

- Using a multilingual training set helps For both character and phoneme based architectures, the model exhibiting the best multilingual generalization is trained with multilingual dataset. In fact, this model significantly outperforms the ones trained on English on low-resource and zero-resource languages without degrading performances on English. With phoneme as intermediate representation, it even improves results on English. On low-resource languages, multilingual trained model obtains results on par with models trained only on the target language (e.g. French trained model on French dataset). It is worth noticing that the multilingual training dataset is only marginally larger than the English one. Performances differences are to be attributed to the additional information the model was able to extract from the diversity of languages seen during training.

- Use phonemes over characters as an intermediate representation has better performances Performances of phoneme based architectures are almost always better than those of their character based counterparts in all our experimental setups. The gap is bigger for models trained on the multilingual dataset than for those trained on monolingual ones. The only models that are not improved are the ones trained and tested on the same languages. Such results show that the use of phoneme as an intermediate representation enables transfer knowledge between language better than character representation.

- Training on multilingual data and a phoneme internal representation yields the best results in all considered cases Training the acoustic model on multilingual data and use a universal phoneme set is a relevant way for improving the generalization capacity of the considered lyrics-to-audio alignment architecture even to zero-

Figure 2. Lyrics-to-audio evaluation on DALI language subset datasets for phoneme and character based architectures. Several training set design strategies are considered. Languages are given by their ISO 639.1 code. Here "source" refers to data language used to train the given model and "target" refers to data language used to evaluate the trained model. When source is equal to target, architectures are trained and tested on the same language. Mean AAE is better if smaller, mean PCO is better if larger. Mean values are displayed using squares

resource scenarios.

6. CONCLUSION

In this paper, we investigated extending state-of-the-art methods in the multilingual context. Focusing on one architecture that seemed fit for generalization, we demonstrated that design choices regarding the training dataset and the acoustic representation space are salient factors. We have shown that using many languages to train the acoustic model and a universal phoneme set improves the multilingual generalization of such architecture. In this work, we have built a dataset using the language distribution found in DALI, which resulted in a largely unbalanced dataset. For comparison, we also conducted experiments with a balanced dataset, in which all 5 languages were equally present. The performance was similar, except for English, when it was significantly degraded. This raises the issue of how to design training sets in a setting where several high-resource languages are available. Although there are no publicly available datasets exhibiting such characteristics, future work should investigate this case. Existing works on multilingual speech processing [11] point towards increasing model complexity to circumvent this. Also, only a small set of languages were considered in this study. Additional experiments on a wider, more diverse set of songs remain to be conducted. Finally, future works should consider the specific case of songs with multilingual lyrics. This problem, known as code-switching, has been studied for speech [21] but never for music. Such a phenomenon is however not uncommon in popular music [10], thus it should be addressed too.

7. REFERENCES

[1] A. M. Kruspe, "Application of Automatic Speech Recognition Technologies to Singing Doctoral Thesis," Ph.D. dissertation, University Fraunhofer, apr 2018. [Online]. Available: http://ieeexplore.ieee.org/document/7178348/

[2] A. Mesaros, "Singing Voice Recognition for Music Information Retrieval," Ph.D. dissertation, Tampere university of technology, 2012. [Online]. Available: https://dspace.cc.tut.fi/dpub/handle/123456789/21404

[3] H. Fujihara, M. Goto, J. Ogata, and H. G. Okuno, "Lyric synchronizer: Automatic synchronization system between musical audio signals and lyrics," *IEEE Journal on Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1252–1261, 2011.

[4] A. Kruspe, "Automatic B**** Detection," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, no. September, 2016, pp. 3–4.

[5] C. W. Wightman and D. T. Talkin, "The aligner: Text-to-speech alignment using markov models," in *Progress in speech synthesis*. Springer, 1997, pp. 313–323.

[6] A. Haubold and J. R. Kender, "Alignment of speech to highly imperfect text transcriptions," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2007, pp. 224–227.

[7] B. Sharma, C. Gupta, H. Li, and Y. Wang, "Automatic lyrics-to-audio alignment on polyphonic music using singing-adapted acoustic models," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 396–400.

[8] D. Stoller, S. Durand, and S. Ewert, "End-to-end Lyrics Alignment for Polyphonic Music Using an Audio-to-Character Recognition Model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019. [Online]. Available: http://arxiv.org/abs/1902.06797

[9] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics transcription in polyphonic music: Does background music help?" *arXiv preprint arXiv:1909.10200*, 2019.

[10] E. E. Davies and A. Bentahila, "Translation and code switching in the lyrics of bilingual popular songs," *The Translator*, vol. 14, no. 2, pp. 247–272, 2008.

[11] S. Watanabe, T. Hori, and J. Hershey, "Language independent end-to-end architecture for joint language and speech recognition," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2017.

[12] J. Cho, M. K. Baskar, R. Li, M. Wiesner, S. H. Mallidi, N. Yalta, M. Karafiat, S. Watanabe, and T. Hori, "Multilingual Sequence-to-Sequence Speech Recognition: Architecture, Transfer Learning, and Language Modeling," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, no. 1, 2019, pp. 521–527.

[13] A. Mesaros and T. Virtanen, "Automatic alignment of music audio and lyrics," in *Proc. Int. Conference on Digital Audio Effects (DAFx)*, 2008, pp. 1–4. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.212.6683{&}rep=rep1{&}type=pdf

[14] M. Mauch, H. Fujihara, and M. Goto, "Integrating Additional Chord Information Into HMM-Based Lyrics-to-Audio Alignment," in *Proc. IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, jan 2012, pp. 200–210. [Online]. Available: http://ieeexplore.ieee.org/document/5876304/

[15] G. Dzhambazov and X. Serra, "Modeling of phoneme durations for alignment between polyphonic audio and lyrics," in *Proc. of the 12th International Conference in Sound and Music Computing (SMC)*, 2015, pp. 281–286.

[16] G. Dzhambazov and A. Srinivasamurthy, "On the Use of Note Onsets for Improved Lyrics-To-Audio Alignment in Turkish Makam Music," in *Proc. 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 716–722.

[17] G. Meseguer-brocal and A. Cohen-hadria, "Dali : a Large Dataset of Synchronized Audio , Lyrics and Notes , Automatically Created Using Teacher-Student Machine Learning Paradigm," in *Proc. International Society on Music Information Retrieval Conference (ISMIR)*, 2018.

[18] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and state-of the art music source separation tool with pre-trained models," in *Proc. Late-Breaking/Demo of International Society of Music Information Retrieval Conference (ISMIR)*, November 2019, deezer Research.

[19] S. Toshniwal, T. N. Sainath, R. J. Weiss, B. Li, P. Moreno, E. Weinstein, and K. Rao, "Multilingual speech recognition with a single end-to-end model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4904–4908.

[20] M. Müller, S. Stüker, and A. Waibel, "Language adaptive multilingual ctc speech recognition," in *Proc. International Conference on Speech and Computer (SPECOM)*. Springer, 2017, pp. 473–482.

[21] K. Li, J. Li, G. Ye, R. Zhao, and Y. Gong, "Towards code-switching asr for end-to-end ctc models," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6076–6080.

[22] H. Sak, F. de Chaumont Quitry, T. Sainath, K. Rao *et al.*, "Acoustic modelling with cd-ctc-smbr lstm rnns," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 604–609.

[23] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ACM International Conference Proceeding Series*, vol. 148, 2006, pp. 369–376.

[24] T. Schultz and A. Waibel, "Language-independent and language-adaptive acoustic modeling for speech recognition," *Speech Communication*, vol. 35, no. 1-2, pp. 31–51, 2001.

[25] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. d'Alché-Buc, "Audio-based detection of explicit content in music," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 526–530.

[26] D. G. Forney, "The viterbi algorithm," *Proc. of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[27] A. Hannun, "Sequence modeling with ctc," *Distill*, vol. 2, no. 11, p. e8, 2017.

[28] A. Flexer, "A closer look on artist filters for musical genre classification," in *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR)*, no. 122, 2007, pp. 16–17.

[29] T. Alumäe, S. Tsakalidis, and R. Schwartz, "Improved multilingual training of stacked neural network acoustic models for low resource languages," in *Proc. Interspeech*, 09 2016, pp. 3883–3887.

[30] G. Dzhambazov, "Knowledge-Based Probabilistic Modeling For Tracking Lyrics In Music Audio Signals," Ph.D. dissertation, Universitat Pompeu Fabra Barcelona, 2017. [Online]. Available: http: //www.tdx.cat/bitstream/handle/10803/404681/tgd. pdf?sequence=1{%}0Ahttp://mtg.upf.edu/node/3751

[31] J. K. Hansen, "Recognition of Phonemes in A-cappella Recordings using Temporal Patterns and Mel Frequency Cepstral Coefficients," in *Proc. 9th Sound and Music Computing Conference (SMC)*, 2012, pp. 494–499.

VOICE-LEADING SCHEMA RECOGNITION USING RHYTHM AND PITCH FEATURES

Christoph Finkensiep[1]   **Ken Déguernel**[1]   **Markus Neuwirth**[2,1]   **Martin Rohrmeier**[1]

[1] École Polytechnique Fédérale de Lausanne   [2] Anton Bruckner University Linz

christoph.finkensiep@epfl.ch, ken.deguernel@epfl.ch,
markus.neuwirth@epfl.ch, martin.rohrmeier@epfl.ch

ABSTRACT

Musical schemata constitute important structural building blocks used across historical styles and periods. They consist of two or more melodic lines that are combined to form specific successions of intervals. This paper tackles the problem of recognizing voice-leading schemata in polyphonic music. Since schema types and subtypes can be realized in a wide variety of ways on the musical surface, finding schemata in an automated fashion is a challenging task. To perform schema inference we employ a skipgram model that computes schema candidates, which are then classified using a binary classifier on musical features related to pitch and rhythm. This model is evaluated on a novel dataset of schema annotations in Mozart's piano sonatas produced by expert annotators, which is published alongside this paper. The features are chosen to encode music-theoretically predicted properties of schema instances. We assess the relevance of each feature for the classification task, thus contributing to the theoretical understanding of complex musical objects.
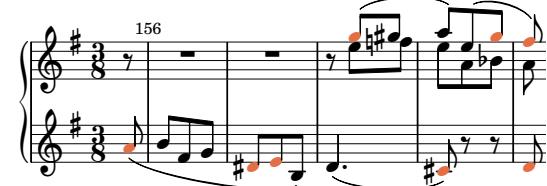
1. INTRODUCTION

Voice-leading schemata are frequently used patterns that can be found across historical periods, ranging from Renaissance, Baroque, and Classical to modern tonal music; examples include such well-known schemata as the Lamento, the Pachelbel, the descending-fifths sequence, and cadences [8, 10, 4, 11]. A schema serves as a template for contrapuntal structure that can be elaborated in multiple ways.

At present, there is only scant quantitative evidence about the frequency and diachronic distribution of schemata across history (e.g., [10, 2]); large-scale, machine-readable datasets on schemata are not yet available. For assessing the prevalence of schemata in a corpus of music, automated recognition of schema instances

(a) A (true) Fonte at the beginning of K281-iii.

(b) A possible candidate for a Fonte in K283-iii.

Figure 1: An example of a Fonte (a) with structural notes highlighted. The task is to decide whether proposed instances such as (b) are true instances or not.

can be a time- and cost-efficient alternative to manually labelled data. However, there are two key challenges for computational approaches when seeking to uncover note patterns in music: (1) the multidimensional (polyphonic) structure of music as opposed to, for example, the sequential structure of written text [17]; (2) the highly flexible nature of these patterns, given that the structural notes in the individual voices can be elaborated in a wide variety of ways.

Voice-leading schemata can be defined as configurations of two or more voices that move together through a sequence of stages, forming particular patterns of successive vertical intervals that occur within a specific tonal context. Consider the example of the Fonte (e.g., [10]): The Fonte is a four-stage pattern involving at least two voices. The bass moves through the scale degrees $\sharp\hat{1} \rightarrow \hat{2} \rightarrow \hat{7} \rightarrow \hat{1}$ of a major scale, while the soprano follows the pattern $\hat{5} \rightarrow \hat{4} \rightarrow \hat{4} \rightarrow \hat{3}$, thus producing the following sequence of vertical intervals: tritone $\rightarrow$ minor third $\rightarrow$ tritone $\rightarrow$ major third. The schema prototype can be elaborated in actual compositions in many different ways. For instance, the notes belonging to one stage can be displaced in time; any number of elaboration notes can be inserted between the structural notes of one stage and between stages... An example illustrating the surface realization of a Fonte is

shown in Figure 1a. While containing the correct interval pattern is a central property of any schema instance, it is not sufficient: the selected notes must also provide the contrapuntal template for its context, so that the notes contained in the time-span covered by the schema candidate can be meaningfully interpreted as ornamentations of the selected notes. Figure 1b shows a candidate for a Fonte instance. The task at hand is to decide whether or not such a candidate is a true schema instance.

To tackle the problem of schema detection, this paper provides two contributions. First, we propose a novel dataset with hand-annotated schemata found in Mozart's piano sonatas (Section 3). Second, we present a binary classifier that recognizes true schema instances among a set of proposed schema candidates based on rhythm and pitch features related to regularity, complexity, salience, and harmonic context (Section 4). We evaluate the impact of these features on the classification task using a logistic regression (Section 5).

2. RELATED WORK

Automated discovery and recognition of musical patterns is a topic of ongoing interest in the MIR community [15, 5, 17, 3, 9, 12, 16]. Voice-leading schemata as a specific class of patterns have so far received only little attention; they have been approached with computational methods only very recently. For instance, Symons [23] has developed an algorithm that recognizes schemata in a small corpus, pointing out the importance of rhythmic regularity. Finkensiep et al. [7] tackle the problem of temporal displacement and free polyphonic textures using a two-dimensional extension of skipgrams, which have previously been proposed by Sears et al. [21, 22]. Recently, Katsiavalos et al. [13] have presented a system that uses heuristics-based time-span reduction to discover and recognize schemata.

Several studies aimed at finding cadences, which can be viewed as a subcategory of voice-leading schemata, and evaluated the features relevant for the classification task. Bigo et al. [1] evaluated a set of 44 features linked with the moment of cadential arrival, which are integrated using a support-vector machine for classifying beats as belonging to a cadence or not. Sears et al. [21] use skipgrams on vertical slices to find cadences using a figured bass-like representation of the notes in each slice. Duane [6] approaches cadences directly as voice-leading patterns by trying to recognize and learn them from melodic motion.

3. DATASET

Our dataset is based on the full set of Mozart's piano sonatas encoded in MusicXML format. These 18 sonatas with 3 movements each (thus 54 movements in total) have been composed between 1774 and 1789 and constitute a prominent sample of the classical style. The pieces in the dataset contain 103,829 notes in total distributed over 7,500 measures, with 244 hand-annotated true instances (0.13%) and 190,994 automatically generated false

Schema	Variant	Occurrences
Do-Re-Mi	.2	5
Fenaroli	.2(.min)	10 (3)
	.2.flipped(.min)	43 (8)
	.2.melcanon(.min)	6 (2)
	.2.basscanon.min	1
Fonte	.2	49
	.2.flipped	2
	.2.majmaj	8
Indugio	.2	9
	.2.voiceex	5
Lamento	.2	2
Lully	.2	2
Morte	.2	1
Prinner	.2	32
Quiescenza	.2	46
	.2.diatonic	6
Sol-Fa-Mi	.2	4

Table 1: List of schemata with their variants and number of occurrences in the Mozart's Piano Sonata dataset.

instances (99.87%) for the selected schema types and subtypes (see Table 1).

3.1 Schema Formalization and Lexicon

For the present study, we selected 10 schema types and 20 subtypes (listed in Table 1) which have been suggested in the literature [10, 4, 20, 19]. The approach presented here assumes that a schema type consists of (1) a fixed number of voices; (2) a fixed number of stages, whereby each stage contains one note per voice; and (3) a characteristic interval pattern between these notes. The prototype for each schema variant (or subtype) is specified using a formal notation. For instance, the prototype of the two-voice Fonte is encoded as:

```
"fonte.2": [["a1", "P5"],
            ["M2", "P4"],
            ["M7", "P4"],
            ["P1", "M3"]]
```

where ".2" indicates the two-voice variant of the Fonte. Each note is given as an interval to some arbitrary reference point. Since all possible transpositions of the schema are considered by the matcher, it is not necessary to know the reference key.

Schema instances are encoded as nested arrays of notes in the same form as the corresponding prototypes. Instances may deviate from the shape of the prototype if (a) a note that would repeat its predecessor (e.g. the second $\hat{4}$ in the Fonte) is held over or missing, or (b) two adjacent voices merge and are represented by a single note on the surface.

3.2 Data Production

The dataset consists of two parts: manual annotations by experts and automatically retrieved candidates for schema instances, i.e. sets of notes with an interval pattern conforming to a schema variant. Both the annotations and the computed candidates share the same encoding format, namely a nested lists of note IDs (one sublist per stage, one note per voice) that corresponds to note elements in a MusicXML representation of the scores. While the manual annotations provide the true instances of the dataset, the false instances consist of all skipgram candidates that do not appear in the annotations. [1] The complete dataset is available on github. [2]

3.2.1 Expert Annotations

Two annotators (the third author and Adrian Nagel) provided their analyses by using a web-based annotation app that was specifically developed for the annotation process. The app displays a score using the Verovio toolkit [18], and allows the user to select individual notes from the musical score to mark schema instances. Instances are automatically checked for conformance with the schema prototype in the lexicon, while permitting the deviations described in Section 3.1. The annotators also considered additional criteria such as harmonic signature, phrase structure, pattern repetition, and form-functional context. [3]

3.2.2 Computing Candidates with Skipgrams

In order to compute all candidates of schemata for the classifier, we base our work on the generalized skipgram model proposed in [7], which enumerates two-dimensional note configurations that occur within certain temporal bounds. We use this algorithm to find configurations with a maximal note displacement of 1 bar per stage and a maximal distance of 1 bar between the onsets of two adjacent stages. The configurations are filtered for a specific interval pattern during enumeration regardless of the local keys. This method provides us with all candidates for a schema instance within a reasonable window. However, due to the exhaustive search and a high number of possible note combinations, our resulting dataset is extremely unbalanced. Because of the high combinatoric complexity, we restrict this study to two-voiced schema variants. Furthermore, we reduced the number of candidates to at most 25 per group of temporally overlapping candidates using a previous version of the model presented here.

4. FEATURES AND SCHEMA CLASSIFICATION

4.1 Musical Features

By using precomputed schema candidates that are known to have the correct interval structure (which is all infor-

mation that we consider for a specific schema type), the problem is narrowed down to deciding whether or not the candidate consists of the structurally important notes. To this end, we have defined a set of features with regard to rhythmic, pitch, and metric information, inspired in part by previous work [10] and that we wish to evaluate with the classifier. These features attempt to measure the *recognizability* of the candidate as a structural pattern, assessing, for example, its complexity, salience, or regularity in various musical dimensions. For a schema candidate C that consists of a number of stages n_s and a number of voices n_v, let $C_{s,v}$ denote the note from stage s and voice v. Each note is represented by an onset, an offset, and a pitch. Whenever pairs of notes are compared, K denotes the numbers of compared note pairs (excluding pairs with missing notes).

The first feature can be considered a rough estimate of the *harmonic or modal signature* of the schema candidate. We define the *harmonic profile* of a candidate as the distribution of pitch-classes (relative to the transposition of the match) of notes that overlap with the time span of the candidate (excluding the matched notes themselves). The `profiledist` is defined as the euclidean distance between a match's pitch profile and the average pitch profile of all true instances of the same schema. Thus, this feature uses training data to derive the prototype profiles instead of defining a harmonic signature a priori.

Three features address the *regularity* of pitch and rhythm between pairs of stages. `rreg` measures the average rhythmic dissimilarity between each pair of stages. For a pair of stages, the rhythmic dissimilarity is defined as the sum of the temporal distance of the notes of the same voice, given the best alignment possible when projecting one stage unto the other. `mreg` is defined very similarly, but here the alignment offset is fixed to whole beats to preserve metric position. Finally, `preg` measures the average pitch dissimilarity between each pair of stages. Similar to rhythmic dissimilarity of a pair of stages, pitch dissimilarity is defined as the sum of the pitch distances of the notes of the same voice, given the best pitch alignment possible when projecting one stage unto the other. These features are defined as

$$\star\texttt{reg} = \frac{1}{K} \sum_{\substack{(s,s') \in stage \\ s \neq s'}} \min_{\delta} \left(\sum_{v=1}^{n_v} |\mu(C_{s,v}) - \mu(C_{s',v}) - \delta| \right),$$

(1)

where μ corresponds to onset for `rreg` and `mreg`, and to pitch for `preg`. For `mreg`, δ is restricted to integer multiples of a beat.

We then define features corresponding to the *complexity* of the candidates in terms of displacement between pairs of notes. `rdsums` and `pdsums` respectively correspond to the average temporal and pitch distance between each note of the same stage. They are defined as

$$\star\texttt{dsums} = \frac{1}{K} \sum_{s=1}^{n_s} \sum_{\substack{(v,v') \in voice \\ v \neq v'}} |\mu(C_{s,v}) - \mu(C_{s,v'})|, \quad (2)$$

---

[1] This includes alternative versions of true instances with several possible note selections.

[2] https://github.com/DCMLab/schema_annotation_data

[3] As detailed in the schema-annotation guidelines (https://github.com/DCMLab/schema_annotation_data/blob/master/manual/manual.pdf).

where μ corresponds to onset for `rdsums` and to pitch for `pdsums`. Similarly, `rdsumv` and `pdsumv` respectively correspond to the average rhythmic and pitch distance between each note of the same voice. They are defined as

$$\star\texttt{dsumv} = \frac{1}{K}\sum_{v=1}^{n_v}\sum_{\substack{(s,s')\in stage \\ s\neq s'}}|\mu(C_{s,v})-\mu(C_{s',v})|, \quad (3)$$

where μ correspond respectively to onset and pitch for `rdsumv` and `pdsumv`.

Another perspective at pitch displacement is provided by `vdist`, which measures the average amount of octave jumps within a voice from one stage to the next, and is defined as

$$\texttt{vdist} = \frac{1}{K}\sum_{v=1}^{n_v}\sum_{s=1}^{n_s-1}\left\lfloor\frac{\text{pitch}(C_{s+1,v})-\text{pitch}(C_{s,v})}{\text{octave}}\right\rfloor. \quad (4)$$

While a certain complexity may be necessary to make a regular pattern recognizable in the first place, a more complex pattern can be more difficult to detect in the presence of other notes. For this reason, `onsets` counts the average number of distinct note onsets in the context of each stage. A low number of onsets allows the stages to be rhythmically displaced while still being recognizable as a unit. Given the number of distinct note onsets D_s for each state s, we have

$$\texttt{onsets} = \frac{1}{n_s}\sum_{s=1}^{n_s}D_s. \quad (5)$$

Finally, we define two features representing the *salience* of the candidate. First, we consider `dur`, which corresponds to the sum of all note durations in the candidate,

$$\texttt{dur} = \sum_{s,v}\text{offset}(C_{s,v})-\text{onset}(C_{s,v}). \quad (6)$$

Then we consider `mweight`, a feature based on metric weight. We define our metric weight function as follows:

$$\text{mw}(C_{s,v}) = \begin{cases} 2 & \text{if } \text{onset}(C_{s,v}) \text{ is on a strong beat.} \\ 1 & \text{if } \text{onset}(C_{s,v}) \text{ is on a weak beat.} \\ \frac{1}{2^p} & \text{if } \text{onset}(C_{s,v}) \text{ is on a subbeat,} \end{cases}$$

where p is the number of prime factors needed to express the subbeat. Given that function, `mweight` corresponds to the average metric weight of each note of the candidate:

$$\texttt{mweight} = \frac{1}{K}\sum_{s=1}^{n_s}\sum_{v=1}^{n_v}\text{mw}(C_{s,v}). \quad (7)$$

4.2 Classification and Evaluation Method

The features described above are used as an input to a logistic regression model, a simple binary classifier model that uses a linear combination of the input features and applies a sigmoid to that score, yielding a value between 0 and 1 that indicates the probability of the input to be a true instance. Since a logistic regression is a special case of a neural network without hidden layers, this approach can be naturally extended to include more layers, allowing for more complex, non-linear feature combinations. However, preliminary experiments have shown that non-linear models (such as simple neural networks and support-vector machines) do not increase model performance and instead lead to overfitting, so we exclude them here.

The input data consists of expert annotations and skip-gram candidates, produced as described in Section 3.2. To get consistent temporal information about the notes, we unfold all repetitions and jumps notated in the scores. Repeated occurrences of notes are disambiguated by selecting for every schema candidate those note occurrences that have a consistent temporal order and cover a minimal time span. Finally, matches that have incomplete stages (due to implicit notes, as described above) are converted into complete instances with missing notes marked explicitly.

To evaluate the model's performance, we use 5-fold cross validation.[4] The pitch histograms used for `profiledist` are computed on the respective training set of each run. In order to get an unbiased model, we follow the advice given in [14] and balance our dataset by upsampling the true instances to match the number of false instances. The model is trained on the balanced training data using the Julia package `GLM.jl`[5] and applied to both balanced and unbalanced test data. In addition, a prior-corrected version of the model (see [14]) is applied to the unbalanced data.

The code for the whole evaluation pipeline is provided online[6], including a notebook[7] that was used to generate all results and figures in this paper.

5. RESULTS AND DISCUSSION

5.1 Classification Performance

The overall performance of the model is shown in Table 2, aggregating over the predictions on all test sets. When applied to data balanced by upsampling, the model achieves a high classification performance with an F-score of 0.894 and a Matthews correlation coefficient (MCC) of 0.787. Since the model is trained on balanced data, applying it to unbalanced data simply scales the number of true positives and false negatives, resulting in a drastically reduced precision. Using the prior correction of the unbalanced dataset results in a very high accuracy; however, it introduces a bias to label matches as non-instances, which results in the false negatives dominating the false positives.

Figure 2 shows how the predicted probability of being a true instance is distributed for instances and non-instances (upper-left corner). Non-instances overwhelm-

---

[4] A 5-fold split was chosen to balance the number of folds and size of the resulting test set.

[5] https://github.com/JuliaStats/GLM.jl
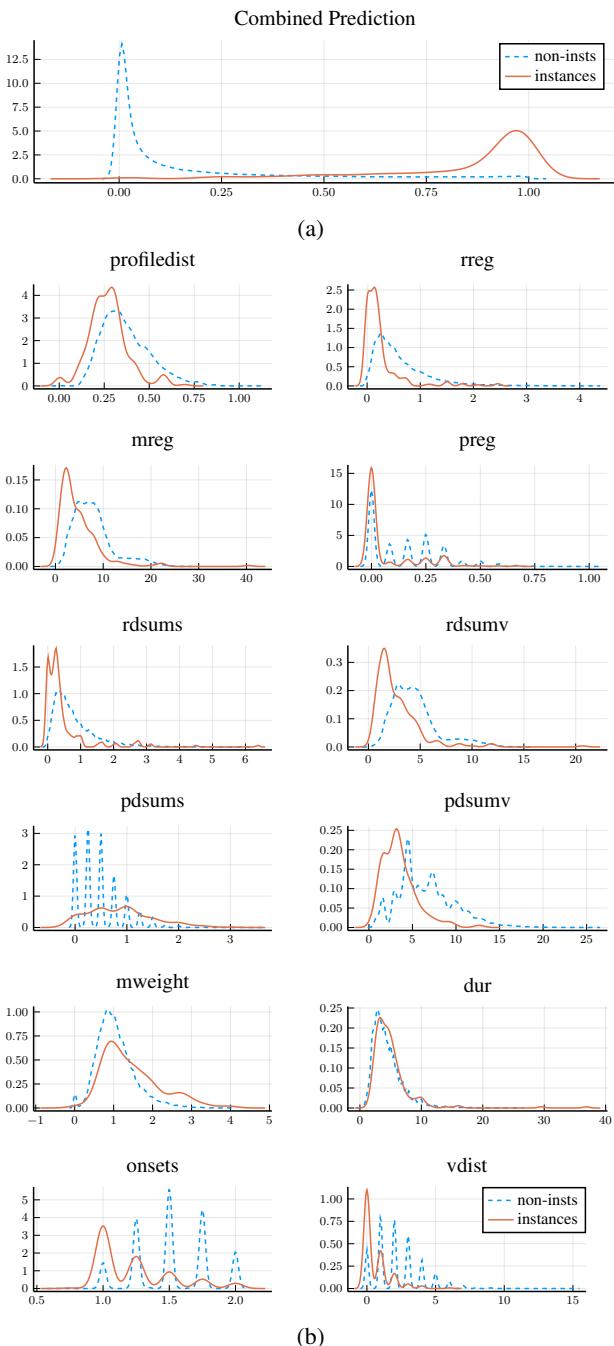
[6] https://github.com/DCMLab/schemata_code/tree/ismir2020

[7] https://github.com/DCMLab/schemata_code/blob/ismir2020/notebooks/ismir2020_classification.ipynb

Condition	TP	TN	FP	FN	accuracy	precision	recall	F-score	MCC
Balanced	171,400	169,867	21,107	19,574	0.893	0.890	0.898	0.894	0.787
Unbalanced	219	169,867	21,107	25	0.889	0.010	0.898	0.020	0.089
Unbalanced (corrected)	15	190,923	51	229	0.999	0.227	0.061	0.097	0.118
Grouped	220	6,009	2,663	12	0.700	0.076	0.948	0.141	0.218
Grouped (corrected)	43	8,596	76	189	0.970	0.361	0.185	0.245	0.245

Table 2: Performance of the model in different conditions. TP = true positives, TN = true negatives, FP = false positives, FN = false negatives, MCC = Matthews correlation coefficient.

Figure 2: Distribution of model prediction (a) and feature values (b) over instances and non-instances as a kernel density estimate. The more the curves tend in opposite directions, the better the two classes are separated.
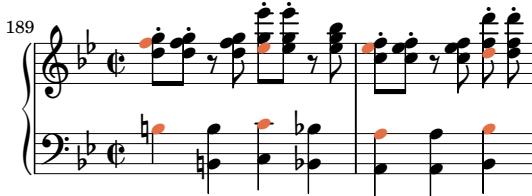
ingly receive low probabilities and instances are typically rated very high. This is in line with the model's good performance on raw data, but it also reveals why imbalance poses a serious problem: while the majority of non-instances are correctly discarded by the model, a minority remains indistinguishable from true instances under the model. When the skipgrams propose many more non-instances than instances, the small part of indistinguishable non-instances becomes huge in relation to the true instances. Note that simply reducing the number of matches does not necessarily improve the situation: taking away the matches with a rating < 0.5 still leaves us with the problematic cases.

A lot of non-instances are proposed as combinatoric variations around true instances. To test whether the problematic cases are variations of true instances or genuine non-instances, we group all matches according to temporal overlap (prior correction is based on the imbalance of the groups here). The results (Table 2) show that grouping drastically increases the performance compared to the ungrouped condition but does not get close to the performance on balanced data, indicating that there is still a significant number of indistinguishable true non-instances.
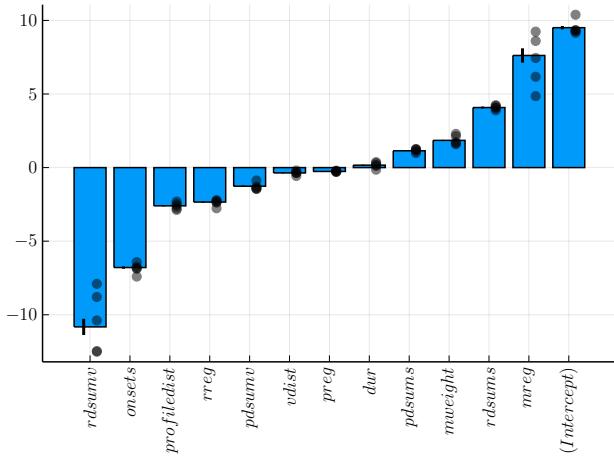
This effect of indistinguishability may be seen as an indicator that our list of features lacks those features that would help resolve the remaining cases and clearly separate the classes. However, it is not clear that finding such features is easily attainable. First, consider that while the existing features are already very informative, the information needed to distinguish the problematic cases would have to be much more precise. Even when the probability of getting a positive result for a non-instance is only 10^{-3}, a true instance proportion of 10^{-3} still leaves a 50% chance that a positive result is a false positive. Second, our annotators conformed to very strict standards in order to discard non-instances, restricting true instances to cases where the schema is a highly plausible template for the musical surface. Such judgments rely on implicit music-theoretical knowledge and intuition, which are difficult to model.

Finally, a look at some highly confident false positives suggests that if schema classification is defined as a binary task (a surface pattern is a schema instance or not), then the performance of this task can hardly been improved.

For example, the excerpt in Figure 1b may not look like a very plausible Fonte at first sight (and was not classified as such by the annotators). However, the last two bars clearly contain the correct contrapuntal pattern for

Figure 3: An ambiguous Fonte match (K333-iii). While intrinsically this is a highly plausible instance (interval pattern, tonal context, melodic parallelism), the context discards it, as the pattern is in fact part of a larger descending-fifths sequence.

Figure 4: The parameters $\vec{\beta}$ of the model trained on the full upsampled dataset (bars) and normalized by multiplication with the average value of the respective feature. Error bars indicate the 95% confidence interval of the fit. Black points indicate the normalized parameters for each model trained during cross validation.

the stages 3 and 4. The beginning can be interpreted as a melodic unfolding of an Em chord that is ornamented by the notes of a B^7 chord, most clearly in the neighbor note d♯ to e (i.e., the bass for the stages 1 and 2 of a Fonte). Therefore, it can be argued that this section shares its contrapuntal structure with the Fonte, even though the typical parallelism is missing. Another, converse, example can be seen in Figure 3: in isolation, the pattern is a clear instance of a Fonte, but it is continued in the manner of a larger descending-fifths sequence, which, depending on the definition used, may discard it as a Fonte. A negative definition like this is very difficult to check under the current paradigm.

5.2 Feature Evaluation

Figure 4 shows the influence of each feature in a model trained on the full balanced dataset. Overall, schemata seem to expose a high regularity and low complexity compared to non-instance candidates. The strong negative factors `rdsumv` and `onsets` disregard candidates with a large temporal extension and a high degree of non-simultaneity. Metric regularity (i.e. rhythmic regularity aligned to the metrical grid) has a strong positive influence,

indicating a preference for a regular temporal organization.

The preference for simultaneity of the notes in the same stage is somewhat contradicted by the moderately positive influence of the `rdsums`, the average note displacement within stages. This is particularly surprising when looking at the distribution of this feature over instances and non-instances (Figure 2b), which shows that instances generally show less displacement than non-instances. One possible explanation of this phenomenon is that the combination of both features (`onsets` and `rdsums`) expresses a general preferences for little displacement, but when the notes are non-simultaneous, then a higher distance is preferred, which may make the structural notes more recognizable.

Less important are features based on pitch (`profiledist`, `pdsum*`, `preg`, and `vdist`) as well as features that indicate basic salience (`dur` and `mweight`). Pitch features are likely of moderate to little importance because most of the relevant pitch-related information is already implied by the schema's interval structure. Interestingly, duration and metric weight (both properties that are taken from each note in isolation) play little to no role, which is confirmed in Figure 2b. This indicates that local properties do not mark notes as structural, this role seems to depend *only* on how the note is used in relation to other notes.

6. CONCLUSION

As the results presented above show, distinguishing between incidental and structural note configurations based on a small number of musically and cognitively motivated heuristics works well in the vast majority of cases. Even if a number of misclassifications remain, a closer look at these cases provides valuable insights into the problem at hand. First, the main limitation of our approach is that the model assesses suggested schema instances individually, without considering, or comparing it to, alternative interpretations. In many cases, the main reason for human experts to reject a candidate does not seem to be a lack of plausibility of the match itself, but rather the availability of a "better explanation", i.e. an alternative analysis of the match's context that identifies a more plausible contrapuntal scaffold. This result is in line with the reduction-based approach of Katsiavalos et al. [13]. Since the features used in this study already proved useful for independent classification, they likely benefit from a general structural-analysis approach, in which schema instances are recognized in reductions of the musical surface.

A second insight concerns the idea of schema itself and its relation to a classification task. From a cognitive perspective, a schema does not need to be instantiated unambiguously or even completely. It is sufficient if listeners recognize the schema as the template for the surface events, or if they understand the composer's intention to evoke the schema. In this regard, *discrete* binary classification into instances and non-instances may be as unattainable as it is undesirable, falling short of the complexity the relationship between schema and realization can exhibit.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] L. Bigo, L. Feisthauer, M. Giraud, and F. Levé. "Relevance of Musical Features for Cadence Detection". In: *Proceedings of the International Society for Music Information Retrieval Conference*. 2018.

[2] V. Byros. "Towards an "Archaeology" of hearing: schemata and eighteenth-century consciousness". In: *Musica Humana* 1.2 (2009), pp. 235–306.

[3] E. Cambouropoulos, T. Crawford, and C. Iliopoulos. "Pattern Processing in Melodic Sequences: Challenges, Caveats and Prospects". In: *Computers and the Humanities* 35 (2001), pp. 9–21.

[4] W. E. Caplin. "Topics and Formal Functions: The Case of the Lament". In: *The Oxford Handbook of Topic Theory* (2014). Ed. by D. Mirka, pp. 415–452.

[5] D. Conklin and M. Bergeron. "Discovery of Contrapuntal Patterns". In: *Proceedings of the International Society on Music Information Retrieval*. 2010, pp. 201–206. ISBN: 978-90-393-5381-3.

[6] B. Duane. "Melodic Patterns and Tonal Cadences: Bayesian Learning of Cadential Categories from Contrapuntal Information". In: *Journal of New Music Research* 48.3 (2019), pp. 197–216.

[7] C. Finkensiep, M. Neuwirth, and M. Rohrmeier. "Generalized Skipgrams for Pattern Discovery in Polyphonic Streams". In: *Proceedings of the International Symposium on Music Information Retrieval*. 2018.

[8] A. Forte. *Tonal Harmony in Concept and Practice*. 3rd ed. New York: Holt, Rinehart and Winston, 1979. ISBN: 978-0-03-020756-3.

[9] M. Giraud, K. Déguernel, and E. Cambouropoulos. "Fragmentations with pitch, rhythm and parallelism constraints for variation matching". In: *International Symposium on Computer Music Multidisciplinary Research (CMMR)*. 2013, pp. 298–312.

[10] R. Gjerdingen. *Music in the Galant Style*. New York: Oxford University Press, 2007.

[11] S. Jan. "Using galant schemata as evidence for universal Darwinism". In: *Interdisciplinary Science Reviews* 38.2 (2013), pp. 149–168.

[12] B. Janssen, W. B. de Haas, A. Volk, and P. van Kranenburg. "Finding repeated patterns in music: state of knowledge, challenges, perspectives." In: *International Symposium on Computer Music Multidisciplinary Research (CMMR)*. 2013, pp. 277–297.

[13] A. Katsiavalos, T. Collins, and B. Battey. "An Initial Computational Model for Musical Schemata Theory". In: *Proceedings of the International Society on Music Information Retrieval*. 2019, pp. 166–172.

[14] G. King and L. Zeng. "Logistic Regression in Rare Events Data". In: *Political Analysis* 9.2 (2001).

[15] O. Lartillot. "Automated Motivic Analysis: An Exhaustive Approach Based on Closed and Cyclic Pattern Mining in Multidimensional Parametric Spaces". In: *Computational Music Analysis*. Ed. by D. Meredith. Springer, 2016, pp. 273–302.

[16] O. Lartillot. "Motivic pattern extraction in symbolic domain". In: *Intelligent music information systems: Tools and methodologies*. IGI Global, 2008, pp. 236–260.

[17] D. Meredith, K. Lemström, and G. A. Wiggins. "Algorithms for Discovering Repeated Patterns in Multidimensional Representations of Polyphonic Music". In: *Journal of New Music Research* 31.4 (Dec. 1, 2002), pp. 321–345. ISSN: 0929-8215. DOI: 10.1076/jnmr.31.4.321.14162.

[18] L. Pugin, R. Zitellini, and P. Roland. "Verovio: a library for engraving MEI music notation into SVG". In: *Proceedings of the International Symposium on Music Information Retrieval*. 2014, pp. 107–112.

[19] J. Rice. "Adding to the Galant Schematicon: The Lully". In: *Forthcoming in Mozart-Jahrbuch* (2014).

[20] J. A. Rice. "The Morte: a Galant Voice-Leading Schema as Emblem of Lament and Compositional Building-Block". In: *Eighteenth-Century Music* 12.2 (2015), pp. 157–181.

[21] D. R. W. Sears, A. Arzt, H. Frostel, R. Sonnleitner, and G. Widmer. "Modeling Harmony with Skip-Grams". In: *Proceedings of the International Society on Music Information Retrieval*. 2017, pp. 332–338. ISBN: 978-981-11-5179-8.

[22] D. R. W. Sears and G. Widmer. "Beneath (or beyond) the Surface: Discovering Voice-Leading Patterns with Skip-Grams". In: *Journal of Mathematics and Music* 0.0 (July 14, 2020), pp. 1–26. ISSN: 1745-9737. DOI: 10.1080/17459737.2020.1785568. URL: https://doi.org/10.1080/17459737.2020.1785568 (visited on 07/29/2020).

[23] J. Symons. "A Cognitively Inspired Method for the Statistical Analysis of Eighteenth-Century Music, as Applied in Two Corpus Studies". PhD thesis. Northwestern University, 2017.

SEMANTICALLY MEANINGFUL ATTRIBUTES FROM CO-LISTEN EMBEDDINGS FOR PLAYLIST EXPLORATION AND EXPANSION

Ayush Patwari, Nicholas Kong, Jun Wang, Ullas Gargi
YouTube Music
{patwaria,kongn,juwanng,ullas}@google.com

Michele Covell, Aren Jansen
Google Research
{covell,arenjansen}@google.com

ABSTRACT

Audio embeddings of musical similarity are often used for music recommendations and autoplay discovery. These embeddings are typically learned using co-listen data to train a deep neural network, to provide consistent triplet-loss distances. Instead of directly using these co-listen–based embeddings, we explore making recommendations based on a second, smaller embedding space of human-intelligible musical attributes. To do this, we use the co-listen–based audio embeddings as inputs to small attribute classifiers, trained on a small hand-labeled dataset. These classifiers map from the original embedding space to a new interpretable attribute coordinate system that provides a more useful distance measure for downstream applications. The attributes and attribute embeddings allow us to provide a search interface and more intelligible recommendations for music curators. We examine the relative performance of these two embedding spaces (the co-listen–audio embedding and the attribute embedding) for the mathematical separation of thematic playlists. We also report on the usefulness of recommendations from the attribute-embedding space to human curators for automatically extending thematic playlists.

1. INTRODUCTION

Automatically annotating music with semantically meaningful and musically relevant attributes is an important effort with a long history [1–5]. It has become especially important as music-streaming services have made large catalogs of recorded music available to people worldwide and as the user interface to these services continues to shift towards voice activation rather than text search or graphical browsing. Describing music using such musical attributes has many applications such as:

- allowing consumers to search for music that satisfies musical, emotional or psychological constraints, using text or voice queries;

- browsing for such music using common usage patterns reflecting activities (*e.g.,* "running") or moods (*e.g.,* "chill");

- sequencing playlists for users that allows them to choose which aspect of musical similarity to maintain, rather than simply following general co-listen patterns;

- providing power users and curators the ability to program specific experiences using higher-order operators.

In this paper, we describe a system for understanding and describing music content. The paper is divided into two main parts. The first part (Section 2) describes how we extract semantically meaningful attributes from features primarily based on audio-spectrogram embeddings trained on co-listen user behavior. The second part (Section 3) explores using these semantic-attribute embeddings both to characterize and to extend professionally curated playlists.

2. EXTRACTING SEMANTICALLY MEANINGFUL ATTRIBUTES FROM CO-LISTEN EMBEDDINGS

We collaborated with the YouTube Music curation team to establish a prioritized list of semantically meaningful audio attributes. Several attributes are subjective, making it difficult to get enough ground truth data to support training. Typical deep networks require many labeled examples, due to the millions of trainable parameters in modern deep networks: for example, ResNet-18 (used by [6]) has around 11 million trainable parameters. We could use existing meta-data for some attributes, such as genre, but other important attributes, such as vocalness (presence of vocals) and energy, are not as prevalent. Even with genres, there is not a consistent set of labels available across different music distributors.

We first review the work of [6]: building on this work allows us to have shallow yet powerful attribute embeddings. In Subsection 2.2, we describe our approach to extracting full-track–level attributes from those co-listen–based audio embeddings. We then discuss our work in measuring attribute *consistency* across the duration of the track (Subsection 2.3). Finally, in Subsection 2.4, we report our accuracy.

2.1 Co-listen–based Audio Embeddings

To allow us to train only comparatively shallow networks, we build on the audio-embedding work done by [6]. In this subsection, we review the approach taken in that previous work.

In [6], an initial audio embedding was obtained using triplet loss from aggregated listening sessions. With the triplet loss, tracks that were (in aggregate) listened to together were trained to be closer in the embedding space than those that were not. The unaveraged embedding is generated using a modified version of Resnet-18, operating on overlapping 3-second audio-spectrogram windows (with a 1-second overlap). For training, [6] used random 3-second samples from the anchor, positive-example, and negative-example tracks. They evaluated by holding back 10% of the 10.5-million audio tracks in their co-listen dataset. When testing on this hold-out set, they achieve over 50% improvement in performance as [2] under the same training regime (0.079 average precision vs. 0.055 for [2]).

In this paper, we use averages of the [6] embeddings as our audio embeddings, with averaging either over the full song duration (Subsection 2.2) or over 10-second tiles (Subsection 2.3).

2.2 Co-Listen Embeddings to Full-Track Attributes

In this paper, we consider the following attributes:

- Genre: A subset of the full international set of genres, including only those deemed important for the US market [1], specifically: Hip-Hop/Rap, R&B/Soul, Blues, Country, Jazz, Rock, Metal, Pop, Dance/Electronic, Alternative/Indie, Latin Urbano, Regional Mexican, Reggae, K-Pop, Korean Ballads, and Classical;

- Valence (or hedonic tone): a measure of the emotional positivity or negativity of the music;

- Vocalness: a measure of the prominence of speaking and singing (or even wordless screaming or humming);

- Energy: a qualitative measure of the intensity (or autonomic arousal) of the music;

- Temporal consistency of energy across a track (Subsection 2.3).

The primary source of ground-truth labels for our attribute-classification models is the team of music experts at YouTube Music. This source means that were are limited to between and 10,000 and 20,000 training examples for the energy, valence, and vocalness classifiers and around 1,000 manual labels per genre for the multi-label genre classifiers.[2]

To allow robust training, even from this small amount of data, we train comparatively small, separate, fully-connected neural nets on top of the (frozen) audio embeddings given by [6]. This is a version of transfer learning but we do not attempt to fine-tune the underlying audio embeddings for our semantic-attribute task, due to the comparatively small amount of training data we have available in our attribute space.

We then create an attribute embedding space using the continuous-valued outputs of the final logits of each of these classifiers, followed by pooled-variance normalization, as will be described in Subsection 3.2.

Details about the classifier network architectures and the training data are given next.

2.2.1 Genre

The genre model is a multi-label classifier (outputting 0 or more labels per video) from a vocabulary of 54 genres. It is trained on a mix of 50,000 manually labeled videos and 6,500,000 labels inferred from the DDEX feed delivered by music labels.[3] The genre-classification network is fully connected with 8 512-wide hidden layers. The input features are:

- Average audio embeddings [6] (across the full track)

- Average video embeddings [7] (across the full track)

- Image embedding of the video thumbnail [8]

- Word embeddings derived from a CBOW model [9] trained on 10B search queries. They are applied to the tokenized title, free-text DDEX genre, and free-text music label name of the video.

- Inferred language of the title [10, 11]

- Video type (Art Track [12], official music video, user-generated content)

While we might have been able to achieve higher music-genre accuracy by learning cross-modal co-embeddings [13–15], we instead use video, image, and word embeddings trained for general video retrieval, without restriction to music-related content. This allows us to re-purpose more general embeddings, again avoiding the overhead of large-network training, just as we have with re-purposing the audio–co-listen embeddings [6].

We trained a genre-classifier network for 2 million steps using Adagrad with a learning rate of 0.05 and a mini-batch size of 64. We did not use regularization. We selected per-class thresholds by choosing the point that maximizes F1 on a separate test set.

---

[1] We restricted our genre set since our evaluation was focused on playlists generated primarily for the US market and since the definition/assignment of genre is not uniform across the globe.

[2] For genres, we also have other sources of label data, as described in Subsection 2.2.1 but that secondary source is significantly less reliable.

[3] The DDEX feed labels must be mapped from the often idiosyncratic genre tags provided by each music label to the 54 genre label set that forms our vocabulary. That mapping is difficult to correctly determine, resulting in noisy training data. We do not use this secondary source of label data at all in evaluation.

Figure 1. Temporal Computed Energy for *Stairway to Heaven*.

2.2.2 Energy

Energy is computed using the output of a regression model using full-track-average audio embeddings [6] as its only input. It was trained on around 20,000 human-labeled examples with ratings in one of 3 buckets (i.e., low, medium, high). These ratings are then converted to scores of 0, $\frac{1}{2}$, and 1. The network is fully connected with 2 hidden layers: the first layer is 128 units wide and the second, 64 units wide. It used Adam optimizer with a decay rate of 0.98 and was trained for 10,000 steps.

2.2.3 Valence

Valence is computed using the output of a regression model, again using full-track-average audio embeddings as its only input. The network is fully connected with 3 hidden layers: the first and third layers are 256 units wide and the second, 512 units wide. It was trained on 10,500 human-labeled examples. As with energy, the training data was human bucketed ratings with 3 distinct levels, from negative (sad or angry), neutral, to positive (happy or content) and the buckets were assigned to 0, $\frac{1}{2}$, and 1. It used Adam optimizer with a decay rate of 0.96 and was trained for 10,000 steps.

2.2.4 Vocalness

Vocalness is computed using the output of a binary classification model, using full-track-average audio embeddings as its only input. It was trained on 18,000 human-labeled examples. The raters were asked to indicate if there were significant lyrics or other vocal elements in the track. Like valence, the network is fully connected with 3 hidden layers: the first and third layers are 256 units wide and the second, 512 units wide. It used Adam optimizer with a decay rate of 0.96 and was trained for 10,000 steps.

2.3 Temporal Inference

While the attributes listed above are often used to describe the entirety of a music track, there can be significant variations in some attributes over the temporal extent of a song. As an example, Figure 1 shows computed energy for the song *Stairway to Heaven*. Generating a single audio embedding representing the whole track via the mean of window samples results in a loss of information on this aspect.

For attributes like this, we shift to performing inference on 10-second segments of audio, using the time-localized audio embeddings. We do this in two steps. As a first step, we train a full-track model of the desired attribute, with the *track-level average* of the local audio embeddings as its

input. With that trained model in hand, we reuse it, running a separate inference on each 10-second audio embedding, to generate time-localized attribute estimates. From this sequence of localized estimates, we compute a track-level estimate using an aggregate heuristic. For example, for the track-level energy, we take the maximum over the moving average of this temporal estimate as follows:

$$\mathrm{E} = \max_{0 \le i < N-W} \frac{1}{W} \sum_{j=i}^{i+W-1} e_j \qquad (1)$$

where N is the the number of 10-second segments in a track, e_j is the raw energy estimate for the jth segment, and W is the window size which also a function of N according to $W = \max\{3, \frac{N}{6}\}$.

In the future, we could train a time-localized regression using explicit labels on the 10-second segments or using the temporal estimates provided by the full-track model as weak labels. However, for the purpose of this paper we restrict ourselves to the method described above.

Separately, using the sequence of local estimates, we measure the attribute's consistency. The local estimate is first smoothed, to give more reliable local estimates of the attribute. For example, for energy, we use a moving average with windows as described above. From that sequence, we can create a consistency measure using:

$$\mathrm{Consistency} = 1 - \frac{\sum_{i=0}^{N-2} |A_{i+1} - A_i|}{\sum_{i=0}^{N-2} A_i}, \qquad (2)$$

where A_i is the attribute value, determined by the smoothed data and centered on the ith 10-second segment of the track.

We improved precision from 85% to 90% using this approach instead of inference on the mean audio embedding, demonstrating the performance improvement possible from aggregating local inferences compared to performing inference on pre-aggregated embeddings. The attribute's consistency measure is also separately useful: for example, it can give playlist curators a deeper description of the acoustic-energy profile, which is needed for task-targeted playlists like "workout" or "focus".

While this approach can be used for other (non-genre) attributes as well, for the results in this paper, we only used it with energy.

2.4 Accuracy of Individual Attribute Detectors

Table 1 describes the accuracy of each of our attribute models.

Energy and valence are regression models. The test set was created from human annotations on a four-point scale. The regression results were evaluated using error thresholds, set according to what was judged acceptable by the curators. Even though the training data for valence used a four-point scale, we used an evaluation threshold that is equivalent to a three-point scale. This coarser scale for valence evaluation was based on the needs of the curators.

Vocalness was trained as a binary classification and was evaluated accordingly.

Attribute	Metric	Quality
Genres: Multi-label classifier.[†]	Human-expert labels[†]	78% precision, 84% recall
Valence (regression, output $\in [0, 1]$)	Prediction < 0.33 from label[‡]	78% accuracy
Vocalness (binary classifier)	Human-labeled ground truth	97% precision, 78% recall
Energy (regression, output $\in [0, 1]$)	Prediction < 0.25 from label[‡]	90% accuracy

[†] The genre classifier was evaluated on the 16 genres used in this paper. The evaluation set was formed from the entries from expert-curated single-genre playlists and their labels were inferred accordingly.
[‡] The thresholds for accuracy were set after consulting with expert human curators who provided musical examples of differences in valence and energy that should be distinguishable.

Table 1. Accuracy of Each Full-Track Attribute Model.

For genre, we created an evaluation set using entries from single-genre playlists authored by curators, with labels inferred accordingly. None of these curated tracks were used in training the genre classifier.

We are able to extract semantic attribute labels from the frozen audio embeddings, both for genre (similar to [6]) and for more qualitative measures (energy, valency, vocalness). For the qualitative measures, this labeling is based solely on the audio embedding.[4] Huang et al. [6] report similar findings. It is somewhat surprising that the semantic information needed to compute these labels are captured by embeddings trained for a completely different task (that is, predicting which songs are listened to together). Based on this observation, we hypothesize that an embedding space formed from these semantic attributes can be used for recommending additions to human-curated playlists. The results that we report in Section 3.2 support this conjecture and strengthen it by suggesting that our attribute-embedding space is better suited for playlist recommendations than the full audio-embedding space.

3. EXPLORING CURATED PLAYLISTS

In this section, we examine the use of our learned musical attributes as a tool for discriminating between and adding to music playlists. In Subsection 3.2, we compare the discriminative power of the attribute-embedding space to that of the audio-embedding space, using the playlists that we describe in Subsection 3.1. Subsection 3.3 then describes a human evaluation of the attribute-based recommendations for playlist extension.

3.1 Corpus of curated playlists

YouTube Music [16] offers playlists with specific themes. Playlist themes are relevant to targeted users or markets and mostly fit into one of the following categories: ephemeral (*e.g.,* event based), seasonal, and canonical. The canonical category includes contextual (*e.g.,* bedtime music), mood (*e.g.,* feel-good favorites), and activity (*e.g.,* workout essentials) playlists. The canonical category is the

most amenable to attribute analysis and automatic augmentation. For this paper, we focused on playlists in the canonical category.

We used several of curated canonical playlists to evaluate how well our attributes characterize and discriminate between the groupings that were created by human experts. Curated playlists are widely served across YouTube Music. All of these curated playlists were publicly available as of April 2020.

For our embedding-space studies, described in Subsection 3.2, we use 17 different human-curated, genre-based playlists. That combined set of playlists had 4,563 entries. To avoid evaluation-set contamination, none of the playlists or their entries were used in the attribute training (Section 2). The names of these 17 playlists are given in Figure 2 and a more complete description is given in [17].

For our playlist-extension studies, described in Subsection 3.3, we use another disjoint set of 5 different human-curated, *vibe*-based playlists. The combined set had 545 entries. General characterizations of these 5 playlists, along with their URL links, are given in [17].

3.2 Analyzing playlists in attribute space
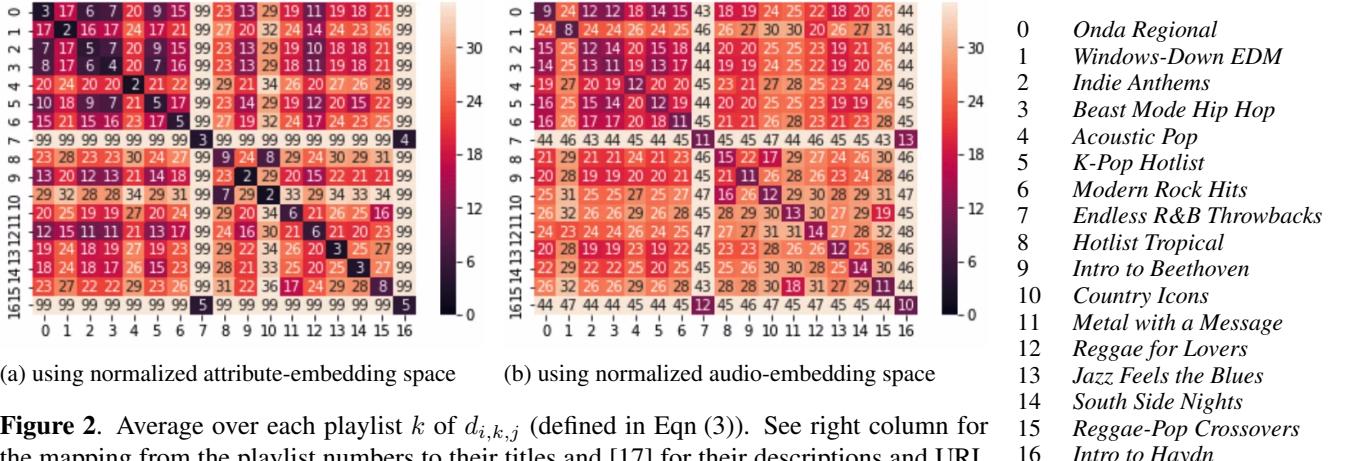
In this section, we investigate how well two different embedding spaces capture the structure of human-curated playlists. The first embedding space is the 128-dimension space generated from [6], renormalized according to the methodology described below. The second is a (renormalized) embedding space formed using the continuous logits that are trained to provide our attribute labels: that is, energy, valence, vocalness, and 16 top-level genres that are typical of music listened to in the US.

We renormalize each space using the Tikhonov-regularized [18] square-root inverse of the pooled variance matrix [19]. The pooled variance matrix is estimated using a sampling of playlists and treating each playlist as a separate cluster (with an independent cluster mean) but with a single shared (pooled) variance matrix. The embedding space is then rotated and scaled, according to the square-root inverse of this pooled variance. We use Tikhonov regularization in this inversion to avoid possible problems with nearly singular variance matrices.[5]

After re-normalization, each playlist is (on average) a Gaussian distribution with an identity-variance matrix, allowing us to directly compare between-playlist distances across the two embedding spaces. We use this distance equivalence throughout this section, to determine how well human-curated playlists are separated in these two embedding spaces. Our hypothesis is that, whichever embedding space gives better separation between authored playlists will also give better suggestions for creating or extending playlists. We will more directly examine how well our suggestions do for playlist generation in the next section. Before moving to that analysis, we compare the mathematical performance of the two spaces in this section.

---

[4] For genre, we provide our networks with information derived from the video content and text in the title and description (see Subsection 2.2.1), as well as the co-listen–based audio embeddings.

[5] We did not observe any near singularities in either the attribute- or the audio-embedding spaces but continued to use it, to avoid issues in the future, when we plan to use a larger group of attributes.

(a) using normalized attribute-embedding space   (b) using normalized audio-embedding space

0	*Onda Regional*
1	*Windows-Down EDM*
2	*Indie Anthems*
3	*Beast Mode Hip Hop*
4	*Acoustic Pop*
5	*K-Pop Hotlist*
6	*Modern Rock Hits*
7	*Endless R&B Throwbacks*
8	*Hotlist Tropical*
9	*Intro to Beethoven*
10	*Country Icons*
11	*Metal with a Message*
12	*Reggae for Lovers*
13	*Jazz Feels the Blues*
14	*South Side Nights*
15	*Reggae-Pop Crossovers*
16	*Intro to Haydn*

Figure 2. Average over each playlist k of $d_{i,k,j}$ (defined in Eqn (3)). See right column for the mapping from the playlist numbers to their titles and [17] for their descriptions and URL links.

We compared the separation of the 17 different human-curated, genre-based playlists from Subsection 3.1 by examining the distances between each playlist entry and all of the playlist centroids:

$$d_{i,k,j} = ||e_{i,k} - m_j||^2 \qquad (3)$$

where $e_{i,k}$ is the embedding-space coordinates for the ith entry in the kth playlist and m_j is the mean of embedding-space coordinates across all N_j entries in the jth playlist: $m_j = \frac{1}{N_j} \sum_{i=0}^{N_j-1} e_{i,j}$. Figure 2 shows the average of these distances for each playlist: that is, $\frac{1}{N_k} \sum_{i=0}^{N_k-1} d_{i,k,j}$. The larger the distance the less "alike" the two playlists appear in that embedding space. Based on Figure 2, the (normalized) attribute embedding space does a better than the (normalized) audio embedding space at separating the playlists, while keeping each individual playlist compact.

We can look at this separation/compactness of playlists in each embedding space, with one summary statistic per playlist entry. We use $\Delta_{i,k} = \min_{j \neq k} d_{i,k,j} - d_{i,k,k}$: the smallest difference between each entry's distance to the closest, "other" mean and its distance to its own mean. Figure 3 shows the histograms of this relative distance measure for the (normalized) attribute-embedding space and the (normalized) audio-embedding space, as well as the histogram of the paired difference between them. For Figure 3-a and Figure 3-b, highly positive values are best and negative values indicate an entry that is closer to a different playlist's mean than to its own. For Figure 3-c, positive values correspond to the attribute-embedding space giving better separation than the audio-embedding space. Using a single-tail, paired Student t-test [20] on this data indicates that the attribute embedding space is significantly better than the audio embedding space with a probability well over 99% (t = 27.24, p = 3e-151). In order to be certain that this high level of significance does not derive from unequal variances across the two embedding spaces, we also ran a single-tail Welch's unequal-variance t-test [21]: this still showed well above 99% certainty (t = 16.24, p = 5e-58).

(a) $\Delta_{i,k}^{att}$ using normalized attribute-embedding space   (b) $\Delta_{i,k}^{aud}$ using normalized audio-embedding space   (c) paired difference: $\Delta_{i,k}^{att} - \Delta_{i,k}^{aud}$ (matched indices)

Figure 3. Histograms of $\Delta_{i,k}$ using the two different embedding spaces and of their difference.

3.3 Extending playlists using attributes

In this section, we explore the application of the learned-attribute space to generating algorithmic candidates that could be used to refresh or extend human curated playlists in the corpus described in Subsection 3.1. One advantage of using the learned-attribute space is the ability for humans to understand, debug, and tweak the automatic method. We run an experiment where we show professional music curators a set of candidates for a playlist and ask them to assign a rating of whether those candidates sufficiently align with its *vibe*.

3.3.1 Playlist selection

We selected 5 playlists from the corpus for this experiment with the premise that they had *consistent vibe* with focus on *context/mood/activity* rather than, for example, *most popular* or *new releases*. We hypothesize that a semantically meaningful attribute space would be better at generating recommendations closer to the *vibe* of such playlists compared to traditional co-listen–based approaches.

3.3.2 Candidate generation

For each playlist, we aggregated attribute scores across its tracks to create a *recipe* consisting of the following:

- Top-N genres contributing to 80% of the cumulative frequency distribution where a genre is assigned to a playlist if its score was above the threshold determined from evaluation in Subsection 2.4.

Playlist	Rating			Total
	Good	Borderline	Bad	
Classical for Sleeping	36%	38%	26%	214
Classic Sunshine Soul	39%	35%	26%	101
Tranquil Spa Day	37%	63%	0%	27
Feeling Good in the 80's	22%	20%	58%	143
90's Rock Relaxation	11%	24%	65%	85

Table 2. Music-curator ratings on recommendations for playlist extension. See [17] for the description and URL link of each playlist.

- Mean and standard-deviation of the real-valued attributes energy, valence, vocalness. We also included tempo in beats-per-minute computed using APM [22] as an additional attribute.

- Other metadata attributes including top-10 artists, earliest and latest release year of tracks on the playlist. These were added as constraints to weed out recommendations too far away from the playlist premise.

This *recipe* is then used to generate a list of tracks classified into the top genres, with real-valued attribute scores at most one standard-deviation away from mean, release year within the earliest and latest release year and performing artist(s) among the top artists. We sort this list by popularity in the last 365 days and prune to generate a final list of recommendations.

3.3.3 Curator assignment

Music curators were shown these recommendations and asked to assign a rating from below options

- Good - "track is not only appropriate for the playlist premise, but also a high-quality recommendation"

- Borderline - "while track's attributes align with the premise, I would not be excited to program it"

- Bad - "I would never program this track to this playlist, because it does not fit the premise". Raters were also asked to note a reason in this case.

3.3.4 Results

The results of the experiment are tabulated in Table 2. We find that, for playlists defined almost solely by mood and emotional affect, the curators found a majority of the tracks good enough to program onto the playlists and some "bad" tracks. For *Tranquil Spa Day* especially, there were no "bad" tracks in the 27 that were rated. This shows that the recipe based on semantic attributes and metadata constraints was a decent heuristic for playlist extension.

For the decade playlists (last 2 rows) the performance was very poor. To have a better understanding, we analyzed the rater notes and found that 77 out of 83 and 45 out of 56 "bad" tracks for *Feeling Good in the 80s* and *90's Rock Relaxation*, respectively, were due to the curators not feeling that the tracks belonged to the correct decade. On

examination, we found that the metadata was indeed incorrect on those tracks. Discounting these tracks with incorrect metadata, our approach again seems to perform decently on these playlists.

For a qualitative study like this, the strongest support is the overall evaluation by the music curators on whether or not the suggestions are useful to have. Even with around one-in-four playlist suggestions being discarded as incorrect, the music curators found that having these automatically generated suggestions available sped up their work on refreshing and extending the vibe-oriented playlists.

4. CONCLUSIONS

We described a system and method to automatically label musical tracks with semantically meaningful attributes, including musical genre, autonomic arousal, valence, and vocalness. These attributes are inferred using models operating on audio embeddings generated by deep neural networks trained on co-listen data, using triplet loss. The attribute models themselves are trained using smaller amounts of labeled data. We show that precision improvements can be obtained by running attribute inference on temporal segments and fusing those scores into a whole-track score compared to running inference on an averaged embedding. This approach also yields temporal consistency attributes that are useful in and of themselves.

We then define a lower-dimensional embedding space established by these semantic musical attributes. We compare these embeddings with the original audio co-listen–trained embeddings in the context of professionally curated playlists. We find that this space better separates a sample of thematic playlists: it matches the semantic similarity implicit in these professionally curated playlists better than the raw audio embedding space.

Unlike previous studies of playlist extension [23, 24], we used these semantic attributes to generate human-readable and -editable recipes for professionally curated playlists. We used those recipes to automatically extend the playlists and measured the quality of those automatic content refreshes via human evaluation.

5. REFERENCES

[1] J. Nam, J. Herrera, M. Slaney, and J. Smith, "Learning sparse feature representations for music annotation and retrieval," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2012.

[2] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2013.

[3] P. Hamel, M. Davies, K. Yoshii, and M. Goto, "Transfer learning in MIR: Sharing learned latent representations for music audio classification and similarity," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2013.

[4] A. van den Oord, S. Dieleman, and B. Schrauwen, "Transfer learning by supervised pre-training for audio-based music classification," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2014.

[5] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2017.

[6] Q. Huang, A. Jansen, L. Zhang, D. Ellis, R. Saurous, and J. Anderson, "Large-scale weakly-supervised content embeddings for music recommendation and tagging," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[7] S.-J. Hwang, J. Lee, B. Varadarajan, A. Gordon, Z. Xu, and A. Natsev, "Large-scale training framework for video annotation," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[8] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of International Conference on Learning Representations Workshop*, 2013.

[10] R. Datta, "PHIL: The probabilistic hierarchical inferential learner," in *10th Annual Bay Area Discrete Mathematics Day*, 2005. http://math.berkeley.edu/~datta/philtalk.pdf.

[11] T. Jauhiainen, M. Lui, M. Zampieri, T. Baldwin, and K. Lindén, "Automatic language identification in texts: A survey," *Journal of Artificial Intelligence Research*, vol. 65, 04 2018.

[12] Y. Help, "What is an Art Track?" *https://support.google.com/youtube/answer/6007071*, 2020.

[13] D. Surís, A. Duarte, A. Salvador, J. Torres, and X. G. i Nieto, "Cross-modal embeddings for video and audio retrieval," in *Proceedings of the European Conference on Computer Vision Workshops*, 2018.

[14] B. Li and A. Kumar, "Query by video: Cross-modal music retrieval," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019.

[15] B. Elizalde, S. Zarar, and B. Raj, "Cross modal audio search and retrieval with joint embeddings based on text and audio," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[16] Y. Music, "https://music.youtube.com," 2020.

[17] U. Gargi, A. Patwari, N. Kong, and J. Wang, "Playlists used for semantically meaningful attributes from co-listen embeddings for playlist exploration and expansion," in *https://github.com/YTMCC/ismir2020/blob/master/playlists.md*, 2020.

[18] A. Tikhonov and V. Arsenin, *Solution of Ill-Posed Problems*. New York: Halsted Press, 1977.

[19] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. New York: Wiley Interscience, 2001.

[20] R. Walpole, R. Myers, S. Myers, and K. Ye, *Probability and Statistics for Engineers and Scientists*. Boston: Prentice Hall, 2012.

[21] G. Ruxton, "The unequal variance t-test is an underused alternative to Student's t-test and the Mann-Whitney U test," *Behavioral Ecology*, vol. 17, no. 4, 2006.

[22] D. Eck, "Beat tracking using an autocorrelation phase matrix," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2007.

[23] C.-W. Chen, P. Lamere, M. Schedl, and H. Zamani, "Recsys challenge 2018: Automatic music playlist continuation," in *Proceedings of the ACM Conference on Recommender Systems*, 2018.

[24] P. Papreja, H. Venkateswara, and S. Panchanathan, "Representation, exploration and recommendation of playlists," in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases Workshops*, 2019.

ASAP: A DATASET OF ALIGNED SCORES AND PERFORMANCES FOR PIANO TRANSCRIPTION

Francesco Foscarin[1]   **Andrew McLeod**[2]   **Philippe Rigaux**[1]
Florent Jacquemard[1,3]   **Masahiko Sakai**[3]

[1] CNAM, Paris, France   [2] EPFL, Lausanne, Switzerland
[3] INRIA, Paris, France   [4] Nagoya University, Nagoya, Japan

`francesco.foscarin@cnam.fr, andrew.mcleod@epfl.ch, philippe.rigaux@cnam.fr`

`florent.jacquemard@inria.fr, sakai@i.nagoya-u.ac.jp`

ABSTRACT

In this paper we present Aligned Scores and Performances (ASAP): a new dataset of 222 digital musical scores aligned with 1068 performances (more than 92 hours) of Western classical piano music. The scores are provided as paired MusicXML files and quantized MIDI files, and the performances as paired MIDI files and partially as audio recordings. Scores and performances are aligned with downbeat, beat, time signature, and key signature annotations. ASAP has been obtained thanks to a new annotation workflow that combines score analysis and alignment algorithms, with the goal of reducing the time for manual annotation. The dataset itself is, to our knowledge, the largest that includes an alignment of music scores to MIDI and audio performance data. As such, it is a useful resource for a wide variety of MIR applications, from those that target the complete audio-to-score Automatic Music Transcription task, to others that target more specific aspects (e.g., key signature estimation and beat or downbeat tracking from both MIDI and audio representations).

1. INTRODUCTION

As data-hungry deep learning models have become more ubiquitous in the field of MIR in recent years (e.g., [19, 22, 37]), large, well-annotated datasets have become increasingly important. Similar trends towards deep learning methods have been seen in related fields such as natural language processing [42] and computer vision [39]. For many tasks in these fields, large datasets can be automatically scraped from the web, and annotated quickly by non-experts (e.g., [34]). Unfortunately, the same can often not be said for tasks in MIR, for many reasons.

First, producing high-quality audio or MIDI data is a non-trivial task, requiring expert performers and expensive equipment that is not always available. Second, the availability of a high-quality digital ground truth is not guaranteed in many cases, particularly for tasks which require a

musical score [1], e.g., for the complete audio-to-score Automatic Music Transcription (AMT) task (see [1] for a recent overview of AMT). Finally, even in the case when both audio/MIDI data and high-quality digital ground truth scores are available, acquiring an alignment between the two is non-trivial. Automatic alignment methods (e.g., [30]) are often not robust enough to deliver highly reliable results and require time-consuming post-processing and cleaning by expert musicians for advanced data usage.

We introduce the Aligned Scores and Performances (ASAP) dataset [2], containing digital musical scores of Western classical piano pieces as both MusicXML and MIDI, aligned at the beat level with audio (from MAESTRO [17]) and MIDI recordings of over 1000 human performances. Regarding the three difficulties outlined above (data creation, digital ground truth availability, and recording-ground truth alignment), we use (1) publicly available MIDI and audio from expert human performances; (2) publicly-available musical scores scraped from the web; and (3) a new workflow to efficiently produce aligned beat, downbeat, key, and time signature annotations with minimal human correction required.

Although ASAP can be used for many tasks, it was designed with two categories specifically in mind: (1) complete audio-to-score AMT and (2) metrical structure-based tasks (e.g., beat and downbeat tracking, tempo estimation, metrical structure alignment, and rhythm quantization) of classical piano performance, from both audio and MIDI.

While a major goal of AMT is to convert an input audio recording into a form of human-readable music notation, the vast majority of AMT systems fall short of such an output. Rather, they convert the input audio recording into some sort of time-frequency representation: either a frame-based multi-pitch detection, where the presence of each pitch is estimated at each point in the input recording (e.g., [21]); or a note-based output such as a piano-roll or MIDI file, where notes are detected each with a pitch, an onset time, and an offset time (e.g., [16]). [3] For these purposes, the ground truth only needs to contain some aligned

---

[1] Although PDF scores are sometimes available, and the field of Optical Music Recognition (see [3] for a recent overview) involves converting these into digital format, this conversion can add errors, and starting from a clean, digital score is generally better if available.

[2] `https://github.com/fosfrancesco/asap-dataset`

[3] This can be post-processed into a musical score, as in the pipeline approach of [29], but such pipelines tend to add noise at each step.

pitch or note presence data—not a full musical score. Existing datasets such as MAPS [8] and the larger MAE-STRO [17] contain appropriate ground truths for this level of transcription (see Section 2).

In recent years, a few systems have been designed to output a more complete musical score directly (e.g., [5, 35]). While each of these works shows promise on this difficult task, neither includes time signatures or key signatures in their output. One [5] uses synthetically generated scores (and synthesized audio), and the other [35] uses audio synthesized from real scores. Since the eventual goal of AMT is a full transcription from human performance to musical score, a large dataset of non-synthetic human performances (containing the tempo deviations and timing intricacies of human performance, as well as real audio) is needed: synthetic data can be useful in the initial phase of model design, but human performance data is required for a truly reliable evaluation. ASAP provides such a dataset.

To our knowledge, no audio-to-score transcription system exists that requires fine-grained alignments between recordings and ground truths, perhaps due to a lack of available data. However, the use of data with even a coarse alignment has been shown to improve performance on the related task of monophonic note-based transcription [31], suggesting that the same should be true for audio-to-score AMT, if a large enough dataset of aligned recordings and scores was available. ASAP provides this time-alignment between the included recordings and ground truth scores.

Regarding metrical tasks, large audio datasets exist, especially for beat and downbeat tracking, enabling sophisticated systems to be trained (e.g., [2]). However, there is an absence of similarly-sized annotated datasets consisting of MIDI data, resulting in a noted lack of training data for metrical tasks from MIDI input (e.g., [25]). Even in the case of audio, much of the existing data is dance music or from other genres with relatively steady tempi compared to the classical piano music contained in ASAP (see Section 4.3 for an analysis of ASAP's tempo changes).

We produce beat and downbeat annotations for every performance in ASAP with a novel workflow that exploits the precise metrical structure information available in a musical score. Projecting this onto each corresponding performance guarantees a robust means to identify the beat and downbeat positions. Working with MIDI allows us to overcome many difficulties found in similar approaches using only audio data (e.g., [32]). The workflow allows us to drastically reduce the time required for manual annotation.

2. RELATED WORK

Some public datasets similar to ASAP—containing combinations of musical scores, MIDI performances, and audio recordings, for AMT and/or beat tracking—already exist. In this Section, we describe those existing datasets in comparison to ASAP, highlighting specifically where ASAP addresses their deficiencies. Table 1 contains a summary of the largest of these datasets in comparison to ASAP. We first describe those containing musical performances (useful for AMT), and follow that with a brief discussion of available datasets for metrical structure-based tasks.

2.1 Performance datasets

There are two datasets which contain multiple performances of many different pieces. The Vienna 4x22 Piano Corpus [11] consists of 22 different performances of each of 4 different pieces, in both audio and MIDI format, aligned to a metrical grid. The CHARM Chopin Mazurka Project [4] dataset contains many recordings of each of 49 different Mazurkas composed by Frédéric Chopin, although the original audio recordings are only referenced, and not available online (instead, many pre-calculated features are provided). While these datasets are valuable for investigating live performance deviations and comparisons between different performances of the same piece, they are not as useful for AMT, since they each consist of a small number of different pieces, leading to likely model overfitting (only 4 different pieces for Vienna 4x22, and only pieces by a single composer in the Mazurka dataset).

The Saarland Music Data (SMD) dataset [28] contains 50 synchronized audio and MIDI recordings of human performers playing a Disklavier piano. The files are not aligned with any musical scores or beat annotations, and the dataset's size is somewhat small compared to other similar datasets. Likely because of its size, SMD has has not been used for AMT in recent work to our knowledge.

CrestMuse PEDB [15] is a dataset based on audio recordings of multiple piano performances of around 100 unique pieces. However, the original audio recordings are not included. Rather, references to commercial CDs which can be purchased, and on which the recordings can be found are given. After a PDF application and pledge are filled out and submitted, access is granted to download the database in about a week. Provided in the dataset are MIDI files, whose notes have been hand-aligned to the referenced recordings; and digital musical scores in an XML-based format, to which the notes and beats of the MIDI files are aligned (using "deviation" XML tags in the score files). Since its initial release, some audio files have been added. However, these are different from the original score-aligned audio recordings, and in some cases are synthesized from MIDI performance. The difficulty of acquiring the audio recordings makes this database rarely used for audio-based tasks such as AMT.

The piano-midi dataset [5] contains 324 quantized MIDI files whose tempo curves have been manually altered with the goal of becoming more human-like. The MAPS dataset [8] contains 210 of these MIDI files (of 119 different pieces), without key and time signatures, each paired with an audio file—some synthesized and some actual recordings. A-MAPS [41] later augmented MAPS with MIDI files containing key signature and time signature annotations. Since the MIDI data is generated from tempo-varied quantized MIDI, rather than actual performance, the MIDI files and recordings do not contain all of the timing variance that would be present in real performance: note onsets and offsets which lie on the same beat in the original musical score also occur simultaneously in the corresponding

---

[4] http://www.mazurka.org.uk/
[5] www.piano-midi.de

Dataset	Size		Performance		Quantized		Annotations		
	Total	Unique	Audio	MIDI	MIDI	Score	Alignment	Metrical	Key
MAPS [8]	269	52	Pseudo†	Pseudo	✓		Full	✓ [41]	✓ [41]
CrestMuse-PEDB [15]	411	≈100	Partial†	✓	✓	✓	Full	✓	✓
SUPRA [36]	478	≈430	Pseudo†	✓					
MAESTRO [17]	1282	≈430	✓	✓					
GTZAN [38]	1000	1000	✓					✓ [24]	Global
Ballroom [12]	685	685	✓					Beat [23]	
Hainsworth [14]	222	222	✓					Beat	
SMC [18]	217	217	✓					Beat	
ASAP	1068	222	520	✓	✓	✓	Beat	Beat	✓

Table 1. An overview of the most relevant datasets for AMT (top section) and metrical tasks (middle section), compared to ASAP (bottom). Alignment refers to the level of alignment between the quantized and performance data. MAPS consists of pseudo-live performance (quantized MIDI with manually altered tempo curves). †MAPS, SUPRA (fully) and CrestMuse-PEDB (partially) include synthesized audio (not real recordings).

MIDI and audio files. In real performance, such events only rarely occur simultaneously. Rather, small timing deviations introduce gaps, overlaps, and other timing variation (see e.g. [26], Figure 4), which are therefore missing (along with ornamentation such as trills, as well as performance errors). Although these datasets contain perfectly-aligned ground truth annotations (which has made MAPS a standard for AMT evaluation since its release), their modest size and the fact that they are not real live performance are drawbacks that we hope to address with ASAP.

The SUPRA dataset [36] contains 478 MIDI files of around 430 different pieces generated from an archive of piano performances in the form of physical piano rolls. SUPRA also contains synthesized audio recordings of each MIDI file, and labels each with a composer and title, but provides no metrical alignment of the pieces.

The MAESTRO dataset [17] contains 1282 real performances of around 430 different pieces from the Yamaha piano e-competition [6]. Each performance is available as a MIDI file and an audio recording with a fine alignment of around 3 ms. Metadata are available for each performance, including the composer and title of each. MAESTRO's size, fine alignment with ground truth, and the fact that it is real performance have made it an excellent source of training and evaluation data for AMT from recording to piano-roll. However, MAESTRO does not contain any note-, beat-, or even piece-level alignment with digital musical scores, required for the complete audio-to-score AMT task (and which ASAP does contain).

2.2 Metrical structure datasets

For metrical structure-based tasks, from live performance MIDI data, annotated datasets from the previous section (particularly piano-midi and CrestMuse-PEDB) are typically used. However, they are relatively small (especially in terms of unique pieces), and piano-midi files in particular do not contain real live performance, as mentioned. For the same tasks from audio data, large annotated datasets exist, enabling sophisticated models to be designed and trained (e.g., [2]). The largest and most widely used (where

---

[6] http://piano-e-competition.com/
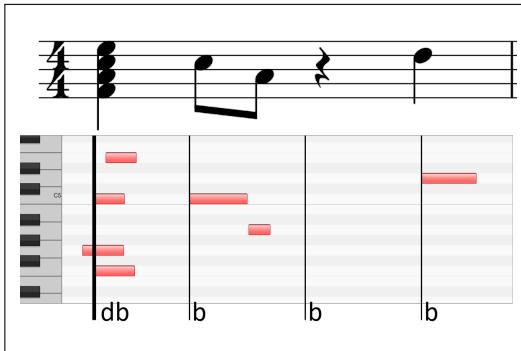
audio files are publicly available, including at least beat annotations) are: GTZAN [38] (1000 audio recordings of various genres with beat, downbeat, and 8th-note annotations), Ballroom [12] (685 audio recordings of ballroom dancing music with beat and downbeat annotations), Hainsworth [14] (222 audio recordings of Western music), and SMC [18] (217 audio recordings of Western music, specifically selected to be difficult for beat tracking). However, the music contained in these datasets tend to have a much steadier tempo than those contained in ASAP (even SMC; see Section 4.3 for a comparison).

3. PRODUCING MUSIC ANNOTATIONS

Annotating a dataset the size of ASAP with ground truth for AMT and related problems such as beat tracking is a time consuming task that can, in principle, only be performed by expert musicians. This severely limits the ease with which one can produce a reliable and finely crafted dataset at the required scale.

For piano music, MIDI and audio performances can be automatically aligned if they are recorded at the same time using an acoustic piano fitted with the proper sensors such as a Disklavier. The main problem then becomes annotating the performances with metrical markings (such as beats and downbeats) and aligning those with a musical score. Holzapfel et al. [18] describe the process used to annotate the SMC dataset in detail, showing how much manual work was required for its beat annotations. ASAP contains more than 92 hours of performance data, and even a skilled musician would need to listen to each multiple times with the proper annotation software in order to annotate it fully (and this time can increase dramatically for complicated pieces with multiple time and key signature changes). Moreover, as highlighted in [40], the manual annotations would still be affected by human subjectivity, requiring a system with multiple annotators and a reconciliation mechanism between them (e.g., [10,18]). We believe that without a large budget and/or the ability to involve a community of expert users willing to spend significant time on the task, producing a large ground truth dataset cannot be achieved through a purely manual approach.

Figure 1. Beat and downbeat annotations produced by our workflow in different cases. The 1st (db) is the median of the onsets of the corresponding notes, the 2nd and the 4th (b) are the onset of the corresponding note, and the 3rd (b) is the mean between the two neighbor annotations.

We therefore propose a workflow (Section 3.1) that allows for the automatic production of annotations from digital musical scores and MIDI performances. The precision of the results is much higher than what would be expected from human-based annotation with a single annotator. Moreover, when the quality of the available scores is high, the workflow does not require any human intervention. Manual intervention is sometimes required to fix problems related to either the digital encoding of the scores or performance errors (Section 3.2).

3.1 Annotation Workflow

The annotation workflow (Figure 3) takes a MIDI performance and a MusicXML score as input, and produces beat, downbeat, key signature change and time signature change annotations for each. Each annotation is aligned to a position (in seconds) in both the MIDI performance and a MIDI score (generated automatically from the MusicXML score), and each downbeat is further aligned with a measure number from the MusicXML score. The workflow is:

1. Expand any repetitions present in the MusicXML score and extract time and key signature changes using music21 [7].
2. Generate the MIDI score using the MuseScore3 MIDI export function.
3. Extract the times of beats, downbeats, and key and time signature changes from the generated MIDI using pretty_midi [33].
4. Align every downbeat from the MIDI score with a measure number in the XML score.
5. Produce a Score2Performance mapping from each note in the MIDI score to each note in the MIDI performance using the algorithm presented in [30].
6. The performance annotations can then be obtained. For each annotation in the MIDI score (from step 3):

   (a) Take the notes with an onset within 20ms of the annotation (there can be multiple notes, e.g. for the downbeat annotation in Figure 1).
   (b) Use the Score2Performance mapping, to obtain the onset times of the corresponding notes in the performance file.

   (c) Compute the median of the onset times of those notes, and use it as the time of the annotation in the performance.
   (d) If no notes are within 20ms of the MIDI score annotation (e.g., in case of a rest), the position is interpolated from neighboring annotations (e.g., the 3rd annotation in Figure 1)

As shown in [13], annotations on rests or multiple non-simultaneous notes (grace-notes, arpeggios) are inherently problematic, even for human annotators. An inspection of our annotations reveals that our workflow generally produces good results. In particular, our use of the median increases robustness while handling non-simultaneous notes.

3.2 Practical issues with erroneous input
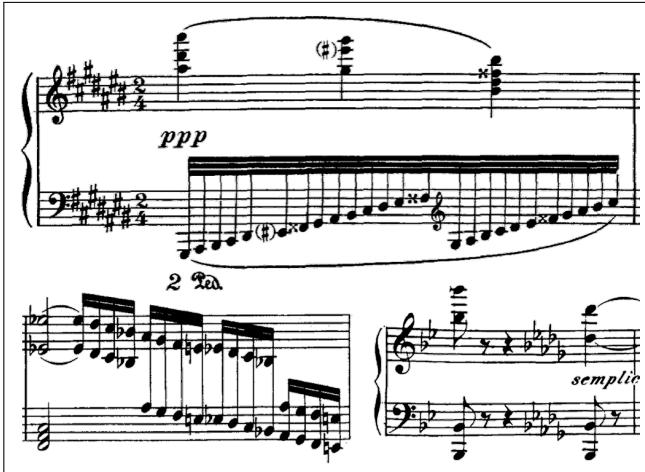
While the mapping between the scores and performances in ASAP is such that they have a very good correspondence in general, local problems can still occur in specific cases. These can be caused either by encoding issues in the XML score, or by performance errors in the MIDI.

For our automatic workflow to produce good results, the *content* level of the XML score must be correctly encoded, although we can ignore problems at the *graphical* level (we refer to the model of the multiple levels of information in a musical score proposed in [9]). Many of the problems that we encountered during the automatic creation of ASAP's annotations are tricks used by editors to fix problems at the graphical level at the expense of correctness at the content level: for example, grace notes entered as regular notes with a smaller font, invisible barlines inserted mid-bar, invisible notes or rests, or note heads changed for visual reasons inconsistent with their actual duration.

In some cases, editors are forced to use such tricks because the original score itself does not follow standard notation rules. Figure 2 shows two examples of incorrect measure duration: one from Ravel's *Ondine* (top) and another from Mozart's *Fantasie in C minor* (in 4/4; bottom left). There are many ways to handle such cases. Possibilities include having invisible tuplet markings, having a different time signature than the one displayed, and having an overflowing measure. The latter two techniques create problems for automatic beat extraction. Figure 2 (bottom right) is an example of a key change in the middle of a bar in Beethoven's *Sonata No.29, Op.106*, 2nd movement. One way to encode this situation is to split the bar into 2 separate bars, but this also creates problems for automatic beat extraction in the form of an extra downbeat.

Fortunately, such problems are generally easy to detect since they often result in a measure where the sum of the events does not match the duration defined by the time signature. To be able to produce correct annotations even in the case of these "faulty" measures (around 3% of measures in ASAP's musical scores), we introduce a manual correction step for the MIDI score annotations (Figure 3). This prevents the propagation of such problems down to the performance annotations.

The alignment that we use to generate the Score2Performance mapping [30] is robust against small errors and

Figure 2. Examples of problems in musical scores, including incorrect bar length (Ravel's *Ondine* (top) and Mozart's *Fantasie in C minor* (in 4/4; bottom left)) and a mid-bar key change (Beethoven's *Sonata No.29, Op.106*, 2nd movement (bottom right)).

Figure 3. Our annotation workflow; "*" indicates manual correction (see Section 3.2)
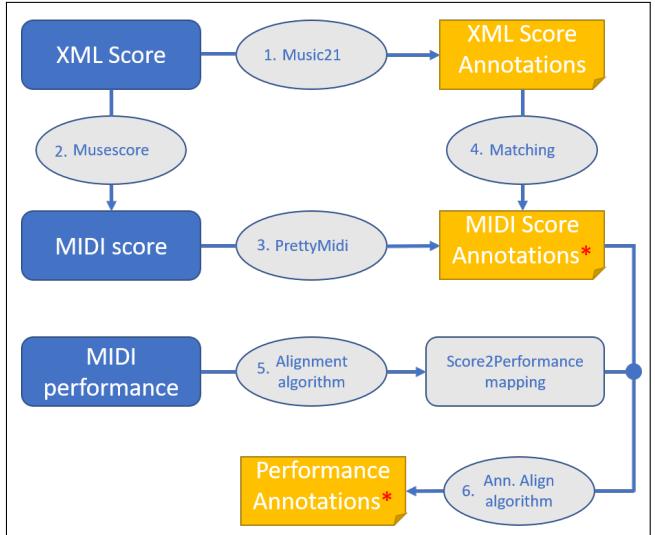
note inversions. Nonetheless, small errors in its alignment exist. As the difference between a performance and MIDI score increases, the chance of having an incorrect alignment also increases. This can occur in the case of embellishments (e.g. long trills, or mordents that can be played from the printed note or from the note above) or major performance mistakes. We try to detect these problems automatically by measuring the inter-beat-intervals of each performance and marking the outliers as possible problems. On these outliers (which occurred in around 400 of ASAP's performances), we introduce a final manual correction step. 43 performances contained significant alignment errors that were corrected, and around 2% of annotations had to be moved by less than 1 second.

4. DATASET OVERVIEW

4.1 Dataset content

ASAP contains 222 distinct musical scores and 1068 unique performances of Western classical piano music from 15 different composers (see Table 2 for a breakdown). 548 of the recordings are available as MIDI only, and all the others (520) are provided as MIDI and audio recordings aligned with approximately 3 ms precision. Each score corresponds with at least one performance (and usually more). Every score and performance in ASAP is labeled with metadata including the composer and the title of the piece. We took care to ensure that any two performances of the same piece are labeled with the same title and composer, and no two distinct pieces in ASAP share both.

Each musical score is provided in both MusicXML [7] and MIDI formats. In the MIDI score, the position of all MIDI events are quantized to a metrical grid according to their position in the MusicXML score. Grace notes are represented in MIDI as notes of very short duration. Repetitions in the score are "unfolded" in the MIDI file such

that some sections of the MusicXML score may be duplicated in the MIDI score. Except for performance mistakes, there is a one-to-one correspondence between the notes in a MIDI performance and its associated MIDI score.

For each performance and MIDI score, ASAP provides the positions (in seconds) of all beats, downbeats, time signature changes and key signature changes. Time signature changes are annotated only on downbeats. In the case of pickup measures (and pickup measures in the middle of the score) we delay the position of the time signature change annotation to the following downbeat. Similarly, key signature changes are annotated only on beats. Each downbeat is also mapped to a specific measure number in the MusicXML score, which allows for a clear score alignment, even in the case of repetitions.

The dataset and the code used to generate annotations, along with a detailed description of the specific formatting of the dataset and usage examples are available online [8].

4.2 Origin of the files

The files in ASAP are drawn from multiple sources. The MusicXML scores are from the MuseScore online library [9], created and uploaded by the users of the MuseScore music notation software. They were first collected and associated to MIDI performances from the Yamaha e-piano competition by the authors of [20]. We manually edited the MusicXML scores using MuseScore3 to correct significant notation errors, and generated quantized MIDI files using MuseScore3's MIDI export utility. The paired audio and MIDI performances come from the MAESTRO dataset [17]. We automatically matched as many of the MAESTRO performances as we could to ones collected by [20], thus associating them with musical scores. The unmatched performances from MAESTRO are not included in ASAP. Finally, we modified 5 of the MIDI performances

---

[7] https://www.musicxml.com/

[8] https://github.com/fosfrancesco/asap-dataset
[9] https://musescore.com/sheetmusic

Composer	XML/MIDI Score	MIDI Perf.	Audio Perf.
Bach	59	169	152
Balakirev	1	10	3
Beethoven	57	271	120
Brahms	1	1	0
Chopin	34	290	109
Debussy	2	3	3
Glinka	1	2	2
Haydn	11	44	16
Liszt	16	121	48
Mozart	6	16	5
Prokofiev	1	8	0
Rachmaninoff	4	8	4
Ravel	4	22	0
Schubert	13	62	44
Schumann	10	28	7
Scriabin	2	13	7
Total	222	1068	520
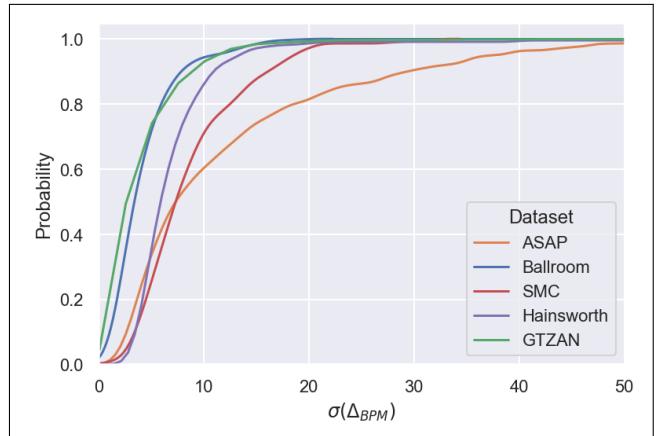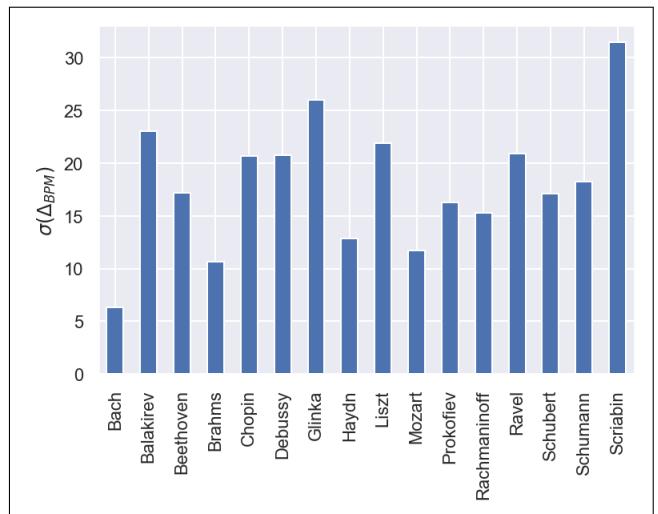
Table 2. The composition of ASAP.

Figure 4. Cumulative Distribution Function of tempo variation $\sigma(\Delta_{BPM})$ of each piece in the compared datasets.

by inserting a missing note at the beginning, and 277 more have been cut to obtain more homogeneous pieces (e.g., the Bach Preludes are separated from the Fugues, even though they sometimes come from the same performance).

4.3 Dataset Statistics

ASAP contains performances of classical piano music, a style that can be challenging for beat tracking systems due to the large tempo variations that are often present. Here, we compare the tempo variation of the pieces in ASAP to that of the pieces of datasets commonly used for beat tracking from audio [12,14,18,38]. To quantify the tempo variation for each piece, we first compute the BPM at each beat based on the amount of time between consecutive beats. Then, we compute Δ_{BPM} at each beat as the difference between consecutive BPMs. Finally, we compute the standard deviation of the set of all Δ_{BPM} values in a particular piece, which we call $\sigma(\Delta_{BPM})$. Figure 4 presents the distribution of these standard deviations for each dataset as a Cumulative Distribution Function, which shows the probability that a randomly chosen piece from each dataset has a $\sigma(\Delta_{BPM})$ less than the given value. From the plot, it can be seen that Ballroom and SMC have generally steadier tempos than the other datasets, and that ASAP's steadiest 40% and 50% of pieces roughly match those of Hainsworth and SMC respectively. However, a large portion of the pieces in ASAP have significantly larger tempo variation than any of the compared datasets.

In ASAP, differences can be observed between composers. Even though the pieces in the dataset fall under the broad term "classical music", ASAP is very diverse, containing pieces of various styles written across different periods. This can be observed from differences in average $\sigma(\Delta_{BPM})$ for each composer, as is shown in Figure 5. The values in this plot generally match musicological intuitions about tempo variation for the different composers.

Figure 5. Average tempo variation $\sigma(\Delta_{BPM})$ of the pieces by each composer in ASAP.

5. CONCLUSION

This paper presented ASAP: a new dataset of aligned musical scores and performances of classical piano music. Downbeat, beat, time signature, and key signature annotations are produced using a novel workflow that exploits information present in the musical score to drastically reduce manual annotation time compared to fully manual annotation. ASAP contains over 1000 annotated MIDI performances of classical piano music, over 500 of which are paired with audio from the MAESTRO dataset. To our knowledge, it is the largest dataset of that contains such a fine-grained alignment between scores and performances.

This work has only scratched the surface of what can be done with ASAP. Future work will present further statistical analyses on the data and baseline model performance on tasks for which it can be used: complete AMT and beat tracking as presented, as well as others such as expressive performance analysis and rendering [4]. For complete AMT in particular, the evaluation method is still an open problem, although proposals have been made (e.g., [6,27]).

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, jan 2019.

[2] Sebastian Böck, Matthew E.P. Davies, and Peter Knees. Multi-task learning of tempo and beat: Learning one to improve the other. In *ISMIR*, 2019.

[3] Jorge Calvo-Zaragoza, Jan Hajic Jr, and Alexander Pacha. Understanding optical music recognition. *Computer Research Repository, abs/1908.03608*, 2019.

[4] Carlos E. Cancino-Chacón, Maarten Grachten, Werner Goebl, and Gerhard Widmer. Computational models of expressive music performance: A comprehensive and critical review. *Frontiers in Digital Humanities*, 5:25, 2018.

[5] Ralf Gunter Correa Carvalho and Paris Smaragdis. Towards end-to-end polyphonic music transcription: Transforming music audio directly to a score. In *WAS-PAA*, pages 151–155, 2017.

[6] Andrea Cogliati and Zhiyao Duan. A metric for music notation transcription accuracy. In *ISMIR*, pages 407–413, 2017.

[7] Michael Scott Cuthbert, Christopher Ariza, and Lisa Friedland. Feature extraction and machine learning on symbolic music using the music21 toolkit. In *ISMIR*, pages 387–392, 2011.

[8] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, aug 2010.

[9] Francesco Foscarin, David Fiala, Florent Jacquemard, Philippe Rigaux, and Virginie Thion. Gioqoso, an online Quality Assessment Tool for Music Notation. In *4th International Conference on Technologies for Music Notation and Representation (TENOR'18)*, Montreal, Canada, May 2018.

[10] Thassilo Gadermaier and Gerhard Widmer. A study of annotation and alignment accuracy for performance comparison in complex orchestral music. *arXiv preprint arXiv:1910.07394*, 2019.

[11] Werner Goebl. Numerisch-klassifikatorische interpretationsanalyse mit dem "Bösendorfer Computerflügel". Master's thesis, Universität Wien, 1999.

[12] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[13] Peter Grosche, Meinard Müller, and Craig Stuart Sapp. What makes beat tracking difficult? a case study on chopin mazurkas. In *ISMIR*, pages 649–654, 2010.

[14] Stephen W Hainsworth and Malcolm D Macleod. Particle filtering applied to musical tempo tracking. *EURASIP Journal on Advances in Signal Processing*, 2004(15):927847, 2004.

[15] Mitsuyo Hashida, Toshie Matsui, and Haruhiro Katayose. A new music database describing deviation information of performance expressions. In *ISMIR*, pages 489–494, 2008.

[16] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. In *ISMIR*, 2018.

[17] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019.

[18] Andre Holzapfel, Matthew EP Davies, José R Zapata, João Lobato Oliveira, and Fabien Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2539–2548, 2012.

[19] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer: Generating music with long-term structure. In *International Conference on Learning Representations*, 2018.

[20] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, Kyogu Lee, and Juhan Nam. VirtuosoNet: A hierarchical RNN-based system for modeling expressive piano performance. In *ISMIR*, pages 908–915, 2019.

[21] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *ISMIR*, pages 475–481, 2016.

[22] Jong Wook Kim and Juan Pablo Bello. Adversarial learning for improved onsets and frames music transcription. In *ISMIR*, pages 670–677, 2019.

[23] Florian Krebs, Sebastian Böck, and Gerhard Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *ISMIR*, pages 227–232, 2013.

[24] Ugo Marchand and Geoffroy Peeters. Swing ratio estimation. In *Digital Audio Effects (DAFx)*, pages 423–428, 2015.

[25] Andrew McLeod, Eita Nakamura, and Kazuyoshi Yoshii. Improved metrical alignment of MIDI performance based on a repetition-aware online-adapted grammar. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 186–190, 2019.

[26] Andrew McLeod and Mark Steedman. HMM-based voice separation of MIDI performance. *Journal of New Music Research*, 45(1):17–26, 2016.

[27] Andrew McLeod and Mark Steedman. Evaluating automatic polyphonic music transcription. In *ISMIR*, pages 42–49, 2018.

[28] Meinard Müller, Verena Konz, Wolfgang Bogler, and Vlora Arifi-Müller. Saarland music data (SMD). In *Late-Breaking and Demo Session of ISMIR*, 2011.

[29] Eita Nakamura, Emmanouil Benetos, Kazuyoshi Yoshii, and Simon Dixon. Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 101–105. IEEE, 2018.

[30] Eita Nakamura, Kazuyoshi Yoshii, and Haruhiro Katayose. Performance error detection and post-processing for fast and accurate symbolic music alignment. In *ISMIR*, pages 347–353, 2017.

[31] Ryo Nishikimi, Eita Nakamura, Satoru Fukayama, Masataka Goto, and Kazuyoshi Yoshii. Automatic singing transcription based on encoder-decoder recurrent neural networks with a weakly-supervised attention mechanism. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165. IEEE, 2019.

[32] Adriana Olmos, Nicolas Bouillot, Trevor Knight, Nordhal Mabire, Josh Redel, and Jeremy R Cooperstock. A high-fidelity orchestra simulator for individual musicians' practice. *Computer Music Journal*, 36(2):55–73, 2012.

[33] Colin Raffel and Daniel P. W. Ellis. Intuitive analysis, creation and manipulation of midi data with pretty_midi. In *ISMIR Late Breaking and Demo Papers*, 2014.

[34] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[35] Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. A holistic approach to polyphonic music transcription with neural netwoks. In *ISMIR*, pages 731–737, 2019.

[36] Zhengshan Shi, Craig Stuart Sapp, Kumaran Arul, Jerry McBride, and Julius O. Smith. SUPRA: Digitizing the stanford university piano roll archive. In *ISMIR*, pages 517–523, 2019.

[37] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-U-Net: A multi-scale neural network for end-to-end audio source separation. In *ISMIR*, pages 334–340, 2018.

[38] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[39] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[40] Christof Weiß, Vlora Arifi-Müller, Thomas Prätzlich, Rainer Kleinertz, and Meinard Müller. Analyzing measure annotations for western classical music recordings. In *ISMIR*, pages 517–523, 2016.

[41] Adrien Ycart and Emmanouil Benetos. A-MAPS: Augmented MAPS dataset with rhythm and key annotations. In *ISMIR Late Breaking and Demo Papers*, 2018.

[42] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.

MOOD CLASSIFICATION USING LISTENING DATA

Filip Korzeniowski[1]   **Oriol Nieto**[1]   **Matthew C. McCallum**[1]
Minz Won[2]   **Sergio Oramas**[1]   **Erik M. Schmidt**[3]
[1] Pandora Media LLC., Oakland, California, USA
[2] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
[3] Netflix Inc., Los Gatos, California, USA

ABSTRACT

The mood of a song is a highly relevant feature for exploration and recommendation in large collections of music. These collections tend to require automatic methods for predicting such moods. In this work, we show that listening-based features outperform content-based ones when classifying moods: embeddings obtained through matrix factorization of listening data appear to be more informative of a track mood than embeddings based on its audio content. To demonstrate this, we compile a subset of the Million Song Dataset, totaling 67k tracks, with expert annotations of 188 different moods collected from AllMusic. Our results on this novel dataset not only expose the limitations of current audio-based models, but also aim to foster further reproducible research on this timely topic.

1. INTRODUCTION

The estimation of moods that a given music track might evoke or empathize with is a relevant task that has been active in the Music Informatics Research (MIR) community for years [20]. This task, which is also known as music emotion recognition, has become even more prominent thanks to the advent of streaming music services with massive collections, where understanding the set of moods of each of their tracks could strongly impact the navigation, discovery, and recommendations of such collections [32]. This task has been typically approached in two different ways: i) regressing a continuous mood space such as the Arousal-Valence one [30], and then clustering such space to obtain a specific mood vocabulary [37]; or ii) classifying a given track into one or more moods, thus becoming a multi-label classification problem with a fixed vocabulary [6], which can be seen as a sub-task of the broader audio tagging problem [27]. In this work, we focus exclusively on the second approach, since it can directly impact search-by-mood applications, while methods like metric learning can potentially overcome the limitation of the fixed vocabulary [5].

Framed under the context of music recommendation, mood recognition is particularly interesting. It has been shown that listener personality correlates not only with musical taste [29, 41], but also with genre [11], which makes the development of psychologically inspired approaches one of the most compelling challenges for recommender systems [32]. Thus, several related techniques have been presented: FocusMusicRecommender [40] makes use of the listener's behavior history to play tracks that are appropriate given the current listener's level of concentration. By incorporating the Five Factor Model [7], collaborative filtering [21] is enhanced with personality embeddings [10]. Moreover, emotions from a microblogging service have been exploited to implement an emotion-aware recommendation system [9]. Such techniques employ data beyond the actual audio signal to enhance mood-based recommenders, inspiring us to make use of listening data to classify moods to potentially improve the navigation and recommendation of large music catalogs.

The contribution of this work to the task of mood prediction is two-fold: i) we assemble a set of 67k tracks from the Taste Profile subset from the Million Song Dataset (MSD) [2] and match them with human-annotated moods available from AllMusic. [1] This is, to the best of our knowledge, the largest expert-annotated mood dataset available. And ii) by running several experiments on this proposed dataset we show how listener data are much more accurate at classifying moods than current audio-based approaches. Similarly to [17], where its authors discuss how lyrics can be useful to predict moods better than actual audio, and following the music recommendation approaches described above, we further argue that listening embeddings yield superior results due to their ability to capture information that is not straightforward to be extracted from pure audio content only.

The rest of the article is structured as follows: in Section 2 we give a formal definition of the mood classification problem. In Section 3 the data employed in this work are described. We then detail the mood classification experiments in Section 4. The results of these experiments are discussed in Section 5. Finally, we draw conclusions and consider potential future directions in Section 6.

---

[1] https://www.allmusic.com/

2. MOOD CLASSIFICATION

Predicting moods evoked by music is often treated as an audio classification problem in the MIR community, [2] where audio data are almost exclusively used as input. In this section we give an overview of this task and its current approaches.

2.1 Problem definition

Mood tagging is a multi-label classification problem, and can be considered a subset of the broader audio tagging task where only those tags that represent moods are considered. Formally, let $\mathbf{x} \in \mathbb{R}^E$ be an embedding representing a given track, where E is the number of dimensions in the embedding. Each track is associated with a set of mood tags from a mood vocabulary $\mathcal{T}$ (e.g., "energetic," "gloomy," "happy"), represented by a binary indicator vector $\mathbf{y} \in \{0,1\}^{|\mathcal{T}|}$. We aim at predicting the set of mood tags associated with the track, using a learnable function f that computes the predicted label vector $\hat{\mathbf{y}} = f(\mathbf{x})$.

Note that $\mathbf{x}$ can be extracted from any source of data representing the track. In our case, we will use audio- and listening-based embeddings.

Other approaches have also framed emotion prediction as a regression problem of an n-dimensional continuous space [37], where the 2D Arousal-Valence model [30] is the most widely used. While this approach has the benefit of considering moods that are not constrained by a specific vocabulary, in this work we focus on the multi-label classification approach due to the direct application to potential user-based scenarios such as search by typing or by voice.
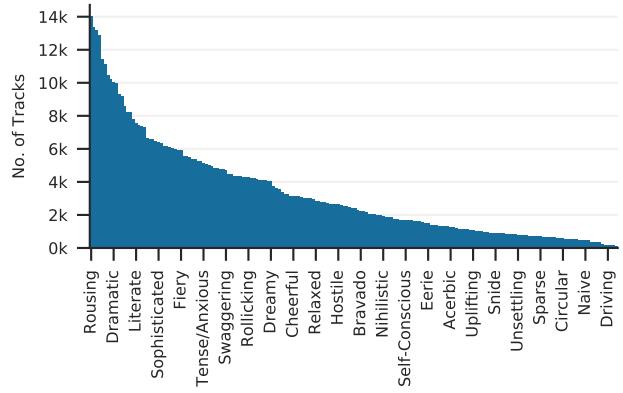
2.2 Current Approaches

The current state of the art largely approaches music mood prediction via audio analysis. Early approaches identified spectral contrast as an informative representation [19], and a number of other authors confirmed this finding as well as a variety of other standard audio features [20, 31, 33, 39]. While the relationship between mood and spectral representations remains non-obvious, previous work has shown that human subjects annotate reconstructions from these representations with reasonable consistency to their original form [35]. Still, the problem remained far from solved.
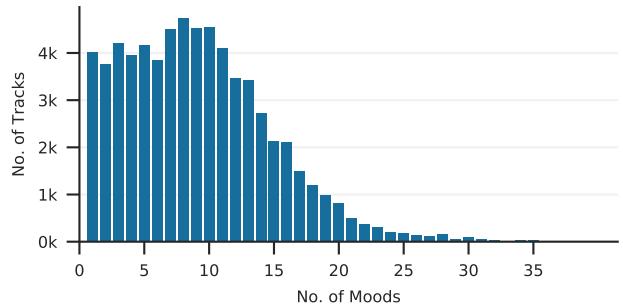
In moving towards increasing model complexity, most approaches have incorporated deep learning methods that seek to learn their own representations [34]. In addition to prediction, audio-based approaches have also been extended to the problem of segmentation [1]. More recent approaches have expanded to multi-modal representations by combining lyrics [8] and others have focused on interpretability of these complex models [6]. At the time of writing, the authors are not aware of any models which leverage features derived from user interactions to estimate the moods of a music track.

3. DATA

The data we collected for this work are derived from various sources: AllMusic provides mood annotations; The

Figure 1: Number of tracks per annotated mood in the AMS. Due to space limitations, only the names of a subset of mood tags are shown.

Figure 2: Number of moods annotated per tracks in AMS.

Echo Nest Taste Profile [25], mapped to tracks in the Million Song Dataset, adds listening data; finally, 7-digital contributes 30s previews as audio data.

We link AllMusic data to the MSD by fuzzy string matching of artist and track names, and requiring track lengths to be within ± 10s. This results in a dataset of 66 993 matched tracks in total, which we call the AllMusic Mood Subset (AMS). As opposed to other music tagging datasets, such as the LastFM Set [4, 15, 17], AMS provides a large vocabulary of mood tags annotated by music experts. While the AllMusic annotations are proprietary, they can be freely consulted on their website and, moreover, are available to be licensed.

Finally, we randomly split the AMS into 80% training, 10% validation, and 10% test, resulting in 53 585, 6695, 6713 tracks respectively. The splits are available online [3] to ensure comparability of future results.

3.1 Mood Data

The mood information that we employ in this work has been human-annotated by experts from AllMusic. These data were previously employed for mood classification [3, 16] and lyrics sentiment detection [24]. The mood tags are annotated at an album level, and we unfold them such that each track is assigned its album-level moods.

The total number of mood tags available is 188. As previous work noted [16], many tags may describe similar

---

Top	Count	Bottom	Count
Rousing	14 018	Melodic	95
Reflective	13 330	Animated	140
Energetic	13 153	Powerful	148
Earnest	12 873	Driving	163
Passionate	11 438	Introspective	176
Confident	11 092	Flowing	218
Amiable	10 424	Positive	307
Intimate	10 188	Stately	310
Dramatic	10 014	Giddy.	315
Playful	9952	Thoughtful	340

Table 1: 10 top and bottom mood tags based on the number of tracks they have been annotated in the AMS.

Figure 3: Consistency ratios for the top 25 most popular moods for each user in the AMS.

moods (such as "Romantic" and "Sensual"), which tend to co-occur, and can be clustered into a smaller number of groups. While we can confirm this by performing manual and/or data-driven explorations on the co-occurrence matrix, we intentionally kept the original annotations. For one, we expect modern machine learning methods to cope with large and possibly overlapping vocabularies. For another, these tags were curated by expert annotators to specifically describe how music feels; while they might characterize similar concepts, they could also provide a more nuanced view of a song's mood.

To give a better notion of the moods in this dataset, in Figure 1 we depict the histogram of number of tracks per mood tag, which follows a typical long-tail distribution. The 10 top and bottom annotated mood tags can be seen in Table 1. As we can see, "Rousing" is the most frequent mood, which appears in 14 018 tracks. On the other hand, "Melodic" is the least frequent one, associated with only 95 tracks. On average across the dataset, there are 3258.6±2961.3 tracks for each tag, with a median of 2385. Furthermore, Figure 2 shows the distribution of number of mood tags per track. It can be seen that most tracks have 13 moods or less, with an average of 9.1±5.7 tags per track and the median centered at 9.
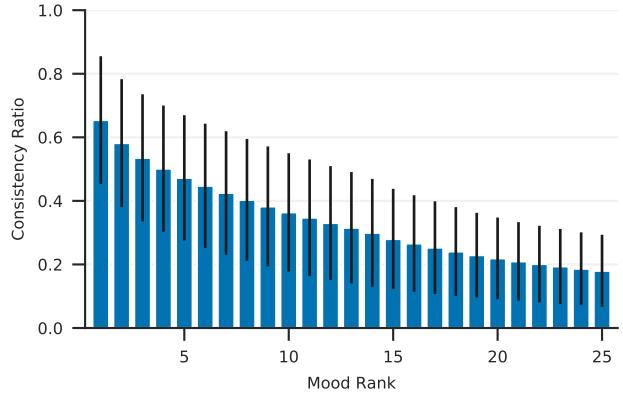
3.2 Audio Data

Since the AMS is a subset of the MSD, we gather the audio data by obtaining the 7-digital 30 second previews associated with all MSD tracks. These are 128kbps mp3 stereo files sampled at 44.1kHz.

3.3 Listening Data

We make use of the Taste Profile from the MSD to obtain listening data. These data contain over 28 million play counts from undisclosed partners associated with $L = 1\,019\,318$ listeners and $S = 384\,546$ tracks.

We motivate the usage of such data in the context of mood classification by showing the relationship between listening habits and the moods of the tracks played, thus arguing that such embeddings are likely to contain relevant data when predicting moods. By mapping the tracks in this set with the moods from the AMS (and thus reducing the

set of listeners down to 1 012 825, the track set down to 66 993, and the play counts down to ∼9 million) we observe that listeners play music that tends to be consistent in terms of its mood. We define the consistency ratio of a mood as the fraction of times it appears in the listening history of a given user. Figure 3 shows the consistency ratio of the nth most popular moods aggregated across users. More specifically, 65.4% of all plays by a given user contain the most popular mood tag for that user; similarly, around 50.1% of a user's plays are annotated with their 4th most popular mood; etc. This exhibits the potential benefits of using listening data, as we confirm in the results of our experiments described next.

4. EXPERIMENTS

As described in Section 2.1, we treat mood prediction as a multi-label classification problem, with a function f predicting mood tags $\hat{y}$ from an input embedding x. The input embedding can stem from different sources, such as listening- or audio-based features; we will refer to this as the *embedding type*. We will use mostly open models trained on publicly available datasets in this work. As we will see, our conclusions follow from these results alone. Furthermore, we will show results for proprietary models trained on in-house listener feedback data. While we acknowledge that these additional results are hardly reproducible without access to our data and methods, they demonstrate how our findings translate to an industrial scale, and are thus a meaningful addition to this work.

4.1 Evaluation Metrics

Our goal is to compare the predictive performance of each embedding type, *i.e.*, given an input embedding of a certain type, how well the predicted moods $\hat{y}$ resemble the true moods y associated with a track. To quantify this, we will use macro-averaged *average precision* as the main evaluation metric, as is commonly used in multi-label classification. Average precision summarizes the precision-recall curve in a single number, and is defined as

$$AP = \sum_n (R_n - R_{n-1}) \cdot P_n, \qquad (1)$$

where R_n and P_n are the recall and precision at the nth threshold at which the recall changes. Note that we use *macro averaging*—we first compute AP for each mood tag, and then average them to calculate the final result.

In our mood prediction setup, there are two main questions we need to consider: how do we arrive at the input embedding $\mathbf{x}$, and how we model and train f. Let us first explore the various embedding types, before we take a detailed look at f.

4.2 Audio-Based Models

Current mood prediction systems typically use audio-based features as input. In this work, we use several audio models, pre-trained on different datasets with varying sizes. This ensures that our results are not specific to a type of model.

4.2.1 Musicnn

We employ Musicnn [28]—a spectrogram-based convolutional neural network (CNN) for audio tagging—as the main pre-trained audio-based baseline. It is openly available[4] and achieves state-of-the-art results. We compare two variants of this model: a smaller one, trained on ~19k tracks from the MagnaTagATune dataset [22], which we will refer to as *MCN-MTT-A*; and a larger one, trained on ~200k tracks from the Million Song dataset, which we will name *MCN-MSD-A*. Both variants come pre-trained to predict 50 tags, a subset of which can be associated with moods. We refer to Musicnn's documentation for further details on its training scheme.

Musicnn is trained to predict tags for 3-second snippets of audio; however, our setup requires a single *embedding per track*. Thus, instead of the final output, we extract the activation of the penultimate layer of the model as embedding. We first compute embeddings of consecutive non-overlapping audio snippets of 3 seconds, and then average all snippet embeddings to form the track-level embedding. This results in a 200-dimensional vector for MCN-MTT-A, and a 500-dimensional vector for MCN-MSD-A. Such global averaging operations are common for music tagging [27].

4.2.2 Short-Chunk CNN

We train a short-chunk CNN [26] from scratch on the 54k training tracks in the AMS. This simple but powerful model feeds a Mel-spectrogram through a 7-layer CNN with 3×3 filters, 2×2 max-pooling layers, and a fully connected layer before the output. For a detailed look into the training regime and architecture, we refer to the original paper.

Since this model was trained directly for mood prediction on the AMS, there is no need for transfer learning as described in Section 4.4. This is a double-edged sword: although the model is focused on the task at hand, it has to learn a large vocabulary of tags from the limited data provided by our dataset. We will refer to this model as *SCC-A*

---

[4] https://github.com/jordipons/musicnn

4.3 Listening-Based Models

In contrast to audio-based models, listening-based ones consider user-song interaction as source data. This *listening data* comes in the form of a sparse feedback matrix $Y \in \mathbb{N}^{L \times S}$, where $y_{l,s}$ is a cell in Y representing the number of times the listener l has either played or rated the song s. The former is called *implicit* feedback, while the latter is referred to as *explicit* feedback. Factorizing Y using factorization rank E (corresponding to the desired embedding dimensionality) yields dense track embeddings $\mathbf{x} \in \mathbb{R}^E$: the input to our mood prediction model.

4.3.1 Taste-Profile Factorization

We use listening data from the complete Taste Profile of 28M play counts to obtain song embeddings by applying weighted matrix factorization using alternating least squares [18] with a rank of $E = 200$ (chosen empirically). These data contain relevant information about the track defined exclusively with implicit feedback: how many times which listeners have listened to which songs. We will call these embeddings *TP-L*.

4.3.2 Proprietary Factorization

Large music streaming services possess much larger and more detailed listening data than openly available resources. To see how the results on open datasets translate to industrial settings, we derive 200-dimensional embeddings from more than 100B in-house explicit user ratings over the whole music catalog, by applying a weighted matrix factorization algorithm. These embeddings will be referred to as *P-L*.

4.4 Transfer Learning
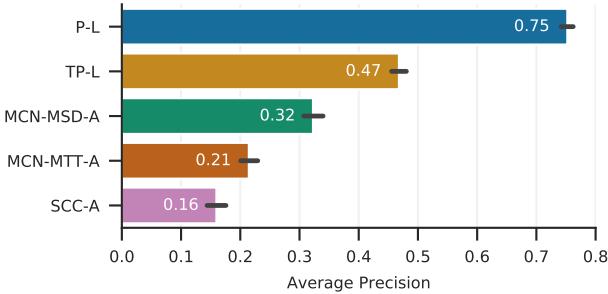
Having computed track-level embeddings $\mathbf{x}$ from various sources, we need to map them to mood tags using a learnable function f. This is a transfer-learning scenario: the input embeddings are obtained from a model trained to solve a different (but related) task, such as collaborative filtering or general audio tagging, and then applied for mood prediction by learning f.

We model f as a multi-layer perceptron (MLP) with a binary indicator vector as output, such that $\hat{\mathbf{y}} = f(\mathbf{x})$, where $\hat{\mathbf{y}} \in [0,1]^{|\mathcal{T}|}$. Thresholding $\hat{\mathbf{y}}$ gives us the set of predicted moods. We train f for each embedding type by minimizing the binary cross-entropy between predicted vectors $\hat{\mathbf{y}}$ and target vectors $\mathbf{y}$ obtained from the true mood tags.

The performance of MLPs heavily depends on the choice of hyper-parameters. To enable a fair comparison, we optimized hyper-parameters for each embedding type individually using Bayesian optimization [36], monitoring average precision on the validation set. To limit the computational cost, we only used TP-L and MCN-MSD-A as input embeddings, since they are the main points of comparison. Each setup enjoyed the same, fixed computational budget of 2 days on a single Tesla M40 GPU, which translates to around 200 trials per setup. Table 2 shows details on the search space and the best found configurations. We

	Domain	TP-L	MCN-MSD-A
Nº layers	[2..4]	4	4
Nº units	[1500..4000]	3909	3933
learning rate	[0.0001, 0.005]	4×10^{-4}	5×10^{-4}
dropout [38]	[0, 0.5]	0.25	0.25
weight decay	[0, 0.0001]	0	1×10^{-6}

Table 2: Hyper-parameters optimized with Bayesian optimization, and best found configurations for each embedding type. Search ranges were defined based on limited initial experiments. For dropout and weight decay, we quantized the interval by 0.125 and 1×10^{-6}, respectively.

Figure 4: Overall results for each model.

see that for both embedding types, the best models reach the upper limit of our search space, which indicates that even larger models might lead to better results. However, we saw diminishing improvements for large models, so we do not expect much further improvement.

We initialize the MLP weights using Kaiming's method [14], and use a rectifier activation function [13] after each layer (the output layer uses a sigmoid). The input is standardized using mean and standard deviation estimated on the training set. We then train f for 100 epochs using a cosine-annealed learning rate [23] (without restarts) and a 1-epoch warm-up phase. During training, we monitor average precision on the validation set to select the best performing model parameters.
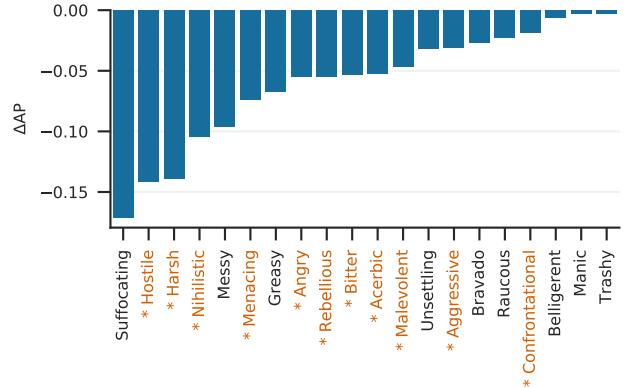
The code to reproduce these experiments is available online. [5]

5. RESULTS
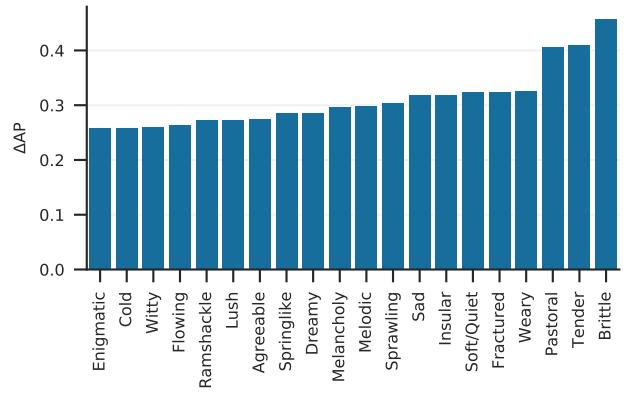
Figure 4 shows the overall results of each embedding type. As mentioned before, our main analysis will be based on the results of open models on publicly available data. We will discuss the results of P-L later.

We see that listening-based embeddings easily outperform audio-based ones (TP-L vs. MCN-MSD-A). We also see a variation within audio-based models. Our experiments were not designed to explain this variation, and the usual suspects offer insufficient clues: for example, dataset size might be an issue (200k for MCN-MSD-A vs. 19k for MCN-MTT-A), but SCC-A was trained on the 54k training tracks from AMS with worse results—here, dataset size relative to vocabulary size might have been the issue. Further experiments, out of scope of this paper, are necessary to understand this in depth.

[5] https://github.com/fdlm/listening-moods

(a) Tags favoring audio-based models.

(b) Tags favoring listening-based models.

Figure 5: Difference of average precision between the best audio-based model (MCN-MSD-A), and the best listening-based model on open data (TP-L). Negative ΔAP means the audio-based embedding performed better. The highlighted tags in (a) belong to the same mood cluster.
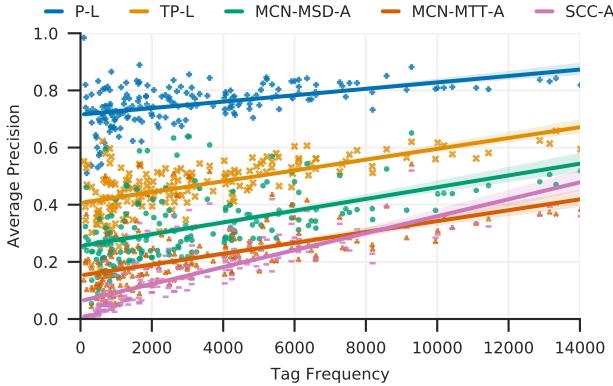
5.1 Tag-Wise Results

Even though the overall results are clear, some tags might be easier to predict from audio than from listening data. To explore this, we subtract the tag-wise average precision of TP-L and MCN-MSD-A, and show the results in Figure 5. Indeed, we find 20 tags for which MCN-MSD-A out-performs TP-L. Moreover, these tags seem to describe related moods. To verify this, we clustered the 188 moods using affinity propagation [12], resulting in 13 clusters. We see that 11 out of the 20 mood tags belong to the same cluster, as highlighted in Figure 5a. In contrast, the tags in Figure 5b come from a wider variety of clusters (not highlighted). This indicates that it is a single, coherent "mood subspace" on which audio data is better suited.

5.2 Results by Tag Frequency

As shown earlier, mood tags in the AMS are unevenly distributed: the least popular tag counts only 95 annotations, while the most popular track 14k. It is reasonable to assume that uncommon tags are more difficult to predict than common ones. To evaluate this, we plot the average precision per tag depending on the tag frequency in Figure 6.

Although we see a direct relation between tag frequency and average precision, the extent is less than we expected.

Figure 6: Results per tag frequency. Dots represent the average precision obtained by a tag, that occurs at a frequency shown on the x axis. Lines represent linear regression models, with shades indicating 95% confidence intervals.

Figure 7: Correlation between tag-wise results of different embeddings. We see that audio-based ones correlate strongly with each other, compared to weaker correlations between listening-based ones.

Furthermore, all embedding types seem to be equally affected: with the exception of SCC-A, the regression slopes of both audio- and listening-based models are notably similar. The exception of SCC-A indicates that tag sparsity may be an issue when training audio models from scratch, but not so when transfer-learning a model that has been trained on more balanced data.

5.3 Results of Proprietary Algorithms

So far, we have discussed the results of open methods on publicly available datasets. However, the attentive reader has noticed that Figure 4 and 6 demonstrate how P-L performs even better than TP-L. To explain the gap between TP-L and P-L, we can point to the different nature and amount of data they were trained on—28M implicit plays for the former, but more than 100B explicit ratings for the latter. The sheer amount of data (a factor of ∼3500) and the stronger signal provided by explicit feedback seem to be remarkably beneficial.

5.4 Consistency of Audio-Based Models

We have shown that listening-based models clearly outperform audio-based models in mood prediction. To demonstrate this, we selected a wide variety of audio models that differed in multiple aspects: network architecture, training datasets, and training regime (pre-trained and trained from scratch). Given these differences, we can ask if there are aspects of mood that current audio models are not capable to capture, but listening-based models can. We try to answer this question by exploring which embeddings capture similar mood information. If an embedding captures similar aspects of mood as another embedding, their tag-wise performance should be correlated—but not necessarily similar in magnitude, as one embedding might just perform better than the other.

We show the correlation in tag-wise performance in Figure 7. The remarkable result is that regardless of their differences, the tag-wise results of all audio-based models are much more correlated than between audio- and

listening-based embeddings. This indicates that audio-based models do capture similar aspects, even if they might not capture it equally well (as the difference between MCN-MSD-A and SCC-A shows). This does not mean that the aspects current audio-based models are missing are not present in the audio at all—just that current models are not able to extract them.

We do not observe a similar pattern for listening-based embeddings: TP-L and P-L show weaker correlation. At this time, we cannot provide a better explanation than referring to the different nature of explicit and implicit feedback data and the sizes of the two datasets.

6. CONCLUSIONS

In this work we have associated 66 993 tracks from the Million Song Dataset with the AllMusic set to yield the AMS, the largest dataset available with the following data modalities: high quality human mood annotations, audio content, and listening data. Furthermore, we have shown how listening data surpass audio-based embeddings when classifying moods in the proposed dataset. The notable differences in performance between listening- and audio-based models suggest that either i) current state-of-the-art audio models are not capable of successfully extracting certain mood information about a given track; and/or ii) such mood information is not necessarily present in the audio content, and thus the usage of other signals such as listening information may be required to obtain more accurate results. With these findings, we encourage researchers to employ data beyond audio content when estimating the mood of a track. In the future, we look to further scrutinize the tags to better understand which moods might be more suitable to be extracted by which type of input representation. Moreover, and along these lines, we would like to address this task in a multi-modal manner, combining different sources to potentially improve performance of this compelling and timely problem.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] Anna Aljanaki, Frans Wiering, and Remco C. Veltkamp. Emotion Based Segmentation of Musical Audio. In *Proc. of the 16th Conference of the International Society for Music Information Retrieval (ISMIR)*, Málaga, Spain, October 2015.

[2] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Anssi Klapuri and Colby Leider, editors, *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, October 2011.

[3] Kerstin Bischoff, Claudiu S. Firan, Raluca Paiu, Wolfgang Nejdl, Cyril Laurier, and Mohamed Sordo. Music Mood and Theme Classification - a Hybrid Approach. In *Proc. of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, October 2009.

[4] Erion Çano and Maurizio Morisio. Music Mood Dataset Creation Based on Last FM Tags. In *Proc. of the 4th International Conference on Artificial Intelligence and Applications (AIAP)*, Vienna, Austria, May 2017.

[5] Jeong Choi, Jongpil Lee, Jiyoung Park, and Juhan Nam. Zero-shot Learning for Audio-based Music Classification and Tagging. In *Proc. of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, November 2019.

[6] Shreyan Chowdhury, Andreu Vall, Verena Haunschmid, and Gerhard Widmer. Towards Explainable Music Emotion Recognition: The Route via Mid-level Features. In *Proc. of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, November 2019.

[7] Paul T. Costa and Robert R. McCrae. *Revised NEO Personality Inventory (NEO PI-R) and NEO Five-Factor Inventory (NEO-FFI): Professional Manual*. Psychological Assessment Resources, 1992.

[8] Rémi Delbouys, Romain Hennequin, Francesco Piccoli, Jimena Royo-Letelier, and Manuel Moussallam. Music Mood Detection Based on Audio and Lyrics with Deep Neural Net. In *Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, September 2018.

[9] Shuiguang Deng, Dongjing Wang, Xitong Li, and Guandong Xu. Exploring user emotion in microblogs for music recommendation. *Expert Systems with Applications*, 42(23):9284–9293, 2015.

[10] Ignacio Fernández-Tobías, Matthias Braunhofer, Mehdi Elahi, Francesco Ricci, and Iván Cantador. Alleviating the new user problem in collaborative filtering by exploiting personality information. *User Modeling and User-Adapted Interaction*, 26(2-3):221–255, 2016.

[11] Bruce Ferwerda, Marko Tkalcic, and Markus Schedl. Personality Traits and Music Genres: What Do People Prefer to Listen To? In *Proc. of the 25th Conference on User Modeling, Adaptation, and Personalization*, Bratislava, Slovakia, July 2017.

[12] Brendan J. Frey and Delbert Dueck. Clustering by Passing Messages Between Data Points. *Science*, 315(5814):972–976, February 2007.

[13] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *Proc. of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, USA, June 2011.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, December 2015.

[15] Xiao Hu and J Downie. Improving mood classification in music digital libraries by combining lyrics and audio. In *Proc. of the 10th ACM International Conference on Digital Libraries (JCDL)*, Surfer's Paradise, Australia, June 2010.

[16] Xiao Hu and J. Stephen Downie. Exploring mood metadata: Relationships with genre, artist and usage metadata. In *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, September 2007.

[17] Xiao Hu and J. Stephen Downie. When lyrics outperform audio for music mood classification: A feature analysis. In *Proc. of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, August 2010.

[18] Yifan Hu, Chris Volinsky, and Yehuda Koren. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, Pisa, Italy, December 2008.

[19] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, Lausanne, Switzerland, August 2002.

[20] Youngmoo E. Kim, Erik M. Schmidt, Raymond Migneco, Brandon G. Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A. Speck, and Douglas Turnbull.

Music emotion recognition: A state of the art review. In *Proc. of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, August 2010.

[21] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):42–49, 2009.

[22] Edith Law, Kris West, Michael Mandel, Mert Bay, and J. Stephen Downie. Evaluation of algorithms using games: The case of music tagging. In *Proc. of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, October 2009.

[23] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *5th International Conference on Learning Representations (ICLR)*, Toulon, France, April 2017.

[24] Ricardo Malheiro, Renato Panda, Paulo Gomes, and Rui Pedro Paiva. Classification and regression of music lyrics: Emotionally-significant features. In *Proc. of the 8th International Conference on Knowledge Discovery and Information Retrieval (KDIR)*, Porto, Portugal, November 2016.

[25] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. The Million Song Dataset Challenge. In *Proc. of the 21st International Conference on World Wide Web (WWW)*, Lyon, France, April 2012.

[26] Minz Won, Andres Ferraro, Dmitry Bogdanov, and Xavier Serra. Evaluation of CNN-based Automatic Music Tagging Models. In *Proc. of the Sound and Music Computing Conference (SMC)*, Torino, Italy, June 2020.

[27] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik Schmidt, Andreas Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *Proc. of the 19th International Society for Music Information Retrieval Conference*, Paris, France, September 2018.

[28] Jordi Pons and Xavier Serra. MusiCNN: Pre-trained convolutional neural networks for music audio tagging. In *Late Breaking of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, November 2019.

[29] Peter J. Rentfrow and Samuel D. Gosling. The do re mi's of everyday life: The structure and personality correlates of music preferences. *Journal of Personality and Social Psychology*, 84(6):1236–1256, 2003.

[30] James A Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178, 1980.

[31] Markus Schedl, Emilia Gomez, Erika S. Trent, Marko Tkalcic, Hamid Eghbal-Zadeh, and Agustin Martorell. On the interrelation between listener characteristics and the perception of emotions in classical orchestra music. *IEEE Transactions on Affective Computing*, 9(4):507–525, 2018.

[32] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, 2018.

[33] Erik M. Schmidt and Youngmoo E. Kim. Modeling Musical Emotion Dynamics with Conditional Random Fields. In *Proc. of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, October 2011.

[34] Erik M. Schmidt and Youngmoo E. Kim. Learning Rhythm and Melody Features with Deep Belief Networks. In *Proc. of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, November 2013.

[35] Erik M. Schmidt, Matthew Prockup, Jeffrey Scott, Brian Dolhansky, Brandon G. Morton, and Youngmoo E. Kim. Relating Perceptual and Feature Space Invariances in Music Emotion Recognition. In *9th Int. Symp. Computer Music Modeling and Retrieval (CMMR)*, London, UK, June 2012.

[36] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, Lake Tahoe, USA, December 2012.

[37] Mohammad Soleymani, Michael N. Caro, Erik M. Schmidt, Cheng Ya Sha, and Yi Hsuan Yang. 1000 Songs for Emotional Analysis of Music. In *Proc. of the 2nd ACM International Workshop on Crowdsourcing for Multimedia (CrowdMM)*, Barcelona, Spain, October 2013.

[38] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[39] Ju-Chiang Wang, Yi-Hsuan Yang, Hsin-Min Wang, and Shyh-Kang Jeng. Modeling the Affective Content of Music with a Gaussian Mixture Model. *IEEE Transactions on Affective Computing*, 6(1):56–68, 2015.

[40] Hiromu Yakura, Tomoyasu Nakano, and Masataka Goto. FocusMusicRecommender: A system for recommending music to listen to while working. In *Proc. of the 23rd International Conference on Intelligent User Interfaces (IUI)*, Tokyo, Japan, March 2018.

[41] E. Zangerle, C. Chen, M. Tsai, and Y. Yang. Leveraging affective hashtags for ranking music recommendations. *IEEE Transactions on Affective Computing (Early-Access)*, 2018.

ULTRA-LIGHT DEEP MIR BY TRIMMING LOTTERY TICKETS

Philippe Esling, Theis Bazin, Adrien Bitton, Tristan Carsault, Ninon Devis

IRCAM - Sorbonne Université, CNRS UMR 9912 - 1, place Igor Stravinsky, Paris, France

`esling@ircam.fr`

ABSTRACT

Current state-of-the-art results in Music Information Retrieval are largely dominated by deep learning approaches. These provide unprecedented accuracy across all tasks. However, the consistently overlooked downside of these models is their stunningly massive complexity, which seems concomitantly crucial to their success.
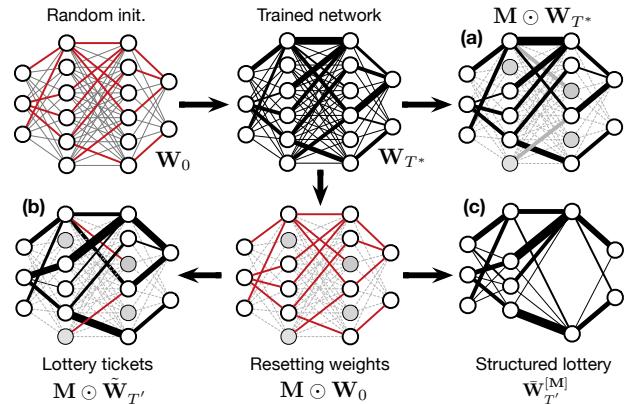
In this paper, we address this issue by proposing a model pruning method based on the *lottery ticket* hypothesis. We modify the original approach to allow for explicitly removing parameters, through *structured trimming* of entire units, instead of simply masking individual weights. This leads to models which are effectively lighter in terms of size, memory and number of operations.

We show that our proposal can remove up to 90% of the model parameters without loss of accuracy, leading to ultra-light deep MIR models. We confirm the surprising result that, at smaller compression ratios (removing up to 85% of a network), lighter models consistently outperform their heavier counterparts. We exhibit these results on a large array of MIR tasks including *audio classification*, *pitch recognition*, *chord extraction*, *drum transcription* and *onset estimation*. The resulting ultra-light deep learning models for MIR can run on CPU, and can even fit on embedded devices with minimal degradation of accuracy. [1]

1. INTRODUCTION

Over the past decades, Music Information Retrieval (MIR) has witnessed a growing interest, with a wide variety of tasks such as *genre classification*, *chord extraction* and *music recommendation* [1] being increasingly implemented in end-user products. Recently, MIR has predominantly improved with machine learning, and almost all state-of-art accuracies are obtained by deep learning models [2]. Although these approaches provide unprecedented results, the major issue in modern deep learning lies in the tremendous complexity and immense size of the models employed. Indeed, deep networks for images can reach up

---

[1] Supplementary results and code to reproduce experiments are available at https://github.com/acids-ircam/lottery_mir

Figure 1. Comparing (a) traditional pruning with (b) the lottery ticket hypothesis, and (c) our structured lottery approach to obtain ultra-light deep networks.

to billions of parameters and new leaps in accuracy seem to only come by worsening this situation. As a showering example of this complexity [3], the inference on a single image in the pervasive ResNet model [4] requires 7.7 GFLOPS [2] . This exploding size leads to profound issues in both the use and understanding of these models. As they are extremely demanding in computation and memory, it precludes their implementation in end-user embedded systems which prevails in audio applications, and also raises some serious environmental issues. Finally, such complexity decreases the potential interpretability of these models.

The idea of eliminating unnecessary weights (*pruning*) was proposed early for neural networks [5]. Most methods are based on masking the smallest-amplitude weights from a large network, as depicted in Figure 1. Other approaches such as *quantization* [6] or *knowledge distillation* [7] have been proposed to decrease the size and energy consumption of trained models with equivalent accuracy. However, keeping the original accuracy of complex large models seems only possible at low compression rates [8]. Furthermore, recent benchmark studies [9] pointed out that most of the proposed methods seems to achieve a similar efficiency in both accuracy and model size.

Recently, the *lottery ticket hypothesis* [10] suggested that randomly-initialized neural networks already contain powerful subnetworks (called *winning tickets*) that could reach the same or higher accuracy than the original networks if they were trained in isolation. Hence, by finding these subnetworks, we could drastically prune most of the

---

[2] FLOPS: floating point operations

weights in large networks and still obtain the same level of accuracy. This implies that the same task could be solved in a very lightweight, memory and energy-efficient way. Furthermore, these subnetworks could be easier to analyze, which could simplify further works towards explainability [11]. Several studies have analyzed different properties of this hypothesis [12–14], as it raises the exciting prospect to obtain much smaller networks that provide a similar accuracy compared to the typically larger state-of-art models. However, this method has two major flaws. First, it has a large training cost, as finding winning tickets seems to only be stable when the training is repeated multiple times over iteratively smaller networks [15]. Second, pruning is done by *masking* the weights, which means that the resulting networks retain the size and computation cost of the original ones, even if most of the weights are unused.

In this paper, we extend the lottery approach to effectively remove weights, obtaining models with a lower size and inference time, while still maintaining a commensurate accuracy. To do so, we introduce a method based on the lottery ticket hypothesis, and we replace the masking operation with a structured pruning operation (termed *trimming* here). The original network capacity is reduced by removing entire computation units (or convolutional channels). This alleviates issues of the original lottery ticket method as, although we still need to repeat the training, it becomes faster at each iteration. We discuss different criteria for selecting the units and their differences to the original lottery ticket hypothesis. Notably, unstructured masking allows to work on local connectivity patterns, whereas trimming can only impact this aspect if we perform *global* selection (ranking units across the network). We show that this approach can be successfully applied across MIR tasks, leading to *ultra-light* deep MIR models. We evaluate the efficiency of replacing the masking operation by our trimming criterion and show that we still obtain commensurate accuracy when removing up to 90% of the model parameters. We also maintain the surprising result [10] that lighter models (removing up to 85% of the network) obtain higher accuracy, while we effectively reduce the model size. We evaluate these results on a large array of MIR tasks including *instrument* [16] and *singing voice classification* [17], *pitch recognition* [18], *automatic chord extraction* [19], *drum transcription* [20] and *onset estimation* [21].

2. STATE-OF-ART

2.1 Model compression and pruning

Various approaches have been proposed for reducing the size of neural network models, while trying to maintain accuracy [5]. These approaches can be globally divided between *pruning* or *compressing* networks. We group in the *compression* category the *distillation* [7] (training a smaller model to fit the internal representations of a larger one) and *quantization* [6] (reducing the size of networks by using lower-resolution weights or binary numbers) approaches. Here, we focus on *pruning*, but note that compression and quantization can be further applied on pruned models.

The goal of *pruning* [5] is to identify and remove weights of a network that are not critical to its accuracy. The original approach to pruning starts by fitting a large and overparametrized network to completion. Then, we aim to mask the less relevant weights in this trained model based on a given selection method. This criterion tries to analyze the usefulness of different parameters, commonly based on their magnitude [22]. Finally, the resulting masked network is *fine-tuned*, trying to restore the accuracy of the original network [9]. Hence, the critical aspect in this approach lies in the method of weight selection. This criterion can perform either a *structured* or *unstructured* and *local* or *global* selection. *Unstructured* pruning acts on individual parameters separately, *structured* pruning aims to effectively remove parts of the networks. Hence, unstructured methods are mostly based on *masking* the weights based on their magnitude [5, 22]. Oppositely, structured pruning aims to remove entire *hidden units* or *convolutional channels* from a network [8, 23].

However, recent studies showed that most pruning methods are mostly equivalent [8]. These approaches usually lead to smaller accuracy than the large network and at low pruning rates, with performance degrading with the amount of weights removed [9], although some are able to maintain (but not outperform) the original accuracy [8].

2.2 Lottery ticket hypothesis

The recently proposed *lottery ticket hypothesis* [10] states that inside a randomly-initialized network, there already exist some considerably smaller subnetworks which would be extremely efficient if trained in isolation. Hence, parts of the weights drawn by random initialization before training already provide a specific topology and parameter configuration that make training particularly effective. The major difference between this approach and the previous magnitude-based selection from which it is inspired [22] is that the selected weights are *reset* to their initialization value before retraining the smaller architecture. Doing so, very small subnetworks (less than 1% of the original network size) could be found across several architectures, even outperforming the larger networks at smaller pruning ratios. For deeper architectures, they further showed [12] that winning tickets should be *rewound* to a given iteration, rather than to initialization values. Interestingly, this seems to confirm that overparameterization is needed to find an optimal solution, but that a lighter solution exists, which is optimized in the compression phase of the training [13,24].

2.2.1 Formalization

We consider a network as a parametric function $f(\boldsymbol{x}; \boldsymbol{W})$, with a set of weights $\boldsymbol{W} \in \mathbb{R}^D$ that are first initialized through sampling $\boldsymbol{W}_0 \sim p(\boldsymbol{W})$. The weights are updated by using a training *algorithm* $\mathcal{A}(i, \boldsymbol{W}_0)$ which maps initial weights $\boldsymbol{W}_0$ to weights $\boldsymbol{W}_i$ at iteration $i \in \{1, .., T\}$, by performing successive updates similar to

$$\boldsymbol{W}_{i+1} = \boldsymbol{W}_i - \eta \nabla_{\boldsymbol{W}} \mathcal{L} \tag{1}$$

with a given loss function $\mathcal{L}$ and learning rate η.

A *subnetwork* of the original network $f(\boldsymbol{x}; \boldsymbol{W})$ can be defined as a tuple $(\boldsymbol{W}, \boldsymbol{M})$ of the original weights $\boldsymbol{W} \in \mathbb{R}^D$ and a mask $\mathbf{M} : \{0, 1\}^D$. Hence, the subnetwork computes the function $f(\boldsymbol{x}; \boldsymbol{M} \odot \boldsymbol{W})$, where $\odot$ denotes the element-wise product.

Lottery Ticket Hypothesis. Given a randomly initialized network $f(\boldsymbol{x}; \boldsymbol{W}_0)$ with $\boldsymbol{W}_0 \sim p(\boldsymbol{W})$, that is trained to reach accuracy a^* in T^* iterations, with final weights $\boldsymbol{W}_{T^*}$, there exists a subnetwork $(\boldsymbol{W}_k, \boldsymbol{M})$ with a given mask $\boldsymbol{M} \in \{0, 1\}^{|\boldsymbol{W}|}$ and iteration $k \ll T^*$, such that if we retrain this subnetwork, it will reach a *commensurate accuracy* $a \geq a^*$ in *commensurate iterations* $T \leq T^* - k$ and *fewer parameters* $\|\boldsymbol{M}\|_0 \ll |\boldsymbol{W}|$.

These highly efficient subnetworks (called *winning tickets*) depend on the original initialization, and can only be identified after full training [12]. Thus, the selected weights and remaining topology form the architecture of the winning ticket. These weights are *reset* to their initialization values *before* the network was trained or *rewound* at an early iteration. The resulting architecture is then retrained until completion, and the whole process is repeated, as described in Algorithm 1.

Algorithm 1 Lottery ticket training with rewinding

1: $\boldsymbol{W}_0 \sim p(\boldsymbol{W})$ ▷ Random initialization
2: $\boldsymbol{M} = \mathbf{1}_{|\boldsymbol{W}|}$ ▷ Initial mask
3: $\boldsymbol{W}_k = \mathcal{A}(k, \boldsymbol{W}_0 \odot \boldsymbol{M})$ ▷ Training for k iterations
4: **while** $\mathcal{C}(\boldsymbol{M}, a, \boldsymbol{W})$ **do**   ▷ Stopping criterion $\mathcal{C}$
5:    $\boldsymbol{W}_T = \mathcal{A}(T, \boldsymbol{W}_k \odot \boldsymbol{M})$ ▷ Train until completion
6:    $r = \mathcal{R}(\{\boldsymbol{W}_{T^*}\})$ ▷ Ranking criterion $\mathcal{R}$
7:    $\boldsymbol{M} = \mathcal{M}(r, \{\boldsymbol{W}_{T^*}\})$ ▷ Masking update $\mathcal{M}$

In their original paper [10], the authors underline the difference between *one-shot* pruning (masking is applied all at once) and *iterative pruning* (repeatedly pruning small parts of the network). They demonstrated that iterative pruning finds smaller architectures that reach higher accuracy than the original network and converge at earlier iterations. They showed on the MNIST dataset, that it was possible to keep the accuracy of large networks, even when masking up to 96.5% of the weights. Their most intriguing result is that smaller networks consistently reach *higher accuracy* than the original ones, even while removing up to 80% of the weights. In a follow-up study [12], they showed that these results could be obtained for deeper architectures, but only through the *rewinding* operation. Another exciting prospect of this hypothesis, is that the resulting subnetworks might encode implicit *inductive biases* for a given task or type of data. In that case, winning tickets could be *transferred* and trained on new tasks, even directly from their extremely lightweight versions [14].

2.3 Music Information Retrieval

Music Information Retrieval (MIR) encompasses all tasks aimed at extracting high-level knowledge from music data. This field has witnessed a flourishing interest, with multiple tasks being increasingly tackled such as *chord extraction*, *drum transcription* and *musical audio classifica-*

tion [1]. Originally, most MIR researches revolved around the idea of extracting a set of hand-crafted features from the signal (such as the Mel-Frequency Cepstral Coefficients), in order to use these as input to machine learning algorithms [25]. Feature-based techniques have been challenged by the advent of deep learning approaches [26], which have shown impressive capacities to learn high-level features on complex data. They simultaneously set new state-of-art results across a wide range of MIR tasks, while opening the path towards unprecedented applications [27].

In this work, we consider a rather broad spectrum of MIR tasks where deep learning approaches are applied. Specifically, we address (i) *audio classification* [17] (finding the class label of audio signals inside a predefined set), (ii) *pitch recognition* [18] (extracting the fundamental frequency of a monophonic audio recording), (iii) *chord extraction* [19] (annotating audio with a given vocabulary of chords), (iv) *onset estimation* [21] (finding events in an audio stream) and (v) *drum transcription* [20] (transforming drums audio signal into a score). We redirect interested readers to [28] for a comprehensive review.

One of the common denominator in deep learning methods applied across all MIR tasks is that their unprecedented accuracy comes at the expense of an increasing size and complexity. Indeed, deep networks for images now reach billions of parameters and leaps in accuracy seem to only come by worsening this situation. An example of this trend in MIR can be seen in the recently proposed CREPE model [18] for pitch extraction. This task was largely handled through the YIN algorithm [29], an extremely simple algorithm, with few parameters and running with very low latency on CPU. For a modest gain in accuracy on the same task, CREPE requires *22 million parameters*, 2.82 GFLOPS and 2.36 seconds on CPU to compute the pitch of a single 4-seconds sample. This exploding size leads to profound issues in both the use and understanding of these models. First, they are extremely demanding in energy consumption and memory, which precludes their implementation in end-user interfaces and also raises serious environmental issues. Furthermore, this complexity stands in the way of any potential interpretability of such models.

3. METHODOLOGY

Here, we first discuss different selection criteria for structured network *trimming*. Then, we discuss different normalization strategies that can allow to perform global selection of units across layers.

3.1 Trimming criteria

In order to perform structured pruning, we need to evaluate the efficiency of *entire units* of computation, rather than individual weights. In the case of convolutional networks, this would amount to analyze the *channels* of each layer. Indeed, channel pruning appears more hardware friendly, and also allows to truly reduce the size of the final model. In the following definitions, we consider that any computation layer can be seen as a weighted transform $f(\boldsymbol{x}, \boldsymbol{W}^{(l)})$,

with a matrix $\boldsymbol{W}^{(l)} \in \mathbb{R}^{n_{out} \times n_{in}}$. Note that we intentionally simplify the notation for more complex layers (*convolutional* or *recurrent*), which embed more complicated matrices. However, we consider in the following that the selection criteria $\mathcal{C}(\boldsymbol{W}^{(l)})$ is computed across the n_{in} dimensions, and that it should produce a vector of n_{out} dimensions. This vector is used to rank the usefulness of different computation units. Hence, after each training iteration, we replace the masking criterion by directly removing parts of the weight matrix for each layer

$$\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l)}_{[C(\boldsymbol{W}^{(l)}),C(\boldsymbol{W}^{(l-1)})]}. \tag{2}$$

Note that we need to carry the pruning criterion from the preceding layer $C(\boldsymbol{W}^{(l-1)})$ in order to reflect potential changes in the structure of the network. All layers in the network that must maintain a given output dimensionality (such as the last layer) are defined as *unprunable*. Following a similar approach than the lottery ticket hypothesis, the remaining weights in the resulting matrix $\boldsymbol{W}^l$ are *rewound* to their values from an earlier iteration [13].

Magnitude. We define a *magnitude-based* criterion, similar to the original lottery [10]. However, we evaluate the overall magnitude of the weights for a complete unit as

$$\mathcal{C}_{mag}(\boldsymbol{W}^{(l)}_i) = \sum_{j=1}^{N_{in}} \left| W^{(l)}_{i,j} \right|. \tag{3}$$

Activation. We can rely on the activation statistics of each unit to analyze their importance. Hence, akin to the previous criterion, we perform a cumulative forward pass through the network after training the model and compute

$$\mathcal{C}_{act}(\boldsymbol{W}^{(l)}_i) = \sum_{k=1}^{\mathcal{D}_v} \left| f(\boldsymbol{x}_k, \boldsymbol{W}^{(l)})_i \right| \tag{4}$$

where we sum across examples of the validation set $\mathcal{D}_v$.

Normalization. An interesting direction proposed in [15] is to consider the *scaling* factor γ in batch normalization layers to evaluate the significance of each layer output. In this criteria, we rely on this *scaling* coefficient as a proxy to the importance of each unit

$$\mathcal{C}_{norm}(\boldsymbol{W}^{(l)}_i) = \left| \gamma^{(l)}_i \right|. \tag{5}$$

Note that this criterion forces each layer to be followed by a normalization layer, from which it can be computed.

4. EXPERIMENTS

We briefly detail the tasks on which we evaluate our method for ultra-light deep MIR. As we address a wide variety of models and datasets, we only provide essential explanations for each. However, unless stated, we follow all implementation details presented in the original papers.

4.1 Tasks

4.1.1 Audio (instrument and voice) classification

Audio classification is one of the seminal and most studied task in MIR [30]. We separate the evaluation into two in-dependent sub-tasks of *singing voice* and *instrument* classification. For both tasks, the model is adapted from the baseline proposed in [17]. The raw input waveform is processed with a stack of 4 dilated 1-dimensional convolutions with batch normalization, ReLU and dropout, followed by 4 fully-connected layers that map to a *softmax*, which outputs a vector of class probabilities. The ground-truth label prediction is optimized with a *cross-entropy* loss. Singing voice classification is performed on mono audio inputs of 3 seconds at 44,100Hz for 10 vocal techniques and a given train/test split ratio [17]. For instrument classification, we rely on the 13 orchestral instruments from URMP [31] and the corresponding recordings from MedleyDB [32]. After silence removal, the combined datasets amount to a total of about 8h30 of isolated instrument recordings. Classification is done on audio inputs of 1.5 seconds at 22,050Hz extracted from the isolated tracks with a single label corresponding to the instrument played.
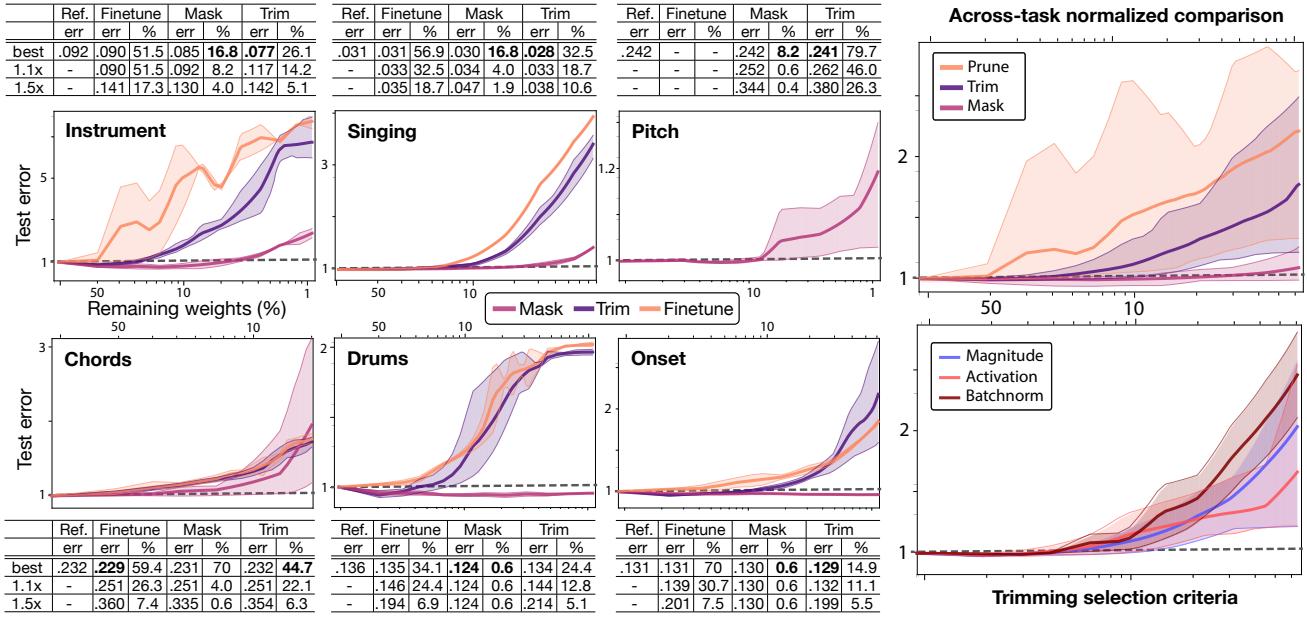
4.1.2 Pitch estimation

The goal of *pitch estimation* is to extract the fundamental frequency of an input audio. For this task, the recently proposed CREPE model [18] requires several large datasets, some of which are not publicly available. However, we only rely here on the open source *NSynth* dataset, which contains single note samples from a range of acoustic and electronic instruments [33]. This leads to 1006 instruments, with different pitches at various velocities available as raw waveforms. All samples last 4 seconds with a sampling-rate of 16kHz. As this incurs an extremely large training time, we use a subsampled dataset, randomly picking 10060 samples (ten notes per instrument). Finally, we trim all samples to their first two seconds to remove silent note tails, ensuring that most inputs to the model are voiced. CREPE is a 6-layer CNN operating directly on waveforms, followed by a single fully connected layer. The model is trained via binary cross-entropy to perform classification over a 360 bin logarithmic frequency scale spanning six octaves from pitch $C1$ to $B7$. The model operates on frames of 1024 samples, which we individually label with the note pitch. We use the *medium* architecture from the CREPE repository [18].

4.1.3 Chord extraction

Automatic chord extraction is defined as labeling segments of an audio signal using an alphabet of musical chords. We perform our experiments based on the model and datasets detailed in [19]. We use the *Beatles* dataset, which contains 180 songs annotated by hand. We rely on a CQT input with hop size 2048, mapped to a scale of 2 bins per semi-tone over 5 octaves starting from $C1$ and containing a total of 105 bins. As input we take 15 successive frames, corresponding to a temporal horizon of approximately 0.7 seconds. We augment the available data by performing all transpositions from -6 to +6 semi-tones. As baseline model, we rely on the CNN architecture described in [19], and evaluate the global accuracy measure.

Instrument

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.092	.090	51.5	.085	**16.8**	**.077**	26.1
1.1x	-	.090	51.5	.092	8.2	.117	14.2
1.5x	-	.141	17.3	.130	4.0	.142	5.1

Singing

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.031	.031	56.9	.030	**16.8**	**.028**	32.5
1.1x	-	.033	32.5	.034	4.0	.033	18.7
1.5x	-	.035	18.7	.047	1.9	.038	10.6

Pitch

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.242	-	-	.242	**8.2**	**.241**	79.7
1.1x	-	-	-	.252	0.6	.262	46.0
1.5x	-	-	-	.344	0.4	.380	26.3

Chords

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.232	**.229**	59.4	.231	70	.232	**44.7**
1.1x	-	.251	26.3	.251	4.0	.251	22.1
1.5x	-	.360	7.4	.335	0.6	.354	6.3

Drums

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.136	.135	34.1	**.124**	**0.6**	.134	24.4
1.1x	-	.146	24.4	.124	0.6	.144	12.8
1.5x	-	.194	6.9	.124	0.6	.214	5.1

Onset

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.131	.131	70	.130	**0.6**	**.129**	14.9
1.1x	-	.139	30.7	.130	0.6	.132	11.1
1.5x	-	.201	7.5	.130	0.6	.199	5.5

Figure 2. Comparing traditional *fine-tuning*, the lottery ticket *masking*, and our structured lottery *trimming* on each MIR task separately for all criteria (left), and across tasks (right, up) and across criteria in trimming (right, down).

4.1.4 Drum transcription

The *drum transcription* task aims at labelling an input audio with onsets of different drum sounds. We rely on an architecture inspired from [21]. The network takes mel-spectrogram inputs processed by 4 layers of 256 padded 2D convolutions of kernel size 5, with unit stride, followed by batch normalization and ReLU. We apply max-pooling at each layer along the frequency dimension. The resulting vector is processed by two linear layers with batch normalization and dropout. Finally, a specific output network of 3 linear layers for each drum sound produces onset probabilities, trained on binary vectors for each drum activation.

To train the network, we use the approach proposed by [34], using a subset of 5000 MIDI drum tracks that we map to random drum sounds to generate waveform recordings. We further rely on the SMT-Drums dataset [35], which provides 104 supplementary polyphonic drum set recordings. For both datasets, we compute a mel-spectrogram of 64 bins, ranging from 20 to 11025 Hz, based on a FFT of window size 2048 and hop size 512.

4.1.5 Onset estimation

Onset estimation [21] aims to detect events in a given audio input. In order to evaluate this task, we rely on the same network presented in the previous section for drum transcription. However, for this task, the last part of the network maps to a single detection subnetwork. We rely on the same drums dataset, but merge all labels to detect event onsets, rather than specific elements of the drumkit.

4.2 Training

All models are trained following their respective procedure and hyperparameters. However, we use a common mini-batch size of 64, the ADAM optimizer with a weight decay penalty of $2e^{-4}$, and an initial learning rate of $1e^{-3}$, which is halved every 10 non-decreasing epochs. We train the models for a number of epochs that is fixed for each task (following the original papers) and keep the model with the best validation score. For the lottery training, we perform *masking* or *trimming* of 30% of the weights at each pruning iteration. We rewind the weights to their values at half of the training epochs. We repeat this process 15 times, leading to models with up to 99.5% of weights removed. This whole lottery training is repeated 5 times, providing the variance and impact of the initialization on the results.

5. RESULTS

5.1 Global evaluation

First, we provide a global evaluation across different tasks, by plotting the respective evolution of the test error as we iteratively remove weights either by classical *fine-tuning*, using the original lottery with *masking*, or our proposed *trimming*. We report for each task the *best* model (lowest test error), *smallest* model (test error at most 1.5 times the original one), and *optimal* model (error at most 1.1 times the original). Results are displayed in Figure 2.

As we can see, classical *fine-tuning* is mostly unable to find more efficient lighter networks and only works at very low pruning rates. Oppositely, our *trimming* approach is able to consistently find networks that are both much smaller and more accurate than reference models. In this regard, the best performances are obtained for *onset detection*, where we find a network with only **14.9%** of the original weights (removing **85.1%** of the weights), while having an error rate of **0.129** (compared to 0.131 for the original). These results hold for almost all tasks: most networks where we trim up to **75%** of the weights pro-

duce lower errors, and we can remove up to **85**% of the weights with minor damage to the test error. Interestingly, the results of *chord extraction* seems to produce the smallest enhancement. This could be explained by the fact that the model has the lowest original number of parameters. Hence, this underlines the crucial need to rely on *largely overparametrized* models to find efficient subnetworks. Regarding the *smallest* models, we are able to remove on average up to 95%, while having a reasonable test error. When comparing our approach to the original lottery *masking*, it seems that masking consistently produces better performances at higher pruning rates, confirming the original results [10] for MIR tasks. However, note that the weights in the masking approach are not removed (however, a fraction of these weights could be removed in a post-processing step). We hypothesize that this resilience to larger pruning ratios stems from the fact that *masking* is able to work on local connectivity patterns, whereas our approach cannot.

5.2 Across-task comparison

5.2.1 Pruning approaches

The lottery ticket hypothesis crucially depends on initialization values for training efficient subnetworks. To evaluate this property across different tasks, we perform the normalized comparison shown in Figure 2 (right, up).

Here, we normalize the error of each task by dividing it by the error of the reference large model, so that its test error is 1. As we can see, using fine-tuning, the approach is unable to obtain subnetworks with higher accuracy, and the error quickly degrades as we remove more weights. Furthermore, it appears that the results are rather unstable, producing large variations in the final test error. Instead, by *rewinding* the weights and *trimming* we consistently obtain smaller subnetworks (up to 75% of the weights removed) that outperform the original models. We are able to apply extensive trimming before the error starts to degrade, globally around 90% across tasks. Hence, it appears that efficient subnetworks can be found solely through the correct combination of connection topology and weights.

5.2.2 Selection criteria

The success of pruning methods depends on the criterion selecting *which* weights should be kept or pruned. Hence, we perform a normalized comparison of different criteria for *trimming*, and display results in Figure 2 (right, down).

Although the global trend seems to be equivalent for most criteria at low pruning ratios, their differences amplify as we remove an increasing amount of weights. Overall, it seems that the *activation* criterion provides the most stable results. Furthermore, it allows to maintain lower error rates, even at higher pruning ratios. However, at lower pruning ratios, it seems that the *magnitude* criterion produces slightly better and more stable results. Finally, the *batchnorm* criterion seems to provide an interesting alternative at low pruning ratios. However, its performance degrades faster than other criteria at very high pruning rates.

task	mod.	error	param	size	FLOPS	mem
inst.	ref	0.092	797 K	10 M	572 M	190 M
	trim	0.117	93.4 K	2.3 M	38.3 M	41.9 M
sing.	ref	0.031	1.4 M	19 M	663 M	194 M
	trim	0.038	144 K	2.7 M	94.4 M	53.2 M
pitch	ref	0.242	5.9 M	49 M	2.8 G	256 M
	trim	0.262	224 K	1.0 M	2.8 M	9.6 M
chord	ref	0.232	416 K	1.4 M	27.2 M	22.1 M
	trim	0.251	91.9 K	0.2 M	1.72 M	589 K
drum	ref	0.136	8.1 M	22 M	3.54 G	667 M
	trim	0.144	1.0 M	3.7 M	87.5 M	10.2 M
onset	ref	0.131	4.7 M	21 M	2.66 G	532 M
	trim	0.132	522 K	3.7 M	87.1 M	8.2 M

Table 1. Comparison between reference models and our trimmed models on *test error*, *number of parameters*, *disk size*, *inference FLOPS* and *memory used* across tasks.

5.3 Resulting model properties

We provide a detailed analysis of the gains provided by our *trimming lottery* for each task. We compare the reference model to the *optimal* one (smallest model within 1.1 times the original test error) found by trimming. We evaluate their *test error*, *number of parameters*, *disk size*, *FLOPS* (required to infer from a single input example) and *memory used* for different MIR tasks, as detailed in Table 1. As discussed previously, we are able to obtain models maintaining the error rates, while having only a small portion of the capacity of the very large models. This can be witnessed in the final properties of the trimmed models. A very interesting observation is that this decrease in parameters amounts to an even larger decrease in the *memory* and *computation power* required. Indeed, while most trimmed models are 10 times smaller than original large models, they use 20 to 50 times less computation power and memory requirements. This can be explained by the fact that most operations are processed across the dimensions of the previous layer. Hence, even small gains in number of parameters can lead to dramatic gains in computation.

6. CONCLUSION

In this paper, we presented a method to obtain ultra-light deep models for MIR, by extending the lottery ticket hypothesis to effectively trim the networks. We have shown that these efficient trimmed subnetworks, removing up to 85% of the weights in deep models, could be found across several MIR tasks. We have also shown that extremely small networks could be found by relying on *masking*, but these do not provide actual enhancement in terms of computation or memory requirements. Oppositely, we have shown that given the non-linear relationship between the number of parameters and computation required, we could find extremely light networks through trimming. These results encourage the crucial implementation of MIR models in embedded audio platforms, which would allow broader end-user applications. The major downside of this approach is its training time, which we partly address by decreasing the cost of each pruning iteration. However, the intriguing prospect of *ticket transfer* [14] could provide such initializations right *at the onset* of training.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[2] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics." in *12th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 403–408.

[3] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "ImageNet training in minutes," in *Proceedings of the 47th International Conference on Parallel Processing*, 2018, pp. 1–10.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[5] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems (NIPS)*, 1990, pp. 598–605.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[8] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *International Conference on Learning Representations*, 2019.

[9] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," *arXiv preprint arXiv:1902.09574*, 2019.

[10] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[11] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Linear mode connectivity and the lottery ticket hypothesis," *arXiv preprint arXiv:1912.05671*, 2019.

[12] ——, "Stabilizing the lottery ticket hypothesis," *arXiv preprint arXiv:1903.01611*, 2019.

[13] J. Frankle, D. J. Schwab, and A. S. Morcos, "The early phase of neural network training," in *International Conference on Learning Representations*, 2020.

[14] A. Morcos, H. Yu, M. Paganini, and Y. Tian, "One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers," in *Advances in Neural Information Processing Systems (NIPS)*, 2019, pp. 4933–4943.

[15] H. You, C. Li, P. Xu, Y. Fu, Y. Wang, X. Chen, R. G. Baraniuk, Z. Wang, and Y. Lin, "Drawing early-bird tickets: Toward more efficient training of deep networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[16] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces." in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[17] J. Wilkins, P. Seetharaman, A. Wahl, and B. A. Pardo, "VocalSet: A singing voice dataset," in *Proceedings of the 19th International Society for Music Information Retrieval (ISMIR) Conference*, 2018, pp. 468–474.

[18] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 161–165.

[19] T. Carsault, J. Nika, and P. Esling, "Using musical relationships between chord labels in automatic chord extraction tasks," in *International Society for Music Information Retrieval (ISMIR) Conference*, 2018.

[20] K. Choi and K. Cho, "Deep unsupervised drum transcription," in *20th International Society for Music Information Retrieval (IISMIR) Conference*, 2019.

[21] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6979–6983.

[22] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1135–1143.

[23] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *International Conference on Learning Representations (ICLR)*, 2017.

[24] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

[25] B. McFee, L. Barrington, and G. Lanckriet, "Learning content similarity for music recommendation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 8, pp. 2207–2218, 2012.

[26] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Feature learning and deep architectures: New directions for music informatics," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 461–481, 2013.

[27] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Flow synthesizer: Universal audio synthesizer control with normalizing flows," *Applied Sciences*, vol. 10, no. 1, p. 302, 2020.

[28] K. Choi, G. Fazekas, K. Cho, and M. Sandler, "A tutorial on deep learning for music information retrieval," *arXiv preprint arXiv:1709.04396*, 2017.

[29] A. De Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[30] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[31] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2019.

[32] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research," in *Proceedings of the 15th International Society for Music Information Retrieval (ISMIR) Conference*, 2014.

[33] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with WaveNet autoencoders," *International Conference on Machine Learning*, vol. 70, pp. 1068–1077, 2017.

[34] M. Cartwright and J. P. Bello, "Increasing drum transcription vocabulary using data synthesis," in *Proc. International Conference on Digital Audio Effects (DAFx)*, 2018, pp. 72–79.

[35] C. Dittmar and D. Gärtner, "Real-time transcription and separation of drum recordings based on NMF decomposition." in *Proc. International Conference on Digital Audio Effects (DAFx)*, 2014, pp. 187–194.

A NEURAL APPROACH FOR FULL-PAGE OPTICAL MUSIC RECOGNITION OF MENSURAL DOCUMENTS

Francisco J. Castellanos   **Jorge Calvo-Zaragoza**   **Jose M. Inesta**

Department of Software and Computing Systems, University of Alicante, Spain

`fcastellanos@dlsi.ua.es, jcalvo@dlsi.ua.es, inesta@dlsi.ua.es`

ABSTRACT

The digitization of the content within musical manuscripts allows the possibility of preserving, disseminating, and exploiting that cultural heritage. The automation of this process has been object of study for a long time in the field of Optical Music Recognition (OMR), with a wide variety of proposed solutions. Currently, there is a tendency to use machine learning strategies based on neural networks because of their high performance and flexibility to adapt to different scenarios by changing only the training data. However, most of the recent literature addresses only specific parts of the traditional OMR workflow such as music object detection or symbol classification. In this paper, we progress one step further by proposing a full-page OMR system for Mensural notation scores that consists of simply two processes, which are enough to extract the symbolic music information from a full page. More precisely, our pipeline uses Selectional Auto-Encoders to extract single staff regions, combined with end-to-end staff-level recognition based on Convolutional Recurrent Neural Networks for retrieving the music notation. The results confirm the adequacy of our method, reporting a successful behavior on two Mensural collections (CAPITAN and SEILS datasets) with a straightforward implementation.

1. INTRODUCTION

The digitization of the content within documents [1] is a process that helps to preserve cultural heritage and enables easier dissemination and knowledge creation. Traditionally, this content digitization is done manually, with an undeniably high cost that is very prone to introduce mistakes as well. In the music context, the development of Optical Music Recognition (OMR) systems promises to perform this task automatically with minimum human involvement. Research efforts have promoted the progress in this field achieving excellent, yet partial results [2–4]; therefore, full digitization of music documents is still to be studied in practical contexts.

Recent advances in machine learning enable new approaches in the OMR field [5]. The use of deep neural networks provides novel ways of avoiding complex multi-step workflows that are considered in legacy OMR research [6]. A successful example of this new trend is the so-called end-to-end approach, that operates at the staff level; in other words, a single step that completely processes the image of a single staff and retrieves the series of symbols that appear therein [7].

While end-to-end strategies can be used to read a sequence of symbols at the staff level, it is still necessary to previously detect all the staves contained in the documents as region blocks, for then transcribing the music content. This staff detection task has been addressed in recent literature [8, 9]. However, these works only assess staff detection as a computer vision problem—i.e., how accurate is, in geometric terms, the region extracted, without considering how useful it is for the subsequent steps. These partial results are not sufficient to determine with certainty the goodness of the approaches within a complete OMR pipeline.
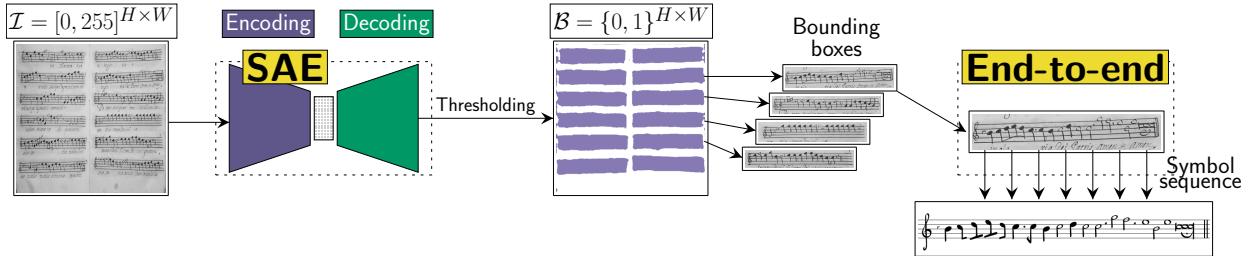
In this work, we carry out a study to determine how the recent advances in OMR interact with each other. Also, we eventually offer, for the first time, results that validate that only two steps—the staff-region detection combined with an end-to-end method—are sufficient to develop a complete page-level OMR system with excellent recognition rates through neural networks.

As we will explain later, this approach is successful if the graphical complexity of the scores follows certain criteria: single-staff systems with a single voice in each. That is why our experiments are restricted to Mensural manuscripts, of great historical interest, where these requirements are common.

2. BACKGROUND

Although the term OMR covers a wide range of scenarios—different research might be carried out according to the notational type or the engraving mechanism of the manuscripts—there has been a general pipeline that addresses the challenge through a series of independent stages that work on different parts of the problem [5].

Traditionally, individual challenges were very complex, so procedures were developed to work on specific manuscripts [10–13]. However, the systems ended up be-

Figure 1: Scheme of the considered methodology.

ing very specific, so previous efforts were hardly reused. In other words, there was little scientific progress in the field.

Recently, the early stages of the process have been reformulated as object detection tasks [14], thus bypassing some of the stages of the traditional workflow. Pacha et al. [15] provided a baseline for direct music-object detection in music score images, experimenting with several models and corpora of different typology.

At the same time, there have been approaches for end-to-end OMR. Such models perform the complete recognition of musical notation from an image, directly providing the sequence of music symbols present therein as output. In addition to high performance, this prevents the need for the training set to be annotated at the symbol-level position and post-process strategies that convert the individually detected elements to the actual music notation. Concerning this formulation, Pugin [16] pioneered the end-to-end approach for printed Mensural notation using Hidden Markov Models (HMM) with the Aruspix system. However, although HMMs represent models that fit perfectly well with the task at issue, other tasks of a similar nature, like handwritten text recognition, experienced a leap in performance using deep neural networks [17]. It has been demonstrated that neural approaches outperform those based on HMM for end-to-end OMR as well [7].

To date, however, there is no existing end-to-end approach that works at the full-page level, but only at the single-staff level. It is not only a challenge to be solved in the field of OMR but also in text recognition—a task that we could consider even simpler. In text recognition, the end-to-end approaches face the recognition process at the line level [18]. For this, there exist line extraction algorithms [19], which enable working at the page level in combination with the line-level end-to-end neural networks. In the case of music, a similar idea is to use staff extraction algorithms combined with the end-to-end staff-level recognition.

Recently, several methods have been proposed to solve the staff detection task [8, 9, 20]. The problem with these works is that they only studied the extraction of staves as a computer vision challenge. Similarly, the end-to-end staff-level neural networks for OMR only experimented with staves detected manually. Therefore, it is not known how well the combination of staff retrieval with staff-level end-to-end neural networks performs in real scenarios.

For all the above, this paper fills a gap in the existing literature and presents, for the first time, a neural full-page

OMR system that takes advantage of recent advances in deep learning to solve the task in just two steps: staff retrieval and end-to-end staff-level recognition. As we will see later, this allows us to provide a general approach that works successfully in different manuscripts by simply providing training data.

3. METHODOLOGY

The proposed methodology outlines an approach by which to evaluate a full page-level OMR system using only two procedures: staff retrieval and end-to-end staff-level recognition. Both of them are solved in a single step each by using deep neural networks. A graphical overview of the complete methodology is shown in Figure 1.

One of the main advantages of our methodology is that it is completely based on machine learning: it is enough to provide annotated examples (of each task) to build new and accurate models—which is usually easier and cheaper than developing a pipeline anew.

Although this approach might not work for arbitrary types of music scores—e.g., recognizing each staff separately does not make that sense for scores that include multi-staff systems—we believe it is worth studying and providing simple, generalizable, and effective solutions in those cases where the structural complexity of the scores makes it possible. Furthermore, our approach is not necessarily restricted to the case of monophony but can be applied in the case where only one voice appears on each staff. In our case study, whose details are available in Section 5, we will apply our methodology to vocal polyphony scores in Mensural notation—where different voices appear independently.
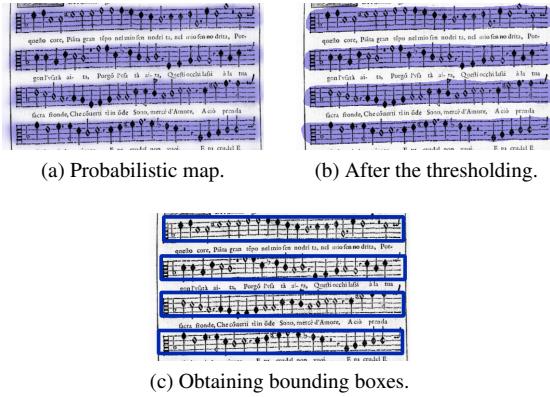
3.1 Staff retrieval

The first step in the considered methodology needs to detect and extract the individual staves. With the premise that all individual staves are compact blocks within the image, we can apply a layout analysis to estimate the probability of each pixel to belong to one of the staves. Previous work [21] presented a Selectional Auto-Encoder (SAE)-based framework focused on performing layout analysis by patches to split the image into different information layers: staff lines, symbols, lyrics and background. Here, we adapt that method to directly detect staff regions. Since the staff blocks are extensive and compact, a patch-wise model may introduce additional errors in their detection. For avoiding

that, we propose to adjust the original image to the input size requirements of the model, being this new resolution enough to discern the different staves.

Let $\mathcal{I} = [0, 255]^{H \times W}$ be a grayscale image [1] with height H and width W in terms of pixels. The SAE model processes $\mathcal{I}$ to return another image $\mathcal{S} = [0, 1]^{H \times W}$ such that $\mathcal{S}_{i,j} \equiv P(\mathcal{I}_{i,j} = \text{'staff'})$—i.e., $\mathcal{S}$ stands for an image with the same size of $\mathcal{I}$, and whose values represent the probability of each pixel of belonging to any staff region. Note that the SAE model is trained through supervised learning mechanisms, hence a set of documents annotated with the location of each staff is required.

After obtaining $\mathcal{S}$, a threshold $\tau \in [0, 1]$ is applied to obtain a binary map $\mathcal{B} = \{0, 1\}^{H \times W}$. Then, the staff regions can be retrieved by performing a connected-component analysis over the map $\mathcal{B}$. Afterwards, we compute the rectangular coordinates of each component for retrieving the bounding boxes. An example of this process can be found in Figure 2.

(a) Probabilistic map.　　(b) After the thresholding.

(c) Obtaining bounding boxes.
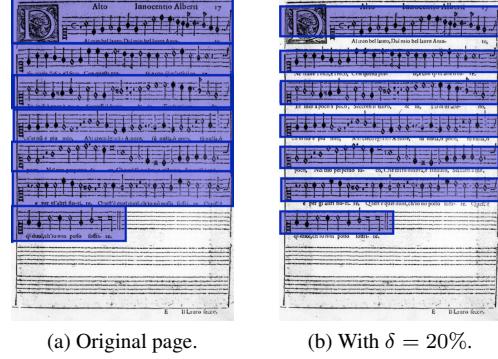
Figure 2: Example of staff-region prediction, with the probabilistic map obtained by the SAE model, the result of applying a threshold to determinate the areas most likely being staves, and finally the bounding box retrieval.

A drawback to this approach is that it requires that the different staves do not overlap with each other, as shown in Figure 3, as that would prevent distinguishing them once $\mathcal{B}$ has been computed. To reduce this possibility, we propose to apply a vertical reduction factor δ, with which the bounding boxes in the ground truth will be trimmed vertically, largely avoiding the overlapping in the annotated documents. Note that, since clipping is necessary to make the subsequent prediction easier, the bounding boxes obtained by the SAE model should be expanded by the same factor after being retrieved. In this way, ideally, the detected bounding boxes will cover the staves completely.

3.2 End-to-end staff-level recognition

Once individual staves are extracted, the symbol recognition at this level can be performed by an end-to-end methodology based on deep neural networks. Within the many options for this, we consider the approach initially

---

[1] This is with no loss of generality, as the approach can be easily extended to deal with color images as well.

(a) Original page.　　(b) With $\delta = 20\%$.

Figure 3: Example of ground truth with overlapping that can be solved by means of applying a reduction factor ($\delta = 20\%$, i.e. 20% top and bottom trims).

proposed by Shi et al. [17], given that it outperformed competing methods for OMR [7].

Given an image $\mathbf{x}$, corresponding to a single-staff region, we want to retrieve the most probable sequence from a fixed alphabet Σ of music symbols. $\mathbf{x}$ can be interpreted as a sequence of frames (single image columns), so the aforementioned problem can be solved by using a recurrent neural network [22]. These networks can provide a probability per frame $P(\sigma \mid x_i), 1 \leq i \leq |\mathbf{x}|, \sigma \in \Sigma \cup \{\epsilon\}$, where ϵ is a special token required to separate consecutive predictions of the same symbol [23].

This stochastic representation of $\mathbf{x}$ can be decoded into an actual sequence of music symbols by first retrieving the most probable sequence of symbols per frame

$$\sigma_i = \arg \max_{\sigma \in \Sigma \cup \{\epsilon\}} P(\sigma \mid \mathbf{x}_i)$$

and then following a greedy approach which merges consecutive frames with the same symbol and removes the frames whose predicted symbol is ϵ [7].

In our case, we add a convolutional neural network on top of the recurrent neural network to automatically learn features that are appropriate for the specific music manuscript at issue [24].

The joint Convolutional Recurrent Neural Network (CRNN) can be trained in an end-to-end fashion by using the so-called Connectionist Temporal Classification (CTC) loss function [23]. Given a ground-truth sample consisting of a single-staff region $\mathbf{x}$ and its corresponding sequence of music symbols σ, CTC is used to modify the network's weights to maximize the probability of retrieving σ from $\mathbf{x}$ without the need of providing a framewise localization of the symbols.

4. EXPERIMENTAL SETUP

4.1 Parameterization of the Neural Networks

In this section, we present the setup of the neural models for both staff retrieval and staff-level symbol recognition. For the first one, we considered the use of SAE due to its high performance and efficiency in the document analysis

Input	Encoding	Decoding	Output
$[0, 255]^{512 \times 512}$	Conv2D$(128, 5 \times 5, $'ReLU'$)$ MaxPool(2×2)	Conv2D$(128, 5 \times 5, $'ReLU'$)$ UpSamp(2×2)	$[0, 1]^{512 \times 512}$
	Conv2D$(128, 5 \times 5, $'ReLU'$)$ MaxPool(2×2)	Conv2D$(128, 5 \times 5, $'ReLU'$)$ UpSamp(2×2)	
	Conv2D$(128, 5 \times 5, $'ReLU'$)$ MaxPool(2×2)	Conv2D$(128, 5 \times 5, $'ReLU'$)$ UpSamp(2×2)	
		Conv2D$(1, 5 \times 5, $'sigmoid'$)$	

Table 1: Detailed description of the selected SAE architecture, implemented as a Fully-Convolutional Network (FCN). 'ReLU' and 'sigmoid' denote the Rectifier Linear Unit and Sigmoid activations, respectively.

task [21]. As of the second process, the symbolic music sequence is obtained by means of a CRNN.

The following notation will be used for the specifications given below: Conv2D$(n, h \times w, $'act'$)$ indicates a two-dimensional convolution operator of n filters and kernel size of $h \times w$ with 'act' denoting the actual activation function; MaxPool$(h \times w)$ represents a down-sampling max-pooling operation with a $h \times w$ window; UpSamp$(h \times w)$ denotes an up-sampling operator of h rows and w columns; BLSTM(n) stands for a bidirectional Long Short-Term Memory unit of n neurons; Dropout(p) represents a dropout operation with a ratio of p neurons; Dense$(n, $'act'$)$ indicates a dense layer of n neurons with 'act' denoting the actual activation function.

4.1.1 Selectional Auto-Encoder

The SAE configuration used in this work is set according to previous works for layout analysis, whose details are given in Table 1. In the staff analysis, the model does not need to predict small details since staves are extensive and compact within the document. For this, we can rescale the original image to the size of the input model, being of enough resolution to differentiate the staves of the ground truth. After some informal testing, we configured the input as an image of 512×512 px. The image rescaling was performed through the OpenCV library.

In addition, as discussed in Section 3.1, a vertical reduction factor δ and a threshold τ to determine the pixels belonging to a staff are necessary. We set $\delta = 20\%$, so the ground-truth staves are top and bottom trimmed by that factor, and $\tau = 0.5$ to indicate that a probability higher or equal to 50% is assumed to represent a pixel from a staff. In our preliminary experiment experiments, δ played an important role for avoiding overlapping, whereas the model was quite robust against different values of τ.

4.1.2 Convolutional Recurrent Neural Network

The CRNN follows the best architecture from the work by Calvo-Zaragoza et al. [7]. It consists of four convolutional layers and max-pooling down-sampling, connected with a recurrent block of two Bidirectional Long Short-Term Memory (LSTM) layers. The specifications of the model are given in Table 2.

Input: $[0, 255]^{64 \times \mathcal{W}}$
Conv2D$(64, 5 \times 5, $'ReLU'$)$, MaxPool$(2 \times 2)$
Conv2D$(64, 5 \times 5, $'ReLU'$)$, MaxPool$(2 \times 2)$
Conv2D$(128, 3 \times 3, $'ReLU'$)$, MaxPool$(2 \times 1)$
Conv2D$(128, 3 \times 3, $'ReLU'$)$, MaxPool$(2 \times 1)$
BLSTM(256), Dropout(0.5)
BLSTM(256), Dropout(0.5)
Dense$(|\Sigma \cup \{\epsilon\}|, $'softmax'$)$

Table 2: Architecture of the CRNN considered for staff-level recognition. 'softmax' indicates the Softmax activation, that normalized the output to a probability over the set of symbols (plus the 'blank' symbol denoted by ϵ). Given that the images are of variable width, this dimension of the input is not specified (indicated as $\mathcal{W}$).

4.2 Corpora

To evaluate our method, we consider the following corpora of Mensural manuscripts:

- The CAPITAN dataset, which encodes a complete *Missa* composed during the second half of the 17th century. Annotations are specifically provided for OMR [25].

- The Symbolically Encoded Il Laurro Secco (SEILS) dataset, which consists of scores from the 16th-century anthology of Italian madrigals *Il Lauro Secco*. Among many formats, the dataset includes the required format to perform OMR [26].

	CAPITAN	SEILS
Engraving	Handwritten	Printed
Pages	97	150
Staves	737	1 278
Running symbols	17 112	31 589
Symbol categories	53	33

Table 3: Corpora statistics.

Page samples from these corpora can be seen in Figure 4. As observed, CAPITAN is handwritten and SEILS is printed. This heterogeneity benefits the verification that the proposed methodology is generalizable to a variety of manuscript types. In addition, some descriptive statistics about the corpora are provided in Table 3.

(a) CAPITAN

(b) SEILS

Figure 4: An example page image from of each dataset considered in the experimentation.

4.3 Evaluation protocol

As of the experimentation, the results have been computed using a 5-fold cross-validation technique (5-CV), each of which takes three data partitions—training, validation and testing—with 60%, 20% and 20% of the whole set of documents, respectively. The training process was performed during 100 epochs, monitoring with the validation partition and reporting the results on the test partition. The experiments have been performed using the Keras v.2.3.1 [27] library with TensorFlow v.1.14 as backend.

In the literature, the experiments are commonly evaluated partially, focusing only on individual processes, regardless of the impact they may have on the transcription into a digital format, which is precisely the ultimate purpose of the OMR field. The main goal of this paper is to evaluate a full-page OMR system combining a staff-retrieval method based on SAE and an end-to-end staff-level recognition. Therefore, we will be able to analyze the effect of staff retrieval in the symbol recognition step.

Nevertheless, experiments have been divided into two parts: an assessment of the staff bounding-box recognition, in which we will present the computation of the average of Intersection over Union (IoU), which provides a measure of the overlapping between the set of retrieved staves and the ground-truth ones (the higher, the better). With regard to the second step of the proposal, the objective is to check the recognition of the sequence of symbols within each staff obtained in the first step. In practical OMR systems, a critical factor to be considered is the number of corrections the user has to perform. Hence, we decided to report the final results in terms of Symbol Error Rate (SER), which is computed as the ratio of editing operations needed to correct the transcription of the symbol sequence (the lower, the better).
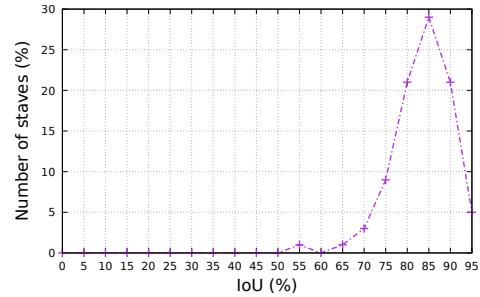
5. RESULTS

Concerning the staff retrieval itself, this step detected all of the 737 staves that CAPITAN contains (considering all the test partitions within the 5-CV) but also retrieved an additional box that did not correspond to a staff. Similarly, the model for SEILS retrieved correctly all the $1\,278$ staves, while 152 regions were detected where there were none. Therefore, whereas all real staves are retrieved, the process also yields some *false positives*, that are supposed to be easily removed in an interactive environment. As a reference in geometric terms, the model for CAPITAN obtained 86.3% of IoU, while SEILS achieved 79.7%. This indicates that the retrieved boxes generally fit well the ground-truth location of the staves. We will see below that this is accurate enough for the task of retrieving the inner symbol sequences.

To complement these numerical results, Figure 5 contains an example of a comparison between a ground-truth staff with the predicted one. Note that, although the IoU obtained in that example is 80.5%, the retrieved staff properly covers the music information. As evidenced in Figure 6, most of the predicted bounding boxes have an IoU between 70% and 95%.

Figure 5: Example of a retrieved staff from SEILS, colored in yellow, compared with the ground-truth box, colored in blue. For this example, the IoU reaches 80.5%.

Figure 6: Average histogram of staves predicted by the SAE model and ordered by IoU (with a granularity of 5%).

We now proceed to present the results of the symbol recognition step. To fully evaluate the two-step proposed method, we should first take into account two main points: first, the staff retrieval SAE can only be trained with ground-truth data, since it constitutes the earlier step in the approach; and second, the end-to-end model would be ideally trained with the bounding boxes predicted in the last step—closer to the real scenario—but it is also possible to directly use the ground-truth staves. While in all cases the sequence of music symbols to be predicted is the same, we can compare the results with different configurations of the input images provided for training the CRNN: ground-truth staff regions (GT); staff regions predicted by the SAE, once trained (Pred.); both ground-truth staff re-

gions and staff regions predicted by the SAE, once trained (GT+Pred.).

Likewise, for the test partition, a real case only contemplates evaluating with the automatically detected staves (real scenario); however, we have also evaluated on ground-truth staves to provide a reference of the loss caused by the automatic staff retrieval.

We report in Table 4 the results of the end-to-end staff-level recognition of symbols, which has been tested with both CAPITAN and SEILS datasets, and each one in turn evaluated with the cases mentioned above.

Data		CAPITAN	SEILS
Training staves	Test staves		
Real scenario			
GT	Pred.	16.8 ± 3.7	5.2 ± 1.4
Pred.	Pred.	14.8 ± 3.6	4.4 ± 0.5
GT+Pred.	Pred.	$\mathbf{11.5 \pm 2.2}$	$\mathbf{3.7 \pm 0.8}$
Reference			
GT	GT	13.2 ± 1.1	4.4 ± 1.2
GT+Pred.	GT	$\mathbf{10.8 \pm 1.1}$	$\mathbf{3.6 \pm 0.9}$

Table 4: Average $\pm$ std. deviation results in terms of SER (%) of a 5-CV experiment for the staff-level end-to-end recognition with different combinations of training and test data during the staff retrieval stage. GT stands for ground-truth staves, while Pred. represents the predicted ones.

First, we focus on the results obtained in the real scenario, i.e. those in which the test staves have been provided by the staff-retrieval step. It can be observed that training with GT achieves successful outcomes, being the SER metric 16.8% and 5.2% for CAPITAN and SEILS, respectively. Results are improved if the training data contains predicted staves instead of GT, with figures that reach 14.8% for CAPITAN and 4.4% for SEILS. The reason behind this phenomenon may come from what is seen in Figure 5: the staff is correctly detected but the box is actually different from the ground-truth one. Therefore, if this difference is also introduced during training, the model is better prepared for what occurs in the real case. Despite this, the experiments reveal that the robustness of the end-to-end model is optimized if a combination of GT and Pred. is performed in the training process, allowing to reduce the symbol error rate until 11.5% and 3.7% for CAPITAN and SEILS datasets, respectively. This combination in the training data seems similar to the typical machine learning strategy called data augmentation [28, 29], given that the Pred. boxes depict variations with respect to the GT ones.

If we analyze the reference results, i.e., those obtained by the end-to-end step tested with GT boxes, we observe that the end-to-end model outperforms the real case, as both training and test data are part of the annotated bounding boxes by the user. Similarly, the data augmentation strategy allows to even improve the results. When comparing to the real scenario, the reference case reports slightly better results, with negligible differences (from 11.5% to 10.8% for CAPITAN and from 3.7% to 3.6% for SEILS, at best).

What is important about the figures above is that they demonstrate that introducing an automatic staff detection step barely affects the overall performance of the system—according to the best values obtained with predictions compared to the best values obtained with ground-truth boxes. Therefore, we can validate our methodology as suitable to deal with the complete recognition of Mensural manuscripts or even any type of musical document that depicts a comparable structure.

Finally, although this is not of special relevance within the scope of this paper, we see that the machine learning models find it easier to deal with printed manuscripts, as the error figures from SEILS are clearly below those from the CAPITAN one. Probably, the regularity of the printed symbols makes the task easier than in the handwritten case.

6. CONCLUSIONS

OMR is an interesting field of study, but most of its research focus on individual steps that avoid evaluating the impact within the full system. In this paper, a full-page OMR system with neural networks has been presented. It is based on the combination of staff-retrieval and symbol sequence recognition steps.

The first step—staff retrieval—has been implemented as a SAE model based on a successful architecture used in previous work for layout analysis. This neural network predicts staff regions as compact blocks, processing the whole image in only one step, and then bounding boxes of predicted staves are extracted. The second step—end-to-end staff-level recognition—transcribes the content of the predicted staves into a digital format, which is the main goal of OMR.

The paper includes a study of the impact of the first step in the final performance in the digitization for two Mensural manuscripts. The methodology has been assessed in terms of SER, which determines the number of corrections that a user should make to have the correct sequence transcribed. The results reveal that ground-truth staves are not the best option for training the end-to-end model in a real case, in which the transcription will be performed from predicted staves. The assessment of the model trained with predicted staves shows performances as good as in the ideal case, in which the training and the test datasets consist of ground truth regions. This means that the precision in staff retrieval is not the most important issue in the symbol recognition task. Furthermore, we observed that training the end-to-end symbol recognizer with a combination of predicted and ground-truth staves provides the best results for both, real and ideal situations, with non-significant differences between them. Therefore, between a fully automatic OMR system and other where the bounding boxes are annotated by the user, the performance hardly varies. We can then conclude that our approach allows transcribing reliably the music content with minimum human effort.

In future works, we plan to keep on researching simple and generalizable strategies for more complex manuscripts, such as those corresponding to polyphonic scores in Western modern notation.

7. ACKNOWLEDGMENT

8. REFERENCES

[1] D. Doermann, K. Tombre *et al.*, *Handbook of Document Image Processing and Recognition.* Springer, 2014.

[2] T. Pinto, A. Rebelo, G. Giraldi, and J. S. Cardoso, "Music score binarization based on domain knowledge," in *Iberian Conference on Pattern Recognition and Image Analysis.* Springer, 2011, pp. 700–708.

[3] A.-J. Gallego and J. Calvo-Zaragoza, "Staff-line removal with selectional auto-encoders," *Expert Systems with Applications*, vol. 89, pp. 138–148, 2017.

[4] Z. Huang, X. Jia, and Y. Guo, "State-of-the-art model for music object recognition with deep learning," *Applied Sciences*, vol. 9, no. 13, p. 2645, 2019.

[5] J. Calvo-Zaragoza, J. H. Jr., and A. Pacha, "Understanding optical music recognition," *ACM Comput. Surv.*, vol. 53, no. 4, Jul. 2020.

[6] D. Bainbridge and T. Bell, "The challenge of optical music recognition," *Computers and the Humanities*, vol. 35, no. 2, pp. 95–121, 2001.

[7] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal, "Handwritten music recognition for mensural notation with convolutional recurrent neural networks," *Pattern Recognition Letters*, vol. 128, pp. 115–121, 2019.

[8] L. Quirós, A. H. Toselli, and E. Vidal, "Multi-task layout analysis of handwritten musical scores," in *Iberian Conference on Pattern Recognition and Image Analysis.* Springer, 2019, pp. 123–134.

[9] A. Pacha, "Incremental supervised staff detection," in *Proceedings of the 2nd International Workshop on Reading Music Systems*, Delft, The Netherlands, 2019, pp. 16–20.

[10] C. Dalitz, G. K. Michalakis, and C. Pranzas, "Optical recognition of psaltic byzantine chant notation," *International Journal of Document Analysis and Recognition*, vol. 11, no. 3, pp. 143–158, 2008.

[11] C. Dalitz and C. Pranzas, "German lute tablature recognition," in *10th International Conference on Document Analysis and Recognition*, 2009, pp. 371–375.

[12] L. J. Tardón, S. Sammartino, I. Barbancho, V. Gómez, and A. Oliver, "Optical music recognition for scores written in white mensural notation," *EURASIP Journal on Image and Video Processing*, vol. 2009, no. 1, p. 843401, 2009.

[13] Y.-H. Huang, X. Chen, S. Beck, D. Burn, and L. Van Gool, "Automatic handwritten mensural notation interpreter: From manuscript to MIDI performance," in *16th International Society for Music Information Retrieval Conference*, M. Müller and F. Wiering, Eds., Málaga, Spain, 2015, pp. 79–85.

[14] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[15] A. Pacha, J. Hajič, and J. Calvo-Zaragoza, "A baseline for general music object detection with deep learning," *Applied Sciences*, vol. 8, no. 9, p. 1488, 2018.

[16] L. Pugin, "Optical music recognitoin of early typographic prints using hidden markov models," in *Proceedings of the 7th International Conference on Music Information Retrieval, Victoria, Canada, 8-12 October*, 2006, pp. 53–56.

[17] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2017.

[18] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in neural information processing systems*, 2009, pp. 545–552.

[19] M. Pastor, "Text baseline detection, a single page trained system," *Pattern Recognit.*, vol. 94, pp. 149–161, 2019.

[20] V. Bosch, J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal-Ruiz, "Sheet music statistical layout analysis," in *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26*, 2016, pp. 313–8.

[21] F. J. Castellanos, J. Calvo-Zaragoza, G. Vigliensoni, and I. Fujinaga, "Document analysis of music score images with selectional auto-encoders," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, 2018, pp. 256–263.

[22] A. Graves, "Supervised sequence labelling with recurrent neural networks," Ph.D. dissertation, Technical University Munich, 2008.

[23] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 369–376.

[24] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proceedings of the 13th European Conference on Computer Vision, Zurich, Switzerland, September 6-12, Part I*, 2014, pp. 818–833.

[25] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal, "Handwritten music recognition for mensural notation: Formulation, data and baseline results," in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15*, 2017, pp. 1081–1086.

[26] E. Parada-Cabaleiro, A. Batliner, and B. W. Schuller, "A diplomatic edition of il lauro secco: Ground truth for OMR of white mensural notation," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, 2019, pp. 557–564.

[27] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[28] Y. Xu, R. Jia, L. Mou, G. Li, Y. Chen, Y. Lu, and Z. Jin, "Improved relation classification by deep recurrent neural networks with data augmentation," in *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, 2016, pp. 1461–1470.

[29] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: When to warp?" in *2016 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2016, Gold Coast, Australia, November 30 - December 2, 2016*, 2016, pp. 1–6.

MODELING PERCEPTION WITH HIERARCHICAL PREDICTION: AUDITORY SEGMENTATION WITH DEEP PREDICTIVE CODING LOCATES CANDIDATE EVOKED POTENTIALS IN EEG

André Ofner and Sebastian Stober

Otto von Guericke University, Magdeburg, Germany

`ofner@ovgu.de stober@ovgu.de`

ABSTRACT

The human response to music combines low-level expectations that are driven by the perceptual characteristics of audio with high-level expectations from the context and the listener's expertise. This paper discusses surprisal based music representation learning with a hierarchical predictive neural network. In order to inspect the cognitive validity of the network's predictions along their time-scales, we use the network's prediction error to segment electroencephalograms (EEG) based on the audio signal. For this, we investigate the unsupervised segmentation of audio and EEG into events using the NMED-T dataset on passive natural music listening. The conducted exploratory analysis of EEG at locations connected to peaks in prediction error in the network allowed to visualize auditory evoked potentials connected to local and global musical structures. This indicates the potential of unsupervised predictive learning with deep neural networks as means to retrieve musical structure from audio and as a basis to uncover the corresponding cognitive processes in the human brain.

1. INTRODUCTION

Studying the human perception of music has received increased interest in Music Information Retrieval (MIR). As humans solve tasks such as beat tracking, genre identification or musical prediction with ease, many MIR methods rely on computational models inspired by human perception. At the same time, studying the brain's response to auditory stimuli is still limited by the lack of resources that map complex musical stimuli to neural processes. Studies in cognitive neuroscience and brain computer interfacing (BCI) on auditory evoked brain states require labor intensive manual preparation and often focus on isolating particular brain responses using sparse stimuli presented individually [1, 2]. While datasets on brain states evoked by natural music exist, they often lack fine-grained annotations of the event structure and corresponding neural activity [3–5].

This entails a demand for efficient and unsupervised mapping techniques between natural music and evoked brain states. Furthermore, there is a need for biologically plausible and multi-modal models for such mapping, as induced brain states are a mixture of stimulus-derived and subjective, cognitive or contextual factors.

We address these challenges with predictive coding, one of the most dominant theoretical frameworks of human perception [6, 7]. Predictive coding offers a comprehensive description of how humans parse and predict sounds and map auditory stimuli to musically meaningful and hierarchically organized units [8]. In predictive coding, the neural response to music is shaped by hierarchically organized expectations [7]. This hierarchy of expectations connects predictions about low-level auditory features to more global context, such as the listener's musical expertise or levels of entrainment during listening [8]. The underlying dependencies between expectancy and uncertainty in predictive coding are particularly interesting in the context of music perception, as music perception can be described as continuously resolving uncertainty and forming new expectations [9–11]. This is in line with evidence on the predictive nature of human music perception, especially within studies on unexpected stimulus deviations and the influence of the listener's expectancy on attention and perceptual precision [9, 11].

Predictive coding offers an efficient algorithmic motif that allows unsupervised learning. Learning in predictive coding systems can be seen as a hierarchy of predictive modules that form predictions over various temporal scales. These predictions can either be about future states in the stimulus domain or about the future of internal states of the systems and are often cast in the context of Bayesian (i.e. probabilistic) inference [12]. In this hierarchical generative model of perception, long-term expectations from temporally stable aspects of music, such as genre or tempo form top-down predictions about the activity of layers closer to the actual auditory information [8]. By propagating the deviations between predictions and observations, the generative model and with that the model of the processed stimulus is updated [7].

Here, we connect predictive coding as a algorithmic motif for unsupervised stimulus representation with deep neural networks and recurrent variational inference in order to segment natural music into units that are musically meaningful. Following the assumption that hierarchical

predictive coding of music explains a substantial amount of evoked brain states, we analyse the retrieved musical structure in terms of the induced neural activity in electroencephalographic signal (EEG). Using the Naturalistic Music EEG Dataset—Tempo (NMED-T) of passive listening to natural music we the demonstrate that the approach allows to locate and visualize event related potentials (ERPs) on local and global scale [3].

2. RELATED WORK

Within the field of MIR, the capacity of predictive coding algorithms to compress and represent auditory information on the sensory level has been exploited for various tasks such as speech re-synthesis or audio compression since many years [13, 14]. The human brain, however, augments such low-level sensory representations with a hierarchy of more abstract, semantic predictions from other brain areas [7]. This aspect of hierarchical predictive learning has found traction in the domain of deep neural networks (DNNs), but so far has been applied mostly to images and video processing [15, 16]. Furthermore, most popular implementations of deep predictive coding often only rely on non-linear transformation of the sensory error and not yet abstract away from pure sensory prediction. Autoregressive modeling of audio has seen tremendous progress in recent years, with a plethora of models performing tasks such as sample level audio prediction or speech synthesis, often with impressive results [17–19]. However, such autoregressive models are computationally expensive and sample-level models still tend to struggle with incorporating more abstract and long-term musical features.

2.1 Auditory evoked potentials and musical structure

Recent years have shown a variety of approaches to studying the human brain's response to auditory stimuli, especially with functional magnetic resonance imaging (fMRI) and electroencephalography (EEG). EEG is especially adequate in the context of music due to its higher temporal resolution. A multitude of auditory features, such as loudness, frequency, tempo and rhythm have been traced in EEG recordings of brain activity during music perception [20–23]. Next to these stimulus-derived aspects, recorded brain activity has further been analysed with respect to more contextual aspects of music perception, such as the listener's attention, which is modulated by aspects such as expertise or engagement [24]. Two extensively researched aspects of the neural response underlying perception potentials are event-related potentials (ERPs) and steady-state evoked potentials (SSEPs) [25, 26]. ERPs and SSEPs differ mainly in their temporal scope: While ERPs are aligned to a single loation (typically the onset of a particular event), SSEPs show frequency alignment to stimulus periodicity over longer time frames [27]. For ERPs, the brain response aligned to the event type of interest is analysed after averaging large amounts of trials [28]. Auditory event-related potentials (AEPs) are modulated by aspects such as rhythm, pitch, timbre or the duration of musical

events, all of which play an important role in human audio segmentation [25, 29–33]. Many of these evoked potentials have been explained in the context of predictive coding as a mixture of bottom-up and top-down mechanisms that are modulated both contextual expectations and the auditory stimulus itself [34, 35]. Similar to ERPs, SSEPs are inspected after averaging over many trials, but don't require zero valued phase offset between stimulus and response. Instead, SSEPs characterize periodic mappings between auditory features and brain response, such as phase locking to perceived frequencies or loudness envelopes. Both ERPs and SSEPs can be related to predictive cognitive processes aiming at structuring the incoming sensory signal into meaningful events in a hierarchical fashion [35, 36].

3. A HIERARCHICAL PREDICTIVE CODING MODEL FOR MUSIC

Predictive coding describes hierarchical predictions of sensory states and hidden states of the network across various time-scales. Sample based predictions about audio requires a model with high temporal resolution that captures the causal dependencies between adjacent samples. Thus, a desired predictive coding model for audio connects low-dimensional predictions over many time-steps with fine-grained predictions at the sensory level. Transforming audio features to high-level representations is a complex task, which is often solved with the non-linear processing found in DNNs. We approach these requirements with a recurrent DNN that generates autoregressive predictions based on long short-term memory (LSTM) [37]. Instead of predicting individual frames, we process mel spectrogram representations of audio. The reduced temporal resolution of spectrograms helps reducing the computational complexity while still capturing fine-grained auditory information. As spectrograms extend into time and frequency, we employ convolutional neural networks (CNNs) to extract features from the spectrograms.

3.1 Autoregressive predictive coding

In order to enable hierarchical predictions across multiple time-scales, we stack multiple LSTM layers and allow each layer to predict the future states of the next lower layer. In line with Bayesian views on brain function and research on the effectiveness of probabilistic recurrent modelling, we express the current state in each layer as Gaussian prior distributions, parameterized by mean and variance parameters [12, 38]. While the lowest layer predicts future audio signal, the network's hidden layers predict future states of the lower layer's representations. More specifically, we first sample the prior distribution of each layer and transform the resulting activation with a convolutional decoder network. The decoder of the lowest layer parameterizes the prediction of expected next spectrogram input window. The decoders in hidden layers output predictions about the mean and variance parameters of the next lower layer. In contrast to the related class of recurrent variational autoencoders (VAE), we do not employ an

encoder network that directly transfers observations to a posterior distribution [39]. Instead, the network processes only the deviation e_t between predicted p_t and observed values o_t with a error encoder network:

$$e_t = p_t - o_t \qquad (1)$$

3.2 Variational inference with deep neural networks

With the previously introduced decoder, error encoder and recurrence networks, the model can be trained to perform variational inference by constructing a variational bound on the data log-likelihood. More specifically, the model is trained to maximize the evidence for its current inferred state, the network's "belief" about what causes an observation. Mathematically speaking, maximizing the model evidence can be expressed as minimizing the complexity of the model's generative model while providing maximally accurate predictions for future audio inputs. The model thus reduces the complexity of states with respect to observations o_t and states s_t at a discrete time step t:

$$Complexity = E_{q(s_{t-1}|o_{\leq t-1})}[KL[q(s_t|o_{\leq t})\|p(s_t|s_{t-1})]] \qquad (2)$$
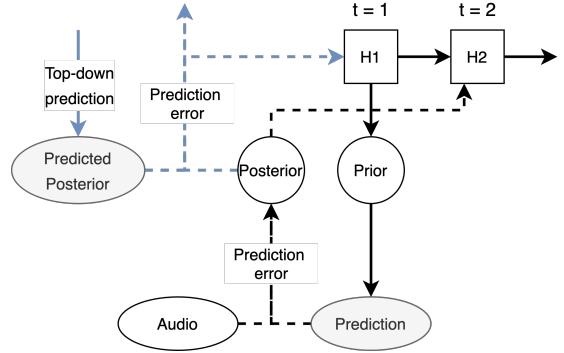
Simultaneously, the accuracy of predicted observations maximized:

$$Accuracy = E_{q(s_t|o_{\leq t})}[\ln p(o_t|s_t)] \qquad (3)$$

The observations in the lowest (sensory) layer refer to the observed audio, while observations in the hidden layers refer to the observed state posteriors in terms of mean and variance. The model optimizes both terms simultaneously for all layers. For this the approximate state posteriors $q(s_{1:T}|e_{1:T}) = \prod_{t=1}^T q(s_t|e_{t-1})$ are inferred by filtering past prediction errors $\{e_t\}_{t=1}^T$. By selecting a pair of adjacent layers and minimizing the accuracy and complexity term between them, this structure allows to form predictions that are consistent between layers, i.e. show small or no top-down prediction error. Such a design prevents error propagation across many layers in a single step. This is a potential drawback and could be improved in future iterations. Throughout all experiments, we used a network with three predictive coding layers. We model $q(s_t|e_{t-1})$ as diagonal Gaussian for all layers with mean and variance parameterized by a convolutional neural network (CNN) with two layers of 64 and 128 units each. The convolutional layers were followed by a dense network of 1024, 512 and 256 units respectively.

3.3 Deterministic transitions with a probabilistic step

The LSTM states of the network are conditioned on its previous states $\{h_t\}_{t=1}^T$, top-down predictions $\{e\_td_t\}_{t=1}^T$ from the next higher layer as well as the prediction error of the last outgoing prediction, the bottom-up prediction error $\{e\_bu_t\}_{t=1}^T$. At each time-step, the bottom-up prediction error is forced to pass a sampling step when updating the prior to the posterior distribution. This means that any incoming sensory information $\{e\_bu_t\}$ must pass

Figure 1. Transitions in a single layer of the predictive coding network. The black pathways show transitions in the lowest layer of the network. Predictions about future audio are conditioned on past states and prediction errors. The blue pathways indicate top-down posterior predictions, which allow to predict the states of lower layers in the network in terms of mean and variance parameters. Multi-step predictions can be generated by updating the recurrent states without sampling new observations.

a stochastic step before being integrated into the deterministic memory states $\{h_t\}$. Figure 1 shows an overview of the transitions in a single layer and the connection to the top-down predictive pathway.

3.4 Model training

The model is trained using a timestep-wise variational evidence lower bound (ELBO) that combines the previously introduced complexity (2) and accuracy (1) terms. Similarly to the objective function in recurrent VAEs [40], the model maximes the ELBO for the approximate posteriors in each layer by accumulating evidence over past timesteps:

$$ELBO(q) = \sum_{t=1}^T \Big(o_{q(s_t|o_{\leq t})}[\ln p(o_t|s_t)]$$
$$- o_{q(s_{t-1}|o_{\leq t-1})}[KL[q(s_t|o_{\leq t})\|p(s_t|s_{t-1})]]\Big) \qquad (4)$$

This structure can be viewed as a hierarchical Kalman filter, making the connection to predictive coding as a Bayesian update scheme or generalized Kalman filtering apparent. We used ReLU activations for all CNNs and hyperbolic tangent activations for the decoder's output layer [41]. In each layer, the prediction error was computed with respect to positive and negative prediction error. Each layer was then ReLU activated before propagation to the encoder networks. For all presented experiments, we trained the model to convergence of the input layer reconstruction loss. For this, we used the Adam optimizer with a learning rate of 10^{-3} [42]. The KL divergence terms for each layer were scaled proportionally to the prediction errors. Furthermore, we weighted the reconstruction losses by 2:1:1 for the employed three layer model.

4. THE NMED-T AND FMA DATASETS

We used the Naturalistic Music EEG Dataset—Tempo (NMED-T) for the evaluations in all presented experiments [3]. NMED-T features EEG recordings from 10 commercially available music pieces, with durations between 270 and 300 seconds, spanning 55 to 150 BPM in various genres. 20 participants were allowed to freely and passively listen to the music, without any additional cognitive load. We used the provided preprocessed version of the EEG data at a sampling rate of 125 Hz. For all presented ERP experiments, we re-referenced the EEG data to the average of all 125 EEG channels and filtered out background noise using a Savitzky-Golay filter before averaging the evoked responses. For network training, we resorted to the "small" partition of the Free Music Archive (FMA) dataset, featuring 8000 songs with 30 seconds duration [43]. We computed magnitude spectrograms for all ten provided audio files of the NMED-T dataset and the FMA audio files before mapping to the mel scale, resulting in mel spectrograms at 125 Hz, equal to the EEG sampling rate. All audio processing steps were done with the librosa library [44]. We tested different mel spectrogram lengths as inputs to the lowest network layer and found lengths between 50 and 150 ms to be the sweet spot with low computation time and without quick overfitting.
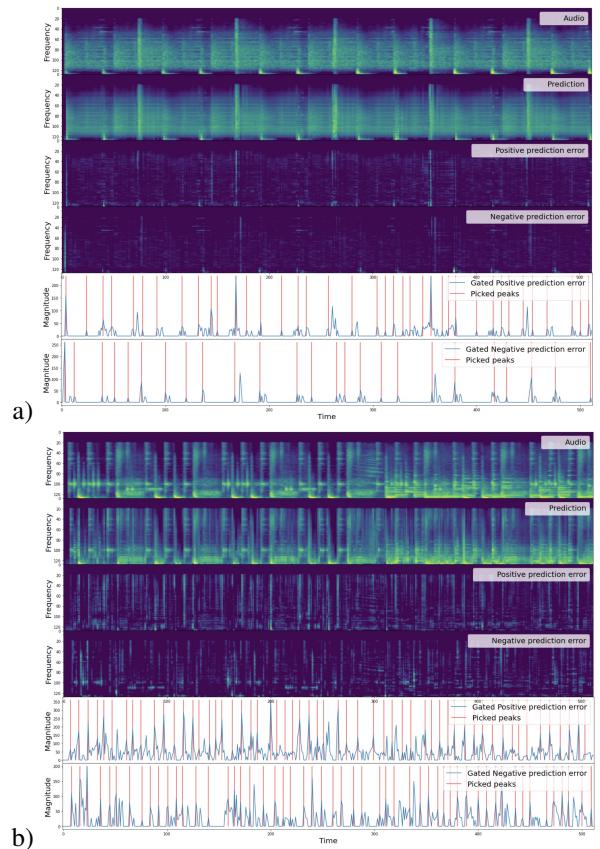
5. EXPERIMENTS

For all following experiments, network training was done first on the FMA dataset followed by a evaluation phase using the NMED-T stimuli. After training on the FMA audio, we froze the network weights and processed the NMED-T audio to generate predictions and corresponding prediction errors. For each processed NMED-T audio stimulus we extracted both positive (PPE) and negative (NPE) valued prediction errors. In this context, PPEs refer to areas where the model predictions are lower than the observed threshold, while NPEs refer to predictions that are higher than the actual values. Predictions were computed in a single pass over each song, i.e. without repeated inference of the current musical context. However, such "active learning" or "active inference" schemes could be explored in the future.

5.1 Deriving segmentation boundaries from prediction errors

In order to inspect the effect of predictive coding at the audio level, we first deactivated the recurrent parts of the lowest layer, forcing the model to express next states as a function of previous observation and the top-down prediction. For model evaluation, we extracted positive and negative prediction errors from each layer of the network. In all layers, we a applied a magnitude threshold to pick peaks from the continuous error response, followed by a peak-picking step that ignores repeated error peaks in a sliding window of fixed size. Both magnitude and window size could be learned by the network itself, leaving

the room for more complex and self-supervised segmentation techniques. All presented experiments use the mean of positive and negative prediction errors, if not further specified. Figure 2 shows two examples for input and predicted audio as well as the corresponding prediction errors and selected peaks. The examples illustrate that autoregressive predictive coding decorrelates large parts of the processed audio in the first layer, by reducing the redundancies in the signal using non-linear weighted predictions based on the past values. This is in line with the spatial and temporal whitening effects described by Rao et al. in the context of center–surround receptive fields in the retina [7]. For the following experiments, we use these sensory predictions to derive segmentation boundaries and explore temporally aligned ERPs in the brain.

Figure 2. Predicted audio and positive and negative prediction errors in the first predictive coding layer for songs with a) 55 and b) 108 BPM. The model generates local predictions about inputs in a sliding window of 50 ms size. This autoregressive and non-linear process removes temporal redundancy in the residual error response. The bottom rows show the thresholded prediction error and picked peaks.

Increasing the weight of the prediction errors in the hidden layer decreased the error magnitudes. This is expected, as the network now learns to include more global temporal context over multiple steps of the lower layer. Ideally, the network learns to predict the rhythmic and timbral structure perfectly and successfully suppresses the prediction error in the first and second layer. If the recurrent parts are

active in the lowest layer, the long-term temporal dependencies can be memorized in the first layer additionally. In our experiments we found that (with a fixed weighting of the prediction errors between layers) deactivating the recurrence in the lowest layer is essential to learning predictive representations in the hidden layers. As visible in Figure 2, the tempo of the song as well as the rhythmic density have influence on the effectiveness of input decorrelation in the lowest layer.
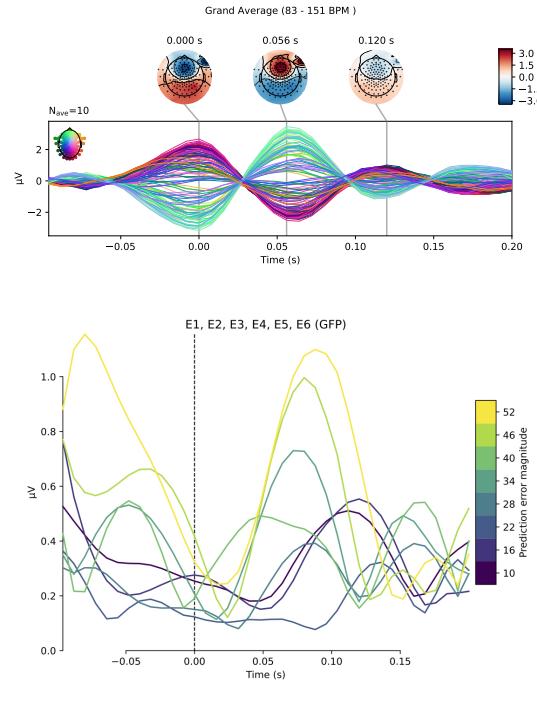
5.2 Grand average ERP

To inspect the possibility to detect ERP events based on the sensory surprise, we extracted the prediction errors from the lowest predictive coding layer and averaged the EEG signal over all trials in all songs and subjects. We were able to derive a total of 242960 trials within 10 songs and 20 subjects using the proposed method. This equates 22140 to 28740 trials per song and between 1108 and 1437 unique event locations per song.

Figure 3 a) shows the grand average ERP for all ten songs in the NMED-T dataset at locations of prediction errors peaks. In comparison to the tempi reported in the orginal NMED-T paper, we sorted the songs between 83 and 151 BPM using beat tracking in the librosa library. The difference between our tempo measures and the ones in the original paper can be explained as being multiples of each other, e.g. 110 BPM being a multiple of 55 BPM. The averaged ERP shows an activity peak for positively correlated channels at the predicted event location, followed by a negative peak around 60 ms after onset. The grand average ERP further shows two smaller peaks around 120 and 170 ms after onset, indicating the presence of surrounding onsets with variable latency. The reduced magnitude of these delayed peaks can be explained by the differences in tempo between songs.Specifically, the difference in peak size between activity close to the predicted onsets and those with greater temporal distance indicates a separation between tempo-independent components (close to the prediction error peak) and attenuated tempo-dependent components. Figure 3 b) shows the grand average ERP in five positively activated channels, sorted by the prediction error magnitude. The magnitude of the first evoked peak after stimulus onset grows proportional with the error magnitude for large error values. For smaller prediction error values, the response shows larger latency. Peaks with similar latency of the evoked activity have magnitudes proportional to the prediction error magnitude. This fits with the assumption that the grand average ERP shows temporally variable peaks induced by differences in tempo.

5.3 Evaluating song-level segmentation with low frequency EEG
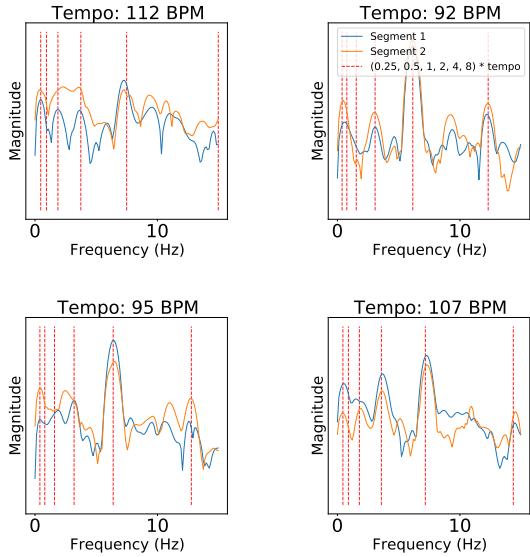
Next to inspecting the predicted ERP responses with the local predictions of the input layer, we want to inspect the possibility to segment stimuli on the song level with the model. For this, we repeat the unsupervised training of the previous experiments, but weight the prediction errors in all layers equally after pretraining for 100000 updates.

a)

b)
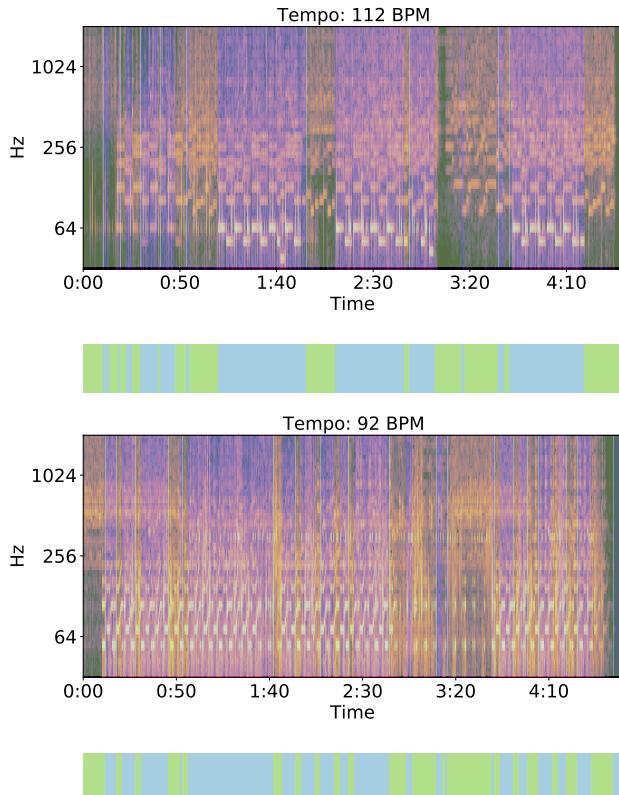
Figure 3. a) Grand average ERP for all songs in the NMED-T dataset at locations of prediction errors peaks generated by the predictive coding network. b) Grand average ERP in five positively correlated channels for trials sorted after the prediction error magnitude of the predictive coding network.

This approach puts more focus on the temporal consistency of the predictions in the hidden layers. Furthermore, we train with multi-step predictions of length 8, i.e. prediction errors are generated with respect to 8 future states at a time. This follows the assumption that both multi-step predictions and increased weighting of the hidden layer prediction errors increase the network's tendency towards more global predictions. In order to evaluate the ability to retrieve meaningful musical structure with the network's predictions, we extracted the timings of prediction errors from the lowest predictive coding layer and used them as starting points of EEG epochs, subsequently averaging the EEG signal over all epochs within each segmented class. Following previous work that illustrates differences in beat processing with SSEPs, we inspect averages of low frequency EEG to detect changes in beat processing or entrainment between the segmented classes derived from prediction errors in the predictive coding network [3]. For this, we use the same epoched data derived from locations computed in the previous experiments but average over all epochs within the bounds of each segmented class.

Here, we want to inspect whether changes in network prediction triggered by peaks in prediction errors show changes that are detectable with SSEPs. To generate binary segmentation, we threshold the prediction error like in the previous steps with a fixed value for each song and switch between segmentation masks when the positive error surpasses the negative error and vice versa. Both

a) SSEPs in low frequency EEG within the segments derived from gated prediction errors of the predictive coding network. Indicated with dashed lines are multiples of the song tempo, ranging from 1 to 16 Hz. Differences between the peaks in the power spectrum of both segments indicate different rhythmic processing between the two segmented classes.

b) Corresponding audio and temporal binary segmentation of two songs derived from gated prediction errors after training the proposed network for unsupervised multi-step prediction. Only the lowest 6 octaves are included for illustration purposes.

Figure 4. Segmented audio and evoked SSEPs in low frequency EEG of the NMED-T dataset.

down-sampling the inputs or increasing to length of multi-step predictions leads to more coarse grained segmentation. We found that using hop lengths up to as much as 16000 successive frames during spectrogram computation were suited to generate song-level segments while simultaneously reducing computation time. Intuitively speaking, the changes in hidden prediction error magnitude reflect the "mid-level" surprise of the network, as the pure sensory surprise is largely minimized in the input layer and the residual errors are further propagated. Future iterations of the model could use learnable error thresholds for improved and self-supervised segmentation. To help visualize the effect of segmentation we reduced spatial EEG dimensionality using Principal Components Analysis (PCA) before averaging the data and analyzed only the first component. Figure 4 a) shows the induced SSEPs in the magnitude of low-frequency EEG for selected songs. Audio and segmentation boundaries for two of these songs are displayed in Figure 4 b). Visible are peaks in the low frequency EEG components within all segmented parts that are aligned with multiples of the song tempo. In most processed songs the noticeable magnitude shifts go along with a stable distribution of the frequencies of evoked peaks, indicating rhythmic differences between the annotated segments which are embedded into the same global tempo.

6. DISCUSSION

This paper explored deep predictive coding for unsupervised audio representation learning inspired by human cognition. We compared the network's prediction errors with evoked potentials in EEG. For this, we related the hierarchical predictions of the model on ten naturalistic musical pieces to onset-aligned evoked potentials captured in EEG. We derived locations for individual musical events from the sensory surprise and inspected steady-state evoked potentials that capture rhythmic differences in the segmented songs. The employed model combines deterministic sequential predictions with probabilistic representations. While the deterministic parts allow to learn regularities over time-scales, the probabilistic elements lessens overfitting and helped shortening training duration. While sensory-level predictions can be employed for local event annotations, the predictions and prediction errors in hidden layers target higher levels of temporal abstraction.

Our results indicate the usefulness of predictive coding for the retrieval of events across the local and global structure of musical works. The model allows to approach audio segmentation jointly with structuring recorded brain activity, forming a basis for retrieval of information about cognitive processes in music perception. This offers an appealing method for researching auditory evoked potentials, as it eases the mapping between stimulus characteristics and connected evoked potentials across time-scales. Future improvements could enhance the capacity of the model, e.g. by allowing the model to segment inputs based on learned error gating.

7. REFERENCES

[1] P. Paavilainen, C. Kaukinen, O. Koskinen, J. Kylmälä, and L. Rehn, "Mismatch negativity (MMN) elicited by abstract regularity violations in two concurrent auditory streams," *Heliyon*, vol. 4, no. 4, 2018.

[2] I. Nambu, M. Ebisawa, M. Kogure, S. Yano, H. Hokari, and Y. Wada, "Estimating the intended sound direction of the user: toward an auditory brain-computer interface using out-of-head sound localization," *PloS one*, vol. 8, no. 2, 2013.

[3] S. Losorelli, D. T. Nguyen, J. P. Dmochowski, and B. Kaneshiro, "NMED-T: A tempo-focused dataset of cortical and behavioral responses to naturalistic music." ISMIR, 2017.

[4] S. Koelstra, C. Muhl, M. Soleymani, J.-S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, and I. Patras, "Deap: A database for emotion analysis; using physiological signals," *IEEE Transactions on Affective Computing*, vol. 3, no. 1, pp. 18–31, 2012.

[5] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahn, "Towards music imagery information retrieval: Introducing the openmiir dataset of EEG recordings from music perception and imagination." in *ISMIR*, 2015, pp. 763–769.

[6] L. Aitchison and M. Lengyel, "With or without you: predictive coding and bayesian inference in the brain," *Current opinion in neurobiology*, vol. 46, pp. 219–227, 2017.

[7] R. P. Rao and D. H. Ballard, "Predictive coding in the visual cortex: a functional interpretation of some extraclassical receptive-field effects," *Nature neuroscience*, vol. 2, no. 1, pp. 79–87, 1999.

[8] P. Vuust and M. A. Witek, "Rhythmic complexity and predictive coding: a novel approach to modeling rhythm and meter perception in music," *Frontiers in psychology*, vol. 5, p. 1111, 2014.

[9] S. L. Denham and I. Winkler, "Predictive coding in auditory perception: challenges and unresolved questions," *European Journal of Neuroscience*, vol. 51, no. 5, pp. 1151–1160, 2020.

[10] M. Heilbron and M. Chait, "Great expectations: is there evidence for predictive coding in auditory cortex?" *Neuroscience*, vol. 389, pp. 54–73, 2018.

[11] S. Koelsch, P. Vuust, and K. Friston, "Predictive processes and the peculiar case of music," *Trends in Cognitive Sciences*, vol. 23, no. 1, pp. 63–77, 2019.

[12] K. Friston and S. Kiebel, "Predictive coding under the free-energy principle," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 364, no. 1521, pp. 1211–1221, 2009.

[13] B. S. Atal and M. R. Schroeder, "Adaptive predictive coding of speech signals," *Bell System Technical Journal*, vol. 49, no. 8, pp. 1973–1986, 1970.

[14] G. Schuller and A. Hännä, "Low delay audio compression using predictive coding," in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2. IEEE, 2002, pp. II–1853.

[15] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," *arXiv:1605.08104*, 2016.

[16] K. Han, H. Wen, Y. Zhang, D. Fu, E. Culurciello, and Z. Liu, "Deep predictive coding network with local recurrent processing for object recognition," in *Advances in Neural Information Processing Systems*, 2018, pp. 9201–9213.

[17] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[18] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," *arXiv:1711.10433*, 2017.

[19] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," *arXiv:1904.03240*, 2019.

[20] S. Nozaradan, I. Peretz, M. Missal, and A. Mouraux, "Tagging the neuronal entrainment to beat and meter," *Journal of Neuroscience*, vol. 31, no. 28, pp. 10 234–10 240, 2011.

[21] S. Nozaradan, I. Peretz, and A. Mouraux, "Selective neuronal entrainment to the beat and meter embedded in a musical rhythm," *Journal of Neuroscience*, vol. 32, no. 49, pp. 17 572–17 581, 2012.

[22] L. K. Cirelli, D. Bosnyak, F. C. Manning, C. Spinelli, C. Marie, T. Fujioka, A. Ghahremani, and L. J. Trainor, "Beat-induced fluctuations in auditory cortical beta-band activity: using EEG to measure age-related changes," *Frontiers in psychology*, vol. 5, p. 742, 2014.

[23] M. S. Treder, H. Purwins, D. Miklody, I. Sturm, and B. Blankertz, "Decoding auditory attention to instruments in polyphonic music using single-trial EEG classification," *Journal of neural engineering*, vol. 11, no. 2, p. 026009, 2014.

[24] A. Aroudi, B. Mirkovic, M. De Vos, and S. Doclo, "Auditory attention decoding with EEG recordings using noisy acoustic reference signals," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 694–698.

[25] G. Plourde, "Auditory evoked potentials," *Best Practice & Research Clinical Anaesthesiology*, vol. 20, no. 1, pp. 129–139, 2006.

[26] S. Morgan, J. Hansen, and S. Hillyard, "Selective attention to stimulus location modulates the steady-state visual evoked potential," *Proceedings of the National Academy of Sciences*, vol. 93, no. 10, pp. 4770–4774, 1996.

[27] P. M. Picciotti, S. Giannantonio, G. Paludetti, and G. Conti, "Steady state auditory evoked potentials in normal hearing subjects: evaluation of threshold and testing time," *Orl*, vol. 74, no. 6, pp. 310–314, 2012.

[28] T.-P. Jung, S. Makeig, M. Westerfield, J. Townsend, E. Courchesne, and T. J. Sejnowski, "Analyzing and visualizing single-trial event-related potentials," in *Advances in neural information processing systems*, 1999, pp. 118–124.

[29] T. W. Picton, *Human auditory evoked potentials*. Plural Publishing, 2010.

[30] R. Näätänen and T. Picton, "The N1 wave of the human electric and magnetic response to sound: a review and an analysis of the component structure," *Psychophysiology*, vol. 24, no. 4, pp. 375–425, 1987.

[31] R. S. Schaefer, P. Desain, and P. Suppes, "Structural decomposition of EEG signatures of melodic processing," *Biological psychology*, vol. 82, no. 3, pp. 253–259, 2009.

[32] M. Meyer, S. Baumann, and L. Jancke, "Electrical brain imaging reveals spatio-temporal dynamics of timbre perception in humans," *Neuroimage*, vol. 32, no. 4, pp. 1510–1523, 2006.

[33] A. Shahin, L. E. Roberts, C. Pantev, L. J. Trainor, and B. Ross, "Modulation of P2 auditory-evoked responses by the spectral complexity of musical sounds," *Neuroreport*, vol. 16, no. 16, pp. 1781–1785, 2005.

[34] T. Baldeweg, "ERP repetition effects and mismatch negativity generation: a predictive coding perspective," *Journal of Psychophysiology*, vol. 21, no. 3-4, pp. 204–213, 2007.

[35] I. Winkler and I. Czigler, "Evidence from auditory and visual event-related potential (erp) studies of deviance detection (mmn and vmmn) linking predictive coding theories and perceptual object representations," *International Journal of Psychophysiology*, vol. 83, no. 2, pp. 132–143, 2012.

[36] S. Nozaradan, I. Peretz, and P. E. Keller, "Individual differences in rhythmic cortical entrainment correlate with predictive behavior in sensorimotor synchronization," *Scientific Reports*, vol. 6, p. 20612, 2016.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," *arXiv:1811.04551*, 2018.

[39] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv:1312.6114*, 2013.

[40] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in neural information processing systems*, 2015, pp. 2980–2988.

[41] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[43] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," *arXiv:1612.01840*, 2016.

[44] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

DECONSTRUCT, ANALYSE, RECONSTRUCT:
HOW TO IMPROVE TEMPO, BEAT, AND DOWNBEAT ESTIMATION

Sebastian Böck
enliteAI, Vienna, Austria
`s.boeck@enlite.ai`

Matthew E. P. Davies
University of Coimbra, CISUC, DEI
`mepdavies@dei.uc.pt`

ABSTRACT

In this paper, we undertake a critical assessment of a state-of-the-art deep neural network approach for computational rhythm analysis. Our methodology is to deconstruct this approach, analyse its constituent parts, and then reconstruct it. To this end, we devise a novel multi-task approach for the simultaneous estimation of tempo, beat, and downbeat. In particular, we seek to embed more explicit musical knowledge into the design decisions in building the network. We additionally reflect this outlook when training the network, and include a simple data augmentation strategy to increase the network's exposure to a wider range of tempi, and hence beat and downbeat information. Via an in-depth comparative evaluation, we present state-of-the-art results over all three tasks, with performance increases of up to 6% points over existing systems.

1. INTRODUCTION

A central concept in much of the work on audio beat tracking is the "tactus" – described as the most comfortable foot-tapping rate when unconsciously tapping to a piece of music. As stipulated by London [1, Ch.1] (and references therein), the tactus is essential for our perception of metre. The tactus by itself carries no information concerning the metrical organisation within a piece of music, but it is informative about both local and global tempo. To perceive metre, we require the hierarchical organisation between at least two levels, and ideally three: a level above the tactus which indicates the longer-term grouping of beats into bars (or measures), and a lower level to describe how the beats are sub-divided – whether in *simple* time (divided by two), or *compound* time (divided by three).

In this sense, we can expand the notion of (unmarked) foot-tapping towards "counting" in time to music. While numerous counting systems exist for the teaching of musical rhythm [2], the "traditional" American system is perhaps the most well-known. For two-level counting, we can mark the three beats of the bar of a waltz as follows: **1** 2 3 **1** 2 3 **1**..., where the **1** indicates the first beat of each

bar, the downbeat. Moving to three-level counting, we can count the sub-divisions of a four beat bar into two as: **1** + 2 + 3 + 4 + **1**..., (*one - and - two - and - three - and - four - and*), and the sub-divisions of the same four beat bar into four as: **1** e + a 2 e + a 3 e + a 4 e + a **1**... (*one - "ee" - and - "ah" - two - "ee" - and - "ah" and so on*).

From the perspective of computational rhythm analysis, we can thus make a distinction between approaches which target one metrical level in isolation, as opposed to those which estimate more than one. Among the single-level approaches, the vast most majority fall within the domain of beat-tracking (e.g [3–6]). When the focus of the analysis moves towards downbeats, this almost exclusively relies on the implicit or explicit modelling of another metrical level, either the beat [7], tatum [8], or a contrast between both [9]. One notable outlier is the downbeat prediction approach of Jehan [10] which relies instead on onset-synchronous analysis.

Concerning the modelling of three simultaneous metrical levels, few published approaches exist. Goto [11] presents a real-time system for estimating the quarter-note, half-note, and measure levels, but doesn't address the sub-beat level. Klapuri et al. [12] on the other hand, address the estimation of tatum, beat, and downbeat, with explicit dependencies between the phase of the beat and the tatum, and the period of the beat and downbeat. For a recent review of beat and downbeat estimation, see [13].

Considering the topic of tempo estimation, which, in most instances, seeks to retrieve a single value to describe a global tempo, existing approaches can be split into two categories: those which make their estimate of the tempo based the post-processing of a sequence of beat times (e.g. for a discussion of techniques, see [14]) and those which treat the task as a classification or regression problem and do not require any prior estimate of the beats [15–18].

In this paper, we seek to work from the perspective of leveraging shared connections in musical structure, and address the simultaneous estimation of three highly interconnected properties of musical rhythm: tempo, beat, and downbeat within a single model. In line with much of the recent literature concerning the extraction of musical information from audio signals [19], we adopt a deep learning approach. We depart from our recent multi-task approach [20] for tempo and beat estimation using a temporal convolutional network (TCN), which was shown to provide state-of-the-art results. We undertake a critical assessment of its constituent parts, and on the basis of our

analysis, adapt it in several ways. At the broadest level, we wish to leverage the benefit of modelling a metrical hierarchy (as opposed to just the beat level) by the inclusion of an additional learning task, downbeat estimation. In terms of the structure of the network itself, we adapt the shallowest layers of the network (i.e. those closest to the musical surface) to provide a better model of harmonic musical sounds. In addition, we propose a novel formulation of the TCN architecture which incorporates an additional dilation rate to each layer as a means to embed understanding of integer ratios modelling the metrical structure.

A peculiar aspect of the evaluation in [20] was the ability of the multi-task model to perfectly estimate the tempo of the HJDB dataset [21] when it was included in the training splits, with good, but noticeably lower performance when it was left as a hold-out test set. Given the characteristic fast tempo of *HJDB*, we speculate that the gap in performance arose due to the lack of any similarly fast-tempo music in the training sets. Following this argument, a secondary motivation of this work is to consider how data augmentation can be used in an efficient way to extrapolate information from regions of the training data which are well-covered in terms of tempo annotations to those which are more sparse.

Via a thorough evaluation across the three tasks of tempo, beat, and downbeat estimation, we demonstrate state-of-the-art performance, and draw attention to the ability of TCN-based approaches to leverage shared representations for multi-task analysis of musical audio signals.

The remainder of this paper is structured as follows. In Section 2 we describe our multi-task formulation and data augmentation strategy in detail. In Section 3 we present an ablation study and comparative evaluation against existing reference systems. Finally, in Section 4 we discuss the impact of the contribution and promising areas for future work.

2. APPROACH

Our earlier multi-task approach for tempo and beat estimation [20] was itself an extension of an earlier TCN-based approach for beat tracking [22]. The core component, which is common to both, is a deep neural network (DNN) architecture based on dilated convolutions, most well-known from *WaveNet* [23]. It is quite striking to consider that an architecture designed for the causal generation of raw audio (primarily for speech synthesis), and with its roots in an auto-regressive process, can find application in a problem cast as binary classification through time, i.e. the classification per frame of the presence or absence of a beat. From an alternative perspective, we may view the strength of the TCN in this problem domain as resulting from multiple connections (both forwards and backwards in time) at different time scales, and thus bearing similarity to much earlier work on the cognitively-inspired use of multi-resolution signal processing for beat tracking [24].

For a detailed description of the existing architecture, we refer to the reader to [20, 22]. In brief, the multi-task approach uses a log-magnitude spectrogram with 81 loga-

rithmically spaced frequency bins and a frame rate of 100 frames per second as input. Overlapping spectrogram snippets are passed through 3 convolution and max pooling layers, followed by 11 dilated convolutional layers whose dilation rate increases by a factor of 2 per layer. The so-called "skip connections" between these layers are provided as an auxiliary output of the TCN and are used to generate a prediction of the tempo across a linear range from $0 - 300$ beats per minute (bpm). The main output of the TCN, a beat activation function, is then processed by a dynamic Bayesian network (DBN) [25] to obtain a final sequence of beat estimates.

In spite of the reported high performance of the multi-task approach on a wide range of musical material for both beat and tempo estimation, we believe it is valuable to question the design decisions of this network and consider the ways in which it could be modified to improve performance. Our focus in this paper is on the core of the network, namely the convolutional and max pooling layers together with the TCN. In a coarse sense, we can consider the convolutional and max pooling layers to relate to more surface-level properties of the music and hence local information, i.e. *what are the spectro-temporal properties of the beats?* with the deeper TCN layers oriented more towards their temporal dependency over longer time scales, i.e. *how is the beat and metrical structure organised over the duration of musical pieces?*

Concerning the first question, a common limitation of beat tracking systems is their ability to reliably detect the beat in music without the presence of drums, as typified, at least in part, by lower reported performance in classical music. Given the high prevalence of rock, pop, jazz, and electronic dance music among existing beat tracking datasets [26], we consider the modelling of harmonic sounds to be important when addressing under-represented musical styles and of crucial importance to reliably detect downbeats in Western music, where harmonic changes often occur at bar boundaries [27]. Regarding the second question, we directly enable the network to learn feature representations which are integer multiplies of each other by deploying multiple concurrent dilated convolutions at each TCN layer, and in so doing embed some implicit hierarchical structure into the model.

In what follows, we describe the specific modifications made to the network. Since the work in this paper explicitly targets the improvement of an existing approach, we allude to performance increases wherever relevant, with detailed results in Section 3.

2.1 Multi-task formulation

Based on the model described above, we add the additional task of downbeat tracking. This can be accomplished in various ways. One option is to model the downbeats and beats jointly as a multi-class problem, i.e. by classifying each input frame to be a beat, a downbeat, or neither. This approach was successfully deployed in [28], but has the downside that it cannot fully leverage the information if a dataset contains only beat or downbeat annotations. Thus

we treat the problem as a multi-label classification problem instead, with the downbeat task treated as a separate binary classification problem with its own output. We model the downbeat output similarly to the beat output as a single *sigmoid* unit which is fed directly from the main TCN output. Whereas the approach in [20] used 16 filters per layer for the multi-task estimation of tempo and beat, with the addition of the downbeat task, we expand the network to include more filters and increase this to 20.

This additional output is then also post-processed with a DBN. Since the beat and downbeat outputs do not define a joint probability density function (i.e. their sum is not guaranteed to be 1 as for multi-class problems), the DBN post-processing used in [28] cannot be applied directly to the combined beat and downbeat activations. Thus the difference of the beat and downbeat activations (limited to positive values) and the downbeat activations are used as state-conditional observations for beats and downbeats, respectively. In Section 3 we refer to this approach as *joint downbeat tracking*.
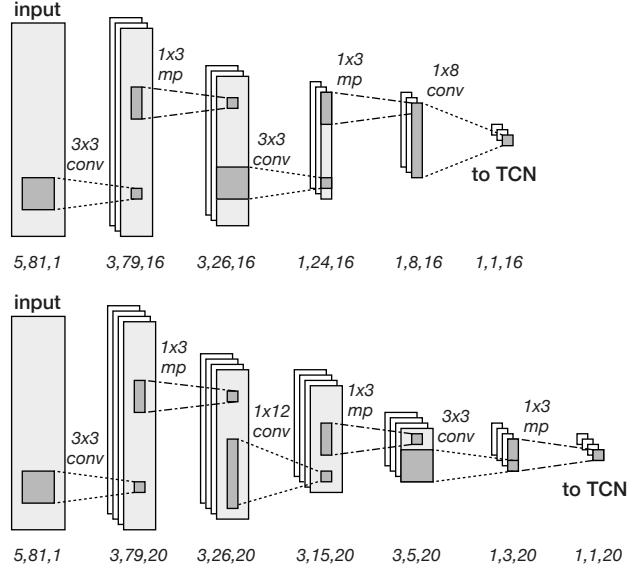
An alternative approach is to first detect the beats and then in a second inference step to find the downbeats given the set of beat predictions. This approach was chosen in [29] and has the most notable advantage that the large joint state space which is required to model multiple bar lengths and tempi at a frame level resolution can be split into two smaller ones. The first one (tracking the beats) only requires multiple tempi to be modelled at the frame level resolution, whereas the second one operates at beat resolution and is completely tempo invariant, thus requiring only very few states. The downside of this approach is that errors made in beat tracking directly propagate to downbeat tracking. In Section 3 we refer to this approach as *sequential downbeat tracking*.

2.2 Conv layers

Both the original beat tracking paper [22] and the multi-task extension tackling global tempo estimation presented in [20] use the same convolutional block to reduce multiple consecutive STFT frames to a one-dimensional feature vector which is then processed by the TCN. Two groups of alternating 3×3 convolution and 1×3 max-pooling layers were used to reduce overlapping spectrogram windows of size 5×81 (time $\times$ frequency) down to 3×26 and 1×8 before these eight bands (roughly representing one octave each) were combined into a singular value with a 1×8 convolution. This feature representation is closely related to the one used in [12] and was shown to work well.

However, a musically motivated reordering of these layers can have a positive effect on the performance of the model. Convolutional filters covering multiple frequency bins but only a single time step have been shown to concentrate on harmonic and timbral features [30] and proven to work well for multiple tasks, including key estimation [31] and automatic music transcription [32]. Moving the "frequency only" convolution in between the two 3×3 convolutions as shown in Figure 1, enables the network to better capture harmonic content across a wider frequency range

instead of detecting local changes in smaller regions of the spectrogram only and then later aggregating them.
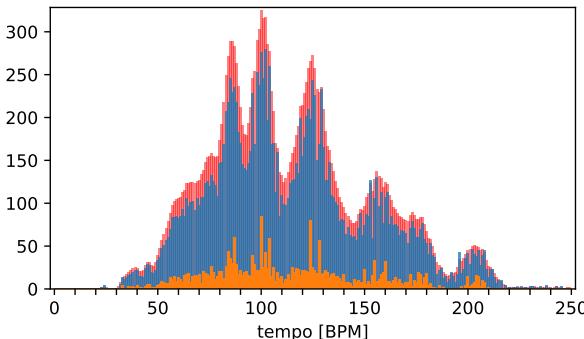
Figure 1: Comparison of the convolution (*conv.*) and max pooling (*mp*) layers. The architecture from [20, 22] (top). Our proposed architecture (bottom). The dimensions of the tensors are shown below each layer.

2.3 TCN layers

From a musical perspective, it is undeniable that discovering downbeats requires more knowledge about the signal than locating beat positions only. Independently of whether this additional knowledge is harmonic or rhythmic in nature, it always requires a longer temporal context. Increasing the temporal context of the TCN by either using larger kernel sizes or adding more layers (with exponentially increasing dilation rates), did not improve any of the tasks under investigation. This observation is not necessarily surprising since the temporal context modelled by the TCN is already about 40 seconds – which should be sufficient to tackle the task of tracking the locations of the downbeats and estimating the length of the bars. Instead, adding a second dilated convolution (with a doubled dilation rate) to each of the TCN layers enables the network to simultaneously model musical properties at various levels which are integer multiples of each other. We discovered that adding a third dilation rate did not further improve performance, but we believe this is very likely an artefact of the data utilised for training, since none of the datasets used have a noticeable number of musical pieces with compound time signatures. The feature maps of the two dilated convolutions are concatenated before spatial dropout [33] and an exponential linear unit (ELU) activation function [34] is applied. In order to keep the output dimensionality of the TCN layer constant, these feature maps are then combined by a 1×1 convolution, which increases the total number of parameters linearly with each TCN layer instead of exponentially.

2.4 Data augmentation

Our approach to data augmentation is both simple and straightforward and similar to the scaling approach applied in [17]. Contrary to other data augmentation strategies, which pre-process the audio signal and manipulate it in various ways (e.g. time stretching, pitch shifting, sample rate conversion to simulate speeding up or slowing down the signal [35, 36]), we do not change the audio signal itself, but instead only change the parameters of the STFT when obtaining a time-frequency representation. To be more precise, we only change the rate at which the overlapping frames of the STFT are obtained from the audio signal by sampling from a normal distribution with 5% standard deviation from the annotated tempo. By changing only the hop size, we obtain spectrograms with varying overlap factors and only the targets have to be adjusted accordingly. Using this data augmentation strategy leads to many more training examples for tempi which are otherwise underrepresented in the data, as can be seen in Figure 2.

Figure 2: Tempo distribution of original tempo annotations (orange, foreground), after data augmentation (blue) and target widening (red, background).

2.5 Network training

Using data augmentation increases the amount of data the network can learn from. However, this also leads to increased training times when using conventional training procedures. Furthermore, the additional downbeat classification layer, the inclusion of a second dilated convolution and the usage of more filters in each of the TCN layers has a notable impact on the size of the model, which now has $116,302$ trainable parameters compared to $29,901$ of [20].

To this end, we make use of the latest training optimisation strategies, namely *RAdam* [37] and *Lookahead Optimization* [38]. The combination of these two drastically reduces the training time (even accounting for the larger number of weights) simultaneously leading to models being less sensitive to different random initialisations. All remaining hyper-parameters were left unchanged. We found the used learning rate of $2e^{-3}$ and clipping the gradients at a norm of 0.5 a sensible choice, as is training on full sequences with a batch size of 1.

We derive the tempo targets in the same way by computing a smoothed and interpolated histogram on the inter

beat intervals. We apply the same target widening strategy to present the network not only the annotated frame and tempo, but also their direct neighbouring frames and ± 2 BPM values as positive targets, albeit with lower weights of 0.5 and 0.25, respectively.
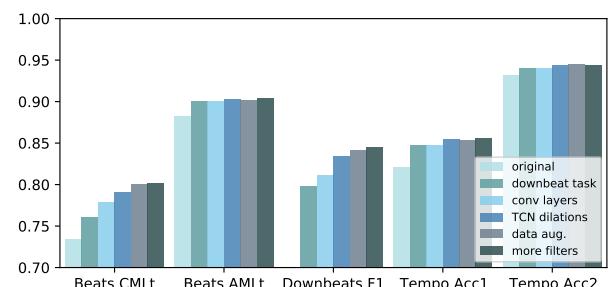
3. EXPERIMENTS AND RESULTS

We use the same datasets as in [20] with the most recent annotations available. *Beatles* [39], *Cuidado* [40], *Hainsworth* [20, 41], *Simac* [42], *SMC* [26], and *HJDB* [21, 28] are used for training and evaluated in an 8-fold cross validation manner. *ACM Mirum* [43, 44], *GiantSteps* [45, 46], and *GTZAN* [47, 48] are used as test datasets. Predictions for the test datasets are obtained by averaging the predictions of the networks trained for cross validation. To enable future comparisons, we make all annotations as well as the beat, downbeat, and tempo estimates available at the accompanying website. [1]

For evaluation, we use the standard metrics used in the literature. For tempo estimation, we report *Accuracy 1* and *Accuracy 2* scores with a tolerance of $\pm 4\%$ as used in [49]. For beat and downbeat tracking, we use *F-measure* and the continuity based metrics *CMLt* (requires beats being tracked at the annotated metrical level) and *AMLt* (allowing alternative metrical levels, such as double/half and triple/third tempo as well as off-beat) with the tolerances as defined as in [39].

3.1 Ablation study

Before reporting comparative evaluation to other methods, we aim to understand how each of the proposed measures outlined in Section 2 contribute to the final performance of the system.

Figure 3: Impact of the improvements proposed for selected evaluation metrics. Mean values over the complete validation set are given.

From Figure 3 it can be seen that the measures undertaken to improve the original system do not contribute the same to the different tasks and the given evaluation metrics. For example, beat tracking *AMLt* and tempo *Accuracy 2* scores increase only marginally, which is best explained by the fact that the baseline system is already performing at a high level on these tasks. However, since these metrics allow metrical ambiguities, it is impossible

---

[1] `https://github.com/superbock/ISMIR2020`

to determine if the system is considering the correct metrical level in the case of beat tracking or the correct tempo octave. Both *CMLt* and *Accuracy 1* require the reported beat locations and tempo to exactly match the annotations (within the allowed tolerance). These metrics therefore better catch the ability of an algorithm to correctly predict the annotated information.

Concentrating on these metrics, it can be seen that additionally modelling and predicting downbeats has a positive effect on beat tracking and tempo estimation. This effect is then strengthened by the modifications made to the convolutional and TCN layers. Using data augmentation and more filters gives a small additional boost. It should be noted that the positive effect of data augmentation on the generalisation capabilities of the network are mostly visible for the task of tempo estimation if "out of tempo distribution" datasets are used for evaluation. Since the validation set is a randomly chosen subset of the training set (and hence has a very similar tempo distribution), the impact is not fully reflected in Figure 3.

3.2 Tempo estimation

Tempo estimation is the task with the most noticeable overall impact of the proposed refinements. While *Accuracy 2* values have been quite high for many systems among all datasets under consideration, the new system is the only one consistently achieving high *Accuracy 1* values as well (Table 1). The system's ability to model several tasks simultaneously and exploit mutual information relevant to all tasks leads to an increased performance of more than 6% points in *Accuracy 1* over the best results reported so far on certain datasets.

	Accuracy 1	Accuracy 2
ACM Mirum		
Gkiokas et al. [50]	0.725	0.979
Percival and Tzanetakis [44]	0.733	0.972
Schreiber and Müller [17]	0.781	0.976
Böck et al. [20]	0.749	0.974
Foroughmand & Peeters [18]	0.733	0.965
Ours	0.841	0.990
GiantSteps		
Gkiokas et al. [50]	0.721	0.922
Percival and Tzanetakis [44]	0.506	0.956
Schreiber and Müller [17] *	0.821	0.971
Böck et al. [20]	0.764	0.958
Foroughmand & Peeters [18] *	0.836	0.979
Ours	0.870	0.965
GTZAN		
Gkiokas et al. [50]	0.651	0.931
Percival and Tzanetakis [44]	0.658	0.924
Schreiber and Müller [17]	0.769	0.926
Böck et al. [20]	0.673	0.938
Foroughmand & Peeters [18]	0.697	0.891
Ours	0.830	0.950

Table 1: Tempo estimation results on unseen test data. Asterisks denote systems which have been trained on a disjoint set of the same source.

3.3 Beat tracking

Although beat tracking performance of existing systems is already very high, the new system sets new high scores in *CMLt* and even exceeds the very high performance values above 0.9 (on *Ballroom*) by more than 4% points. Other systems achieve such high scores only under the less strict *AMLt* metric, which also permits metrical errors, including double/half, triple/third tempo, and off-beat. This highlights the capability of the system to track beats exactly at the annotated metrical level.

	F-measure	CMLt	AMLt
Ballroom			
Böck et al. [28]	0.938	0.892	0.953
Elowsson [51] ‡	0.925	0.903	0.932
Davies and Böck [22]	0.933	0.881	0.929
Ours (beat tracking)	0.956	0.935	0.958
Ours (joint tracking)	0.962	0.947	0.961
Hainsworth			
Böck et al. [5]	0.884	0.808	0.916
Elowsson [51] ‡	0.742	0.676	0.792
Davies and Böck [22]	0.874	0.795	0.930
Ours (beat tracking)	0.904	0.851	0.937
Ours (joint tracking)	0.902	0.848	0.930
SMC			
Böck et al. [5]	0.529	0.428	0.567
Elowsson [51] ‡	0.375	0.225	0.332
Davies and Böck [22]	0.543	0.432	0.632
Ours (beat tracking)	0.552	0.465	0.643
Ours (joint tracking)	0.544	0.443	0.635
GTZAN			
Böck et al. [5]	0.864	0.768	0.927
Davies and Böck [22]	0.843	0.715	0.914
Ours (beat tracking)	0.883	0.808	0.930
Ours (joint tracking)	0.885	0.813	0.931

Table 2: Beat tracking results on datasets used for training with 8-fold cross validation (top), and on unseen test data (bottom). ‡ was trained on *Ballroom* data only.

In Table 2 it can also be seen that joint modelling of beats and downbeats (in the DBN) can be beneficial for music with constant meter and steady tempo (e.g. *Ballroom*), whereas it negatively impacts performance for expressive music as contained in *Hainsworth* and *SMC*.

3.4 Downbeat tracking

For the task of downbeat tracking the systems, performance can be clearly separated into two main categories: i) the systems of Durand et al. [8] and Fuentes et al. [9], which explicitly model harmonic features (using chroma features as input for the neural network) and ii) the ones of Böck et al. [28] and ours which learn harmonic features implicitly. Whereas the former show better performance on pop music (e.g. the *Beatles* dataset) where downbeats often coincide with harmonic changes, they perform less well on data where bars are mostly defined based on rhythm.

	F-measure	CMLt	AMLt
Ballroom			
Böck et al. [28]	0.863	0.834	0.931
Durand et al. [8]	0.797	0.616	0.916
Fuentes et al. [9]	0.83	-	-
Ours (sequential tracking)	0.900	0.894	0.953
Ours (joint tracking)	0.916	0.913	0.960
Hainsworth			
Böck et al. [28]	0.684	0.628	0.832
Durand et al. [8]	0.664	0.500	0.804
Fuentes et al. [9]	0.67	-	-
Ours (sequential tracking)	0.713	0.686	0.855
Ours (joint tracking)	0.722	0.696	0.872
Beatles			
Böck et al. [28]	0.831	0.730	0.858
Durand et al. [8]	0.847	0.722	0.875
Fuentes et al. [9]	0.86	-	-
Ours (sequential tracking)	0.829	0.748	0.860
Ours (joint tracking)	0.837	0.742	0.862
GTZAN			
Böck et al. [28]	0.640	0.577	0.824
Durand et al. [8]	0.607	0.480	0.774
Ours (sequential tracking)	0.654	0.619	0.817
Ours (joint tracking)	0.672	0.640	0.832

Table 3: Downbeat tracking results on datasets used for training with 8-fold cross validation (top), and on unseen test data (bottom).

Regarding the question of whether *joint downbeat tracking* or *sequential downbeat tracking* is superior, Table 3 shows a consistent advantage for processing beats and downbeats simultaneously. The only exception is the *Beatles* dataset, which contains some music with changing metre. Due to memory constraints, joint downbeat tracking cannot model these metre changes. Modelling them is computationally only feasible with sequential downbeat tracking, which may further benefit from sub-beat modelling, as used in [9].

4. DISCUSSION AND CONCLUSIONS

In this paper we address the multi-task estimation of three inter-related properties of musical metre: tempo, beat, and downbeat. Our approach is somewhat unconventional as we do not propose a new method from scratch, but instead we deconstruct, analyse, and then reconstruct an existing approach as a means to further the state of the art. By pairing our methodology with an ablation study, we are able to directly observe the impact of the implemented changes, and in turn, to observe the cumulative gains in performance. Via our evaluation, it is clear that there is no "magic bullet" among our proposed modifications, yet their combination is clearly effective. Furthermore, we must accept that when the baseline performance is already high, the margin for improvement is somewhat limited.

By close inspection of the performance of our approach in comparison both to the baseline and other existing systems, we consider the main impact of our approach as constituting a "closing of the gap" between stricter and more lenient evaluation metrics across each of the tasks. For tempo estimation, our approach is the first to exceed 0.83 for *Accuracy 1* across three large reference datasets, which are completely unseen to our training scheme. Likewise, when considering the positive impact for beat tracking, we find the clearest improvements in the evaluation metric which enforces tracking at the annotated metrical level. Since the relative improvements under the more lenient metrics are much smaller, we do not believe that our approach has unlocked the means to accurately infer the tempo, beat, or downbeat in extremely challenging musical examples. Reference to the incremental improvements for the *SMC* dataset for beat tracking can immediately attest to this. Indeed, the lack of improvement for this kind of musical material may require the reformulation of the inference techniques used to recover the final outputs, rather than intervention at the point of training the networks. Alternatively, they may require a fundamentally different way in which to present targets to the network which is better able to model temporal uncertainty in the annotations. We consider both of these to be promising areas for future work in order to address more challenging data in a robust way.

Ultimately, we believe the main contribution of our work rests in the increased reliability of the good predictions made by the model across these three tasks. It is well-established within music cognition that the perception of tempo, beat, and metre is ambiguous and varies among listeners; therefore within the MIR community, it is easy to justify the use of "multiple-choice" evaluation methodologies. However, this evaluation practice explicitly masks the fact that for almost any piece of music, at least some of these allowed options will be much less reasonable than others. Thus, in the absence of a multi-level annotation methodology in which the set of allowed annotations are specific to individual pieces of music, the only way to guarantee a high-quality prediction (in an unsupervised way) is to aim to maximise performance under stricter evaluation metrics. The alternative is to perform a subjective assessment of beat and downbeat performance via listening to clicks mixed with the audio signals. Given the large amount of musical material in existing datasets, this remains a daunting prospect. However, by restricting this kind of supervised analysis to the subset of excerpts which are accurate only when allowing for alternative interpretations of the annotations, we may move towards a closer estimate of the true performance of these systems. In addition, this kind of partial subjective evaluation could act as a means to "bootstrap" the specification of alternative hypotheses on a per-excerpt basis.

5. ACKNOWLEDGMENTS

6. REFERENCES

[1] J. London, *Hearing in Time: Psychological Aspects of Musical Meter.* Oxford University Press, 2004.

[2] S. L. Gage, "An analysis and comparison of rhythm instructional materials and techniques for beginning instrumental music students," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 1994.

[3] E. D. Scheirer, "Tempo and beat analysis of acoustic musical signals," *The Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.

[4] B. McFee and D. P. W. Ellis, "Better beat tracking through robust onset aggregation," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, 2014, pp. 2154–2158.

[5] S. Böck, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles." in *Proc. of the 15th Intl. Society for Music Information Retrieval Conf.*, 2014, pp. 603–608.

[6] T. Cheng, S. Fukayama, and M. Goto, "Convolving Gaussian Kernels for RNN-Based Beat Tracking," in *Proc. of the 26th European Signal Processing Conf.*, 2018, pp. 1919–1923.

[7] G. Peeters and H. Papadopoulos, "Simultaneous beat and downbeat-tracking using a probabilistic framework: Theory and large-scale evaluation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1754–1769, 2011.

[8] S. Durand, J. P. Bello, B. David, and G. Richard, "Robust downbeat tracking using an ensemble of convolutional networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 76–89, 2017.

[9] M. Fuentes, B. McFee, H. C. Crayencour, S. Essid, and J. P. Bello, "Analysis of common design choices in deep learning systems for downbeat tracking," in *Proc. of the 19th Intl. Society for Music Information Retrieval Conf.*, 2018, pp. 106–112.

[10] T. Jehan, "Downbeat prediction by listening and learning," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005, pp. 267–270.

[11] M. Goto, "An audio-based real-time beat tracking system for music with or without drum-sounds," *Journal of New Music Research*, vol. 30, no. 2, pp. 159–171, 2001.

[12] A. Klapuri, A. Eronen, and J. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 1, pp. 342–355, 2006.

[13] M. Fuentes, "Multi-Scale Computational Rhythm Analysis: A Framework for Sections, Downbeats, Beats, and Microtiming," Ph.D. dissertation, Université Paris-Saclay, 2019.

[14] H. Schreiber, "Data-driven approaches for tempo and key estimation of music recordings," Ph.D. dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2020.

[15] A. J. Eronen and A. P. Klapuri, "Music tempo estimation with k-nn regression," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 1, pp. 50–57, 2009.

[16] H. Schreiber and M. Müller, "A post-processing procedure for improving music tempo estimates using supervised learning." in *Proc. of the 18th Intl. Society for Music Information Retrieval Conf.*, 2017, pp. 235–242.

[17] ——, "A single-step approach to musical tempo estimation using a convolutional neural network," in *Proc. of the 19th Intl. Society for Music Information Retrieval Conf.*, 2018, pp. 100–105.

[18] H. Foroughmand and G. Peeters, "Deep-rhythm for global tempo estimation in music," in *Proc. of the 20th Intl. Society for Music Information Retrieval Conf.*, 2019, pp. 636–643.

[19] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[20] S. Böck, M. E. P. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *Proc. of the 20th Intl. Society for Music Information Retrieval Conf.*, 2019, pp. 486–493.

[21] J. Hockman, M. E. P. Davies, and I. Fujinaga, "One in the Jungle: Downbeat detection in Hardcore, Jungle, and Drum and Bass," in *Proc. of the 13th Intl. Society for Music Information Retrieval Conf.*, 2012, pp. 169–174.

[22] M. E. P. Davies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," in *Proc. of the 27th European Signal Processing Conf.*, 2019.

[23] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR abs/1609.03499*, 2016.

[24] L. M. Smith, "A multiresolution time-frequency analysis and interpretation of musical rhythm," Ph.D. dissertation, University of Western Australia, 2000.

[25] F. Krebs, S. Böck, and G. Widmer, "An efficient state space model for joint tempo and meter tracking," in *Proc. of the 16th Intl. Society for Music Information Retrieval Conf.*, 2015, pp. 72–78.

[26] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon, "Selective sampling for beat tracking evaluation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 9, pp. 2539–2548, 2012.

[27] H. Papadopoulos and G. Peeters, "Joint estimation of chords and downbeats from an audio signal," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 138–152, 2010.

[28] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proc. of the 17th Intl. Society for Music Information Retrieval Conf.*, 2016, pp. 255–261.

[29] F. Krebs, S. Böck, M. Dorfer, and G. Widmer, "Downbeat tracking using beat-synchronous features and recurrent neural networks," in *Proc. of the 17th Intl. Society for Music Information Retrieval Conf.*, 2016, pp. 129–135.

[30] J. Pons, T. Lidy, and X. Serra, "Experimenting with musically motivated convolutional neural networks," in *Proc. of 14th Intl. Workshop on Content-Based Multimedia Indexing*, 2016.

[31] H. Schreiber and M. Müller, "Musical tempo and key estimation using convolutional neural networks with directional filters," in *Proc. of the Sound and Music Computing Conf.*, 2019, pp. 47–54.

[32] R. Kelz, S. Böck, and G. Widmer, "Deep polyphonic ADSR piano note transcription," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, 2019, pp. 129–135.

[33] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2015, pp. 648–656.

[34] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proc. of the 4th Intl. Conf. on Learning Representations*, 2016.

[35] B. McFee, E. Humphrey, and J. Bello, "A software framework for musical data augmentation," in *Proc. of the 16th Intl. Society for Music Information Retrieval Conf.*, 2015, pp. 248 – 254.

[36] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks." in *Proc. of the 16th Intl. Society for Music Information Retrieval Conf.*, 2015, pp. 121–126.

[37] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proc. of the 8th Intl. Conf. on Learning Representations*, 2020.

[38] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, "Lookahead optimizer: k steps forward, 1 step back," in *Proc. of the 33rd Conf. on Neural Information Processing Systems*, 2019.

[39] M. E. P. Davies, N. Degara, and M. D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," Centre for Digital Music, Queen Mary University of London, Tech. Rep. C4DM-TR-09-06, 2009.

[40] F. Gouyon and P. Herrera, "Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptors," in *Audio Engineering Society Convention 114*, 2003.

[41] S. Hainsworth and M. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP Journal on Applied Signal Processing*, vol. 15, pp. 2385–2395, 2004.

[42] F. Gouyon, "A computational approach to rhythm description — audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing," Ph.D. dissertation, Universitat Pompeu Fabra, 2005.

[43] G. Peeters and J. Flocon-Cholet, "Perceptual tempo estimation using GMM-regression," in *Proc. of the 2nd ACM workshop on music information retrieval with user-centered and multimodal strategies (MIRUM)*, 2012, pp. 45–50.

[44] G. Percival and G. Tzanetakis, "Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1765–1776, 2014.

[45] P. Knees, A. Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. Le Goff, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections." in *Proc. of the 16th Intl. Society for Music Information Retrieval Conf.*, 2015, pp. 364–370.

[46] H. Schreiber and M. Müller, "A crowdsourced experiment for tempo estimation of electronic dance music," in *Proc. of the 19th Intl. Society for Music Information Retrieval Conf.*, 2018, pp. 409–415.

[47] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[48] U. Marchand and G. Peeters, "Swing ratio estimation," in *Proc. of the 18th Intl. Conf. on Digital Audio Effects*, 2015, pp. 423–428.

[49] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, p. 1832–1844, 2006.

[50] A. Gkiokas, V. Katsouros, G. Carayannis, and T. Stafylakis, "Music tempo estimation and beat tracking by applying source separation and metrical relations," in *Proc. of the 37th IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, 2012, pp. 421–424.

[51] A. Elowsson, "Beat tracking with a cepstroid invariant neural network," in *Proc. of the 17th Intl. Society for Music Information Retrieval Conf.*, 2016, pp. 351–357.

EXPLORING ACOUSTIC SIMILARITY FOR NOVEL MUSIC RECOMMENDATION

Derek Cheng
Cornell University
dsc252@cornell.edu

Thorsten Joachims
Cornell University
tj@cs.cornell.edu

Douglas Turnbull
Ithaca College
dturnbull@ithaca.edu

ABSTRACT

Most commercial music services rely on collaborative filtering to recommend artists and songs. While this method is effective for popular artists with large fanbases, it can present difficulties for recommending novel, lesser known artists due to a relative lack of user preference data. In this paper, we therefore seek to understand how content-based approaches can be used to more effectively recommend songs from these lesser known artists. Specifically, we conduct a user study to answer three questions. Firstly, do most users agree which songs are most acoustically similar? Secondly, is acoustic similarity a good proxy for how an individual might construct a playlist or recommend music to a friend? Thirdly, if so, can we find acoustic features that are related to human judgments of acoustic similarity? To answer these questions, our study asked 117 test subjects to compare two unknown candidate songs relative to a third known reference song. Our findings show that 1) judgments about acoustic similarity are fairly consistent, 2) acoustic similarity is highly correlated with playlist selection and recommendation, but not necessarily personal preference, and 3) we identify a subset of acoustic features from the Spotify Web API that is particularly predictive of human similarity judgments.

1. INTRODUCTION

Suppose you want to recommend a novel artist B to a friend who you know likes some artist A. Which of B's songs should you recommend that they listen to first? As we will argue in the following, the answer to this question involves concepts of acoustic similarity, playlist context, and personal preference as they all relate to the task of music recommendation [18].

The motivation for asking this question comes from an application we are developing called Localify [1] that recommends artists from a listener's hometown based on the listener's favorite artists [7]. While most of these favorite artists will tend to be popular, well-known artists, the vast

[1] https://www.localify.org

majority of local artists are likely to be relatively obscure *long-tail* artists [1, 3, 11]. There is often limited user preference data (listening histories, like/dislike ratings) associated with these long-tail artists and even less associated with their individual songs.

Automatic playlist generation is one of the core components of our locally-focused music recommender system. As we know from recommender systems research, providing the user with a contextual explanation of the recommendation is important because it provides information that is both useful for decision making and for developing trust in the system [13, 21]. We also know from music psychology research that listeners tend to prefer familiar music [6, 15]. To this end, our playlist algorithm creates a list of songs by alternating between a song by a favorite artist and a song by a local artist. This allows the algorithm to balance exploiting familiarity from known artists with exploring novel artists from the local music scene.

Most commercial recommender systems make use of listening histories from a large number of listeners to make accurate recommendations [2]. This technique is referred to as collaborative filtering (CF). CF systems suffer the cold-start problem [19]: little or no historical user preference data exists for new or obscure artists. As a result, a CF-based recommender system cannot recommend these artists with sufficient confidence. An alternative to CF systems are content-based (CB) recommender systems that make use of the audio signal for recommendation.

Although many of the local artists we want to recommend are relatively obscure, we have informally found that our CF system is able to accurately recommend local artists in most cases. However, it tends to do a poor job of recommending relevant songs for many of our obscure local artists. To this end, we have been exploring the suitability of CB recommendation based on acoustic similarity between a given reference song and a set of candidate songs in order to improve our playlist algorithm.

1.1 Research Questions

To better understand the role of acoustic similarity in recommending long-tail artists, we address the following three research questions in this paper:

RQ1 Are human judgments about acoustic similarity consistent across users?

RQ2 Is acoustic similarity a good proxy for how an individual might construct a playlist, recommend music to a friend, or prefer one song over another?

RQ3 If so, what are some of the measurable acoustic properties that correlate with how humans judge acoustic similarity?

The first question acknowledges that music is subjective and people will naturally have differing opinions [5]. However, if there is little or no consistency in how listeners perceive songs and compare them to one another, then modeling music on the basis of acoustic similarity will be unreliable.

While the answer to the second question may seem obvious, many listeners have eclectic tastes and expect some amount of variety, so it is possible that having music that is too acoustically similar could result in homogeneous playlists and boring recommendations. Additionally, listeners bring non-musical context to bear when listening to music. This includes the meaning of the lyrics, the artist's social persona, visual media (album covers, music videos), trust in the source of the recommendation, the listener's emotional state, etc. In addition, an individual's perception of music is influenced by a variety of socio-demographic factors including age, race, ethnicity, gender, social class, family, political beliefs, and values [12]. To this end, it is entirely possible that the acoustic content is only a small part of the equation when it comes to making a successful recommendation [17].

We address the first two questions by asking individuals to choose among two unknown candidate songs (B1 and B2) from an unknown artist B in the context of a third familiar reference song (A1) by a known artist A. Specifically, we ask which one (B1 or B2) is more acoustically similar to A1, which one they would recommend to a friend who likes A1, which one they would include in a playlist after A1, and which one they personally prefer. Throughout this paper, we will refer to a specific set of three songs $\{A1, B1, B2\}$ as a *song tuple*, and we asked a large number of test subjects to each evaluate a subset of 12 total song tuples (see Table 1) based on their genre preferences.

For the third question, there has been a good deal of research within the music information retrieval community that focuses on using digital signal processing and machine learning to estimate acoustic similarity (e.g., MIREX Audio Similarity Task [2], [10,14,22]). We will not advance the state-of-the-art in this paper but rather simply explore how acoustic properties such as estimated tempo, danceability, and valence correlate with a human listener's judgments related to acoustic similarity. For this, we both analyze open-ended responses from listeners and examine correlations between 11 song-level audio features that we obtain from the public Spotify API for each of the songs in our study.

1.2 Related work

Our work is related to other studies that explore how people interact with music recommender systems. One recent study by Lee et al. [13] found that there are many factors, including aesthetic qualities, familiarity, trust in the recommender, and contextualization, that affect whether a person will adopt a music recommendation. This is consistent with our findings that while acoustic similarity is important for music recommendation, other non-content-based information also plays a big role (see Table 2.)

Zhang et al. [23] stress the importance of serendipity and warn about the dangers of self-reinforcing "filter bubbles" when music recommender systems focus too much on optimizing accuracy. Our recommender system embraces these ideas by attempting to introduce novel artists through locally-focused music recommendation. In addition, the design of our survey was influenced by their suggestion that a recommender system should be akin to having a trusted friend recommend music.

Our application also reflects the findings of Jun et al. [8] who suggest that blending songs in a specific order can improve the quality of the playlist. Specifically, we would like to play a familiar song followed by an acoustically similar song by a local artist so that they flow together. The idea is that a user will be more likely accept the local music recommendation if it sounds similar to something they already enjoy.

Finally, we refer the reader to both Lee et al. [13] and Laplante [12] for recent and comprehensive literature reviews on studies related to human-centered music recommendation.

2. METHODOLOGY OF STUDY

In this section, we describe both how we selected the song tuples and how we designed the user study.

2.1 Song Tuple Selection

We began by collecting 12 song tuples of 3 songs each, for a total of 36 songs. We first selected 4 genres (pop, rock, hip hop, and R&B) with 3 tuples assigned to each genre. Each tuple consisted of a song by a popular artist from its genre, as well as two songs by a relatively lesser-known artist from the same genre. We refer to the popular artist and song respectively as the A artist and A1 song, and the lesser-known artist and songs respectively as the B artist and B1 & B2 songs. For example, in one of the song tuples from the pop genre, the A artist is Billie Eilish and the A1 song is *bad guy*, while the B artist is Gabbie Hanna and the B1 & B2 songs are *Honestly* and *Butterflies*.

The A artists were determined by finding popular artists associated with each of our four genres using the Spotify API [3]. The A1 songs were chosen from the most popular tracks for each artist to maximize the likelihood of them being recognized by our study's participants. The B artists were chosen from a large corpus of artists from our own application such that each B artist was listed as being related to artist A according to the Spotify Web API [4]. We also ensured that the B artists and their B1/B2 songs had limited popularity so that they would likely not be familiar to our study's participants to better simulate long-tail music recommendation.

---

[2] https://www.music-ir.org/mirex

[3] https://developer.spotify.com/documentation/web-api/reference/artists/get-artist/

[4] https://developer.spotify.com/documentation/web-api/reference/artists/get-related-artists/

While we generally selected the most popular songs for both the A and the B artists, we skipped over songs in the popularity ranking under the following conditions. For A1 song selection, we skipped over a song if:

1. The song is a cover of a song by another artist
2. The song is actually by another artist where the current artist is only featured
3. The song was very recently released and thus high in popularity but still relatively low in number of streams

For B1/B2 song selection, criteria 1 and 2 were applied as well. In addition, we also skipped over a potential B1/B2 song if it had an excessively high number of streams due to being a "one hit wonder" so as to reduce the likelihood that test subjects would recognize it.

2.2 Survey Design

Our three section study was conducted through an online Qualtrics survey. In the first section, each participant was asked to provide demographic and psychographic data (e.g., age, gender, time spent listening to music daily, preferred streaming services). Additionally, we asked participants for the two genres they were most familiar with from our four genres of pop, rock, hip hop, and R&B.

The second section collected quantitative data regarding the participants' preferences for playlist selection, recommendation to a friend, acoustic similarity, and personal preference. In this section, participants were presented with six song tuples — three song tuples associated with each of their two selected genres — and asked to answer questions regarding each song tuple. The song tuples were shown by genre, with the ordering of the song tuples within each genre as well as the ordering of the genres themselves being randomized. Additionally, within each song tuple, the ordering of the B1 and B2 songs was also randomized for every participant.

For each song tuple, the participants were first asked to listen to clips of the first 30 seconds of the A1 song and the two B1/B2 songs. The length of 30 seconds was chosen firstly to minimize fatigue and restrict the survey to a reasonable length, and secondly to give the user a sufficiently long sample to form a solid impression of the music [20]. Once they had listened to the clips, they were prompted to answer the following questions regarding the two B1/B2 songs:

1. If you were creating a playlist with Song A1 and either B1 or B2, which one would you pick?
2. If you had a friend who likes Song A1 by Artist A and you wanted to introduce them to Artist B, which song would you recommend to them first?
3. Which song is most acoustically similar to Song A1?
4. Which song do you prefer?

In addition to choosing either B1 or B2, participants were also allowed to answer "About the same" if they could not decide. Finally, we also asked participants if they recognized any of the songs or artists presented in the song tuple, for which they could answer "Yes", "No", or "Maybe".

In the third section, participants were asked the following open-ended questions:

1. When deciding to pick specific songs for a playlist, what do you consider to be most important?
2. When deciding songs to recommend to a friend, what do you consider to be most important?
3. When comparing songs in terms of acoustic similarity, what do you consider to be most important?

These three questions were intended to collect qualitative data regarding how participants made their decisions in the second section of the survey.

3. SURVEY DATA

Participants were recruited on a voluntary basis with no compensation via email lists and social media from the authors' academic and social circles, based primarily within the United States. We received responses from 113 participants, with 103 of these considered valid. From these participants, the youngest represented age group was 17 or younger, while the oldest was 61 - 70, with the median age group being 21 - 25. 58.25% of participants identified as male, while 41.75% identified as female. Participants indicated usage of seven different streaming services, with Spotify being by far the most popular. Some participants also indicated usage of older music playback technologies, such as iTunes libraries, CDs, and vinyl records.

3.1 Song Tuple Responses

From the 103 valid responses, we counted a total of 612 song tuple evaluations, with 317 of these considered valid. We considered a song tuple evaluation valid based on the following two criteria. Firstly, the participant must recognize the A1 song and must not recognize the B1 and B2 songs. This is because our set of questions regarding a song tuple were intended to be asked under the assumption that the A1 song was known, and the B1 and B2 songs were both unknown. Secondly, the participant must spend at least 60 seconds evaluating the tuple. We defined 60 seconds per tuple as the threshold at which a participant is considered to have faithfully answered our questions. Based on these criteria, we discarded responses which did not contain any valid tuple evaluations, and we discarded any invalid tuple evaluations from the remaining responses. Each song tuple was evaluated by at least 14 participants, with a mean of 26.4 participants.

3.2 Qualitative Feedback

One author coded the responses to the three open-ended questions from the third section of the survey. Initial categories (e.g., texture content, rhythmic content, context, preference, playlist mix) and subcategories (e.g., energy, tempo, variety) were formed after making a first pass over 95 non-empty responses out of a total of 103 survey responses. Each response was then coded according to these categories and subcategories. The results are found in Table 2.

The main takeaway from our coding exercise is to note that, as expected, acoustic similarity (Q1) is almost entirely related to audio content, while playlist song selection (Q2) and song recommendation (Q3) involve both audio content

Table 1. The 12 song tuples across four genres, consisting of 3 songs per tuple, used in our survey. For each song tuple, the first song listed is the well-known reference song (A1), followed by the more acoustically similar B1/B2 song, and then the less acoustically similar B1/B2 song. The first line for each of our four concepts (acoustic similarity, playlist selection, recommendation, and personal preference) represents the number of participants respectively that selected the first place song / indicated that they were the same / selected second place song based on acoustic similarity. The second line represents the p-value for a binomial hypothesis test in which the null hypothesis assumes that B1/B2 songs are equally likely to be selected by a participant. Song tuples are sorted by these p-values for acoustic similarity. Bold font indicates statistically significant differences at the $\alpha < 0.05$ level. Italics indicate that participants generally preferred the less acoustically similar song.

Genre	Artist	Song	Acoustic Similarity	Playlist Selection	Recommend	Preference
Rock	The Beatles	Here Comes The Sun	45 / 12 / 5	42 / 15 / 5	42 / 15 / 5	31 / 18 / 13
	Aviator Stash	Hype	**0.000**	**0.000**	**0.000**	**0.006**
	Aviator Stash	Tyler the Beat				
Hip Hop	Nicki Minaj	Anaconda	19 / 0 /1	14 / 3 / 3	14 / 5 / 1	*5 / 6 / 9*
	Mulatto	B*tch From Da Souf	**0.000**	**0.010**	**0.001**	*0.244*
	Mulatto	Longway				
Rock	Paramore	Still into You	14 / 2 / 1	13 / 3 / 1	12 / 1 / 4	8 / 7 / 2
	Tonight Alive	Lonely Girl	**0.001**	**0.002**	0.056	0.088
	Tonight Alive	The Other Side				
Pop	Post Malone	rockstar	22 / 8 / 5	17 / 8 / 10	15 / 11 / 9	*10 / 10 / 15*
	Lil Xan	Lies	**0.001**	0.126	0.156	*0.195*
	Lil Xan	Color Blind				
Pop	Billie Eilish	bad guy	29 / 5 / 10	29 / 6 / 9	28 / 5 / 11	20 / 13 / 11
	Gabbie Hanna	Honestly	**0.002**	**0.001**	**0.006**	0.079
	Gabbie Hanna	Butterflies				
Hip Hop	Cardi B	I Like It	13 / 2 / 3	12 / 2 / 4	8 / 4 / 6	*6 / 5 / 7*
	Kash Doll	Doin Too Much	**0.017**	0.056	0.367	*0.419*
	Kash Doll	No Lames				
Rock	Imagine Dragons	Believer	13 / 5 / 3	8 / 10 / 3	10 / 8 / 3	*3 / 9 / 9*
	The Score	Unstoppable	**0.017**	0.161	0.070	*0.107*
	The Score	Legend				
Hip Hop	Drake	One Dance	14 / 2 / 4	13 / 3 / 4	11 / 6 / 3	7 / 9 / 4
	Kahiem Rivera	Smokin' Weed with the Devil	**0.023**	**0.036**	**0.044**	0.322
	Kahiem Rivera	Good Winter				
Pop	The Weeknd	Starboy	21 / 5 / 10	13 / 8 / 15	18 / 6 / 12	*13 / 6 / 17*
	Myer Clarity	Love Me When I'm High	**0.041**	0.279	0.161	*0.223*
	Myer Clarity	All the Way Down				
R&B	Beyoncé	Halo	9 / 2 / 3	12 / 0 / 2	11 / 0 / 3	7 / 4 / 3
	Keri Hilson	Energy - Main Final	0.107	**0.011**	**0.044**	0.234
	Keri Hilson	Got Your Back				
R&B	Camila Cabello	Havana	9 / 2 / 4	9 / 2 / 4	10 / 0 / 5	10 / 1 / 4
	Ally Brooke	Lips Don't Lie	0.175	0.175	0.183	0.122
	Ally Brooke	No Good				
R&B	Frank Ocean	Thinkin Bout You	9 / 1 / 5	10 / 2 / 3	9 / 3 / 3	10 / 2 / 3
	Syd	Getting Late	0.244	0.070	0.107	0.070
	Syd	Over				

Table 2. Coded responses from 95 participants to questions about acoustic similarity, playlist selection, and music recommendation. The numbers in parenthesis reflect the number responses for each label.

Q1: When comparing songs in terms of **acoustic similarity**, what do you consider to be most important?	
Textural content (71)	Instrumentation (37), Vibe/tone (15), Vocal tone (13), Production/mix (6)
Rhythmic content (65)	Tempo (37), Beat/rhythm (24), Bass line (3), Repetitiveness (1)
Dynamic content (13)	Energy (8), Dynamic range(2), Volume (2), Brightness/intensity (1)
Musicological concepts (8)	Genre/style (4), Mood/expression (4)
Harmonic content (7)	Key (5), Chords (2)
Context (3)	Lyrics (3)

Q2: When deciding to pick specific songs for a **playlist**, what do you consider to be most important?	
Textural content (49)	Vibe/tone (25), Acoustic similarity (18), Instrumentation (5), Vocal tone (3)
Musicological concepts (32)	Mood/expression (19) , Genre/style (12), Time period (1)
Preference (17)	Personal preference (9), Personal mood (6), Catchy (1)
Playlist mix (17)	Flow between songs (10), Variety (7)
Context (13)	Lyrics (7), Theme (6)
Rhythmic content (13)	Tempo (9), Beat (4)
Dynamic content (7)	Energy (7)

Q3: When deciding songs to **recommend** to a friend, what do you consider to be most important?	
Preference (59)	Friend will like (32), Personal preference (16), Interesting to me (5), Catchy (5), Originality (1)
Textural content (30)	Acoustic similarity (20), Vibe/tone (5), Instrumentation (3), Vocal tone (2)
Musicological concepts (11)	Genre/style (11)
Context (6)	Lyrics (3), Meaning (2), Artist background (1)
Rhythmic content (3)	Beat (3)

as well as information that is not directly related to audio content such as personal preference, artist background, and lyrical meaning. That is, while audio similarity is important when creating playlists and recommending music, it is not the only factor that listeners use to make decisions.

4. DISCUSSION OF RESULTS

In this section, we address the three questions that we initially posed using the data that we collected from our user study.

4.1 RQ1: Consistency of Acoustic Similarity Judgments

The first question we explore is the extent to which judgments about acoustic similarity are consistent across many listeners. To do this, we examine how often one of the B1/B2 songs is designated as being more acoustically similar to the A1 song for each of the 12 song tuples. The third column of Table 1 reports the raw counts of how often the *winning* B1/B2 song is designated as being more, equally, or less acoustically similar by our study's participants. We also conducted a binomial hypothesis test where the null hypothesis assumes that both B1/B2 songs are equally similar to the A1 song.

In 9 of the 12 tuples, the similarity judgment was significantly ($\alpha < 0.05$) pointing in one direction [5], suggesting that there was a winner between the B1/B2 songs. The other three tuples showed majorities at or above 64% of the vote. These three tuples were the three R&B tuples that received the fewest number of evaluations, reducing the statistical power of the tests. Overall, the results suggest that listeners are somewhat consistent in their judg-

Table 3. Correlation coefficients for 317 song tuple trials when comparing pairs of acoustic similarity, playlist selection, song recommendation, and personal preference.

	Playlist	Recommend	Pref.
Aco. Similarity	0.716	0.595	0.387
Playlist		0.596	0.386
Recommend			0.116

ment of acoustic similarity even when comparing songs by similar artists.

4.2 RQ2: Relationship between Playlist Selection and Recommendation

To answer our second research question, we look at the correlation between how participants voted for B1 or B2 relative to A with respect to assessing acoustic similarity, song selection for playlist creation, music recommendation, and personal preference. In Table 3, we report the correlation coefficients between pairs of these four concepts. We also conducted a two-tailed hypothesis test for each of these correlation coefficients and found all six to be highly statistically significant, with five p-values less than 0.001 and the p-value for recommendation and preference equal to 0.038.

We see that acoustic similarity is most highly correlated with playlist selection, suggesting that listeners consider acoustic similarity to be important when constructing playlists. This high correlation is also supported by the qualitative feedback (see Table 2) in which *acoustic similarity* is explicitly mentioned as being an important consideration by 18 participants. The participants also mention a large number of acoustic concepts related to texture, rhythm, and dynamics.

---

[5] When applying a Bonferroni correction for multiple hypothesis tests, we observe that 5 of the 12 tuples have p-values less than $\alpha < 0.004$.

We also observe a high level of correlation between acoustic similarity and recommendation. This is consistent with the coded qualitative feedback (see Table 2) in which 20 participants explicitly mention the role of acoustic similarity in music recommendation. However, the slightly lower level of correlation might be attributed to the fact that preference seems to play a greater role in recommendation than acoustic similarity.

Finally, we note a lower level of correlation with personal preference. This is unsurprising since in the previous two cases, we asked participants to make judgments between B1 and B2 relative to the A1 song. It is reasonable that one of the B1/B2 songs would fit more naturally, but that the participant would personally prefer the other B1/B2 song. This, in fact, occurred in 5 of our 12 tuples (see the right most column of Table 1) suggesting that making a recommendation relative to a reference song is different from simply picking preferred songs in isolation.

4.3 RQ3: Acoustic Similarity with Acoustic Features

Here, we explore how a set of song-level content-based features are related to human judgments about acoustic similarity. The specific set of features that we use are obtained using the *Audio Features for a Track* endpoint from the Spotify Web API [6]. These eleven mid-level song features include danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, and tempo.

Our approach involved conducting a one-tailed paired t-test over the 12 song tuples for each of the 11 audio features. For each tuple, we calculate the magnitude (absolute value) of the difference of an audio feature between the the winning B1/B2 song and the reference track A1, and the magnitude of the difference between the losing B1/B2 song and A1. Here, our alternative hypothesis is that the deviation between the winning B1/B2 song and A1 will be smaller than losing B1/B2 and A1 for a given feature. This would imply that the audio feature encodes information that is used by listeners to assess acoustic similarity.

None of the t-tests for any of the 11 audio features revealed any statistically significant differences at the 0.05 confidence level, suggesting that there is no obvious single audio feature that can be used directly to predict acoustic similarity. However, five features shown in Table 4 have a p-value less than $\alpha < 0.15$, suggesting that these features may be related to acoustic similarity. We should note that our sample size with $n = 12$ tuples is small and, as a result, the power of our hypothesis test was limited. A future study with a larger number of song tuples (i.e., more statistical power) may result in more statistically significant results.

It is interesting to note that two of these features, valence and energy, roughly correspond to the first two dimensions of Russell's classic circumspect model of affect [16] that is frequently used to model mood in emotion [9]. Tempo is a rhythmic audio feature and was explicitly mentioned by a large number of our study's participants as being important for assessing acoustic similarity (see Table 2)

Table 4. Spotify audio features for p-values less than $\alpha < 0.15$.

Acoustic Feature	p-value
valence	0.07
speechiness	0.11
tempo	0.12
liveness	0.13
energy	0.13

as well as for song selection when creating playlists. The other two features, speechiness and liveness, are less typical features that were engineered by researchers first at The Echo Nest [7] and now at Spotify [8] to describe the texture of a song. Taken together, this set of five features reflects song texture, rhythmic properties, and perceived mood, suggesting that acoustic similarity is likely to be multi-faceted.

5. CONCLUSION

In this paper, we have explored the relationships between the concepts of acoustic similarity, song selection for playlist creation, music recommendation, and personal preference. Through our user study, we have found that:

1. There is a degree of consistency among human judgments of acoustic similarity.
2. Acoustic similarity is highly correlated with playlist selection and recommendation, but not personal preference.
3. While certain acoustic features obtained seem to be related to acoustic similarity, additional evidence (i.e., more tuples) is necessary to support this with statistical certainty.

These findings are especially significant for the task of recommending songs by obscure, long-tail artists. They provide empirical support for the usage of content-based recommender systems when lack of user preference data precludes the effective functioning of collaborative filtering-based recommender systems. This can apply to both more general recommendation tasks, as well as specific ones like next-song selection for playlist generation.

Building upon the findings described in this paper, a potential avenue for further investigation would be another user study, in a similar vein to our study, but covering a much wider range of music than the 36 songs included in our study, and perhaps with fewer human evaluations per song tuple. This would allow us to determine with statistical certainty whether specific acoustic features encode information about acoustic similarity, therefore providing insight into what acoustic features should be taken into account when building a content-based recommender system [4]. Ultimately, the findings from this paper combined with additional research will aid in the development of a more effective novel-artist recommender system for Localify and other similar music recommendation services.

---

[6] https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

[7] https://github.com/echonest/pyechonest

[8] https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

6. ACKNOWLEDGMENTS

All survey data collected for this paper can be found at the author's website.[9] We have also created a shared Spotify playlist[10] that contains the 36 songs that were used in this study to help the reader better interpret our survey results. This project is supported by NSF grant IIS-1615706/1615679 and IIS-1901168/1901330.

7. REFERENCES

[1] Chris Anderson. *The long tail: Why the future of business is selling less of more*. Hachette Books, 2006.

[2] Luke Barrington, Reid Oda, and Gert RG Lanckriet. Smarter than genius? human evaluation of music recommender systems. In *ISMIR*, volume 9, pages 357–362. Citeseer, 2009.

[3] Oscar Celma. Music recommendation. In *Music recommendation and discovery*, pages 43–85. Springer, 2010.

[4] Joseph Cleveland, Derek Cheng, Michael Zhou, Thorsten Joachims, and Douglas Turnbull. Content-based music similarity with siamese networks. In *Machine Learning for Media Discovery (ML4DM) Workshop at the International Conference on Machine Leanring (ICML)*, 2020.

[5] Arthur Flexer and Taric Lallai. Can we increase inter- and intra-rater agreement in modeling general music similarity? 2019.

[6] Patrick G Hunter and E Glenn Schellenberg. Interactive effects of personality and frequency of exposure on liking for music. *Personality and Individual Differences*, 50(2):175–179, 2011.

[7] Vianca Hurtado, Thorsten Joachims, and Douglas Turnbull. Changing how we value local music using personalized recommendation. In *ACM Tapia*, 2020.

[8] Sanghoon Jun, Daehoon Kim, Mina Jeon, Seungmin Rho, and Eenjun Hwang. Social mix: automatic music recommendation and mixing scheme based on social network analysis. *The Journal of Supercomputing*, 71(6):1933–1954, 2015.

[9] Youngmoo E Kim, Erik M Schmidt, Raymond Migneco, Brandon G Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A Speck, and Douglas Turnbull. Music emotion recognition: A state of the art review. In *ISMIR*, volume 86, pages 937–952, 2010.

[10] Peter Knees and Markus Schedl. *Music similarity and retrieval: an introduction to audio-and web-based strategies*, volume 36. Springer, 2016.

[11] Dominik Kowald, Markus Schedl, and Elisabeth Lex. The unfairness of popularity bias in music recommendation: A reproducibility study. In *European Conference on Information Retrieval*, pages 35–42. Springer, 2020.

[12] Audrey Laplante. Improving music recommender systems: What can we learn from research on music tastes? In *ISMIR*, pages 451–456, 2014.

[13] Jin Ha Lee, Liz Pritchard, and Chris Hubbles. Can we listen to it together?: Factors influencing reception of music recommendations and post-recommendation behavior. 2019.

[14] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning content similarity for music recommendation. *IEEE transactions on audio, speech, and language processing*, 20(8):2207–2218, 2012.

[15] Adrian C North and David J Hargreaves. Subjective complexity, familiarity, and liking for popular music. *Psychomusicology: A Journal of Research in Music Cognition*, 14(1-2):77, 1995.

[16] J. A. Russell. A circumspect model of affect. *Journal of Psychology and Social Psychology*, 39(6):1161, 1980.

[17] Matthew J Salganik, Peter Sheridan Dodds, and Duncan J Watts. Experimental study of inequality and unpredictability in an artificial cultural market. *Science*, 311(5762):854–856, 2006.

[18] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. Music recommender systems. In *Recommender systems handbook*, pages 453–492. Springer, 2015.

[19] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, 2018.

[20] Yading Song, Simon Dixon, Marcus T Pearce, and Andrea R Halpern. Perceived and induced emotion responses to popular music: Categorical and dimensional models. *Music Perception: An Interdisciplinary Journal*, 33(4):472–492, 2016.

[21] Nava Tintarev and Judith Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender systems handbook*, pages 479–510. Springer, 2011.

[22] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[23] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: Introducing serendipity into music recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 13–22, 2012.

---

[9] https://dougturnbull.org/index.php/publications/
[10] https://open.spotify.com/playlist/
4zuHH222bWJFMBaMv3TiyY?si=LmI1wkWGRDqf5xZn7ssCEA

DRUMGAN: SYNTHESIS OF DRUM SOUNDS WITH TIMBRAL FEATURE CONDITIONING USING GENERATIVE ADVERSARIAL NETWORKS

Javier Nistal
Sony CSL
Paris, France

Stefan Lattner
Sony CSL
Paris, France

Gaël Richard
LTCI, Télécom Paris
Institut Polytechnique de Paris, France

ABSTRACT

Synthetic creation of drum sounds (e.g., in drum machines) is commonly performed using analog or digital synthesis, allowing a musician to sculpt the desired timbre modifying various parameters. Typically, such parameters control low-level features of the sound and often have no musical meaning or perceptual correspondence. With the rise of Deep Learning, data-driven processing of audio emerges as an alternative to traditional signal processing. This new paradigm allows controlling the synthesis process through learned high-level features or by conditioning a model on musically relevant information. In this paper, we apply a Generative Adversarial Network to the task of audio synthesis of drum sounds. By conditioning the model on perceptual features computed with a publicly available feature-extractor, intuitive control is gained over the generation process. The experiments are carried out on a large collection of kick, snare, and cymbal sounds. We show that, compared to a specific prior work based on a U-Net architecture, our approach considerably improves the quality of the generated drum samples, and that the conditional input indeed shapes the perceptual characteristics of the sounds. Also, we provide audio examples and release the code used in our experiments. [1]

1. INTRODUCTION

Drum machines are electronic musical instruments that create percussion sounds and allow to arrange them in patterns over time. The sounds produced by some of these machines are created synthetically using analog or digital signal processing. For example, a simple snare drum can be synthesized by generating noise and shaping its amplitude envelope [1] or, a bass drum, by combining low-frequency harmonic sine waves with dense mid-frequency components [2]. The characteristic sound of this synthesis process contributed to the cult status of electronic drum machines in the '80s.

---

[1] https://github.com/SonyCSLParis/DrumGAN

Data-driven processing of audio using Deep Learning (DL) emerged as an alternative to traditional signal processing. This new paradigm allows us to steer the synthesis process by manipulating learned higher-level latent variables, which provide a more intuitive control compared to conventional drum machines and synthesizers. In addition, as DL models can be trained on arbitrary data, comprehensive control over the generation process can be enabled without limiting the sound characteristic to that of a particular synthesis process. For example, Generative Adversarial Networks (GANs) allow to control drum synthesis through their latent input noise [3] and Variational Autoencoders (VAE) can be used to create variations of existing sounds by manipulating their position in a learned timbral space [4]. However, an essential issue when learning latent spaces in an unsupervised manner is the missing interpretability of the learned latent dimensions. This can be a disadvantage in music applications, where comprehensible interaction lies at the core of the creative process. Therefore, it is desirable to develop a system which offers expressive and musically meaningful control over its generated output. A way to achieve this, provided that suitable annotations are available, is to feed higher-level conditioning information to the model. The user can then manipulate this conditioning information in the generation process. Along this line, some works on sound synthesis have incorporated pitch-conditioning [5,6], or categorical semantic tags [7], capturing rather abstract sound characteristics. In the case of drum pattern generation, there are neural-network approaches that can create full drum tracks conditioned on existing musical material [8].

In a recent study [9], a U-Net is applied to neural drum sound synthesis, conditioned on continuous perceptual features describing timbre (e.g., boominess, brightness, depth). These features are computed using the Audio Commons timbre models. [2] Compared to prior work, this *continuous* feature conditioning (instead of using categorical labels) for audio synthesis provides more fine-grained control to a musician. However, this U-Net approach learns a deterministic mapping of the conditioning input information to the synthesized audio. This limits the model's capacity to capture the variance in the data, resulting in a sound quality that does not seem acceptable in a professional music production scenario.

In this paper, we build upon the same idea of conditional generation using continuous perceptual features,

---

[2] https://github.com/AudioCommons/ac-audio-extractor

but instead of a U-Net, we employ a Progressive Growing Wasserstein GAN (PGAN) [10]. Our contribution is two-fold. First, we employ a PGAN on the task of conditional drum sound synthesis. Second, we use an auxiliary regression loss term in the discriminator as a means to control audio generation based on the conditional features. We are not aware of previous work attempting *continuous* sparse conditioning of GANs for musical audio generation. We conduct our experiments on a dataset of a large variety of kick, snare, and cymbal sounds comprising approximately 300k samples. Also, we investigate whether the feature conditioning improves the quality and coherence of the generated audio. For that, we perform an extensive experimental evaluation of our model, both in conditional and unconditional settings. We evaluate our models by comparing the Inception Score (IS), the Fréchet Audio Distance (FAD), and the Kernel Inception Distance (KID). Furthermore, we evaluate the perceptual feature conditioning by testing if changing the value of a specific input feature yields the expected change of the corresponding feature in the generated output. Audio samples of DrumGAN can be found on the accompaniment website (see Section 4).

The paper is organized as follows: In Section 2 we review previous work on audio synthesis, and in Section 3 we describe the experiment setup. Results are presented in Section 4, and we conclude in Section 5.

2. PREVIOUS WORK

Deep Generative modeling is a topic that has gained a lot of interest during the last years. This has been possible partly due to the growing amount of large-scale datasets of different modalities [5, 11] coupled with groundbreaking research on generative neural networks [10, 12–15]. In addition to the methods listed in the introduction focusing on drums sound generation, a number of other studies have applied deep learning methods to address general audio synthesis. Autoregressive models for raw audio have been very influential in the beginning of this line of research, and still achieve state of the art in different audio synthesis tasks [5, 12, 16, 17]. Approaches using Variational Auto-Encoders [13] allow manipulating the audio in latent spaces learnt i) directly from the audio data [4], ii) by imposing musically meaningful priors over the structure of these spaces [7, 18, 19], or iii) by restricting such latent codes to discrete representations [20]. GANs have been extensively applied to synthesis of speech [21] and domain adaptation [22, 23] tasks. The first of its kind applying adversarial learning to the synthesis of musical audio is WaveGAN [3]. This architecture was shown to synthesize audio from a variety of sound sources, including drums, in an unconditional way. Recent improvements in the quality and training stability of GANs [10, 24, 25] resulted in methods that outperform WaveNet baselines on the task of audio synthesis of musical notes using sparse conditioning labels representing the pitch content [6]. A few other works have used GANs with rather strong conditioning on prior information for tasks like Mel-spectrum

inversion [26] or audio domain adaptation [27, 28]. Recently, other promising related research incorporates prior domain-knowledge into the neural network, by embedding differentiable signal processing blocks directly into the architecture [29].

3. EXPERIMENT SETUP

In this Section details are given about the conducted experiment, including the data used, the model architecture and training details, as well as the metrics employed for evaluation.

3.1 Data

In the following, we briefly describe the drum dataset used throughout our experiments, as well as the Audio Commons feature models, with which we extract perceptive features from the dataset.

3.1.1 Dataset

For this work, we make use of an internal, non-publicly available dataset of approximately 300k one-shot audio samples aligned and distributed across a balanced set of kick, snare, and cymbal sounds. The samples originally have a sample rate of 44.1kHz and variable lengths. In order to make the task simpler, each sample is shortened to a duration of one second and down-sampled to a sample rate of 16kHz. For each audio sample, we extract perceptual features with the Audio Commons timbre models (see Section 3.1.2). We perform an 90% / 10% split of the dataset for validation purposes. The model is trained on the real and imaginary components of the Short-Time Fourier Transform (STFT), which has been shown to work well in [30]. We compute the STFT using a window size of 2048 samples and 75% overlapping. The generated spectrograms are then simply inverted back to the signal domain using the inverse STFT.

3.1.2 Audio-Commons Timbre Models

The Audio Commons project [3] implements a collection of perceptual models of features that describe high-level timbral properties of the sound. These features are designed from the study of popular timbre ratings given to a collection of sounds obtained from Freesound [4]. The models are built by combining existing low-level features found in the literature (e.g., spectral centroid, dynamic-range, spectral energy ratios, etc), which correlate with the target properties enumerated below. All features are defined in the range [0-100]. We employ these features as conditioning to the generative model. For more information, we direct the reader to the project deliverable.[3]

- **brightness**: refers to the clarity and amount of high-pitched content in the analyzed sound. It is computed from the spectral centroid and the spectral energy ratio.

---

[3] https://www.audiocommons.org/2018/07/15/audio-commons-audio-extractor.html
[4] https://freesound.org/

- **hardness**: refers to the stiffness or solid nature of the acoustic source that could have produced a sound. It is estimated using a linear regression model on spectral and temporal features extracted from the attack segment of a sound event.

- **depth**: refers to the sensation of perceiving a sound coming from an acoustic source beneath the surface. A linear regression model estimates depth from the spectral centroid of the lower frequencies, the proportion of low frequency energy and the low-frequency limit of the audio excerpt.

- **roughness**: refers to the irregular and uneven sonic texture of a sound. It is estimated from the interaction of peaks and nearby bins within frequency spectral frames. When neighboring frequency components have peaks with similar amplitude, the sound is said to produce a 'rough' sensation.

- **boominess**: refers to a sound with deep and loud resonant components. [5]

- **warmth**: refers to sounds that induce a sensation analogous to that caused by the physical temperature. [5]

- **sharpness**: refers to a sound that might cut if it were to take on physical form. [5]

3.2 Architecture Design and Training Procedure
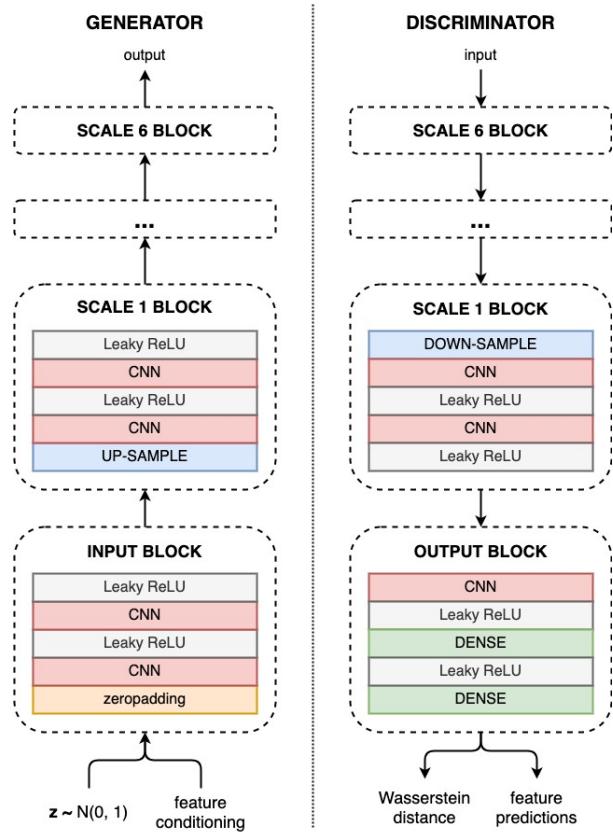
In the following, we will introduce the architecture and training of DrumGAN, and will briefly describe the baseline model against which DrumGAN is evaluated.

3.2.1 Generative Adversarial Networks

Generative Adversarial Networks (GAN) are a family of training procedures inspired by game theory, in which a generative model competes against a discriminative adversary, that learns to distinguish whether a sample is real or fake [14]. The generative network, or Generator (G), estimates a distribution p_g over the data x by learning a mapping of an input noise p_z to data space as $G_\theta(z)$, where G_θ is a neural network implementing a differentiable function with parameters θ. Inversely, the discriminator $D_\beta(x)$, with parameters β is trained to output a single scalar indicating whether the input comes from the real data p_r or from the generated distribution p_g. Simultaneously, G is trained to produce samples that are identified as real by the discriminator. Competition drives both networks until an equilibrium point is reached and the generated examples are indistinguishable from the original data. For a Wasserstein GAN, as used in our experiments, the training criterion is formally defined as

$$\min_G \max_D \Gamma(D, G) = \frac{1}{N} \sum_i D(x^i) - D(G(z^i)). \quad (1)$$

---

[5] Description of the calculation method for this feature is not available to the authors at current time.

Figure 1. Proposed architecture for DrumGAN (see Section 3.2 for details).

3.2.2 Proposed Architecture

In the proposed architecture, the input to G is a concatenation of the 7 conditioning features c, described in Section 3.1.2, and a random vector z with 128 components sampled from an independent Gaussian distribution. The resulting vector is fed through a stack of convolutional and box up-sampling blocks to generate the output signal $x = G_\theta(z; c)$. In order to turn the 1D input vector into a 4D convolutional input, it is zero-padded in the time- and frequency-dimension (i.e., placed in the middle of the convolutional input with $128 + 7$ convolutional maps). As depicted in Figure 1, the generator's input block performs this zero-padding followed by two convolutional layers with ReLU non-linearity. Each scale block is composed of one box up-sampling step at the input and two convolutional layers with filters of size $(3, 3)$. The number of feature maps decreases from low to high resolution as {256, 128, 128, 128, 64, 32}. Up-sampling of the temporal dimension is just performed after the 3rd scale block. We use a Leaky ReLU as activation functions and apply pixel normalization after every convolutional step (i.e., normalizing the norm over the output maps at each position). The discriminator D is composed of convolutional and down-sampling blocks, mirroring G's configuration. Given a batch of either real or generated STFT audio, D estimates the Wasserstein distance between the real and generated distributions [24], and predicts the perceptual features ac-

companing the input audio in the case of a real batch, or those used for conditioning in the case of generated audio. In order to promote the usage of the conditioning information by G, we add an auxiliary Mean Squared Error (MSE) loss term to the objective function, following a similar approach as in [31]. We use a gradient penalty of 10.0 to satisfy the Lipschitz continuity condition of Wasserstein GANs. The weights are initialized to zero and we apply layer-wise normalization at run-time using He's constant [32] to promote an equalized learning. A mini-batch standard deviation layer before the output block of D encourages G to generate more variety and, therefore, reduce mode collapse [25].

3.2.3 Training Procedure

Training follows the procedure of Progressive Growing of GANs (PGANs), first used for image generation [10], which has been successfully applied to audio synthesis of pitched sounds [6]. In a PGAN, the architecture is built dynamically during training. The process is divided into training iterations that progressively introduce new blocks to both the Generator and the Discriminator, as depicted in Figure 1. While training, a blending parameter α progressively fades in the gradient derived from the new blocks, minimizing possible perturbation effects. The models are trained for 1.1M iterations on batches of 30, 30, 20, 20 12 and 12 samples, respectively for each scale. Each scale is trained during 200k iterations except the last one, which is trained up to 300k iterations. We employ Adam as the optimization method and a learning rate of 0.001 for both networks.

3.2.4 The U-Net Baseline

As mentioned in the introduction, we compare DrumGAN against a previous work tackling the exact same task (i.e., neural synthesis of drums sounds, conditioned on the same perceptual features described in Section 3.1.2), but using a U-Net architecture operating in the time domain [9]. The U-Net model is trained to deterministically map the conditioning features (and an envelope of the same size as the output) to the output. The dataset used thereby consists of 11k drum samples obtained from Freesound[6], which includes kicks, snares, cymbals, and other percussion sounds (referred to as *Freesound drum subset* in the following).

3.3 Evaluation

Assessing the quality of synthesized audio is hard to formalize making the evaluation of generative models for audio a challenging task. In the particular case of GANs, where no explicit likelihood maximization exists, a common evaluation approach is to measure the model's performance in a variety of surrogate tasks [33]. As described in the following, we evaluate our models against a diverse set of metrics that capture distinct aspects of the model's performance.

---

[6] www.freesound.org

3.3.1 Inception Score

The *Inception Score (IS)* [25] penalizes models that generate examples that are not classified into a single class with high confidence, as well as models whose examples belong to only a few of all the possible classes. It is defined as the mean KL divergence between the conditional class probabilities $p(y|x)$, and the marginal distribution $p(y)$ using the class predictions of an Inception classifier (see Eq. 2). We train our Inception Net[7] variant to classify kicks, snares and cymbals, from Mel-scaled magnitude STFT spectrograms using the same train/validation split of 90% / 10%, used throughout our experiments. As additional targets, we also train the model to predict the extracted perceptual features described in Section 3.1.2 (using mean-squared error cost).

$$IS = \exp\big(E_x[KL(p(y|x)||p(y))]\big) \qquad (2)$$

3.3.2 Fréchet Audio Distance (FAD)

The *Fréchet Audio Distance* compares the statistics of real and generated data computed from an embedding layer of a pre-trained VGG-like model[8] [34]. FAD fits a continuous multivariate Gaussian to the output of the embedding layer for real and generated data and the distance between these is calculated as:

$$FAD = ||\mu_r - \mu_g||^2 + tr(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r \Sigma_g}) \quad (3)$$

where (μ_r, Σ_r) and (μ_g, Σ_g) are the mean and covariances of the embedding of real and generated data respectively. The lower the FAD, the smaller the distance between distributions of real and generated data. FAD is robust against noise, consistent with human judgments and more sensible to intra-class mode dropping than IS.

3.3.3 Kernel Inception Distance (KID)

The KID measures the dissimilarity between samples drawn independently from real and generated distributions [35]. It is defined as the squared Maximum Mean Discrepancy (MMD) between representations of the last layer of the Inception model (described in Section 3.3.1). A lower MMD means that the generated p_g and real p_r distributions are close to each other. We employ the unbiased estimator of the squared MMD [36] between m samples $x \sim p_r$ and n samples $y \sim p_g$, for some fixed characteristic kernel function k, defined as

$$
\begin{aligned}
MMD^2(X,Y) = \quad & \frac{1}{m(m-1)}\sum_{i\neq j}^{m} k(x_i, x_j) \\
+ \quad & \frac{1}{n(n-1)}\sum_{i\neq j}^{n} k(y_i, y_j) \\
- \quad & \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i, y_j).
\end{aligned}
\qquad (4)
$$

Here, we use an inverse multi-quadratic kernel (IMQ) $k(x,y) = 1/(1 + ||x-y||^2/2\gamma^2)$ with $\gamma^2 = 8$ [37], which

---

[7] https://github.com/pytorch/vision/blob/master/torchvision/models/inception.py
[8] https://github.com/google-research/google-research/tree/master/frechet_audio_distance

has a heavy tail and, hence, it is sensitive to outliers. We borrow this metric from the Computer Vision literature and apply it to the audio domain. We train a separate inception model on the FreeSound drum subset used for the U-Net baseline experiments (see Section 3.2.4). This is done to allow comparison of the inception-based metrics with DrumGAN. Since the FreeSound drum subset doesn't contain annotations of the instrument type, we train our variant on just the feature regression task, and restrict our comparison to KID and FAD, as these metrics do not compare class probabilities but embedding distributions.

3.3.4 Feature Coherence

We follow the methodology proposed by [9] for evaluating the feature control coherence. The goal is to assess whether increasing or decreasing a specific feature value of the conditioning input yields the corresponding change of that feature in the synthesized audio. To this end, a specific feature i is set to 0.2 (low), 0.5 (mid), and 0.8 (high), keeping the other features and the input noise fixed. The resulting outputs $x_{\text{low}}^i, x_{\text{mid}}^i, x_{\text{high}}^i$ are then evaluated with the Audio Commons Timbre Models (yielding features fx^i). Then, it is assessed if the feature of interest changed as expected (i.e., $fx_{\text{low}}^i < fx_{\text{mid}}^i < fx_{\text{high}}^i$). More precisely, three conditions are evaluated: E1: $fx_{\text{low}}^i < fx_{\text{high}}^i$, E2: $fx_{\text{mid}}^i < fx_{\text{high}}^i$, and E3: $fx_{\text{low}}^i < fx_{\text{mid}}^i$. We perform these three tests 1000 times for each feature, always with different random input noise and different configurations of the other features (sampled from the evaluation set). The resulting accuracies are reported.

4. RESULTS AND DISCUSSION

In this section, we briefly describe our subjective impression when listening to the model output, and we will give an extended discussion on the quantitative analysis, including the comparison with the baseline U-Net architecture.

4.1 Subjective Evaluation and Generation Tests

The results of the qualitative experiments discussed in this section can be found on the accompaniment website. [9] In general, conditional DrumGAN seems to have better quality than its unconditional counterpart and substantially better than the U-Net baseline (see Section 3.2.4). In the absence of more reliable baselines, we argue that the perceived quality of DrumGAN is comparable to that of previous state-of-the-art work on adversarial audio synthesis of drums [3].

We also perform radial and spherical interpolation experiments (with respect to the Gaussian prior) between random points selected in the latent space of DrumGAN. Both interpolations yield smooth and perceptually linear transitions in the audio domain. We notice that radial interpolation tend to change the percussion type (i.e., kick, snare, cymbal) of the output, while spherical interpolation affects other properties (like within-class timbral characteristics and envelope) of the synthesized audio. This gives a hint on how the latent manifold is structured.

---

[9] https://sites.google.com/view/drumgan

	IS	KID	FAD
real data	2.26	0.05	0.00
train feats	**2.19**	0.39	0.77
val feats	2.18	**0.35**	0.76
rand feats	2.09	1.36	**0.70**
unconditional	**2.19**	1.07	1.00

Table 1. Results of Inception Score (IS, higher is better), Kernel Inception Distance (KID, lower is better) and Fréchet Audio Distance (FAD, lower is better), scored by DrumGAN under different conditioning settings, against real data and the unconditional baseline. The metrics are computed over 50k samples, except for *val feats*, where 30k samples are used (i.e., the validation set size).

	KID	FAD
real data	0.04	0.00
real feats	1.45	3.09
rand feats	13.94	3.17

Table 2. Results of Kernel Inception Distance (KID) and Fréchet Audio Distance (FAD), scored by the U-Net baseline [9] when conditioning the model on feature configurations from the real data and on randomly sampled features. The metrics are computed over 11k samples (i.e., the Freesound drum subset size).

4.2 Quantitative Results

4.2.1 Scores and Distances

Table 1 shows the DrumGAN results for the Inception Score (IS), the Kernel Inception Distance (KID), and the Fréchet Audio Distance (FAD), as described in Section 3.3. These metrics are calculated on the synthesized drum sounds of the model, based on different conditioning settings. Besides the unconditional setting of DrumGAN (*unconditional*), we use feature configurations from the train set (*train feats*), the valid set (*valid feats*), and features randomly sampled from a uniform distribution (*rand feats*). The IS of DrumGAN samples is close to that of the real data in most settings. This means that the model outputs are clearly assignable to either of the respective percussion-type classes (i.e., low entropy for kick, snare, and cymbal posteriors), and that it doesn't omit any of them (i.e., high entropy for the marginal over all classes). The IS is slightly reduced for random conditioning features, indicating that using uncommon conditioning configurations makes the outputs more ambiguous with respect to specific percussion types. While FAD is a measure for the perceived quality of the individual sounds (measuring co-variances within data instances), the KID reflects if the generated data overall follows the distribution of the real data. Therefore, it is interesting to see that *rand feats* cause outputs which overall do not follow the distribution of the real data (i.e., high KID), but the individual outputs are still plausible percussion samples (i.e., low FAD). This quantitative result is in-line with the perceived quality of

Feature	U-Net			DrumGAN		
	E1	E2	E3	E1	E2	E3
brightness	**0.99**	**0.99**	**1.00**	0.74	0.71	0.70
hardness	0.64	**0.65**	0.59	0.64	0.64	**0.62**
depth	**0.94**	0.65	**0.94**	0.79	**0.72**	0.74
roughness	0.63	0.59	0.57	**0.72**	**0.68**	**0.67**
boominess	**0.98**	**0.82**	**0.98**	0.80	0.74	0.77
warmth	**0.92**	**0.79**	**0.91**	0.76	0.71	0.71
sharpness	0.63	0.77	0.45	**0.84**	**0.82**	**0.82**
average	**0.83**	**0.76**	**0.78**	0.76	0.72	0.72

Table 3. Mean accuracies for the feature coherence tests on samples generated with the baseline U-Net [9] and DrumGAN.

the generated samples (see Section 4.1). In the unconditional setting, both KID and FAD are worse, indicating that feature conditioning helps the model to both generate data following the true distribution, overall, as well as in individual samples.

Table 2 shows the evaluation results for the U-Net architecture (see Section 3.2.4). As the train / valid split for the Freesound drum subset (on which the U-Net was trained) is not available to the authors, the U-Net model is tested using the features of the full Freesound drum subset (*real feats*), as well as random features. Also, we do not report the IS for the U-Net architecture, as it was trained on data without percussion-type labels, making it impossible to train the inception model on such targets. As a baseline, all metrics are also evaluated on the real data on which the respective models were trained. While evaluation on the real data is straight-forward for the IS (i.e., just using the original data instead of the generated data to obtain the statistics), both KID and FAD are measures usually *comparing* the statistics between features of real and generated data. Therefore, for the *real data* baseline, we split the real data into two equal parts and compare those with each other in order to obtain KID and FAD. The performance of the U-Net approach on both, KID and FAD is considerably worse than that of DrumGAN. While the KID for *real feats* is still comparable to that of DrumGAN (indicating a distribution similar to that of the real data), the high FAD indicates that the generated samples are not perceptually similar to the real samples. When using random feature combinations this trend is accentuated moderately in the case of FAD, and particularly in the case of the KID, reaching a maximum of almost 14. This is, however, intelligible, as the output of the U-Net depends only on the input features in a deterministic way. Therefore, it is to expect that the distribution over output samples changes fully when fully changing the distribution of the inputs.

4.2.2 Feature Coherence

Table 3 shows the accuracy of the three feature coherence tests explained in Section 3.3.4. Note that, as both models were trained on different data, the figures of the two models are not directly comparable. However, also reporting the figures of the U-Net approach should provide some context on the performance of our proposed model. In ad-

dition, as both works use the same feature extractors and claim that the conditional features are used to shape the same characteristics of the output, we consider the figures from the U-Net approach a useful reference. We can see that for about half the features, the U-Net approach reaches close to 100% accuracy. Referring to the descriptions on how the features are computed it seems that the U-Net approach reaches particularly high accuracies for features which are computed by looking at the global frequency distribution of the audio sample, taking into account spectral centroid and relations between high and low frequencies (e.g., brightness and depth). U-Net performs considerably worse for features which take into account the temporal evolution of the sound (e.g., hardness) or more complex relationships between frequencies (e.g., roughness). While DrumGAN performs worse on average on these tests, the results seem to be more consistent, with less very high, but also less rather low accuracy values (note that the random-guessing baseline is 0.5 for all the tests). The reason for not performing better on average may lie in the fact that DrumGAN is trained in an adversarial fashion, where the dataset distribution is enforced, in addition to obeying the conditioned characteristics. In contrast, in the U-Net approach the model is trained deterministically to map the conditioning features to the output, which makes it easier to satisfy the simpler characteristics, like generating a lot of low- or high-frequency content. However, this deterministic mapping results in a lower audio quality and a worse approximation to the true data distribution, as it can be seen in the KID and FAD figures, described above.

5. CONCLUSIONS AND FUTURE WORK

In this work, we performed percussive sound synthesis using a GAN architecture that enables steering the synthesis process using musically meaningful controls. To this end, we collected a dataset of approximately 300k audio samples containing kicks, snares, and cymbals. We extracted a set of timbral features, describing high-level semantics of the sound, and used these as conditional input to our model. We encouraged the generator to use the conditioning information by performing an auxiliary feature regression task in the discriminator and adding the corresponding MSE loss term to the objective function. In order to assess whether the feature conditioning improves the generative process, we trained a model in a completely unsupervised manner for comparison. We evaluated the models by comparing various metrics, each reflecting different characteristics of the generation process. Additionally, we compared the coherence of the feature control against previous work. Results showed that DrumGAN generates high-quality drum samples and provides meaningful control over the audio generation. The conditioning information was proven useful and enabled the network to better approximate the real distribution of the data. As future work, we will focus on scaling the model to work with audio production standards (e.g., 44.1kHz sample rate, stereo audio), and implement a plugin that can be used in a conventional Digital Audio Workstation (DAW).

6. REFERENCES

[1] R. Gordon, "Synthesizing drums: The snare drum," *Sound On Sound*, Jan. 2002. [Online]. Available: https://www.soundonsound.com/techniques/synthesizing-drums-snare-drum

[2] ——, "Synthesizing drums: The bass drum," *Sound On Sound*, Jan. 2002. [Online]. Available: https://www.soundonsound.com/techniques/synthesizing-drums-bass-drum

[3] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *Proc. of the 7th International Conference on Learning Representations, ICLR*, May 2019.

[4] C. Aouameur, P. Esling, and G. Hadjeres, "Neural drum machine: An interactive system for real-time synthesis of drum sounds," in *Proc. of the 10th International Conference on Computational Creativity, ICCC*, June 2019.

[5] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *Proc. of the 34th International Conference on Machine Learning, ICML*, Sydney, NSW, Australia, Aug. 2017.

[6] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," in *Proc. of the 7th International Conference on Learning Representations, ICLR*, May 2019.

[7] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Universal audio synthesizer control with normalizing flows," *Journal of Applied Sciences*, 2019.

[8] S. Lattner and M. Grachten, "High-level control of drum track generation using learned patterns of rhythmic interaction," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, New Paltz, NY, USA, Oct. 2019.

[9] A. Ramires, P. Chandna, X. Favory, E. Gómez, and X. Serra, "Neural percussive synthesis parameterised by high-level timbral features," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, May 2020.

[10] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations, ICLR*, May 2018.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, June 2009.

[12] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Proc. of the 9th ISCA Speech Synthesis Workshop*, Sunnyvale, CA, USA, Sept. 2016.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. of the 2nd International Conference on Learning Representations, ICLR*, Banff, AB, Canada, Apr. 2014.

[14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of the 28th International Conference on Neural Information Processing Systems NIPS*, Montreal, Quebec, Canada, Dec. 2014.

[15] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proc. of the 33rd International Conference on Machine Learning, ICML*, New York City, NY, USA, June 2016.

[16] S. Vasquez and M. Lewis, "Melnet: A generative model for audio in the frequency domain," *CoRR*, vol. abs/1906.01083, 2019.

[17] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, Calgary, AB, Canada, Apr. 2018.

[18] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces," in *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR*, Paris, France, Sept. 2018.

[19] ——, "Generative timbre spaces with variational audio synthesis," in *Proc. of the 21st International Conference on Digital Audio Effects DAFx-18*, Aveiro, Portugal, Sept. 2018.

[20] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *CoRR*, vol. abs/2005.00341, 2020.

[21] Y. Saito, S. Takamichi, and H. Saruwatari, "Statistical parametric speech synthesis incorporating generative adversarial networks," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 26, pp. 84–96, 2018.

[22] T. Kaneko and H. Kameoka, "Parallel-data-free voice conversion using cycle-consistent adversarial networks," *CoRR*, vol. abs/1711.11293, 2017.

[23] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "Timbretron: A wavenet(cyclegan(cqt(audio))) pipeline for musical timbre transfer," in *Proc. of the 7th International Conference on Learning Representations, ICLR*, New Orleans, LA, USA, May 2019.

[24] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Proc. of the International Conference on Neural Information Processing Systems, NIPS*, Long Beach, CA, USA, Dec. 2017.

[25] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Proc. of the International Conference on Neural Information Processing Systems, NIPS*, Barcelona, Spain, Dec. 2016.

[26] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Proc. of the Annual Conference on Neural Information Processing Systems, NIPS*, Vancouver, BC, Canada, Dec. 2019.

[27] E. Hosseini-Asl, Y. Zhou, C. Xiong, and R. Socher, "A multi-discriminator cyclegan for unsupervised non-parallel speech domain adaptation," in *Proc. of the 19th Annual Conference of the International Speech Communication Association*, Hyderabad, India, Sept. 2018.

[28] D. Michelsanti and Z. Tan, "Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification," in *Proc. of the 18th Annual Conference of the International Speech Communication Association, Interspeech*, Stockholm, Sweden, Aug. 2017.

[29] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: differentiable digital signal processing," in *Proc. of the 8th International Conference on Learning Representations, ICLR*, Addis Ababa, Ethiopia, Apr. 2020.

[30] J. Nistal, S. Lattner, and G. Richard, "Comparing representations for audio synthesis using generative adversarial networks," in *Proc. of the 28th European Signal Processing Conference, EUSIPCO2020*, Amsterdam, NL, Jan. 2021.

[31] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proc. of the 34th International Conference on Machine Learning, ICML*, Sydney, NSW, Australia, Aug. 2017.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE International Conference on Computer Vision, ICCV*, Santiago, Chile, Dec. 2015.

[33] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," in *Proc. of the 4th International Conference on Learning Representations, ICLR*, San Juan, Puerto Rico, May 2016.

[34] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A metric for evaluating music enhancement algorithms," *CoRR*, vol. abs/1812.08466, 2018.

[35] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD gans," in *Proc. of the 6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada, Apr. 2018.

[36] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, pp. 723–773, 2012.

[37] R. M. Rustamov, "Closed-form expressions for maximum mean discrepancy with applications to wasserstein auto-encoders," *CoRR*, vol. abs/1901.03227, 2019.

A DEEP LEARNING BASED ANALYSIS-SYNTHESIS FRAMEWORK FOR UNISON SINGING

Pritish Chandna[1]   **Helena Cuesta**[1]   **Emilia Gómez**[2,1]

[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona
[2] European Commission, Joint Research Centre, Seville

`{pritish.chandna, helena.cuesta, emilia.gomez}@upf.edu`

ABSTRACT

Unison singing is the name given to an ensemble of singers simultaneously singing the same melody and lyrics. While each individual singer in a unison sings the same principle melody, there are slight timing and pitch deviations between the singers, which, along with the ensemble of timbres, give the listener a perceived sense of "unison". In this paper, we present a study of unison singing in the context of choirs; utilising some recently proposed deep-learning based methodologies, we analyse the fundamental frequency (F0) distribution of the individual singers in recordings of unison mixtures. Based on the analysis, we propose a system for synthesising a unison signal from an *a cappella* input and a single voice prototype representative of a unison mixture. We use subjective listening tests to evaluate perceptual factors of our proposed system for synthesis, including quality, adherence to the melody as well the degree of perceived unison.

1. INTRODUCTION

Throughout history, singing has been an important cultural activity for humans, serving for propagation of beliefs and ideas amongst the masses as well as for social entertainment. The social aspect led to gatherings of people singing in a group, which evolved into polyphonic ensemble singing with multiple voices singing counterpoint melodies in complex harmonies. A group of people singing in such an ensemble is commonly termed as a choir and the focus of our study is on one setting of such choirs consisting of four voices known as Soprano, Alto, Tenor and Bass (SATB). Each voice within an SATB ensemble has its own function and melodic range in the whole. SATB is one of the most widely studied, documented, and practiced forms of choirs with numerous dedicated conservatories across Europe, highlighting the cultural importance of the art form. Within each of the SATB voices, it is common to have multiple singers of similar

vocal range singing the same melody simultaneously, in a form known as unison singing. While all the singers in a unison sing the same melody, it is impossible for a group of two or more people to perfectly synchronize and sing the exact same pitch line. Each singer has their own natural micro-deviations, both in terms of timing and pitch, from the prescribed score and their own distinct timbre. The combination of micro-deviations and the ensemble of timbres leads to the perception of unison, wherein several singers are perceived to be singing a single pitch contour [1], and is the main focus of our study.

Pitch and fundamental frequency (F0) are related but not equivalent terms. While the F0 generally refers to the physical frequency of vibrations of the vocal folds for a singing voice signal, pitch refers to an abstract perceptual concept which has been found to be closely correlated to the F0. Frequency is usually measured in Hertz, representing the number of cycles of a periodic signal per second, whereas pitch is described in terms of perceptual units like *cents*. The cent is a unit defined on a logarithmic scale, as a measure of the ratio between the frequency in Hertz and a base frequency, commonly chosen to be $440\,\mathrm{Hz}$, as shown in Equation 1.

$$f_{cents} = 1200 \cdot log_2 \frac{f_{hertz}}{440} \qquad (1)$$

Thus defined, the cent is correlated to the perceptually relevant musical unit of an equally tempered semitone. Specifically, one semitone spans $100\,\mathrm{cents}$. Examined individually, the pitch of the singers in a unison can be represented by the F0 of each individual singer's vocal signal, this can be tested by synthesising a time-varying sinusoid with the frequency of the signal. However, when the individual signals are added the resultant pitch is not merely the sum of F0 value, and the methodology of synthesising a sum of sinusoids as with single singers fails to produce the same perceptual result due to physical phenomena such as beating, among others. Past studies have utilized artificial unison mixes created by the use of a vowel only singing voice synthesizer to study the perception of a single pitch contour in a unison [1]. Other areas of past research related to unison singing include single voice to unison synthesis models, based on creating voice clones with variations in the input [2, 3]. Fuelled by the deep learning revolution, singing voice synthesizers have evolved over the

last few years, allowing us to take a step further both into exploring the perception of unison and into effective solo voice to unison synthesis. We build on the work done by Ternström [1] by leveraging recently proposed synthesis methodologies to synthesize a single voice prototype representing the melodic and linguistic content of a unison mixture. This allows use to further test the hypothesis of a single F0 contour representative of the perceived pitch of a unison via subjective listening tests. We also verify the author's findings by analysing a set of real recordings of unison singing. In addition, we propose a methodology combining previous research and recently developed techniques to synthesize a unison mixture from a single voice input. We follow the basic methodology used by Schnell et al. [2] to create voice clones with variations in three aspects; pitch, timing and timbre, and use perceptual evaluation tests to evaluate the effect of each of these parameters on the perception of the sense of a unison.

The rest of the paper is structured as follows. Section 2 discusses previous works pertinent to our study. We then present the analysis of the choir recordings in Section 3, including a description of the dataset of choir recordings, the methodology used for the analysis and the results of the analysis. The synthesis methodology we use for synthesizing voice clones and the single voice prototype of the unison mixture is described in Section 4. Section 4.3 presents the perceptual evaluation methodology used and the results of the perceptual tests. Finally, we present a discussion on our findings in the analysis of the choir and the perceptual evaluation of the synthesis in Section 6.

2. RELATED WORK

We divide the description of related works into three sections: past studies into the analysis of the perception of unison, previous works on synthesising unison mixtures from choirs and recently proposed deep-learning based methodologies what we will use for analysis and synthesis.

2.1 Analysis Of Unison Perception

The perception of pitch dispersion has previously been studied in [1], wherein the author used synthesized singing voice stimuli to investigate the preferences of expert listeners in unisons. In the study, pitch dispersion is defined as the bandwidth of the fundamental frequency and its harmonic partials across individual singers in a unison. It is suggested that this dispersion is related to the *flutter*—small variations in F0 that are too fast to be perceived as pitch variations. The concept of pitch *scatter* is presented in the study as the standard deviation over voices in the mean F0: the average F0 computed over the duration of each tone of a song. The study concludes that a *scatter* of 0 cents–5 cents was preferred by the participants while a *scatter* of 5 cents–14 cents was seen as the limit of tolerance before dissonance was reported. In addition, the author also highlights several differences between solo and ensemble singing. For instance, a single performer produces tones with well-defined properties: pitch, timing,

loudness, timbre, while an ensemble of performers produces sounds with statistical distributions of each of these properties.

A similar method for modelling *scatter* in choir sections was presented by Cuesta et al. [4] using small windows to compute the standard deviation between individual F0s in the unison. This study used real recordings of choral singing instead of synthesised stimuli, presenting a mathematical model for dispersion rather than a perceptual evaluation. For the dataset used in the research, F0 (or pitch) dispersion was found to be in the range of 20 cents–30 cents for all SATB voice sections, being slightly larger in the Bass.

Another recent study focused on the analysis of F0 in vocal music is work by Weiss et al. [5], where the authors proposed an approach to measure intonation quality of choir recordings. They create an ideal 12-tone equal temperament grid, and then calculate the deviation of each F0 and its partials to their theoretical position in the grid. The overall deviation is computed as a weighted sum of each partial's deviation. This method enables the analysis of the overall intonation of a full choir recording, but does not account for the deviations within sections of the choir.

2.2 Unison Synthesis

Signal processing techniques have previously been utilised to synthesize choir unison by adding several clones of a monophonic *a cappella* signal with uncorrelated pitch, timing, and timbre deviations. Most particularly, Pitch Synchronous Overlap Add (PSOLA) methods [2] have been exploited as an analysis-synthesis framework to decompose the vocal signal into a set of constituent waveforms representing successive pitch periods of the signal. Pitch and timing deviations are added to the vocal signal using time stretching and pitch shifting techniques to create voice clones, which are combined to form the output unison signal.

Other proposed methodologies for creating a unison output from an *a cappella* signal include morphing the spectral and pitch components of the vowels of the input signal as in [3]. The methodology's effectiveness is constrained to low tempo inputs . Random modulation of beating partials to create a choral effect [6] has also been used.

2.3 Deep Learning For Analysis and Synthesis

For our work, we build on the work done in [1] and [4], modelling the perceptual pitch contour of a unison mixture as a single F0 contour. To this end, we use a recently proposed Convolutional Representation for Pitch Estimation (CREPE) methodology [7] for extracting F0 contours from real world recordings of individual singers in a choir setting as well the F0 contour of unison mixture created by combining the individual voices. This methodology uses a series of convolutional operations on the waveform of the input signal and outputs a probability distribution over a discrete representation of the underlying F0 contour of the signal across a series of time-frames.

To synthesize the single voice prototype representing a unison mixture output and the voice clones for creating a unison mixture from a single voice input, we adapt the methodology proposed by Chandna et al. [8], which allows for the re-synthesis of a solo single voice from a musical mixture input via the underlying linguistic features. This methodology builds on the idea of re-synthesizing a vocal signal from a musical mixture by estimating the parameters of a vocoder synthesizer [9] and uses an encoder built of a bank of bi-directional long short-term memory (LSTM) recurrent neural networks (RNNs) to estimate a continuous representation of the underlying linguistic features present in the input mixture signal. The continuous representation is singer-independent and language agnostic, and was initially proposed for zero-shot voice conversion via an autoencoder network [10]. The linguistic features can then be used to generate the spectral envelope of the vocal signal in the mixture, providing the singer identity. The authors of [8] proposed two decoders for this process, a Singer Dependent Network (SDN) which takes the singer identity as a one-hot vector, and a Singer Independent Network (SIN) which intrinsically learns the singer identity from the given input. The spectral envelope is then combined with the F0, extracted via an external algorithm, to synthesize the vocal signal. While the original framework was proposed for extracting a singing voice from a pop/rock musical mixture, we adapt the SDN network for synthesising a unison mixture from an *a cappella* input and the SIN network for synthesizing an *a cappella* singing voice from a unison mixture. The adaptations we apply are described in Section 4. The SIN and SDN models [8] were trained on a proprietary dataset with 12 hours of data comprising 205 songs by 45 male and female singers, and we have obtained a copy of the trained model with permission from the relevant authorities for our study.

3. ANALYSIS OF CHOIR RECORDINGS

We analyse the variations between individual singers in a unison in terms of variance in pitch and timing. Below, we present the dataset that we use in our analysis, followed by the methodology used for analysis, and finally the results of our analysis.

3.1 Datasets

We analyse the Choral Singing Dataset (CSD) [4], which includes monophonic recordings of 3 choral pieces: *Niño Dios d'Amor Herido*, written by Francisco Guerrero, *Locus Iste*, written by Anton Bruckner, and *El Rossinyol*, a Catalan popular song. There are 16 different singers for each song with 4 singers for each of the four parts; Soprano, Alto, Tenor and Bass. The dataset also includes manually corrected F0 annotations for each track.

3.2 Analysis Methodology

To analyse inter-singer variance in **pitch**, the first step is the extraction of an F0 contour from a unison mixture of singers. We aim to study the behavior of a monophonic

F0 extractor in such cases, assuming that we have a sufficiently balanced unison performance, where the contribution of each singer is similar in terms of volume and energy. To this end, we use CREPE [7] to extract the fundamental frequency of the unison mixture created by summing and normalizing all corresponding individual singers in each vocal part of the recordings. This is termed as $EstF0_U$.

We then measure the resemblance of the estimated $EstF0_U$ to each of the manually annotated F0 tracks and to the mean $F0_m$ [1] . We use standard evaluation metrics for melody extraction including *Raw Pitch Accuracy (RPA), Overall Accuracy (OA), Voicing Recall (VR)* and *Voicing False Alarm (VFA)* between the $EstF0_U$, the average the mean $F0_m$, and each individual singer curve, $GTF0_i$ [2]

Once we have verified the accuracy of the extraction system, we build a statistical model for the individual contours in the unison, as suggested by [1]. In our model, the framewise F0 of an individual singer, $F0_i$, can be represented as a distribution of values around the mean $F0_m$ with a deviation of F_{devi}, as shown in Equation 2

$$F0_i = F0_M + F_{devi} \qquad (2)$$

This equation also allows us to define the $F0_{i+1}$ of a singer in terms of the $F0_i$ of another singer in the unison as

$$F0_{i+1} = F0_m + F_{devi+1}$$
$$F0_{i+1} - F0_i = F_{devi+1} - F_{devi}$$
$$F0_{i+1} = F0_i + F_{devi+1} - F_{devi}$$
$$F0_{i+1} = F0_i + \Delta F0_s \qquad (3)$$

Where we define $\Delta F0_s$ as the inter-singer deviation, represented by Equation 4. For each pair of singers in the unison, we compute the frame-wise difference between the corresponding F0 contours in cents. For this calculation, only frames with positive F0 values, also known as *voiced* frames, were considered. We average these inter-singer deviations across time and songs, and obtain a single value for each group, i.e., SATB.
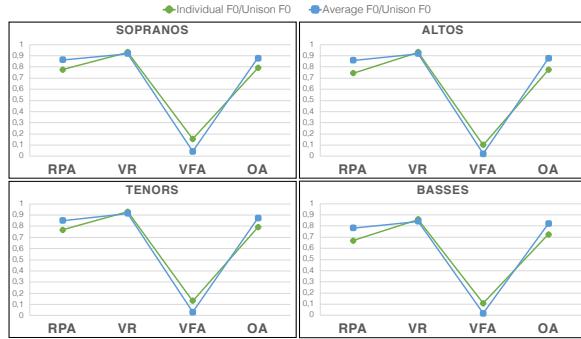
$$\Delta F0_s = \frac{\displaystyle\sum_{i=1}^{n} \sum_{j=i+1}^{n} \left| F0_i - F0_j \right|}{\binom{n}{2}} \qquad (4)$$

where the sub-index s indicates the choir section, $s \in [S, A, T, B]$, and n is the number of singers. In our use case, $n = 4$.
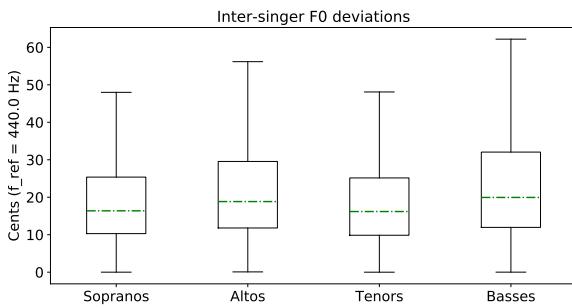
To study **timing** deviations, we focus in the transitions from *voiced* to *unvoiced*, and vice-versa—where singers are not entirely in sync. We call these regions *transition regions*, where some of the singers in the mixture are voiced

---

[1] Note that the average F0 value has to be adjusted for timing differences between the individual singers. To this end, we define the average to be zero (unvoiced frame) if and only if all individual values for that frame are zero. for all other cases, the average is calculated only accounting for the non-zero values.

[2] We use the `mir_eval` library [11] for this evaluation, and we use a pitch tolerance of 30 cents

Figure 1. Resemblance of the estimated unison $EstF0_U$ estimation to each individual $GTF0_i$ contour (green) and the average (blue) using pitch evaluation metrics averaged across each choir section.

Figure 2. Inter-singer deviations in cents averaged across the whole dataset for each choir section. Deviations are calculated using Equation 4.

and others are unvoiced, with a positive or zero F0. We measure the length of all the transition regions in every unison from the CSD, and then average across choir sections.

3.3 Analysis Results

A summary of the results of the comparison between the fundamental frequency extracted by CREPE, $EstF0_U$, and the manually corrected fundamental frequency, $GTF0_i$ is illustrated in Figure 1, along with a comparison with the mean, $F0_m$. We observe that all sections follow the same pattern with similar metric values, and the unison F0 estimated by CREPE, $EstF0_U$, is closer to the average $F0_m$, than to the individual contours. In addition, all metrics improve when we compare the average F0 curve to the extracted F0 contour from the unison: RPA, VR and OA are higher in the blue plots, while VFA is lower. We can thus use the pitch estimated by CREPE, $EstF0_U$, as a representative of the mean single pitch contour perceived in a unison mixture [1].

The calculated $\Delta F0_s$ is shown in Figure 2. We observe an inter-singer deviation in the range of $0\,\text{cents}$–$50\,\text{cents}$, with a mean of around $20\,\text{cents}$. This value, representing the inter-singer deviation in the unison mixtures, is comparable to the pitch dispersion studied by Cuesta et al. [4]. While the methodology for modelling is different, these re-

Section	Average Timing Deviation $\pm$ Standard Deviation
Soprano	0.134 ± 0.039 sec
Alto	0.093 ± 0.0024 sec
Tenor	0.100 ± 0.021 sec
Bass	0.124 ± 0.021 sec

Table 1. Timing deviations averaged across the CSD. These values measure the time span in which all singers in the unison transition from voiced to unvoiced, and vice-versa, averaged across all transitions in each song.

sults are in accordance with their reported per-section pitch dispersion: larger in the bass section, smaller in the sopranos, and very similar for altos and tenors.

Table 1 shows the results of the timing analysis. We observe an average timing deviation of $0.1\,\text{s}$ between the voices in the unison for all parts of the choir.

4. SYNTHESIS METHODOLOGY

We present two synthesis models, Solo To Unison (STU) to synthesize voice clones for creating a unison mixture from a single voice input, and Unison to Solo (UTS) for synthesizing single voice prototype representing the melodic and linguistic content of a unison mixture input [3]. Similar to the work presented by Schnell et al [2], we decompose the input signal into the F0, harmonic spectral envelope, and aperiodicity envelope. However, instead of using Pitch Synchronous Overlap Add (PSOLA) methods, we utilize the WORLD vocoder [12], which has been shown to be an effective vocoding system for singing voice synthesis [13, 14, 14]. Similar to [14], we use truncated frequency warping in the cepstral domain [15] to reduce the dimensions of the harmonic components from 1024 to 60 log Mel-Frequency Spectral Coefficients (MFSCs) with an all-pole filter with warping coefficient $\alpha = 0.45$. In addition, we use bandwise aperiodic analysis to reduce the dimensionality of aperiodic features to 4. For the rest of this paper, we refer to these 64 features together as the spectral envelope.

4.1 Unison to Solo (UTS)

As shown in Figure 3, we first perform a short-time Fourier transform (STFT) to extract a spectrogram from the input waveform. The magnitude part of the spectrogram is passed through the encoder proposed in [8] to extract a continuous representation of the linguistic features present in the unison mixture input. The linguistic features are decoded via the SIN network [8] to generate the spectral envelope for vocal synthesis. This envelope is combined

---

[3] Audio examples are provided as complementary material at `https://pc2752.github.io/unison_analysis_synthesis_examples/` and the source code with the trained models are available at `https://github.com/MTG/content_choral_separation`

with the pitch contour output from CREPE [7], $F0_U$ to synthesise the single voice prototype representing the unison mixture input.

4.2 Solo to Unison (STU)

The analysis part of the STU case follows a similar methodology, as we extract the linguistic features and the $EstF0_i$ contour from the input *a cappella* voice signal. To create voice clones with pitch and timing deviations, we add randomly sampled noise from a normal distribution with a mean of 0 and a variable standard deviation, termed as std. This represents the inter-singer deviation, $\Delta F0_s$, and allows us to model the $F0_{i+1}$ of the clone as per Equation 3. Timing deviations are added by shifting the voiced portions of the input signal or the portions between tow blocks of silence of more than $80\,\text{ms}$ by a variable amount, randomly sampled from a normal distribution of mean 0 and standard deviation ts. The values of std and ts are based on our analysis of the Choral Singing Dataset presented in Section 3.3

Finally, for variations in timbre, we generate the spectral envelope of a variable number singers, ns, of the same gender as the input using the SDN network proposed in [8]. This is based on our analysis presented in Section 3.3. There was no overlap between singers in the set used for training the synthesis model and the singers in the Choral Singing Dataset used for evaluation. The various voice clones are added together and normalized in amplitude to produce the final unison output. We evaluate various combinations of std, ts, and ns on their impact of the perception of unison.

4.3 Perceptual Evaluation Methodology

We used subjective listening tests with low and high anchors, as modified versions of the MUSHRA-methodology [16] to evaluate subjective criteria of the synthesis produced by our analysis synthesis framework.

While there are several aspects that could be evaluated, we focused on three keys aspects: adherence to melody, perception of unison, and audio quality. For each aspect, the participants were presented with 4 questions, one for each part of the SATB choir, and were asked to rate the test cases in the question on a continuous scale of 1–5 with respect to a presented reference. The test case and references provided pertained to the the same section of the song and were between $7\,\text{s}$–$10\,\text{s}$ each. The parameters used for these tests are described below for each aspect.

4.3.1 Adherence to melody and lyrics

For this aspect, we wanted to see the similarity of the perceived pitch contour of the output for both the UTS and STU cases to that of a ground truth unison mixture. To this end, the reference provided to the participant was a ground truth unison sample made by summing the corresponding four individual singers of a part to form a unison mixture. This reference is referred to as *REFU*. The participants were asked to rate test samples which included the single voice prototype of the unison as output by the UTS

system, referred to as *UTS*. In addition, we evaluated the output of STU with a pitch deviation with parameter std set to $50\,\text{cents}$, the acceptable limit of pitch deviations, as shown by our analysis in Section 3.3 and suggested by [1]. Four singers were used for generating this test case, with parameter ns set to 4, and it is referred to as *STU_PS*. We also evaluated the output of the UTS system with both pitch and timing deviations with parameter ts set to $40\,\text{ms}$. While our analysis in Section 3.3 suggests that higher values of ts could have been used, we found that increasing the value beyond $40\,\text{ms}$ leads to a unacceptable level of degradation in output quality. We refer to this test case as *STU_PTS*. We also provided a lower anchor of a sample of the same length from another vocal part.

4.3.2 Perception of unison

Unison is a loosely defined perceptual aspect, the cognition of which we aim to study here. For this, we provide a reference of a ground truth unison sample created in the same manner as described above, *REFU*. Given this, participants were asked to rate outputs from the STU system based on their similarity to the reference in terms of the perception of unison. In addition to the *STU_PTS* and *STU_PS* cases with pitch, timing and timbre variance, we also tested the case for just timing and singer variation, referred to as *STU_TS* and a case with just pitch and timing deviations, referred to as *STU_PT*, timbral changes were not done for the voice clones used for creating this test case. The *a cappella* sample of a single singer singing the same example as the reference was provided as a lower anchor.
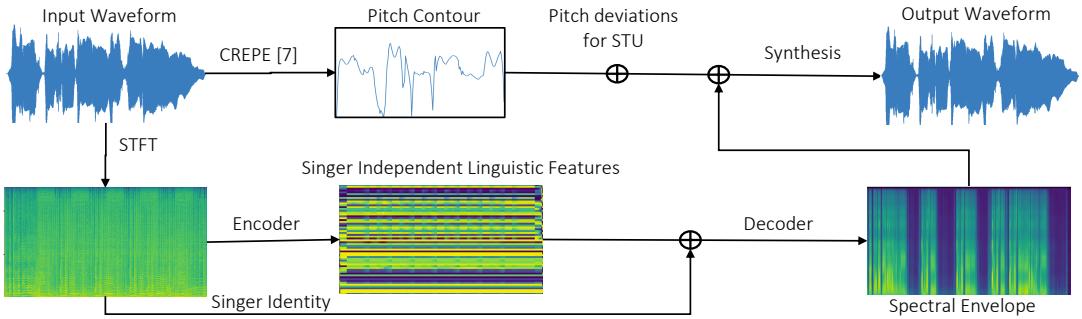
4.3.3 Audio Quality

Audio quality is another subjective measure that is well defined in literature but not easily understood by non-expert participants. For the evaluation, we set an upper limit of audio quality to the resynthesis of a single voice recording with the WORLD vocoder *REFS* and a lower limit to the resynthesis of a unison mixture with the same *RESSYN-THU*. The test cases provided to the participants were the same as those provided for the adherence to melody case, except that the lower anchor was changed.

4.4 Perceptual Evaluation Results

There were 17 participants in our evaluation, of which 10 had prior musical training. To account for inter-participant variance in subjective evaluation, the opinion score for each question was normalized over ratings for the reference and the lower anchor before calculating the mean opinion scores (MOS) and the standard deviations in opinion scores, presented in Table 2.

The subjective nature of the perceptual aspects evaluated must be taken into account for the evaluation and the mean opinion scores are indicative of preferences rather than absolute measures of quantity. It can be observed that the perceived adherence to melody for the prototypical *a cappella* voice synthesized by the UTS model has higher preference than the STU models, although a high variance

Figure 3. The synthesis framework. The magnitude part of the spectrogram of the input is passed through the encoder from [8] to extract linguistic content. For the UTS case, this is passed to the decoder along with a learned embedding to the SIN decoder [8] to generate the spectral envelope. For STU, the linguistic features are passed to the SDN decoder along with a one-hot vector of the same gender as the input. F0 is extracted from the input waveform via CREPE [7]. Both the envelope and the F0 are used to synthesize the output voice. For the STU case, timing deviations are further added before summing with the input and normalizing.

Test Case	Adherence To Melody	Unison Perception	Audio Quality
UTS	3.6 ± 0.93		2.1 ± 0.65
STU_PS	3.3 ± 0.83	2.6 ± 0.85	2.8 ± 0.45
STU_PTS	2.9 ± 1.14	3.2 ± 0.96	3.1 ± 0.63
STU_TS		2.3 ± 1.11	
STU_PT		3.0 ± 1.23	

Table 2. Mean Opinion Score (MOS) $\pm$ Standard Deviation for the perceptual listening tests across the test cases provided. The models shown corresdond to the Unison to Solo (UTS) model, the Solo to Unison with pitch, timing and singer variations, indicated by the addition of the letters P,T and S as suffixes to the abbreviation, respectively. The scores for each question were normalized by the responses to the upper and lower limits for the responses defined in section 4.3.

is observed in the ratings for the same. The unison perception evaluation shows that the variations in either timing or pitch alone are not as preferred as variations in both aspects together. Timbre variations do not have as significant an effect on perception of unison as variances in timing and pitch. The evaluation of audio quality shows room for improvement in the synthesis of the voice signals. This can partly be attributed to the use of the WORLD vocoder [12] and we believe that this can be improved on in the future using recently proposed neural synthesis techniques.

5. CONCLUSIONS

We have presented an analysis of the Choral Singing Dataset, building on the work presented in [1]. In accordance with the analysis done by [4], we observe deviation in the range of $0\,\text{cents}$–$50\,\text{cents}$ between the F0 contours of the individual singers in the unison mixtures in the dataset. We further note an timing deviation of $0.1\,\text{s}$ between singers in unison in the dataset.

We then used this analysis along with recently proposed deep-learning based methodologies to present a synthesis system for a unison mixture from a single voice input and a single voice prototype synthesis representing the melodic and linguistic content of a unison mixture input. Based on these systems, we were able to conduct a perceptual evaluation of the unison, further supporting the claim of [1] that the a mixture of different voices singing in unison is perceived to have a single pitch. In addition, we found that pitch and timing deviations together are important for the perception of the unison, and that variations in either aspect alone is insufficient for such. However, timbre variations were not found to be as relevant.

We present this work as the first step into the analysis of an under-explored research area, hoping to fuel further discussion on the topic. While interesting from an academic standpoint, the systems we present also have several commercial applications such as creating a unison choral effect to be used in music production as well as for transposition and transcription, in conjunction with the work presented in [17]. We also plan to incorporate the presented work with [18], for complete source separation for choral recordings.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] S. Ternström, "Perceptual evaluations of voice scatter in unison choir sounds," *STL-Quarterly Progress and Status Report*, vol. 32, pp. 041–049, 1991.

[2] N. Schnell, G. Peeters, S. Lemouton, P. Manoury, and X. Rodet, "Synthesizing a choir in real-time using pitch synchronous overlap add (psola)." in *ICMC*, 2000.

[3] J. Bonada, "Voice solo to unison choir transformation," in *Proceedings of the 118th Audio Engineering Society Convention*, Barcelona, Spain, May 2005.

[4] H. Cuesta, E. Gómez, A. Martorell, and F. Loáiciga, "Analysis of intonation in unison choir singing," in *Proceedings of the International Conference of Music Perception and Cognition (ICMPC)*, Graz, Austria, 2018, pp. 125–130.

[5] C. Weiss, S. J. Schelcht, S. Rosenzweig, and M. Müller, "Towards measuring intonation quality of choir recordings: A case study on bruckner's locus iste." in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 276–283.

[6] M. B. Dolson, "A tracking phase vocoder and its use in the analysis of ensemble sounds," Ph.D. dissertation, California Institute of Technology, 1983.

[7] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 161–165.

[8] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "Content based singing voice extraction from a musical mixture," in *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2020, pp. 781–785.

[9] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "A vocoder based method for singing voice extraction," in *Proceedings of the 44th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2019)*, IEEE. Brighton, UK: IEEE, 2019. [Online]. Available: https://arxiv.org/abs/1903.07554

[10] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, "Autovc: Zero-shot voice style transfer with only autoencoder loss," in *International Conference on Machine Learning*, 2019, pp. 5210–5219.

[11] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.

[12] M. Morise, F. Yokomori, and K. Ozawa, "World: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[13] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "WGANsing: A multi-voice singing voice synthesizer based on the wasserstein-gan," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[14] M. Blaauw and J. Bonada, "A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs," *Applied Sciences*, vol. 7, no. 1313, 12/2017 2017.

[15] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai, "Mel-generalized cepstral analysis-a unified approach to speech spectral estimation," in *3rd International Conference on Spoken Language Processing (ICSLP)*, 1994.

[16] C. Völker, T. Bisitz, R. Huber, B. Kollmeier, and S. M. Ernst, "Modifications of the multi stimulus test with hidden reference and anchor (mushra) for use in audiology," *International journal of audiology*, vol. 57, no. sup3, pp. S92–S104, 2018.

[17] H. Cuesta, B. McFee, and E. Gómez, "Multiple F0 Estimation in Vocal Ensembles using Convolutional Neural Networks," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montreal, Canada (Virtual), 2020.

[18] D. Petermann, P. Chandna, H. Cuesta, J. Bonada, and E. Gómez, "Deep Learning Based Source Separation Applied To Choir Ensembles," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montreal, Canada (Virtual), 2020.

ESSENTIA.JS: A JAVASCRIPT LIBRARY FOR MUSIC AND AUDIO ANALYSIS ON THE WEB

Albin Correya[1]   **Dmitry Bogdanov**[1]   **Luis Joglar-Ongay**[1,2]   **Xavier Serra**[1]

[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain

[2] SonoSuite, Barcelona, Spain

`albin.correya@upf.edu, dmitry.bogdanov@upf.edu`

ABSTRACT

Open-source software libraries for audio/music analysis and feature extraction have a significant impact on the development of Audio Signal Processing and Music Information Retrieval (MIR) systems. Despite the abundance of such tools on the native computing platforms, there is a lack of an extensive and easy-to-use reference library for audio feature extraction on the Web. In this paper, we present *Essentia.js*, an open-source JavaScript (JS) library for audio and music analysis on both web clients and JS-based servers. Along with the Web Audio API, it can be used for efficient and robust real-time audio feature extraction on the web browsers. *Essentia.js* is modular, lightweight, and easy-to-use, deploy, maintain and integrate into the existing plethora of JS libraries and Web technologies. It is powered by a WebAssembly back-end of the Essentia C++ library, which facilitates a JS interface to a wide range of low-level and high-level audio features. It also provides a higher-level JS API and add-on MIR utility modules along with extensive documentation, usage examples, and tutorials. We benchmark the proposed library on two popular web browsers, Node.js engine, and Android devices, comparing it to the native performance of Essentia and Meyda JS library.

1. INTRODUCTION

The Web has become one of the most used computing platforms with billions of web pages served daily, and JS being an essential part of building modern web applications. Using HTML, CSS, and JS, developers can make dynamic, interactive, and responsive web pages by implementing custom client-side scripts. At the same time, the developers can also use cross-platform run-time engines like Node.js [1] to write server-side code in JS. The flexibility of using JS on both server and client ends of web

---

[1] https://nodejs.org

applications arguably makes it one of the most used computer programming languages in the recent years [2]. JS is also widely used as an entry-level programming language by the thinkers from design, art, computer graphics, architecture, and spaces in between where audio processing and analysis can be relevant.

With the adoption of both HTML5 and the W3C Web Audio API specifications [14], modern web browsers are capable of audio processing, synthesis, and analysis without any third-party dependencies on proprietary software. This allows developers to move most of the audio processing code from the server to the client and can provide better scalability and deployment, considering that the web-client has computational resources for the required processing. Web Audio API provides a JS interface to various predefined nodes for audio processing, synthesis, and analysis. Even though the provided capabilities are limited, the API includes the *ScriptProcessorNode* for developers to add custom JS code for audio processing. [2] The design of this node has been criticized by the audio developer community since it runs the JS audio processing code on the main UI thread, which can result in unreliable performance and audio glitching [15]. As an alternative, *AudioWorklet* [10] has been introduced to mitigate this design issue to a great extent by allowing running custom audio processing code on a dedicated audio thread. It also allows bi-directional communication between the audio thread and the control thread of Web Audio API asynchronously using the web message ports.

Despite all of these recent developments of Web Audio technologies, the Audio Signal Processing and MIR communities still lack reliable and modular software tools and libraries that could be easily used for building audio and music analysis applications for web browsers and JS run-time engines. To the best of our knowledge, Meyda [11] and JS-Xtract [18] are the few available JS libraries that are ready-to-use and have implementations of a limited set of MIR audio features. [3] The lack of more extensive alternatives is not surprising, given that writing a new JS audio analysis library from scratch would require a lot of effort. Also, JS code for audio processing are prone to performance issues due to the *just-in-time* (JIT) compilation and garbage collection overhead, which can be critical for

---

[2] https://www.w3.org/TR/webaudio/#scriptprocessornode
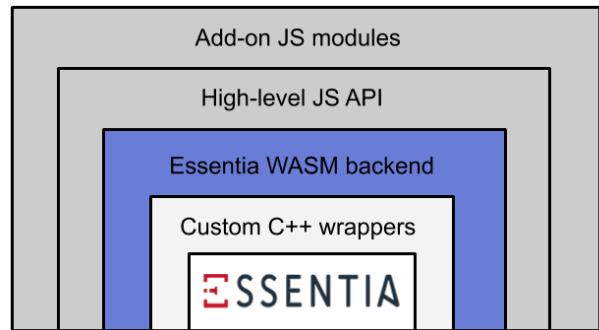
[3] As of May 2020, Meyda only has 20 MIR algorithms.

real-time audio and music analysis tasks. Due to these reasons, researchers and developers often rely on server-side audio processing solutions using the existing native MIR tools for writing the required web applications.

Over the last two decades, the existing software tools for audio analysis have been mostly written in C/C++, Java and Python and delivered as standalone applications, host application plug-ins, or as software library packages. Software libraries with a Python API, such as Essentia [7], Librosa [23], Madmom [6], Yaafe [22] and Aubio [8], have been especially popular within the MIR community due to rapid prototyping and rich environment for scientific computations. Meanwhile, the libraries with a native C/C++ back-end offered faster analysis [24] and were often required for writing industrial audio applications. Various web applications for audio processing and analysis have been developed using these tools. Spotify API [4] (formerly Echonest), Freesound API [13] and AcousticBrainz [25] are examples of web services providing precomputed audio features to the end users via a REST API. Besides, numerous web applications were built for aiding tasks such as crowd sourcing audio annotations [9, 12], audio listening tests [19, 26] and music education platforms [1, 21] to mention a few. All these services manage their audio analysis on the server, which may require a significant effort and resources to scale to many users.

With the recent arrival of WebAssembly (WASM) support on most of the modern web browsers [16], one can safely port the existing C/C++ audio processing and analysis code into the Web Audio ecosystem using compiler toolchains such as Emscripten. [5] WASM is a low-level assembly-like language with a compact binary format that runs with near-native performances on modern web browsers or any WebAssembly-based stacks without compromising security, portability and load time. WASM code was found to be comparatively faster than JS code [17] and generates no garbage from the code and can run within *AudioWorkletProcessor*. [6] This makes it an ideal solution to the problems that were previously hindering us from building efficient and reliable JS MIR libraries for the web platform [20]. However, taking full advantage of all these features can be challenging because it requires understanding concurrent programming wrapped with several JS APIs and dealing with cross-compilation and linking of the native code. Even for experienced developers, compiling native code to WASM targets, generating JS bindings, and integrating them in a regular JS processing code pipeline can be cumbersome. Hence, it is essential that an ideal JS MIR software library should encapsulate and abstract all these steps and be packaged as a compact build which is easy-to-use and extendable using a high-level JS API. Besides the JS API, the potential users might also need utility tools that are often required while building MIR-based projects, such as plotting audio features on an HTML page, ready-to-use feature extractors, and possible integration with web-based

Figure 1: Overview of the *Essentia.js* library in terms of its abstraction levels.

machine learning frameworks.

In [24], the authors evaluated a wide range of MIR software libraries in terms of *coverage*, *effort*, *presentation*, *time-lag* and found Essentia [7] [7] to be an overall best performer with respect to these criteria. Essentia is an open-source library for audio and music analysis released under the AGPLv3 license providing a wide range of optimized algorithms (over 250 algorithms) that are successfully used in various academic and industrial large-scale applications. Essentia includes both low-level and high-level audio features, along with some ready-to-use features extractors. And, it provides an object-oriented interface to fine tune each algorithm in detail. Given all these advantages and that the code repository is consistently maintained by its developers, it is a good potential choice for porting into WASM target for the web platform.

In this paper, we present *Essentia.js*, an open-source JS library for audio and music analysis on the web, released under the AGPLv3 license. It allows building audio analysis and MIR applications for web browsers and JS engines such as Node.js. It provides straightforward integration with the latest W3C Web Audio API specification allowing efficient real-time audio feature extraction on the web browsers. Web applications written using the proposed library can also be cross-compiled to native targets such as for PCs, smartphones, and IoT devices using the JS frameworks like Electron [8] and React Native. [9]

The rest of the paper is organized as follows. Section 2 outlines the design choices, software architecture and various components of *Essentia.js*. An overview of potential use-cases and usage examples are detailed in Section 3. A detailed benchmarking of *Essentia.js* across and against various platforms and alternative JS libraries can found in Section 4. Finally, we conclude in Section 5.

2. ESSENTIA.JS

Essentia.js is much more than just JS bindings to the Essentia C++ library. It was developed with coherent design and functional objectives that are necessary for building an efficient and easy-to-use MIR library for the Web. In this

section, we discuss our design choices, the architecture, and various components of *Essentia.js*. Figure 1 shows an overview of these components.

2.1 Design and Functionality

We chose the following goals and design decisions for developing the library:

- **User-friendly API and utility tools**. The Web is a ubiquitous computing platform, and an ideal JS MIR library should provide a simple, user-friendly API while being highly configurable for advanced use cases. *Essentia.js* ships with a simple JS API and add-on utility modules with a fast learning curve for new users. The main JS API is composed of a singleton class with methods implementing most of the functionality (each method is an algorithm in Essentia). These methods are automatically generated from the upstream C++ code and documentation using code templates and scripting as described in Sections 2.2 and 2.3. We also provide add-on modules with helper classes for feature extraction and visualisation that can be used for rapid prototyping of web applications. A quick preview of the JS API can be seen in Listing 2.

- **Modularity and extensibility**. Just like Essentia itself, the *Essentia.js* codebase is modular by design. It contains a large amount of configurable algorithms that can be arranged into the desired audio processing chains. The add-on utility modules shipped with the library leverage this functionality to build custom feature extractors.

- **Web standards compliance.**Web browsers provide a standard set of tools for loading and processing audio files using the HTML5 Audio element [10] and the Web Audio API. *Essentia.js* rely on these standard features for loading audio files or for streaming real-time audio from various device sources. It also provides seamless integration with the latest tools in the Web Audio ecosystem such as AudioWorklets, Web Workers, [11] WASM and *SharedArrayBuffer*. In addition, JS conforms to the ECMAScript specification [12] and it is evolving fast with new features added to the language every year. For backward and forward compatibility of our JS code, we wrote our JS API using Typescript (Section 2.3).

- **Lightweight and few dependencies**. It is important for a JS library to be lightweight since the load times of JS code can directly impact the UI/UX and performance of web applications. This includes having fewer dependencies, which also makes the library much more maintainable. Taking this into account, *Essentia WASM backend* is built without any third-party software dependencies of the Essentia library except for Kiss FFT. [13] It includes

the majority of the algorithms in Essentia, [14] while the few excluded algorithms can be still integrated into the WASM backend by compiling and linking with the required third-party dependencies using our build tools (Section 2.5). Besides, all the JS code in the library is passed through a code compression process to achieve lightweight builds for the web browsers. With all these efforts we were able to achieve builds of *Essentia.js*, including the WASM backend and the JS API, as small as 2.5MB approximately. We also provide tools for custom lightweight builds of the library that only include a subset of the selected algorithms to further reduce the build size (Section 2.5).

- **Reproducibility**. Using the WASM backend of Essentia ensures identical analysis results across various devices and native platforms on which Essentia has been previously extensively used and tested. Remarkably, *Essentia.js* allows reproducing a large amount of existing code and research based on Essentia as well as, to a certain extent, other libraries. In particular, it is possible to use *Essentia.js* to reproduce common input audio representations for the existing machine learning models, enabling their usage in web applications.

- **Easy installation**. *Essentia.js* is easy to install and integrate with new or existing web projects. It is available both as a package on NPM [15] and as static builds on our public GitHub repository. In addition, we also provide continuous delivery network (CDN) through open source web services.

- **Extensive documentation**. We provide a complete API reference together with the instructions to get started, tutorials, and interactive web application examples. [16] The documentation is built automatically using JS-doc [17] and the algorithm reference is generated from the upstream Essentia C++ documentation using Python scripts.

2.2 Essentia WASM backend

As already mentioned, the core of the library is powered by the *Essentia WASM backend*. It contains a lightweight WASM build of Essentia C++ library along with custom JS bindings for using it in JS. This backend is generated in multiple steps.

Firstly, the Essentia C++ library is compiled to LLVM assembly [18] using Emscripten. Emscripten [28] is a LLVM to JS compiler which provides a wide range of tools for compiling the C/C++ code-base or the intermediate LLVM builds into various targets such as *asm.js* [19] and WASM. Secondly, we need a custom C++ interface in order to generate the corresponding JS bindings which would allow us access the algorithms in Essentia on the JS side. We used

---

[10] https://www.w3.org/html/wiki/Elements/audio
[11] https://w3c.github.io/workers/
[12] http://ecma-international.org/ecma-262
[13] https://github.com/mborgerding/kissfft

[14] As of May 2020, over 200 algorithms are supported.
[15] https://www.npmjs.com
[16] https://mtg.github.io/essentia.js
[17] https://jsdoc.app
[18] https://llvm.org
[19] http://asmjs.org

Embind [4] for generating this C++ interface that binds Essentia native code to JS.

Writing custom JS bindings for all Essentia algorithms can be tedious considering their large amount. Hence, we created Python scripts to automate the generation of the required C++ code for the C++ wrapper from the upstream library Python bindings. Using this scripts, it is possible to configure which algorithms to include in the bindings by their name and category. Therefore, it is possible to create extremely lightweight builds of the library with only a few specific algorithms required for a particular application. The *Essentia WASM backend* is built by compiling the generated wrapper C++ code and linking with the precompiled Essentia LLVM using Emscripten.

Essentia WASM backend provides compact WASM binary files along with the JS bindings to over 200 Essentia algorithms. We provide these binaries and a JS glue code for both asynchronous and synchronous import of Essentia WASM backend, covering a wide range of use cases. The build for asynchronous import can be instantly loaded into a HTML page. The synchronous-import build supports the new ES6 style class imports characteristic of the modern JS libraries. This build can be also seamlessly integrated with AudioWorklet and Web Workers for better performance demanding web applications.
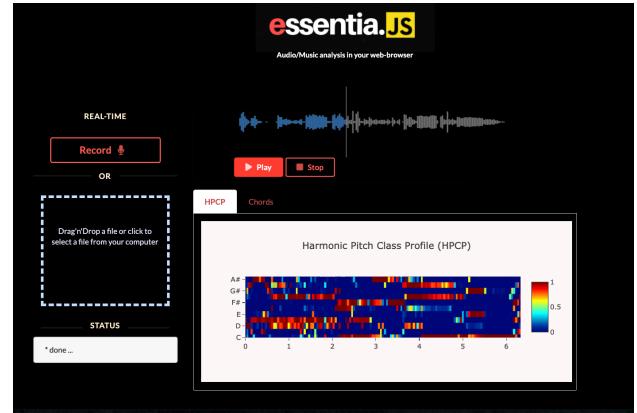
2.3 High-level JS API

Even though it is possible to use the *Essentia WASM backend* with its bindings directly, they have limitations due to the specifics of using *Embind* with Essentia: a user needs to manually specify all parameter values for the algorithms because the default values are not supported. This is cumbersome and to solve this issue we developed a high-level JS API written using Typescript [5]. Typescript is a typed superset of JS that can be compiled to various ECMA targets of JS. In addition, this gives us the benefit of having a typed JS API which can provide better exception handling. Again we used custom Python scripts and code templates to automatically generate the Typescript wrapper in a similar way to the C++ wrapper for the WASM backend. The high-level JS API of *Essentia.js* provides a singleton class *Essentia* with all the algorithms and helper functions encapsulated as its methods All the algorithm methods are configurable similarly to the Essentia's C++/Python API itself. Listing 1, shows an example of using the high-level API of *Essentia.js*.

2.4 Add-on utility modules

Essentia.js is shipped with a few add-on modules to facilitate common MIR tasks. These add-on modules are written entirely in Typescript using the *Essentia.js* high-level JS API. Currently, we provide two add-on modules:

- **essentia.js-extractor** contains predefined feature extractors for common MIR tasks, computing BPM, beat positions, pitch, predominant melody, key, chords, chroma features, MFCC, etc. Each extractor implements the entire processing chain starting from audio input and outputs the resulting track-level or frame-level features. These extractors are configurable as well.

Figure 2: Screenshot of a example web application that use *Essentia.js* and its add-on modules.

- **essentia.js-plot** provides helper functions for visualization of MIR features, allowing both real-time and offline plotting in a given HTML element. It uses the *Plotly.js* data visualization library, which has a design layout and functionalities much alike of Matplotlib, [20] and is easily configurable. Currently, we provide object-oriented classes for plotting basic MIR features like melody/pitch contours, spectrograms, chroma, MFCC, etc. The module is functionally similar to the *display* module in Librosa [23].

A full reference of the modules can be found in the documentation of the library. Both modules can be easily extended with more functionalities as per the demands of the user community.

2.5 Build and packaging tools

We provide tools for custom builds and packaging of *Essentia.js* for the developers and the end-level users:

- **Command-line interface (CLI)**. We provide CLI for building *Essentia.js* using some customised shell scripts.

- **Docker**. We provide a Docker image with static builds of Essentia with Emscripten and other development dependencies required for building *Essentia.js*.

- **Web application**. We also host a website for building custom *Essentia.js* online. [21] It allows users to select specific set algorithms and build settings.

The official *Essentia.js* builds are bundled using Rollup [22] and then packaged and hosted using NPM.

3. GETTING STARTED

In this section, we outline several usage examples and application scenarios for getting started with *Essentia.js*.

The library can be imported into a web application using the following methods:

- **HTML <script> tag**. The most simple way to use *Essentia.js* is by using it with the HTML *<script>* tag.

---

[20] https://matplotlib.org
[21] https://mtg.github.io/essentia.js-builder
[22] https://rollupjs.org

- **NPM**. *Essentia.js* can be also installed from NPM using the command `npm install essentia.js`.

- **ES6 class imports**. *Essentia.js* can be imported using the ES6 class style imports in JS. This allows to integrate the library into any modern JS framework. Listing 1 shows an example of using ES6 style imports for an offline feature extraction task.

- **CDN**. We also provide CDN links for instantly serving *Essentia.js* online using free third-party web services such as Jsdelivr [23] and Unpkg. [24]

There are a lot of potential web applications that can be built with *Essentia.js*. The library provides algorithms for typical sound and music analysis tasks, including spectral, tonal, and rhythmic characterization. In particular, it is suitable for onset detection, beat tracking and tempo estimation, melody extraction, key and chord estimation, sound and music classification, cover song similarity, loudness metering, and audio problems detection among others. Figure 2 shows the screenshot of an example web application that we include with the library. Below we outline some of the common application use cases of the library. We provide an extensive collection of analysis examples on our website. [25]

3.1 Offline feature extraction

Many MIR use cases rely on an offline audio analysis and feature extraction. Listing 1 shows a simple JS example of using the library for offline analysis of pitch and onsets. For features computed on overlapping frames, *Essentia.js* provides the *FrameGenerator* method similarly to Essentia's Python API. Frames generated by this method can be used as an input to other algorithms in the processing chain. The offline feature extraction can be run inside a Web Worker to improve the efficiency in performance-demanding web applications.

3.2 Real-time feature extraction

Essentia.js can be used for efficient real-time audio/music analysis in web browsers along with the Web Audio API. This can be done by using the *ScriptProcessorNode* or the newly introduced *AudioWorklet* in the Web Audio API:

- **ScriptProcessorNode** allows users to provide custom JS code for audio feature extraction in its `onprocess` callback. Even though the *ScriptProcessorNode* is deprecated according to the current W3C Web Audio API specifications, it is still widely used by the developers because of its cross-browser support.

- **AudioWorklet** design pattern [10] allows users to write high-performance real-time audio analysis on a dedicated audio thread. Users can write custom analysis code by extending the *AudioWorkletProcessor* and further abstract it as a node in the Web Audio API graph

```javascript
// Imports Essentia WASM backend
import {EssentiaWASM} from 'essentia-wasm.module.js';
// Imports Essentia.js core API
import Essentia from 'essentia.js-core.es.js';

// Creates Essentia.js instance
const essentia = new Essentia(EssentiaWASM);

// Instance of Web Audio API AudioContext
const audioContext = new AudioContext();
// URL of an audio file
let audioURL = "https://freesound.org/data/previews
    /328/328857_230356-lq.mp3";

// Decode audio file as Float32 typed array
const audioData = await essentia.
    getAudioChannelDataFromURL(audioURL, audioContext,
    0); // audioContext, channel number

// Convert audioData array into vector
const audioVector = essentia.arrayToVector(audioData);

// Onset detection with SuperFluxExtractor algorithm
let bt = essentia.SuperFluxExtractor(audioVector);
console.log(bt.onsets);

// Pitch estimation with PitchYinProbabilistic
    algorithm
let pyIn = essentia.PitchYinProbabilistic(audioVector,
    4096, 256); // frameSize, hopSize

console.log(pyIn.pitch);

// Shutdown Essentia.js instance and free memory
essentia.shutdown();
essentia.delete();
```

Listing 1: A simple example of offline audio feature extraction using *Essentia.js* via ES6 style imports.

using *AudioWorkletNode*. [26] Currently, the only limitation is that it is only supports in the latest Firefox and Chromium-based web browsers.

3.3 Machine learning applications

In the recent years, machine learning (ML) techniques, especially deep learning have been widely used in a wide range of MIR tasks. With the support of WebGL and WASM, modern web browsers are also capable of running ML applications. *Essentia.js* can be easily integrated with popular JS ML frameworks such as *TensorFlow.js* [27] and *Onnx.js* [27] for training and inference. Pre-trained audio ML models using features computed with Essentia as an input (e.g., mel-spectrograms, Constant-Q transform, or chroma) can be easily ported and used for inference in web browsers. In particular, Essentia now ships with a collection of pre-trained TensorFlow models for music audio tagging and classification [3]. These models can be run for inference using *Essentia.js* and *TensorFlow.js* libraries. Our *essentia.js-extractor* add-on module provides a mel-spectrogram extractor for computing the inputs to these models.

4. BENCHMARK

We tested the performance of *Essentia.js* in terms of the analysis time for common MIR audio features on various

---

[23] https://www.jsdelivr.com
[24] https://unpkg.com
[25] https://mtg.github.io/essentia.js/examples

[26] https://www.w3.org/TR/webaudio/
#audioworkletnode
[27] https://github.com/Microsoft/onnxjs

(a) *Essentia.js*

(b) *Meyda*
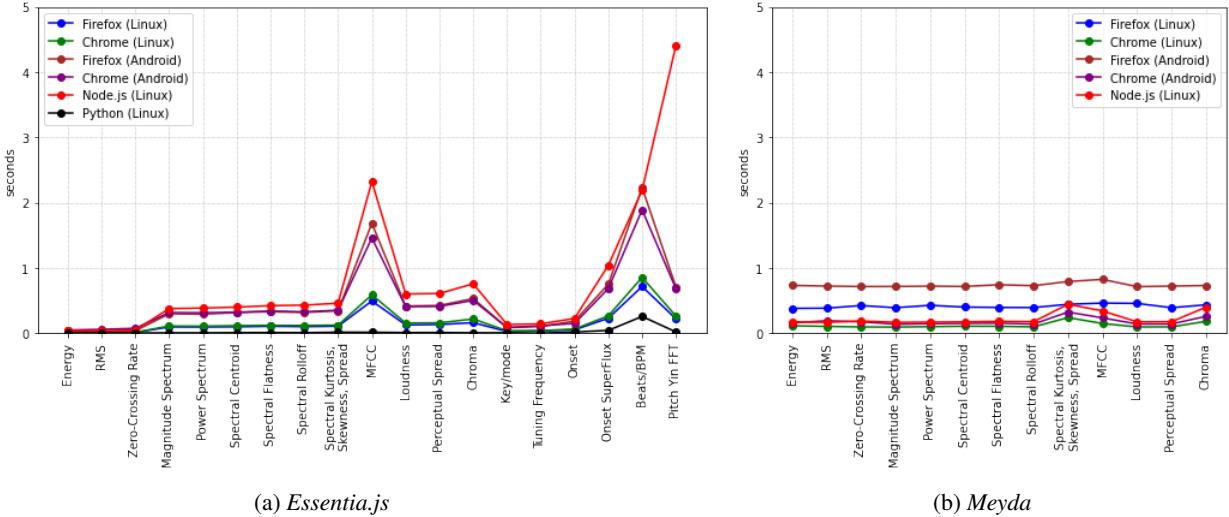
Figure 3: Average analysis times (in seconds) for common audio features on a 5-second music clip. "Python (Linux)" corresponds to the analysis baseline using native Essentia with Python bindings.

platforms, and compared it to the native Essentia library. In addition, we measured the analysis times for features available in *Meyda* and compared them to their *Essentia.js* counterparts. To this end, we built a set of test suites using the JS library *benchmark.js* and implemented the equivalent features using both libraries. In our benchmark we measure the time it takes for the entire processing chain to compute a feature given a 5-second audio segment as an input. The code used by *Essentia.js* is equivalent to the one for *Essentia* used in Python. The benchmarking of Python implementation was done using the library *pytest* with the *benchmark extension*. We provide the code and website to reproduce these experiments online. [28]

The results are reported in Figure 3. They include tests on five different environments:

- Linux with Chrome 84.0.4147.89 run with disabled extensions.

- Linux with Firefox 78.0.2 in private browsing mode.

- Android 9 (LineageOS 16) with Chrome 84.0.4147.89 in incognito mode.

- Android 9 (LineageOS 16) with Firefox Nightly 200727 06:00

- Linux with Node.js v.13.13.0.

The Linux computer used for all runs is a 2017 DELL XPS-15 with a 2.80GHz x 8 Intel Core i7-7700HQ processor, 16GB of RAM and Ubuntu 19.04 as OS. The mobile phone is a Xiaomi Redmi Note 7 Pro with a Snapdragon Octa-core 1.7 GHz processor and 6GB RAM. All the tests were done with the same 5 seconds audio file.

As we can see from Figure 3, the results shows that the performance of *Essentia.js* algorithms were considerably slower when running on Node.js and Firefox and Chrome on Android compared to Firefox and Chrome on Linux. Interestingly, Node.js performed worse than Firefox and

Chrome on Android, which was not expected. This is probably because different vendors have slightly different implementations of WASM support in their platforms or due to some other reasons yet to be found. In addition, WASM is a relatively new technology in active development. [29] Many proposals for improving its performance such as SIMD optimizations and multi-thread support are under way. We also aim to do detailed benchmarking of real-time use cases and using the Web Audio API Audio Worklets in our future work.

5. CONCLUSIONS

We have presented *Essentia.js*, an open-source JavaScript library for music/audio analysis on the Web. It is based on the Essentia C++ library which is commonly used in MIR, ported to JS via WASM, and compatible with the latest technologies in the Web Audio ecosystem. To the best of our knowledge, this is the most comprehensive library for audio analysis and MIR, which can be run on web browsers as well as JS server applications. We hope that the library will contribute to the creation of new online music technology tools in educational, industrial, and creative contexts. The source code of the library is publicly available in our Github repository. [30] Everyone is encouraged to contribute to the library.

In our future work, we will focus on improving the performance of the library along with expanding the add-on modules, particularly on providing pre-trained audio ML models for audio analysis, classification, and synthesis on the web client. We also aim to develop interesting web applications that go beyond typical MIR tasks to attract and build a diverse user community. The detailed information about the library is available at the official web page. [31] It contains the complete documentation, usage examples and tutorials needed for one to get started.

---

[28] https://mtg.github.io/essentia.js-benchmarks

[29] https://webassembly.org/roadmap/

[30] https://github.com/MTG/essentia.js

[31] https://essentia.upf.edu/essentiajs

6. REFERENCES

[1] MusicCritic: An automatic assessment system for musical exercises. https://musiccritic.upf.edu. Accessed: 2020-09-04.

[2] Stack Overflow Annual Developer Survey. https://insights.stackoverflow.com/survey. Accessed: 2020-15-04.

[3] Pablo Alonso-Jiménez, Dmitry Bogdanov, Jordi Pons, and Xavier Serra. Tensorflow audio models in Essentia. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, pages 266–270, 2020.

[4] Chad Austin. CppCon 2014: Embind and Emscripten: Blending C++11, JavaScript, and the Web Browser. https://www.youtube.com/watch?v=Dsgws5zJiwk. Accessed: 2020-15-04.

[5] Gavin Bierman, Martín Abadi, and Mads Torgersen. Understanding typescript. In *European Conference on Object-Oriented Programming (ECOOP 2014)*, pages 257–281. Springer, 2014.

[6] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. Madmom: A new python audio and music signal processing library. In *ACM International Conference on Multimedia (MM 2016)*, pages 1174–1178, 2016.

[7] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 493–498, 2013.

[8] Paul M Brossier. The aubio library at MIREX 2006. In *Music Information Retrieval Evaluation Exchange (MIREX 2006)*, 2006.

[9] Mark Cartwright, Ayanna Seals, Justin Salamon, Alex Williams, Stefanie Mikloska, Duncan MacConnell, Edith Law, Juan P Bello, and Oded Nov. Seeing sound: Investigating the effects of visualizations and complexity on crowdsourced audio annotations. *Proceedings of the ACM on Human-Computer Interaction*, 1(CSCW):1–21, 2017.

[10] Hongchan Choi. *AudioWorklet: The future of web audio*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 2018.

[11] Jakub Fiala, Nevo Segal, and Hugh A. Rawlinson. Meyda: an audio feature extraction library for the Web Audio API. In *Web Audio Conference (WAC 2015)*, pages 1–6, 2015.

[12] Eduardo Fonseca, Jordi Pons Puig, Xavier Favory, Frederic Font Corbera, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. Freesound datasets: a platform for the creation of open audio datasets. In *International Society for Music Information Retrieval Conference (ISMIR 2017)*, pages 486–93.

[13] Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. In *ACM International Conference on Multimedia (MM 2013)*, page 411–412, New York, NY, USA, 2013.

[14] W3C Audio Working Group. W3C Web Audio API specifications. https://www.w3.org/TR/webaudio. Accessed: 2020-15-04.

[15] W3C Technical Architecture Group. Web Audio API Design Review. https://github.com/w3ctag/design-reviews/blob/master/2013/07/WebAudio.md. Accessed: 2020-05-04.

[16] Andreas Haas, Andreas Rossberg, Derek L Schuff, Ben L Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. Bringing the web up to speed with webassembly. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2017)*, pages 185–200, 2017.

[17] David Herrera, Hanfeng Chen, Erick Lavoie, and Laurie Hendren. Numerical computing on the web: Benchmarking for the future. In *ACM SIGPLAN International Symposium on Dynamic Languages (DLS 2018)*, pages 88–100, 2018.

[18] Nicholas Jillings, Jamie Bullock, and Ryan Stables. JS-Xtract: A realtime audio feature extraction library for the Web. In *International Society for Music Information Retrieval Conference (ISMIR 2016) Late Breaking Demo*, 2016.

[19] Nicholas Jillings, David Moffat, Brecht De Man, and Joshua D. Reiss. Web Audio Evaluation Tool: A browser-based listening test environment. In *Sound and Music Computing Conference (SMC 2015)*, 2015.

[20] Jari Kleimola and Oliver Larkin. Web audio modules. In *Sound and Music Computing Conference (SMC 2015)*, 2015.

[21] Anand Mahadevan, Jason Freeman, Brian Magerko, and Juan Carlos Martinez. Earsketch: Teaching computational music remixing in an online web audio based learning environment. In *Web Audio Conference (WAC 2015)*, 2015.

[22] Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gaël Richard. Yaafe, an easy to use and efficient audio feature extraction software. In *International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 441–446, 2010.

[23] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Python in Science Conference (SciPy 2015)*, 2015.

[24] David Moffat, David Ronan, and Joshua D. Reiss. An evaluation of audio feature extraction toolboxes. In *International Conference on Digital Audio Effects (DAFx 2015)*, pages 1–7, 2015.

[25] Alastair Porter, Dmitry Bogdanov, Robert Kaye, Roman Tsukanov, and Xavier Serra. Acousticbrainz: a community platform for gathering music information obtained from audio. In *International Society for Music Information Retrieval Conference (ISMIR 2015)*, 2015.

[26] Michael Schoeffler, Fabian-Robert Stöter, Bernd Edler, and Jürgen Herre. Towards the next generation of web-based experiments: A case study assessing basic audio quality following the ITU-R recommendation BS. 1534 (MUSHRA). In *Web Audio Conference (WAC 2015)*, pages 1–6, 2015.

[27] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, et al. Tensorflow. js: Machine learning for the web and beyond. In *Conference on Systems and Machine Learning (SysML 2019)*, 2019.

[28] Alon Zakai. Emscripten: an llvm-to-javascript compiler. In *ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2011)*, pages 301–312, 2011.

ON THE CHARACTERIZATION OF
EXPRESSIVE PERFORMANCE IN CLASSICAL MUSIC:
FIRST RESULTS OF THE *CON ESPRESSIONE* GAME

Carlos Cancino-Chacón[1,4] **Silvan Peter**[2] **Shreyan Chowdhury**[2]
Anna Aljanaki[3] **Gerhard Widmer**[1,2]

[1] Austrian Research Institute for Artificial Intelligence, Vienna, Austria
[2] Institute of Computational Perception, Johannes Kepler University Linz, Austria
[3] Institute of Computer Science, University of Tartu, Estonia
[4] RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion, University of Oslo, Norway

`carlos.cancino@ofai.at`, `silvan.peter@jku.at`

ABSTRACT

A piece of music can be expressively performed, or interpreted, in a variety of ways. With the help of an online questionnaire, the *Con Espressione* Game, we collected some 1,500 descriptions of expressive character relating to 45 performances of 9 excerpts from classical piano pieces, played by different famous pianists. More specifically, listeners were asked to describe, using freely chosen words (preferably: adjectives), how they perceive the expressive character of the different performances. In this paper, we offer a first account of this new data resource for expressive performance research, and provide an exploratory analysis, addressing three main questions: (1) how similarly do different listeners describe a performance of a piece? (2) what are the main dimensions (or axes) for expressive character emerging from this?; and (3) how do measurable parameters of a performance (e.g., tempo, dynamics) and mid- and high-level features that can be predicted by machine learning models (e.g., articulation, arousal) relate to these expressive dimensions? The dataset that we publish along with this paper was enriched by adding hand-corrected score-to-performance alignments, as well as descriptive audio features such as tempo and dynamics curves.

1. INTRODUCTION

In the Western classical music tradition, music exists at an interplay of creative intentions of composer, performers and listeners. Composers encode their ideas using written notation (i.e., musical scores), and performers bring these ideas to life, guided by the expression markings, performance traditions, and their own creative imagination. Each performance can sound very different from the next.

Much of the research on musical expression has focused on what pieces express through attributes of their musical structure [1–3]. There has been much focus on the expression of *emotion* in particular (e.g., [4–8]); however, a comprehensive description of the expressive character of a performance includes additional, not specifically emotion-related aspects. The aim of this research is to find the dimensions of musical expression that can be attributed to a performance, as perceived and described in natural language by listeners.

Within the classical music tradition, there is already a practice of assigning verbal descriptors to aspects of musical expression. For example, instructions related to expressive character are sometimes marked on the score by the composer with a fixed set of (mostly) Italian terms (e.g., *Allegro*, *dolce*). Many of those terms describe emotions, but they describe a wide range of other aspects as well, including movement analogies and metaphors (e.g., free, flowing) [9].

Using an online questionnaire, the *Con Espressione* Game (CEG), which is part of the research project of the same name [10], we collected verbal descriptors of expressive performances. In this paper we present first results on three main questions: (1) can we observe any consistency in the way listeners perceive and describe *expressive character*?; (2) can we identify main descriptive dimensions along which these characterizations can be organized?; and (3) how are these dimensions related to measurable qualities (or parameters) from audio recordings, or to performance information extracted from these?

The rest of this paper is structured as follows: Section 2 points to some related work. Section 3 describes the CEG and the collected data. We continue the paper with individual sections addressing the three main questions we want to study: Section 4 presents results on how similar/consistent the characterizations of the participants are, Section 5 focuses on analyzing the main descriptive dimensions. Section 6 presents results relating performance and audio features to the expressive dimensions. Section 7 discusses the results of these experiments. Finally, Section 8 concludes the paper.

2. RELATED WORK

In this section we provide a few pointers to related work on research on music emotion and expressive performance, and on characterizing musical expression. A full overview is beyond the scope of this paper.

From a musicological and philosophical viewpoint, the study of musical expression has focused on music as in composition [1,2], although there is a trend in recent times to include and recognize both the role of the performer and the role of the listener. Some recent papers have focused on developing mid-level features that capture qualities of musical recordings that relate to perceptual aspects of the music [11, 12] . For an overview of literature on emotion in music, we refer the reader to [3] and [13].

Computational models of expressive performance are a means to study principles of performance in quantitative terms. In Western classical music, much of this work has focused on establishing relationships between structural aspects of a musical score and quantifiable aspects of a performance such as expressive timing and dynamics. At the intersection of research on music emotion and computational models of performance is the study of the relation between performance parameters (timing, dynamics) and emotion [4–8]. For an overview of computational modeling of music performance, we refer the reader to [14].

Particularly relevant to the current study are adjective lists for describing musical expressiveness, including the seminal work by Hevner [15] and its more recent updates [16, 17]. Hevner identified several clusters of adjectives that describe musical expression. Many of these clusters relate to emotions, and include aspects such as *'happy'* and *'sad'*. Schaerlaeken et al. [18] conducted a large study to investigate the use of metaphors for describing Western classical music. They propose the Geneva Musical Metaphors Scale (GEMMES), which comprises 5 metaphorical scales including aspects like *flow, movement, force, interior* and *wandering*. The focus of their study was on characterization of different pieces of music, rather than on description of the character of different expressive performances. A related study [9] presents a perception-based clustering of expressive musical terms (i.e., performance directives such as *catabile* and *leggiero*), relating these terms to locations in Russell's arousal–valence space [19]. Murari et al. [20] study of listeners' characterization of music using non-verbal sensory scales.

3. THE *CON ESPRESSIONE* GAME

We collected our data through a Web-based questionnaire ("CEG"), where participants listened to several performances of each of 9 classical piano pieces. They were then asked to describe the performance in free text (preferably adjectives, as many as they liked), concentrating on the performative aspects and not on the piece itself. The target phenomenon thus is what we would call the *expressive character* of performances. Users could also select their favorite performance of each piece. Upfront, a few general questions were asked regarding their level of mu-

Figure 1. A word cloud of the terms in the dataset.

sical education, how often they listen to classical music, and whether they play the piano. The order of pieces and performances was randomized. Users could stop at any time.

3.1 Pieces and Responses

The CEG contains 45 performances of 9 excerpts (see Table 2). [1] The length of the excerpts was between 27 seconds and 188 seconds. The CEG was launched on the 3rd of April 2018. The questionnaire was filled out by 194 participants, out of which 88% had some kind of music education – on average 11.7 years; 179 participants answered in English, 12 in German and three in each of Russian, Spanish and Italian. On average, participants listened to the performances of 4.5 out of 9 pieces, 27 participants listened to all the 45 excerpts.

3.2 Meta-data and Annotations

We enriched the dataset by adding score-to-performance alignments for all performances in the dataset, as well as descriptive performance features. The alignments were produced by manually annotating the position of the beats in the audio files using Sonic Visualiser [21]. The scores were encoded manually and exported in MusicXML format following publicly available editions. [2] From these alignments we compute performance features such as tempo and loudness curves (see Section 6.1). The dataset is available online. [3]

4. HOW SIMILARLY DO LISTENERS DESCRIBE A PERFORMANCE OF A PIECE?

Probably the first question that arises concerns the similarity of the descriptions in the dataset, i.e., whether there are

---

[1] `con-espressione.cp.jku.at`

[2] from `https://imslp.org`.

[3] The dataset and code necessary to reproduce the experiments reported in this paper can be found in `https://cpjku.github.io/con_espressione_game_ismir2020/`.

Composer	Piece	#	Pianists
Bach	Prelude No.1 in C, BWV 846 (WTC I)	7	Gieseking, Gould, Grimaud, Kempff, Richter, Stadtfeld, MIDI
Mozart	Piano Sonata K.545 C major, 2nd mvt.	5	Gould, Gulda, Pires, Uchida, MIDI
Beethoven	Piano Sonata Op.27 No.2 C# minor, 1st mvt.	6	Casadesus, Lazić, Lim, Gulda, Schiff, Schirmer
Schumann	Arabeske Op.18 C major (excerpt 1)	4	Rubinstein, Schiff, Vorraber, Horowitz
Schumann	Arabeske Op.18 C major (excerpt 2)	4	Rubinstein, Schiff, Vorraber, Horowitz
Schumann	Kreisleriana Op.16; 3. Sehr aufgeregt (ex. 1)	5	Argerich, Brendel, Horowitz, Vogt, Vorraber
Schumann	Kreisleriana Op.16; 3. Sehr aufgeregt (ex. 2)	5	Argerich, Brendel, Horowitz, Vogt, Vorraber
Liszt	Bagatelle sans tonalité, S.216a	4	Bavouzet, Brendel, Katsaris, Gardon
Brahms	4 Klavierstücke Op.119, 2. Intermezzo E minor	5	Angelich, Ax, Serkin, Kempff, Vogt

Table 1. Performances used in the *Con Espressione* Game.

commonalities in the way listeners describe and like performances. In this section we present a series of analyses that provide different perspectives on the data.

4.1 Frequency and Distribution of Terms

Users provided 1,515 individual descriptions for a total of 3,166 terms, of which 1,415 (approx. 45%) are unique. Figure 1 shows a word cloud of the terms appearing in the dataset. Taken all answers together, each performance was described using at least 47 and at most 114 terms. The average number of terms per piece varies between 60.3 for Liszt's *Bagatelle sans tonalité* and 98.4 for Bach's *Prelude in C*. Each performance is characterized by at least 44 and at most 91 unique terms. The average number of unique terms per piece varies between 51.0 for Liszt's Bagatelle and 78.4 for Bach's Prelude.

The terms *'dynamic'*, *'expressive'*, *'fast'*, *'like'*, *'loud'*, *'mechanical'*, *'slow'*, and *'soft'* appear in at least one performance of every piece. 420 terms appear more than once and, coincidentally, also for different performances. Only 40 terms appear in more than ten performances. The most frequently used terms *across performances* are *'dynamic'*, *'expressive'*, and *'soft'*, which appear in at least 22 performances. In increasing order of occurrence, the terms *'playful'*, *'boring'*, *'dynamic'*, *'mechanical'*, *'slow'*, *'soft'*, *'expressive'*, and *'fast'* are used at least 40 times, with this last term being the most used with 64 occurrences.

4.2 Complexity of the Descriptions

An interesting question is whether there is a relation between listeners' musical backgrounds and the complexity of their answers. In particular, we are interested in determining whether listeners with more musical experience with Western classical music describe performances of this kind of music in a more complex language. As a measure of the complexity of the descriptions we use the Dale-Chall readability score [22], a measure that takes into account the number and commonality of words (defined as words that would be familiar to American students) in a weighted sum. Intuitively, we can understand this measure as a way of combining the length (i.e., number of terms) of the answer and the use of specialized vocabulary (such as musical terms). A larger score means a more complex answer. To test whether listeners with more musical training describe performances in a more complex language, we conduct a linear regression (years of musical training

vs. answer complexity). This analysis reveals a positive correlation that we assessed for statistical significance using a one-tailed Wald test (Pearson's $r = 0.27$, $W = 2.32$, $p = 0.01$, $R^2 = 0.07$), which suggests a small effect. We also test whether listeners that often *listen* to classical music describe performances in more complex terms, and find a small, non-significant correlation. These results present weak evidence supporting the idea that listeners with more musical experience describe the expressive character in more complex ways.

4.3 Listeners' Preferences

To determine how similar the preferences of the listeners are, i.e., if they like (or dislike) similar performances we compute a χ^2-test for each piece to determine if there are clear preferred performances, or if all performances of the same piece are equally liked. There are only three pieces that reject the null hypothesis (the frequency of preferred performances is flat) at a level $\alpha = 0.01$, namely the performance of Bach's piece by Richter; and the performances of the two excerpts of Schumann's Arabeske (in this case, the preferred performances were both by Vorraber). Two additional pieces reject the null hypothesis at level $\alpha = 0.05$: Schumann's Kreisleriana performed by Brendel and Mozart's piece by Pires. Both Gould's performances and the deadpan MIDI performances (for both Bach and Mozart) were the least preferred for this piece. It is important to emphasize that this analysis only indicates the presence of preferred performances; it does not identify what the preferred performance is.

4.4 Semantic Similarity of the Descriptions

We use Li et al.'s [23] method for estimating the semantic similarity of short sentences, to compute the pairwise similarity between descriptions of performances. Similarity between individual terms is estimated as the semantic distance between words in WordNet [24], and the overall sentence similarity is weighted with corpus statistics. Intuitively, this method quantifies the overlap in terms (and very directly related synonyms of these terms) between the answers of the participants. The average similarity of the descriptions of a performance of a piece by the same pianist (i.e., *intra-performance*) is 16%, the average similarity of descriptions of a performance and the descriptions of other performances of the same piece by different pianists (i.e., *inter-performance*) is 15%; and the average similarity

of descriptions of a performance and performances of other pieces (i.e., *inter-piece*, excluding performances of the same piece by other pianists) is 15%. To test whether the differences between these groups are significant, we performed a one-way ANOVA ($F(132, 2) = 9.8$, $p < 0.001$, $\eta^2 = 0.13$), which suggests a medium-small effect. We use t-tests with Bonferroni correction to test the pairwise differences (3 tests, $\alpha = .02$). These tests suggest that the average intra-performance similarity is larger than both the average inter-performance (one-tailed $t(88) = 3.4$, $p < 0.001$, Cohen's $d = .76$) and inter-piece similarity (one-tailed $t(88) = 3.6$, $p < 0.001$, Cohen's $d = .71$). The difference between the inter-performance and inter-piece similarities is not statistically significant (two-tailed $t(88) = 0.0$, $p = 0.99$, Cohen's $d = 0.0$). These results suggest that listeners describe a performance of a piece in more similar terms than they describe other performances of the same piece by different pianists. A possible explanation for the fact that there is little variation in how listeners describe performances of different pieces could be that listeners have a limited vocabulary with which to distinguish the difference in expressive character.

5. WHAT ARE THE MAIN DIMENSIONS FOR EXPRESSIVE CHARACTER?

Most of the terms in Hevner's adjective checklist [15], as well as the main five dimensions of expressive performance markings identified by Sulem et al. [9] and main factors of the GEMMES [18] are present in listeners' responses. However, the characterizations go beyond the aforementioned clusters in at least three aspects: many terms are non-emotional, technical, or disapproving. Non-emotional terms are e.g., those that are hard to unambiguously place in the arousal–valence space (unlike Hevner's or Sulem et al.'s clusters) such as *'clean'*, *'metallic'* or *'[t]his is Glenn Gould, obviously'*. Technical terms include terms that describe playing techniques such as *'legato'*, *'staccato'*, or more generally *'fast'*, *'loud'*, and *'mechanical'*. Disapproving terms include descriptions with negative connotations such as *'boring'*, *'sterile'*, or *'robotic'*.

Regarding automated analysis, characterization of expressive performance is not a common case in natural language processing (NLP). Also, the meanings of many terms in the context of expressive performance are slightly different from their common usage. Preliminary tests indicate that learned occurrence-based semantics on related or general topic corpora largely fail to represent more than superficial similarity for this dataset.

In order to identify the main dimensions of terms, we compute a principal component analysis (PCA) on the occurrence matrix of the dataset. The data is preprocessed in several steps: answers in other languages are translated to English and terms are stemmed. A term is omitted if (1) it shows up less than three times in the dataset (its contribution to the global variance is minimal) or (2) it is used for all interpretations of the same piece (its piecewise entropy is zero; for instance, a participant wrote *'i love mozart'* (sic) for all performances of the Mozart piece).

Table 2 shows the terms that have the strongest loading on the dimensions in the above PCA. Dimension 1 carries intuitive meaning: its extremes reach from *'hectic'* and *'agitated'* to *'gentle'* and *'calm'*. The other three dimensions are harder to connect to a clear semantic dimension. Note for instance the terms *'cold'* and *'warm'*; both influence dimension 4 strongly in the same direction. Figure 2 displays the terms used to describe Mozart's Sonata in the space spanned by the dimensions 1 and 2. The performances themselves are embedded in the space as the centroid of their respective terms. Three clusters emerge, with the deadpan MIDI and Glenn Gould clearly sticking out, and Mitsuko Uchida slightly more towards the *'calm'* and *'sad'* end of Dim.1 (an impression confirmed by listening).

6. HOW DO MEASURABLE/COMPUTABLE PERFORMANCE FEATURES RELATE TO THE EXPRESSIVE CHARACTER DIMENSIONS?

In this section we study whether there is a systematic relationship between the expressive character dimensions described in Section 5 and measurable or computed performance qualities that can be extracted directly from the audio files or from the score-to-performance alignments. In the rest of this article, we refer to these measurable or computed performance qualities as *performance features*.

6.1 Description of Performance Features

6.1.1 Performance Parameters
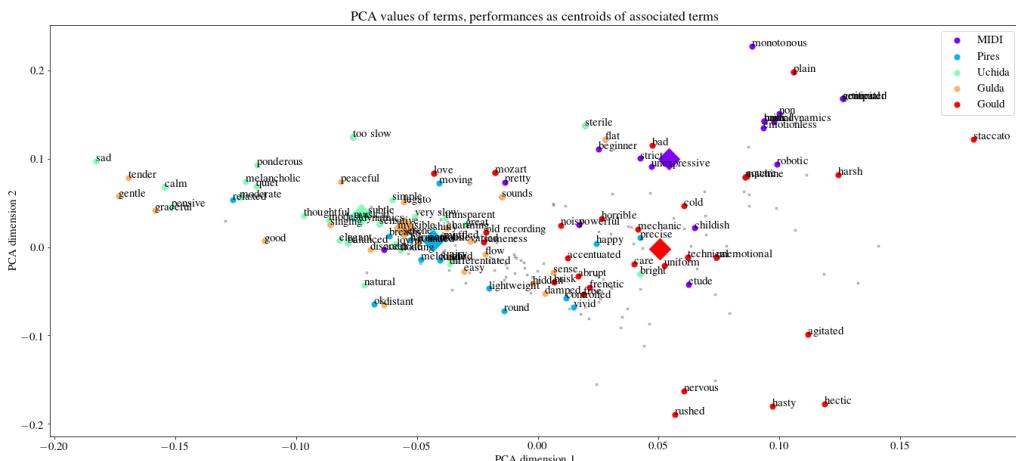
We consider two performance parameters, tempo and dynamics curves, to relate to the expressive dimensions described above. The tempo curves are extracted directly from the hand-corrected score-to-performance alignments by computing the inter-beat intervals. For computing loudness we use the loudness curve computed from the MIR Toolbox [25] using a perceptually weighted smoothing of the signal energy. For inter- and intra-piece comparisons, we calculate the average value, standard deviation, kurtosis and skewness of these curves. Average tempo/dynamics provides an indicator of how fast/loud a performance is, the standard deviation quantifies the tempo/loudness deviations, kurtosis provides a measure of how extreme these deviations are, and skewness indicates how asymmetric the tempo/loudness values are (e.g., whether the piece is regularly faster or slower than the average tempo).

6.1.2 Mid-level Features

Mid-level features are perceptual qualities of music such as articulation, rhythmic clarity and modality that describe overall properties of musical excerpts and are intuitively clear to listeners [11]. We extract the seven mid-level features described in [12], using the deep convolutional network architecture from [26] (the A2Mid variant, specifically). The 7 mid-level features are *melodiousness*, *articulation*, *rhythmic complexity*, *rhythmic stability*, *dissonance*, *tonal stability*, and *minorness* (see [12] for a detailed description of the features). We train our model on the mid-level features dataset [12], which contains 5000

| Dimension 1 | | | | Dimension 2 | | | |
positive loading		negative loading		positive loading		negative loading	
hectic	0.17	sad	-0.20	rushed	0.22	hard	-0.19
staccato	0.15	gentle	-0.18	nervous	0.20	stumbling	-0.18
hasty	0.15	tender	-0.18	too fast	0.17	staccato	-0.17
agitated	0.14	calm	-0.16	bit	0.16	ponderous	-0.14
irregular	0.14	graceful	-0.16	hasty	0.15	monotonous	-0.13
Dimension 3				**Dimension 4**			
positive loading		negative loading		positive loading		negative loading	
monotonous	0.22	heavy	-0.14	ok	0.24	cold	-0.15
bad	0.17	graceful	-0.13	happy	0.21	warm	-0.14
warm	0.16	smooth	-0.12	joyful	0.19	floating	-0.14
peaceful	0.16	ponderous	-0.12	free	0.15	blurred	-0.14
beautiful	0.15	soaring	-0.10	breathy	0.14	mysterious	-0.13

Table 2. Terms with strongest loadings for each expressive character dimension.

Figure 2. A visualization of the first two dimensions recovered by the PCA. Dots represent terms used in the CEG. Colored terms from characterizations of Mozart's Sonata K 545, grey terms from other pieces. The terms are colored according to the performance they characterize.

audio snippets of 15 seconds each, and use the trained model to predict the mid-level features of the piano performances without any fine-tuning, as there is too little data for a supervised fine-tuning step. To improve the validity of the transfer, we incorporate unsupervised domain adaptation [27] during the training phase. Since the pieces in the CEG are piano performances, we use a separate private collection of non-annotated piano music as the data source for domain adaptation. We observe more variation in the mid-level predictions between the performances while using a domain-adapted model than a non-domain-adapted one, which indicates that it is a useful step in the pipeline.

6.1.3 High-level Features

As high-level emotion-related descriptors, we choose the common *arousal* and *valence* dimensions [19, 28] and aim to predict these from the audio recordings, to then relate them to the expressive character dimensions. We train a dynamic arousal-valence prediction network using the DEAM dataset [29]. We tested a VGG-like model, similar to the one described in Section 6.1.2, and we observed that when the network is pre-trained on the mid-level dataset

and extended with a multi-layer GRU-RNN (Gated Recurrent Unit Recurrent Neural Network) that is trained on the DEAM dataset, we get the best results. To improve the results further, we use the recently released receptive-field regularized ResNet [30] for the pre-training phase, since it appears to give better results for short audio snippets than the VGG-like variant. The inputs to our network are Mel-spectrograms and we process 2-second segments of the spectrogram with 0.5-second hops. As in the case of the expressive performances, in order to compare the predicted arousal and valence curves for inter- and intra-piece comparisons, we compute average, standard deviation, kurtosis and skewness of these curves for each performance of each piece.

6.2 Analysis with Multiple Linear Regression

To study the relation between the performance parameters and mid- and high- level features to the expressive character dimensions described in Section 5 we use multiple linear regression (MLR) analysis. In this analysis, the *dependent variables* are each of the expressive character dimensions (Dimensions 1 to 4) and the *independent vari-*

Dimension 1		Dimension 2		Dimension 3		Dimension 4	
PP ($R^2 = 0.24$)		PP ($R^2 = 0.18$)		PP ($R^2 = 0.26$)		PP ($R^2 = 0.24$)	
loudness avg	0.51^{***}	loudness sk	0.45^{**}	loudness std	-0.53^{**}	beat period k	-0.34^{*}
						loudness std	-0.44^{*}
MF ($R^2 = 0.39$)		MF ($R^2 = 0.00$)		MF ($R^2 = 0.00$)		MF ($R^2 = 0.29$)	
rhythmic complexity	-0.74^{*}	minorness	0.15	articulation	-0.15	rhythmic complexity	0.52^{*}
tonal stability	-0.94^{**}					tonal stability	0.84^{***}
articulation	0.46^{*}						
HF ($R^2 = 0.22$)		HF ($R^2 = 0.00$)		HF ($R^2 = 0.36$)		HF ($R^2 = 0.09$)	
valence sk	0.48^{**}	valence avg	0.14	valence k	0.42^{**}	valence k	-0.33^{*}
				arousal avg	-1.24^{***}		
				valence std	0.27^{*}		
				valence avg	-0.82^{*}		

Table 3. Multiple Linear Regression Analysis. PP, MF and HF refer to performance parameters, mid- and high- level features, respectively. avg, std, k and sk denote average, standard deviation, kurtosis and skewness. The values are the regression coefficients (indicating the contribution of that feature to the model). R^2 is the adjusted coefficient of determination for the whole model. *, **, and *** indicate statistical significance at levels $\alpha = .05, .01$ and $\alpha < .001$, respectively.

ables are the performance features described above. We carry out $4 \times 3 = 12$ MLRs for each expressive character dimension (4 in total) and subset of performance features (expressive parameters, mid- and high-level features). Each of these regressions investigates whether each subset of performance features (expressive parameters, mid-/high-level) can significantly predict the position of the pieces in the expressive character dimensions. The position of each piece in the 4D expressive character space is computed as the centroid of all of its terms in this space. For each of these MLRs we perform a variable selection using the Zheng-Loh method [31]. The results are summarized in Table 3. The MLR results indicate that the expressive parameters are significant predictors of all 4 expressive character dimensions, with medium effect sizes (R^2). Mid-level features are only significant predictors of Dimensions 1 and 4. High-level features are only significant for Dimensions 1 and 3. Thus, Dimension 1 (the *'gentle'/'calm'* vs. *'hectic'/'agitated'* axis, see Section 5) seems systematically related to our performance features at all three levels, which further corroborates its significance.

7. DISCUSSION

In Section 4.2 we observed a small positive relationship between the complexity of verbal descriptions and listeners' musical training. We expect that stronger evidence of a relationship would emerge if musical training were better controlled for (our sample had few listeners with < 5 years of training) and the complexity measure were further developed to account for specialized musical terms. Our analysis of listeners' preferred performances in Section 4.3 revealed that the deadpan performances and performances by Glenn Gould were least well-liked. Prior research has suggested that listeners prefer quantitatively average expressive performances [32], which might explain partially the lack of enthusiasm for Gould's idiosyncratic playing.

The results in Sections 4.4 and 5 suggest that listeners tend to describe performances of the same piece similarly, although there is some variability (e.g., a performance can

be described both as *'beautiful'* or *'bad'* by different listeners; cf. both *'cold'* and *'warm'* being negatively correlated with Dimension 4). An important issue is that NLP methods for assessing similarity between the descriptions are not really suitable for analyzing performance descriptions, where each term is loaded with complex meaning [4] as well as many cross-domain mappings (e.g., metaphors).

The results in Section 6.2 reveal relationships between performance features and expressive dimensions that conform to musical intuition, with the effects being most pronounced for expressive character Dimension 1 (which is also the one that we find easiest to interpret, see table 2). For instance, the analysis suggests that louder performances or performances with large outliers in the valence curve would be perceived as more irregular and agitated, while softer performances or performances without large outliers in valence would be perceived as calm or graceful.

8. CONCLUSIONS AND FUTURE WORK

This paper has introduced the CEG dataset and presented some exploratory analysis addressing three main questions related to inter-listener agreement, main emerging description dimensions, and relations between user characterizations and measurable performance parameters.

Future work will focus on a more in-depth analysis of the question of semantic similarity. As discussed in Section 5, the description of expressive character includes many nuances that are not well suited to be analyzed with generic NLP methods, given how loaded with meaning certain terms are. We plan to investigate methods like *pile sorting* [33] with expert musicians to devise a meaningful semantic clustering of the terms. Furthermore, we plan to collect more human annotations (e.g., mid- and high-level features) as a basis for a more systematic comparison.

---

[4] For example, the performance of the Mozart piece by Austrian-trained Japanese pianist Mitsuko Uchida was described by a participant as 'Russian pianist'. To understand this description, it is necessary to have the concept of the Russian School of performance.

9. ACKNOWLEDGMENTS

10. REFERENCES

[1] S. Davies, "Philosophical Perspectives on Music's Expressiveness," in *Music and emotion: Theory and research.*, P. N. Juslin and J. A. Sloboda, Eds. Oxford University Press, 2001, pp. 23–44.

[2] C. Rosen, *Music and Sentiment.* New Haven and London: Yale University Press, 2010.

[3] P. N. Juslin and J. A. Sloboda, "Music and emotion," in *The Psychology of Music*, 3rd ed., D. Deutsch, Ed. London, UK: Academic Press, 2013, ch. 15, pp. 583–645.

[4] P. N. Juslin, "Communicating emotion in music performance: a review and theoretical framework," in *Music and Emotion: Theory and Research.* Oxford University Press, 2001, pp. 309–337.

[5] ——, "Emotional communication in music performance: A functionalist perspective and some data," *Music Perception: An Interdisciplinary Journal*, vol. 14, no. 4, pp. 383–418, 1997.

[6] P. N. Juslin and R. Timmers, "Expression and communication of emotion in music performance," in *Handbook of music and emotion: Theory, research, applications*, P. N. Juslin and J. A. Sloboda, Eds. New York: Oxford University Press, 2010, pp. 453–489.

[7] P. Juslin, "Five facets of musical expression: a psychologist's perspective on music performance," *Psychology of Music*, vol. 31, no. 3, pp. 273–302, 2003.

[8] P. N. Juslin, A. Friberg, and R. Bresin, "Toward a computational model of expression in music performance: The GERM model," *Musicae Scientiae*, vol. 5, no. 1, pp. 63–122, 2001.

[9] A. Sulem, E. Bodner, and N. Amir, "Perception-based classification of expressive musical terms: Toward a parameterization of musical expressiveness," *Music Perception: An Interdisciplinary Journal*, vol. 37, no. 2, pp. 147–164, 2019.

[10] G. Widmer, "Getting Closer to the Essence of Music: The *Con Espressione* Manifesto," *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 2, pp. 1–13, Jan. 2017.

[11] A. Friberg, E. Schoonderwaldt, A. Hedblad, M. Fabiani, and A. Elowsson, "Using listener-based perceptual features as intermediate representations in music information retrieval," *The Journal of the Acoustical Society of America*, vol. 136, no. 4, pp. 1951–1963, 2014.

[12] A. Aljanaki and M. Soleymani, "A Data-Driven Approach to Mid-level Perceptual Musical Feature Modeling," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, Paris, France, 2018.

[13] Y.-H. Yang and H. H. Chen, "Machine recognition of music emotion: A review," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 1–30, 2012.

[14] C. Cancino-Chacón, M. Grachten, W. Goebl, and G. Widmer, "Computational Models of Expressive Music Performance: A Comprehensive and Critical Review," *Frontiers in Digital Humanities*, vol. 5, p. 25, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fdigh.2018.00025

[15] K. Hevner, "Experimental Studies of the Elements of Expression in Music," *The American Journal of Psychology*, vol. 48, no. 2, pp. 246–268, April 1936.

[16] P. R. Farnsworth, "A Study of the Hevner Adjective List," *The Journal of Aesthetics and Art Criticism*, vol. 13, no. 1, pp. 97–103, September 1954.

[17] E. Schubert, "Update of the Hevner Adjective Checklist," *Perceptual and Motor Skills*, vol. 96, pp. 1117–1122, 2003.

[18] S. Schaerlaeken, D. Glowinski, M.-A. Rappaz, and D. Grandjean, ""Hearing Music as ...": Metaphors Evoked by the Sound of Classical Music," *Psychomusicology: Music, Mind and Brain*, vol. 29, no. 2-3, pp. 100–116, 2019.

[19] J. A. Russell, "A Circumplex Model of Affect," *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[20] M. Murari, A. Rodà, S. Canazza, G. D. Poli, and O. D. Pos, "Is vivaldi smooth and takete? nonverbal sensory scales for describing music qualities," *Journal of New Music Research*, vol. 44, no. 4, pp. 359–372, 2015. [Online]. Available: https://doi.org/10.1080/09298215.2015.1101475

[21] C. Cannam, C. Landone, and M. Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," in *Proceedings of the ACM Multimedia 2010 International Conference*, Firenze, Italy, October 2010, pp. 1467–1468.

[22] E. Dale and J. S. Chall, "A Formula for Predicting Readability," *Education Research Bulletin*, vol. 27, no. 1, pp. 11–20+28, January 1948.

[23] Y. Li, D. McLean, Z. Bandar, J. D. O'Shea, and K. Crockett, "Sentence Similarity Based on Semantic Nets and Corpus Statistics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, pp. 1138–1150, August 2006.

[24] G. A. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[25] Lartillot, Olivier and Toiviainen, Petri, "A Matlab toolbox for musical feature extraction from audio," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-07)*, Bordeadux, France, 2007.

[26] S. Chowdhury, A. Vall, V. Haunschmid, and G. Widmer, "Towards Explainable Music Emotion Recognition: The Route via Mid-level Features," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, Delft, The Netherlands, 2019.

[27] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32th International Conference on Machine Learning (ICML 2015)*, Lille, France, 2015.

[28] N. H. Frijda, *The emotions*. Cambridge, UK: Cambridge University Press, 1986.

[29] A. Aljanaki, Y.-H. Yang, and M. Soleymani, "Developing a benchmark for emotional analysis of music," *PloS one*, vol. 12, no. 3, 2017.

[30] K. Koutini, H. Eghbal-zadeh, and G. Widmer, "Receptive-field-regularized cnn variants for acoustic scene classification," in *Proceedings of the Accepted at Detection and Classification of Acoustic Scenes and Events 2019 (DCASE Workshop 2019)*, New York, NY, USA, 2019.

[31] X. Zheng and W.-Y. Loh, "A consistent variable selection criterion for linear models with high-dimensional covariates," *Statistica Sinica*, vol. 7, no. 2, pp. 311–325, 1997. [Online]. Available: http://www.jstor.org/stable/24306081

[32] B. Repp, "The aesthetic quality of a quantitatively average music performance: Two preliminary experiments," *Music Perception*, vol. 14, no. 4, pp. 419–444, 1997.

[33] R. Trotter and J. M. Potter, "Pile sorts, a cognitive anthropological model of drug and aids risks for navajo teenagers: Assessment of a new evaluation tool," *Drugs & Society*, vol. 7, no. 3-4, pp. 23–39, 1993.

TOWARDS A FORMALIZATION OF MUSICAL RHYTHM

Martin Rohrmeier
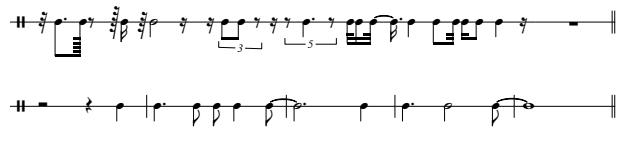
Digital and Cognitive Musicology Lab, École Polytechnique Fédérale de Lausanne

`martin.rohrmeier@epfl.ch`

ABSTRACT

Temporality lies at the very heart of music, and the play with rhythmic and metrical structures constitutes a major device across musical styles and genres. Rhythmic and metrical structure are closely intertwined, particularly in the tonal idiom. While there have been many approaches for modeling musical tempo, beat and meter and their inference, musical rhythm and its complexity have been comparably less explored and formally modeled. The model formulates a generative grammar of symbolic rhythmic musical structure and its internal recursive substructure. The approach characterizes rhythmic groups in alignment with meter in terms of the recursive subdivision of temporal units, as well as dependencies established by recursive operations such as preparation and different kinds of shifting (such as anticipation and delay). The model is formulated in terms of an abstract context-free grammar and applies for monophonic rhythms and harmonic rhythm.

1. INTRODUCTION

Temporality lies at the very heart of music, and the play with rhythmic and metrical structures constitutes a major device across musical styles and genres. However, there is comparably less research on rhythm itself than on other temporal structures of music. For instance, there is a lot of work on modeling beat and beat inference [1–3], tempo estimation [4] as well metrical structure [5] and its inference [6–10]. There has been major theoretical work differentiating between grouping and meter [11], and between rhythm and meter [5, 12]. In comparison, there is less formalization work on musical rhythm [13–15], and some major studies such as the GTTM [11] or [5] avoid a formal characterization of musical rhythm. The purpose of this theoretical paper is to address this gap and to provide a generative model of musical rhythm in terms of an abstract context-free grammar that generates rhythmic structure in alignment with metrical structure.
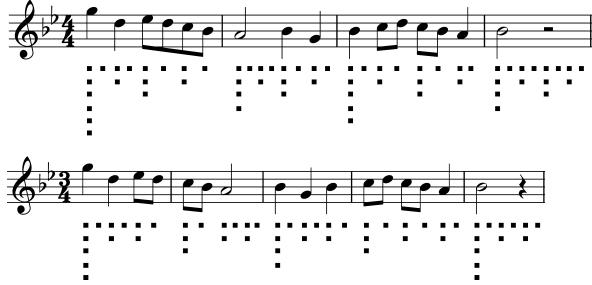
Figure 1: Two different series of onsets and durations.

2. MOTIVATION

Rhythm is commonly thought of as a series of onsets and durations of musical events. While duration patterns are essential for musical rhythm, the core idea of the model is to capture that rhythmic structures, especially those in tonal music, are more than (fully) freely placed onsets over time and that the concept of rhythm involves different kinds of dependencies that are constituted between its musical events. To illustrate this point, Figure 1 displays two different series of onsets and durations. Only the lower one looks like a plausible candidate for a rhythm from a tonal piece of music. There are several points underpinning this distinction. In essence, it is argued that rhythm is understood involving an *interpretation* in terms of hierarchical dependencies of temporal events and their assignment to the metrical grid, which result in a surface projection of patterns of onsets and durations.

One central point is that the rhythmic Gestalt is fundamentally defined by its relation to metrical structure; rhythm cannot be separated from a metrical interpretation. This point is illustrated by Figure 2. Both rhythms have the identical sequence of onsets and durations, but different metrical structures associated with them, and this results in both rhythms sounding very different. In particular, the ways in which events are linked to weak and strong metrical beats and also their underlying meter have a major impact on the interpretation of a given pattern of durations.

A second major point lies in the fact that we characterize rhythmic structures in terms of an *interpretation* by event dependencies and transformations. For instance, certain events lead to other events; we understand a certain event or group of events as an upbeat to (or preparation of) another event; and we understand certain events as subdivisions of longer units (such as triplets). Furthermore, when we speak about syncopation, anticipation, or delay, it means that certain events occur earlier or later, implying that there is an (underlying) position where these events would have been expected normally before the transformation (shift) [13, 16]. Rhythmic events are also recognized as grouped.

Figure 2: The alignment with metrical structure may yield two very different rhythmic interpretations for the same pattern of onsets and durations.

Generally, it is useful to distinguish different levels of musical time [17–20]. Rhythmic structure, in the aspect that is modeled here, lives in *idealized* time. The metrical grid presumes an underlying isochronic beat, and rhythmic patterns are related by simple integer ratios in relation to the grid and the beat. The rhythmic structure at this symbolic level in relation to the symbolic beat is different from the level of tempo variation, expressive timing, swing, groove, performance errors and other subsymbolic variations of timing.

In sum, we generally conceive of rhythms as structured both in terms of an associated metrical structure as well as in terms of event dependencies such as the ones mentioned. In contrast, events occurring with purely random onsets and durations (like the first example in Figure 1) sound erratic—which in turn means that they have no *interpretation* in terms of the dependencies outlined. It is the purpose of the proposed model to express the various rhythmic dependencies at the deep structure that give rise to the patterns of event onsets and durations observed at the surface.

One common observation in rhythmic structure is that events may reach into the timespan of other events. This is particularly common with preparations before an event, anticipations or syncopations that may enter during the timespan of the directly preceding event. This may cause the preceding event to be shortened, which we refer to as *time stealing* when it is discussed below.

2.1 Related literature

Numerous approaches have addressed rhythmic and metrical structure in music [5, 21]. While there are several research directions in terms of rhythmic corpus studies [22–27] and mathematical analyses [28–30], there is less research proposing formal theoretical frameworks generating rhythmic structure. Differentiating metrical structure from grouping, the GTTM [11] laid a foundation for the understanding of meter that is still in place today. Several endeavors have been devoted to implementing the GTTM in a computational way [31, 32].

Several computational approaches to rhythm have proposed sequential models such as Markov models, HMMs or other graphical models [33, 34]. More recently various hierarchical approaches and probabilistic grammars have been used for rhythmic inference and transcription problems [15,35–37]. These approaches are essentially built on recursive subdivision (split). [14] proposed an algorithmic model of rhythm using transformations of syncopation, figural, and density (split) based on the transformation vector proposed by [13]. From the perspective of mathematical music theory, rhythmic structure has also been modeled in terms of subdivision of a graph [38].

The present model extends previous approaches [14,15, 39] by characterizing an overarching abstract context-free grammar of recursive rhythmic dependencies. It is based on five abstract operations of splits, preparations, and shifts using a tripartite representation of rhythmic categories, and models rhythmic conflicts using the concept of *time stealing*. As a grammar-based generalized model of rhythm, it can be naturally integrated with syntactic models of harmony [39–48] for modeling harmonic rhythm.

3. THE FORMALISM

3.1 Metrical structure

Metrical structure has been famously modeled by [11] with a recursive grid of metrical weights and a notation adopted from metrical phonology in linguistics [49–53]. Examples of the metrical grid are shown in Figures 2 and 3. In the grid, each level m is characterized by a multiple (2 or 3) of the period of the subordinate level $m - 1$ and an offset. Generally, the subdivision for regular meters is binary or ternary [5]; for irregular meters the formalization would need to be extended to combinations of twos and threes (e.g. $\frac{7}{8} = \frac{3}{8} + \frac{2}{8} + \frac{2}{8}$).

The metrical grid can be characterized as follows for regular meters: the beat level is marked with m_0 and the beginning of the segment or piece is indicated by the index 0, and locations are indicated in reference to m_0. Each higher metrical level i is characterized by the tuple (π_i, o_i), the regularity $\pi_i \in \{2, 3\} \cdot \pi_{i-1}$, and the offset $o_i := o_{i-1} + a\pi_{i-1}, with a \in \mathbb{N}_0$, $o_0 = 0$ and $\pi_0 = 1$. This ensures that higher levels can only subselect beats established in all lower metrical levels. Metrical levels below the beat ($i < 0$) are characterized in the same way with $\pi_i \in \{\frac{1}{2}, \frac{1}{3}\} \cdot \pi_{i+1}$, and the offset $o_i = 0$. Accordingly, a metrical grid M is fully defined by the list (or series) of all tuples $M := (\pi_i, o_i)$. The metrical grid is potentially infinite in duration and has an arbitrary number of metrical levels i. Most commonly, almost all values of π are 2, except for the levels 1 and 2 in ternary meters.

The metrical weight at position t is characterized as:

$$W_{M:=(\pi_i, o_i)}(t) = \sum_i \left(1 - sign(\mid t - o_i \mid \mod \pi_i)\right) \ (1)$$

where the $1 - sign(\cdot)$ function is used to compute 1 for a position falling on the metrical grid and 0 otherwise. Furthermore, a subsegment of a metrical grid M is characterized by $M_{[a,b]}$, where a and b denote the beginning and end locactions of the open or closed subsegment interval.

3.2 Generative rhythm

The formalism is modeled employing abstract context-free grammars [47]. The grammar consists of four parts: $\mathcal{G} = (C, \Sigma, P, C_0)$, non-terminal rhythmic categories C, terminal symbols Σ, production rules P, and, in this case, a set of start symbols $C_0 \subseteq C$. All sets C, Σ, C_0 are infinite.

A rhythmic category consists of a tuple (t, m): a timespan t and the metrical grid m associated with the timespan. The timespan represents a formalization of the rhythmic time interval (which is going to be generatively subdivided). It is itself a triple $[a : b : c]$ that combines three parts: a downbeat part of a duration b, also called the *body* of t, and an initial offset (upbeat) of duration a, and final offset (coda) of duration c. Durations are defined in beats ($\in \mathbb{Q}$) including fractions of beats, such as $\frac{1}{8}$. The offset parts can be positive or negative. If positive, a time segment is added to the core length of b; if negative, b is shortened by that amount. The total duration of t is $a + b + c$. While in practice, a and c are each mostly no longer than $\frac{b}{2}$, it is avoided to postulate such a restriction theoretically rather than empirically. However, $-b < a < b$, $-b < c < b$, and $a + c <= b$ are required.

The set of surface symbols Σ consists of musical events that are characterized by pairs (l, m) of event durations $l \in \mathbb{Q}, l \leq b$ in beats with associated metrical weights $m \in \mathbb{Q}$. The separate modeling of the event length in addition to the timespan is important because a realized event may in turn occupy a shorter duration than its timespan (e.g. a quarternote in a half-note timespan, or a staccato note). This makes it possible to model the type of rests that can occur in this case. Since the focus in this paper is about the rhythmic grammar, other features of the musical event (like pitch) will be left out of account.

The set of start symbols $C_0 \subseteq C$ consists of rhythmic categories that do not have a coda offset:

$$C_0 = \{k \mid k = (t = [a : b : c], m) \in C \wedge c = 0\} \quad (2)$$

Because the symbol space is infinite, rules do not constitute rewrite operations over symbols (as in classical context-free grammars), but as rewrite functions. The set of generative production rules P is characterized as

$$P \subseteq \{r \mid r : C \to (C \uplus \Sigma)^*\}, \quad (3)$$

where the rules are functions that map sets of categories onto sets of categories or surface symbols, establishing an abstraction over different paramaterized instantiations of analogous rules.

The set of core rules constitutes the heart of the generative formalism. For the generation of rhythmic structure, five main rules are assumed: *split*, *prepare*, *(c-split)*, *anticipate*, and *delay*.
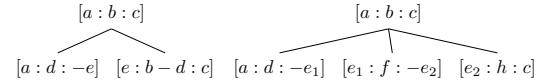
Split rule. The main rule of the formalism is *split*, which subdivides the body of the rhythmic category, while the outer upbeat and coda parts remain identical. The split can induce *timestealing* such that a timespan protrudes into

an adjacent one by an upbeat or coda of length e. Note that it does not result in an update of the length of the timespan body b, but of its upbeat or coda part. This maintains the core durations at the deep structure. Split can subdivide a timespan into two or three parts; other subdivisions, such as four or five, etc., require multiple split operations.

$$
\begin{aligned}
&([a : b : c], m_0) \longrightarrow ([a : d : -e], m_1)\,([e : b - d : c], m_2) \\
&\quad \mid 0 < d < b, m_1 = m_{0[0, a+d-e[}, m_2 = m_{0[a+d-e, a+b+c]}
\end{aligned}
\quad (4)
$$

$$
\begin{aligned}
&([a : b : c], m_0) \longrightarrow \\
&([a : d : -e_1], m_1)\,([e_1 : f : -e_2], m_2)\,([e_2 : h : c], m_3) \\
&\quad \mid d + f + h = b, \quad m_1 = m_{0[0, a+d-e_1[}, \\
&m_2 = m_{0[a+d-e_1, a+d+f-e_2[}, m_3 = m_{0[a+d+f-e_2, a+b+c]}
\end{aligned}
\quad (5)
$$
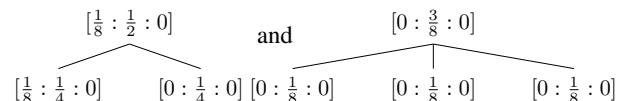
The working of these rules may be visualized by the subtrees they produce (ignoring the metrical assignment):

It is further assumed that the majority of these split operations divide the body of the timespan equally ($d = f = \frac{b}{2}$, for even b) or into simple integer ratios. The actual instantiation of these splits in practice is, however, not a matter that should be decided a-priori at the level of the formalism. Note, for instance, that an unequal subdivision like 3:1 in the context of a long note and an upbeat to the next bar, is not required since such cases are rather expressed with the upbeat (u-split) rule that is explained next.

Also note that for categories where the metrical accent lies at the beginning of the body, it is sufficient to metrically characterize the split segments by the metrical weight at the downbeat (onset) of the body. The notation $m = u \oplus v$ is used to denote the weight of the downbeat in terms of the metrical level u generated by the split operation plus the metrical weight v inherited from the parent node in the tree. Figure 3 and 4 illustrate this notation.
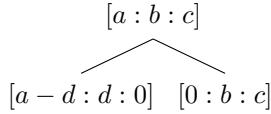
As an example, a halfnote split into two quarter-notes or dotted quarter-note split into three eighth-notes would be expressed like this:
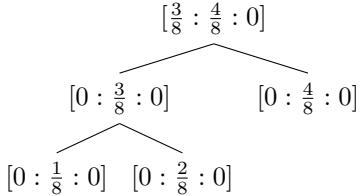
Prepare (U-Split) rule. The second core rule models upbeat structures. It takes the upbeat part of the timespan of a rhythmic category and generates an own rhythmic category from it:

$$
\begin{aligned}
&([a : b : c], m_0) \longrightarrow ([a - d : d : 0], m_1)([0 : b : c], m_2) \\
&\quad \mid 0 < d \leq a, m_1 = m_{0[0, a[}, m_2 = m_{0[a, b+c]}
\end{aligned}
\quad (6)
$$

The corresponding tree fragment looks like this:

$$[a:b:c]$$
$$[a-d:d:0] \quad [0:b:c]$$

For example, a half-note that is prepared by a combined upbeat of an eighth and a quarternote would be expressed as follows:

$$\left[\tfrac{3}{8}:\tfrac{4}{8}:0\right]$$
$$\left[0:\tfrac{3}{8}:0\right] \qquad \left[0:\tfrac{4}{8}:0\right]$$
$$\left[0:\tfrac{1}{8}:0\right] \quad \left[0:\tfrac{2}{8}:0\right]$$

One may postulate a corresponding counterpart to the upbeat split in the *prepare* rule; this rule (*c-split*) would instantiate a rhythmic category from the coda-part of a given rhythmic category. While it is unclear if such a rule would indeed be required for tonal rhythm (i.e. a phenomenon of "post-paration" as opposite to "pre-paration"), still the rule is listed even though it may be dropped from the formalism (or be found to not occur empirically):

$$([a:b:c], m_0) \longrightarrow ([a:b:0], m_1)\,([c-d:d:0], m_2)$$
$$\mid 0 < d \le c, m_1 = m_{0[0,a+b[}, m_2 = m_{0[a+b,a+b+c]} \quad (7)$$

Shift rules. Two further rhythmic phenomena do not relate to subdivision but to the shift of events, such as in the context of syncopations. In these cases, a rhythmic category c may be shifted to occur early or late. The corresponding rules are *anticipate (e-shift)* and *delay (l-shift)*. These rules are unary rules that transform a rhythmic category rather than creating a new one.

$$\text{e-shift: } ([a:b:c], m_0) \longrightarrow ([0:b:a+c], m_0) \quad (8)$$

$$\text{l-shift: } ([a:b:c], m_0) \longrightarrow ([a+c:b:0], m_0) \quad (9)$$

Surface rules. Finally, from the set of recursive generated rhythmic subdivisions and transformations a rhythmic surface will be generated. If events are shortened by *timestealing*, the lengths of the core are updated (by the first two rules). In order to ensure that all upbeat parts and codas have been instantiated either with events or shifts, surface symbols can only be generated for categories that have an empty upbeat and coda part. Once generated, surface symbols cannot reenter the generative process.

$$([a:b:c], m) \longrightarrow ([0:b+a:c], m) \text{ for } a < 0 \quad (10)$$
$$([a:b:c], m) \longrightarrow ([a:b+c:0], m) \text{ for } c < 0 \quad (11)$$
```
surface:
```
$$([0:b:0], m) \longrightarrow (b, W_m(0)) \quad (12)$$
$$([0:b:0], m) \longrightarrow (l, W_{m_{[0,l[}}(0))(\epsilon, b-l, W_{m_{[l,b]}}(0)) \quad$$
$$\mid 0 < l \le b \quad (13)$$

In other words, the surface rule yields the rhythmic surface duration $l = b$ or $l \le b$ as well as its metrical weight

m. If the event is shorter than its timespan b, the surface rule also creates a rest event ϵ that fills up the remaining space, so that the subsequent events are not affected by the shortening.

The surface rule is designed in such a way that all lengths of all surface events add up to the full length of the entire musical segment from the start symbol:

$$\sum l_i = a + b \quad \text{for } c_0 = [a:b:0] \in C_0 \quad (14)$$

An illustrative example of this sum can be reconstructed from the surface-note durations in Figure 3 and 4.

4. EXAMPLES

4.1 Melodic rhythm

A first detailed analysis is carried out on the first two bars of the jazz standard "Blue Bossa". The tree analysis based on the generative model is displayed in Figure 3 together with a corresponding analysis that visualizes the recursive rhythmic subdivisions and shifts of the same generation using musical score lines. All durations are encoded following common music notation, i.e. $\frac{1}{2}$ refers to a half-note, 1 to a whole note, or $\frac{3}{8}$ to a dotted quarter-note; $\frac{1}{4}$ refers to the quarter-note beat level. The figure characterizes the metrical (sub)grids of each category employing the $m = u \oplus v$ notation.
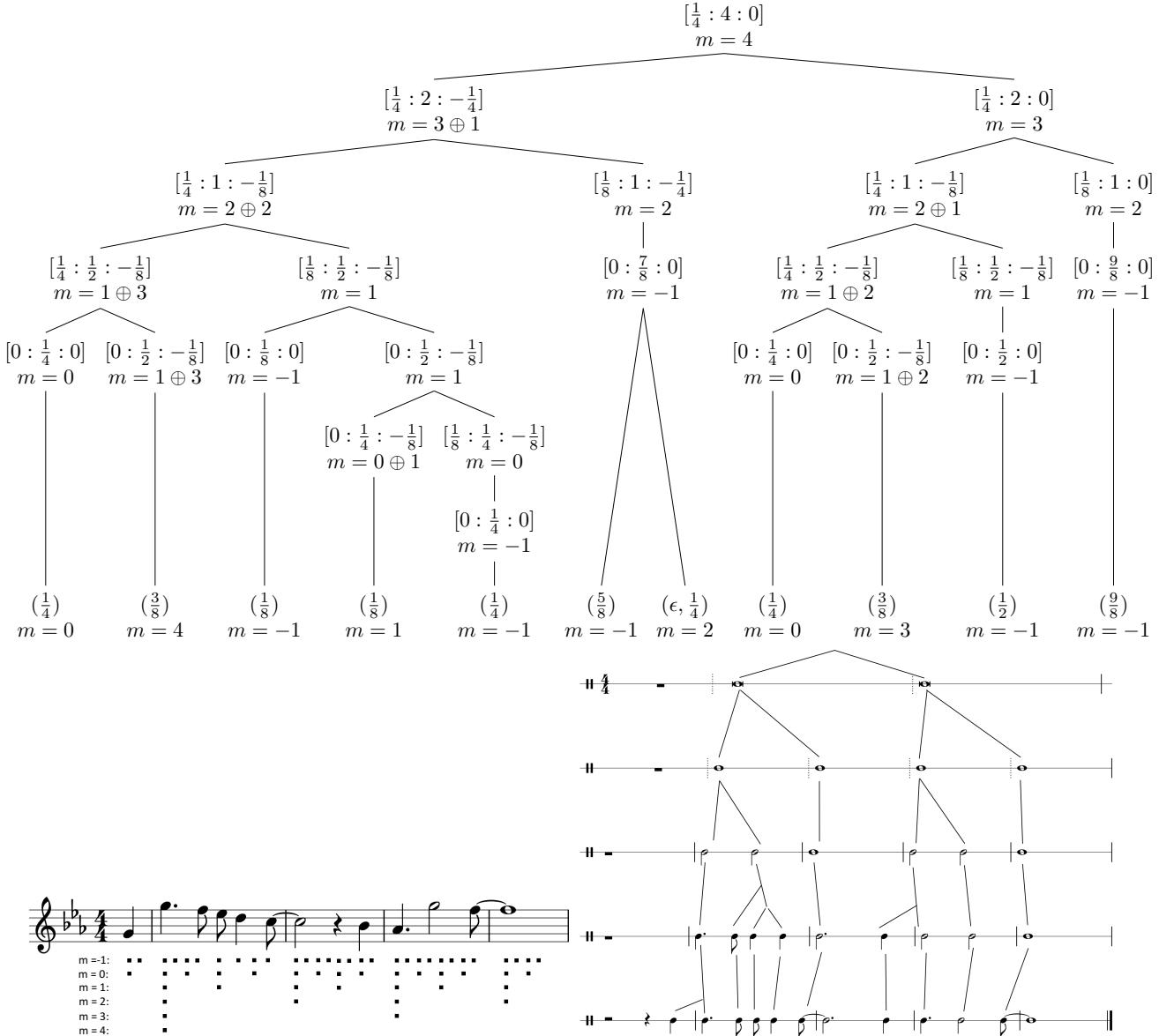
Several observations can be made based on the figure. All of the applications of *split* illustrate that timespans may be subdivided with equal subdivision of the core, yet resulting in unequal timespan durations based on time-stealing effects encoded in the upbeat and coda parts of the timespan category. The derivation of the sixth note provides an example where an event at the metrical whole-note level is syncopated by an eighth note and at the surface instantiated shorter than its timespan resulting in the surface generation of an additional rest.

Further, the first, third, and seventh note may be understood as upbeats to the subsequent events. The second halves of measures 1 and 3 have syncopations in which the shifted events reach into the timespan of the previous events, causing the notes on beats 3 and 1, respectively, to be shortened by one eighth note.

4.2 Harmonic rhythm

The formalism proposed has a different application in the modeling of harmonic rhythm. For instance, this concerns modeling the harmonic-rhythmic structure as it is contained in leadsheets. A major difference between the previous case of melodic rhythm is that harmonic rhythm in leadsheets may well employ (harmonic) upbeats, yet no rhythmic shifts in the sense of syncopation, anticipation or delay; also time overlaps that involve the timespan coda have not been observed. A computational version of this (sub)model has been proposed in [39].

Figure 4 shows an example analysis of the first 8 bars of the harmonic phrase of "Blue Bossa". The tree analysis displays the harmonic syntactic dependencies following [47, 54, 55] in conjunction with the harmonic rhythm
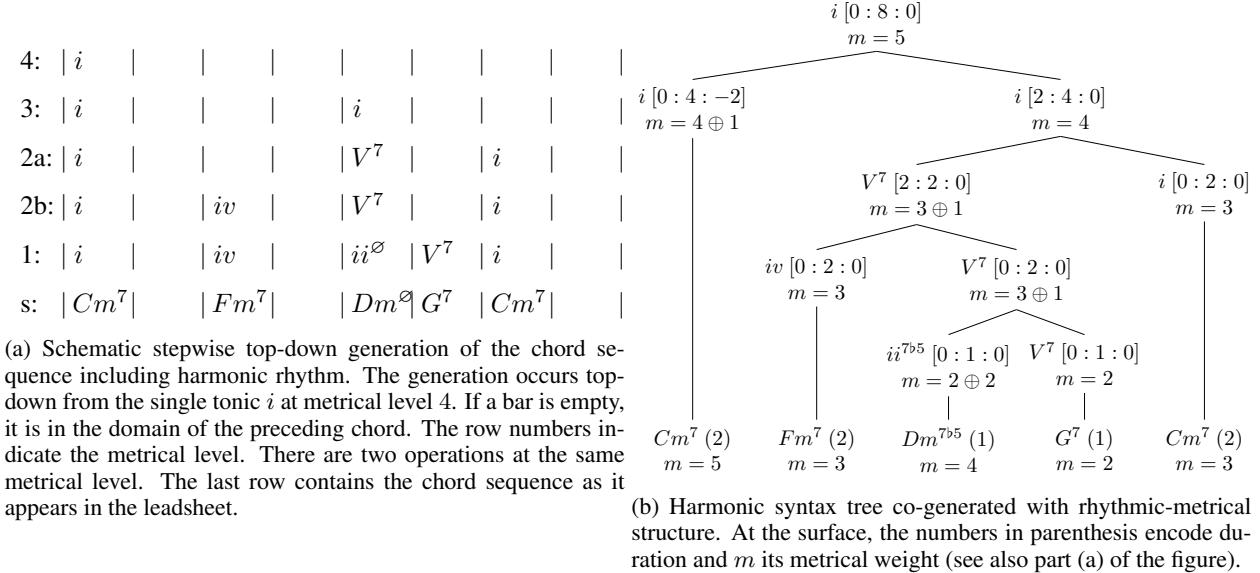
Figure 3: A rhythmic analysis of the first four bars of the melody of the Jazz standard "Blue Bossa".

and the metrical levels. This requires a product grammar as defined by [39] for the formulation of the coordination between harmonic and rhythmic structures—which is an application of the formalism as proposed here. The tonic i at the highest level is equally split into two tonic timespans at level 3. When the preparing dominant V^7 is inserted, it causes to take up half of the space of the timespan of i and causes i to appear later. When iv is introduced preparing V^7 it is introduced at the same metrical level (level 2) and reaches into the time domain of the initial i (time-stealing). By analogy, the introduction of $ii^\varnothing$ takes up the half of the V^7 time domain. Accordingly, the analysis reveals that the metrical domains of the chords in the hierarchical analysis are not identical with the position where the chords occur on the surface. Figure 4 (a) displays the stepwise joint derivation of harmonic syntactic dependencies and harmonic rhythm.

5. DISCUSSION

The contribution of this paper is to characterize the recursive internal structure of musical rhythms using a formal grammar. This goes beyond the GTTM, which does not propose a model of rhythm, and further argues that the inference of the hierarchical rhythmic deep structure is central to music cognition. Because of the joint representation of rhythmic and metrical structure in the model, a parser of the proposed abstract grammar of musical rhythm instantiates rhythmic interpretation and metrical inference at the same time.

In this formalism, the concept of *timestealing* is proposed. It is modeled at the highest metrical level it affects, and the split operation already sets up the timespans for subsequent preparation or shift operations in the upbeat or coda parts of the timespan category. This modeling ensures that all operations remain context-free and could be implemented and parsed efficiently with a parser of abstract

4: | i | | | | | | | |

3: | i | | | | i | | | |

2a: | i | | | | V^7 | | i | |

2b: | i | | iv | | V^7 | | i | |

1: | i | | iv | | $ii^\varnothing$ | V^7 | i | |

s: | Cm^7| | Fm^7 | | $Dm^\varnothing$ | G^7 | Cm^7 | |

(a) Schematic stepwise top-down generation of the chord sequence including harmonic rhythm. The generation occurs top-down from the single tonic i at metrical level 4. If a bar is empty, it is in the domain of the preceding chord. The row numbers indicate the metrical level. There are two operations at the same metrical level. The last row contains the chord sequence as it appears in the leadsheet.

$i\,[0:8:0]$
$m=5$

$i\,[0:4:-2]$
$m=4\oplus 1$

$i\,[2:4:0]$
$m=4$

$V^7\,[2:2:0]$
$m=3\oplus 1$

$i\,[0:2:0]$
$m=3$

$iv\,[0:2:0]$
$m=3$

$V^7\,[0:2:0]$
$m=3\oplus 1$

$ii^{7\flat5}\,[0:1:0]$
$m=2\oplus 2$

$V^7\,[0:1:0]$
$m=2$

$Cm^7\,(2)$
$m=5$

$Fm^7\,(2)$
$m=3$

$Dm^{7\flat5}\,(1)$
$m=4$

$G^7\,(1)$
$m=2$

$Cm^7\,(2)$
$m=3$

(b) Harmonic syntax tree co-generated with rhythmic-metrical structure. At the surface, the numbers in parenthesis encode duration and m its metrical weight (see also part (a) of the figure).

Figure 4: Syntactic analysis of the harmony and its rhythm in the first phrase of the Jazz standard "Blue Bossa" in C minor.

context-free grammars. This would not be guaranteed if the upbeat feature would only be instantiated at the upbeat or shift operation, for instance. If such an upbeat would reach across a border at a different metrical level the information to adapt the coda part of the adjacent subtree would have to perculate through the tree in a context-sensitive fashion, thus resulting in a model of much higher computational complexity. Such an instance could, for example, be observed in the syncopation of the sixth note in Figure 3. The eighth note syncopation of a note at the whole-note level results in a shift of the corresponding right neighbor at the quarternote level, accordingly the information would have to traverse one node up and four nodes down the tree to reach the right node. Further, a generation of upbeats and shifts without timespan reservation may result in the generation of impossible structures if both sides expand in a unrestricted context-free fashion.

Because of the hierarchical modeling of shifts of timespans, it is not necessary to include "hacks" such as binding-over of events as in musical notation (as in the long notes C and F in Figure 3) since the logic of syncopation can be modeled directly. With the formalism and the upbeat feature it is further possible to model the rhythmic displacement of an entire group of events, such as the syncopation of four quarternotes by an eighth note.

The tripartite representation of a timespan with upbeat, core, and coda parts makes it possible to maintain the simple split ratios at the deep structure. It also models the normalized locations where syncopations originated, as well as the overarching timespan that a deeper event dominates even though it may only occur at a different surface position (such as the tonic or dominant symbols in Figure 4). Maintaining simpler deep structure relations aggregates of similar rules, which establishes theoretical parsimony and facilitates probabilistic modeling and inference.

In the presented model rhythmic structure is generated in alignment with meter. It is possible to devise a variant

of the model such that metrical structure is *co-generated* jointly with rhythm rather than having it defined with the start category. The additive $m = u \oplus v$ characterization as used in Figure 3 and 4 defines metrical weight with a current metrical level and a part inherited from the parent. Split, prepare and shift rules can be redefined in such a way that they recursively generate each successive metrical level. While such an approach has advantages for complex and irregular rhythms, it would require additional constraints to ensure metrical consistency across independent context-free subtrees.

The proposed formalism models the generation of a single rhythmic sequence. For musical structures with multiple streams or voices, additional parallel trees can be instantiated which need to fulfill the constraint that their derived metrical structures are aligned. Moreover, complex (non-Western) rhythms and meters can be modeled with an extended model of meter that allows for non-isochronous or additive subdivisions [5]. Application of a computational implementation of the model would be measures of rhythmic complexity based on the derivation tree as well as rhythmic similarity based on largest embeddable common subtrees as for instance employed in [43].

The fact that rhythmic relations instantiate upbeats and splits is closely related to the core syntactic principles of preparation and prolongation [54]. This is corroborated by the fact that harmonic syntax and rhythm are found to be highly correlated in computational modeling [39].

Finally, there is also a close relation between such recursive rhythm and grouping; in fact, the higher order rhythmic categories reflect or *constitute* the grouping structure, refining the concept from the GTTM [11]. Crucially, it has not been argued that this model holds for all rhythms found in musical practice. Rather, the formalism models rhythmic interpretability based on the deep structural rhythmic dependencies outlined; while music is highly flexible, certain complex rhythms (Figure 1) may not be interpretable.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] A. Holzapfel, F. Krebs, and A. Srinivasamurthy, "Tracking the 'odd': Meter inference in a culturally diverse music corpus," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. ISMIR, 2014, pp. 425–430.

[2] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio." in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 227–232.

[3] H. Merchant, J. Grahn, L. Trainor, M. Rohrmeier, and T. Fitch, "Finding the beat: a neural perspective across humans and non-human primates," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 370, no. 1664, p. 20140093, 2015.

[4] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[5] J. London, *Hearing in time*. Oxford: Oxford University Press, 2004.

[6] C. Lee, "The rhythmic interpretation of simple musical sequences: Towards a perceptual model," in *Musical structure and cognition*, P. Howell, I. Cross, and R. West, Eds. London: Academic Press., 1985, pp. 53–69.

[7] ——, "The perception of metrical structure: Experimental evidence and a model," in *Representing Musical Structure*, P. Howell, R. West, and I. Cross, Eds. London: Academic Press, 1991, pp. 59–127.

[8] D. Temperley and D. Sleator, "Modeling meter and harmony: A preference-rule approach," *Computer Music Journal*, vol. 23, no. 1, pp. 10–27, 1999.

[9] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 342–355, 2005.

[10] B. van der Weij, M. T. Pearce, and H. Honing, "A probabilistic model of meter perception: Simulating enculturation," *Frontiers in Psychology*, vol. 8, p. 824, 2017.

[11] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. Cambridge, MA: MIT Press, 1983.

[12] C. Hasty, *Meter as Rhythm*. Oxford: Oxford University Press, 1997.

[13] G. Sioros and C. Guedes, "Syncopation as transformation," in *Sound, Music, and Motion*, M. Aramaki, O. Derrien, R. Kronland-Martinet, and S. Ystad, Eds. Springer International Publishing, 2014, vol. Vol. 8905, pp. 635–658.

[14] G. Sioros, M. E. P. Davies, and C. Guedes, "A generative model for the characterization of musical rhythms," *Journal of New Music Research*, vol. 47, no. 2, pp. 114–128, 2018.

[15] F. Foscarin, F. Jacquemard, and P. Rigaux, "Modeling and learning rhythm structure," in *Proceedings of the Sound and Music Computing Conference (SMC)*, 2019.

[16] D. Temperley, "Syncopation in rock: a perceptual perspective," *Popular Music*, vol. 18, no. 1, pp. 19–40, 1999.

[17] E. Clarke, "Levels of structure in the organization of musical time," *Contemporary Music Review*, vol. 2, no. 1, pp. 211–238, 1987.

[18] ——, "Rhythm and timing in music," in *The Psychology of Music*, 2nd ed., D. Deutsch, Ed. San Diego: Academic Press, 1999, pp. 473–500.

[19] H. Honing, "Structure and interpretation of rhythm and timing," *Tijdschrift voor Muziektheorie*, vol. 7, no. 3, pp. 227–232, 2002.

[20] ——, "Structure and interpretation of rhythm in music," in *The Psychology of Music*, 3rd ed., D. Deutsch, Ed. Academic Press, 2013, pp. 369–404.

[21] D. Temperley, *The Cognition of Basic Musical Structures*. Cambridge, MA: MIT Press, 2001.

[22] M. Mauch, "A Corpus-based study of rhythm patterns," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 163–168.

[23] A. Holzapfel, "Relation between surface rhythm and rhythmic modes in turkish makam music," *Journal of New Music Research*, vol. 44, no. 1, pp. 25–38, 2015.

[24] H. V. Koops, A. Volk, and W. B. De Haas, "Corpus-based rhythmic pattern analysis of ragtime syncopation," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. ISMIR press, 2015, pp. 483–489.

[25] K. Salley and D. T. Shanahan, "Phrase rhythm in standard jazz repertoire: A taxonomy and corpus study," *Journal of Jazz Studies*, vol. 11, no. 1, pp. 1–39, 2016.

[26] J. London, R. Polak, and N. Jacoby, "Rhythm histograms and musical meter: A corpus study of malian percussion music," *Psychonomic bulletin & review*, vol. 24, no. 2, pp. 474–480, 2017.

[27] D. Huron, *Sweet Anticipation: Music and the Psychology of Expectation*. Cambridge, Massachusetts: MIT Press, 2006.

[28] E. Amiot, *Music through Fourier space*. Berlin: Springer, 2016.

[29] G. T. Toussaint, "A mathematical analysis of african, brazilian, and cuban clave rhythms," in *Proceedings of BRIDGES: Mathematical Connections in Art, Music and Science*, 2002, pp. 157–168.

[30] E. D. Demaine, F. Gomez-Martin, H. Meijer, D. Rappaport, P. Taslakian, G. T. Toussaint, T. Winograd, and D. R. Wood, "The distance geometry of music," *Computational Geometry: Theory and Applications*, vol. 42, no. 5, pp. 429–454, 2009.

[31] M. Hamanaka, K. Hirata, and S. Tojo, "Automatic generation of grouping structure based on the GTTM," in *Proceedings of the International Computer Music Conference (ICMC)*, 2004, pp. 141–144.

[32] ——, "Implementing "A Generative Theory of Tonal Music"," *Journal of New Music Research*, vol. 35, no. 4, pp. 249–277, 2006.

[33] M. Pearce, "The construction and evaluation of statistical models of melodic structure in music perception and composition," Ph.D. dissertation, City University, London, 2005.

[34] C. Raphael, "Automated rhythm transcription," in *Proceedings of the 2nd International Society for Music Information Retrieval Conference (ISMIR)*, 2001, pp. 99–107.

[35] F. Foscarin, F. Jacquemard, P. Rigaux, and M. Sakai, "A parse-based framework for coupled rhythm quantization and score structuring," in *Mathematics and Computation in Music*, 2019.

[36] A. McLeod and M. Steedman, "Meter detection in symbolic music using a lexicalized pcfg," in *Proceedings of the 14th Sound and Music Computing Conference (SMC)*, 2017, pp. 373–379.

[37] ——, "Meter detection and alignment of midi performance," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 113–119.

[38] J. Yust, *Organized Time: Rhythm, Tonality, and Form*. Oxford: Oxford University Press, 2018.

[39] D. Harasim, T. J. O'Donnell, and M. A. Rohrmeier, "Harmonic syntax in time: Rhythm improves grammatical models of harmony," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*. ISMIR, 2019, pp. 335–342.

[40] M. Steedman, "The Blues and the abstract truth: Music and mental models," in *Mental Models in Cognitive Science: Essays in Honour of Phil Johnson-Laird*, J. Oakhill and A. Garnham, Eds. Psychology Press, 1996, ch. 15, pp. 305–318.

[41] S. Tojo, Y. Oka, and M. Nishida, "Analysis of chord progression by HPSG," in *Proceedings of the 24th IASTED International Conference on Artificial Intelligence and Applications*, Innsbruck, Austria, 2006, pp. 305–310.

[42] M. Rohrmeier, "Towards a generative syntax of tonal harmony," *Journal of Mathematics and Music*, vol. 5, no. 1, pp. 35–53, 2011.

[43] B. De Haas, M. Rohrmeier, R. Veltkamp, and F. Wiering, "Modeling Harmonic Similarity Using a Generative Grammar of Tonal Harmony," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, G. et al., Ed., 2009.

[44] W. B. De Haas, "Music information retrieval based on tonal harmony," Ph.D. dissertation, Utrecht University, 2012.

[45] M. Granroth-Wilding and M. Steedman, "Statistical Parsing for Harmonic Analysis of Jazz Chord Sequences," *Proceedings of the International Computer Music Conference (ICMC)*, vol. 2012, pp. 478–485, 2012.

[46] M. Granroth-wilding and M. Steedman, "A Robust Parser-Interpreter for Jazz Chord Sequences A Robust Parser-Interpreter for Jazz Chord Sequences," *Journal of New Music Research*, vol. 43, no. 4, pp. 355–374, 2014.

[47] D. Harasim, M. A. Rohrmeier, and T. J. O'Donnell, "A generalized parsing framework for generative models of harmonic syntax," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 152–159.

[48] D. Quick and P. Hudak, "Grammar-based automated music composition in Haskell," *Proceedings of the first ACM SIGPLAN workshop on Functional Art, Music, Modeling & Design - FARM '13*, pp. 59–70, 2013.

[49] N. Chomsky and M. Halle, *The sound pattern of English*. New York: Harper & Row, 1968.

[50] M. Liberman and A. Prince, "On stress and linguistic rhythm," *Linguistic inquiry*, vol. 8, no. 2, pp. 249–336, 1977.

[51] B. Hayes, *Metrical stress theory: Principles and case studies*. University of Chicago Press, 1995.

[52] J. Katz, "Hip-hop rhymes mirror phonological typology," *Lingua*, vol. 160, pp. 54–73, 2015.

[53] U. Reich and M. Rohrmeier, "Batidas latinas – rhythm and meter in spanish and portuguese and other forms of music," in *Phonological Typology of Syllable and Word Languages in Theory and Practice*, ser. (linguae & licerae), J. C. Reina and R. Szczepaniak, Eds. Berlin / New York: Walter de Gruyter, 2014, pp. 391–420.

[54] M. Rohrmeier, "The syntax of jazz harmony: Diatonic tonality, phrase structure, and form," *Music Theory and Analysis (MTA)*, vol. 7, no. 1, pp. 1–63, 2020.

[55] M. Rohrmeier and M. Neuwirth, "Towards a syntax of the classical cadence," in *What is a Cadence*, M. Neuwirth and P. Bergé, Eds. Leuven University Press, 2015, pp. 287–338.

Paper Session 5

PRACTICAL EVALUATION OF REPEATED RECOMMENDATIONS IN PERSONALIZED MUSIC DISCOVERY

Brian Manolovitz
University of Miami
bmm157@miami.edu

Mitsunori Ogihara
University of Miami
m.ogihara@miami.edu

ABSTRACT

Studies have shown that repeated exposures to novel songs cause an increase in a person's memory and liking. These studies are commonly verified through self-reporting emotion-based surveys. This paper proposes the "retention rate" as an additional parameter for evaluation, which examines the rate at which the listener revisits the novel items. The authors hypothesize that when a person listens to novel (i.e., both unfamiliar and interesting) pieces of music, the retention rate will be proportional to the number of times the discovery engine suggests the pieces to her, as long as they remain novel. The authors have tested the hypothesis through a six-week human-subject experiment which simulates a real-world listening environment and a follow-up survey. During the experiment period, each subject received, through Discover Weekly in Spotify, suggestions for novel songs up to three times and provided evaluation. One month after the evaluation experiment, the human-subjects answered whether they had revisited the novel songs. Through the analysis of the response and survey data, the researchers conclude that the more times a listener is exposed to a song during the discovery process, the more likely she is to return to the song.

1. INTRODUCTION

The arrival of online music streaming services, such as Spotify[1] and Apple Music[2], has greatly changed the way people listen to music. They allow their users to make dynamic selections of music from vast libraries and thus provide an improved exposure outlet for musicians. By adopting a music streaming service, listeners are far more likely to broaden their tastes and explore songs and artists appearing in the long tail of the popularity distribution [3]. The instant accessibility to a myriad of songs through streaming platforms addresses the long-tail problem [13], where most listening data corresponds to few songs and the vast majority of songs have very little listening data, especially through use of collaborative filtering (CF). CF is a technique that finds a group of users whose tastes and activities show substantial similarity to each other and makes recommendations based upon what the other members of the group liked. The more people in the group who enjoy a piece, the more likely that the system recommends it, which gives rise to a recommendation bias towards popular songs, though other techniques can help correct this issue.

Personalized music discovery tools, such as Spotify's Discover Weekly playlist, aim at recommending music independent of user groups. They utilize the personal listening history of a user and try to suggest new and interesting songs specific to her interests. These "serendipitous recommendations" [9] are useful for extracting music from the long tail, which would otherwise be difficult for the user to find. They make use of content-based filtering techniques, which determine song similarity through the audio features and are a potential solution to the popularity contest that tends to be created by collaborative filtering methods.

However, even when presented with novel songs, the onus is on the user to remember to revisit them, usually by saving them or adding them to a playlist. The goal of the user during the experience of a new piece may not be to record what she liked for future relistening, but instead, she may want to listen to something in the background [4]. As a result, some songs may disappear not only from the memory of the user but also from her song collection, despite that the listener enjoyed them on first listen. These songs in oblivion create missed opportunities to expand both the listening repertoire of the user and the audience of the artist. The goal of this study is to show that when repeating recommendations during the music discovery process, the listeners are significantly more likely to revisit the discovered songs. This addresses the long-tail problem by focusing on data saturation, rather than item selection, in order to help items break out of the long tail.

Missed opportunities also arise when a user has a neutral or uncertain initial response to a song and discards it, since subsequent listens may have yielded a more favorable response. The music domain has the somewhat unique characteristic that users are expected to revisit songs many times. Studies have shown that repeated listens to a piece of music cause an initial increase in liking, which subsides with satiation. Simultaneously,

[1] www.spotify.com
[2] www.apple.com/apple-music

memory steadily increases throughout [7,11,14,15]. This is not an issue in the typical radio format for music discovery, where songs are presented many times upon their release, ensuring listeners will become familiar. Though streaming services have adopted the radio format to an extent [5], this study should provide justification for expanding its use in the music discovery tools.

In the present paper, the authors studied whether repeated recommendations of novel songs cause users to return to the songs later. A six-week human-subject experiment with a follow-up survey was conducted, which simulated a real-world listening environment. During the experiment period, the subjects were provided, through Discover Weekly in Spotify, suggestions for novel songs. Each novel song appeared no more than three times during the experiment. Each subject provided her response to each song on the list. One month after the conclusion of the six-week experiment, the human-subjects answered whether they had revisited the novel songs. Although it is typical to evaluate the effects of repeated recommendations with respect to a person's familiarity and preference to the songs, this study will introduce two additional factors, retention rate and forgetting rate, which allow the standard listening behaviors of the subjects to dictate the results. This paper will describe the details of the experiment and present the analysis of the data that was collected.

2. RELATED WORKS

When examining research on novelty and the effects of repeated recommendations, it is essential to trace everything back to [1], where Berlyne coined the inverted-U theory for collative variables. This theory postulates that as a collative variable (i.e., familiarity, complexity) increases, a person's liking increases to a point, then decreases, creating an inverted-U shape. Chmiel and Schubert [2] examined the validity of this theory throughout the past several decades of research. They found that, in general, the theory holds in the results of the studies they surveyed. It is, therefore, safe to assume that the first time a person listens to a song will not be their most enjoyed listening experience. Vargas and Castells [16] point out that flaws exist in evaluating novel recommendations solely based on the accuracy of the selected songs. They offer an alternative strategy that accounts for the ranking of the chosen items and their relevance to the user.

The problem of recommending long-tail items is a popular topic in the current recommendation systems research. Park [12] proposed an adaptive clustering method that clusters items based on their popularity but chose only the data objects in the long tail for clustering. This method performed better than prior approaches, both in terms of performance and the system's ability to recommend long-tail items. Wang et. al [17] utilized the users' experience level to control the extent to which long-tail items are recommended, finding that more experienced users were more open to the items in the long tail. [6] adapts the diversification of recommendation lists based on the perceived preferences of the user towards diversity and penalizes the inclusion of popular items while increasing accuracy. Using a multi-objective simulated

annealing process, the resulting recommendation lists performed very well, compared to existing methods. Finally, [10] extended an existing tripartite graph approach for long-tail recommendations by expanding full genres, allowing more connections between items and genres. The results showed an increase in diversity and recall scores over existing methods.
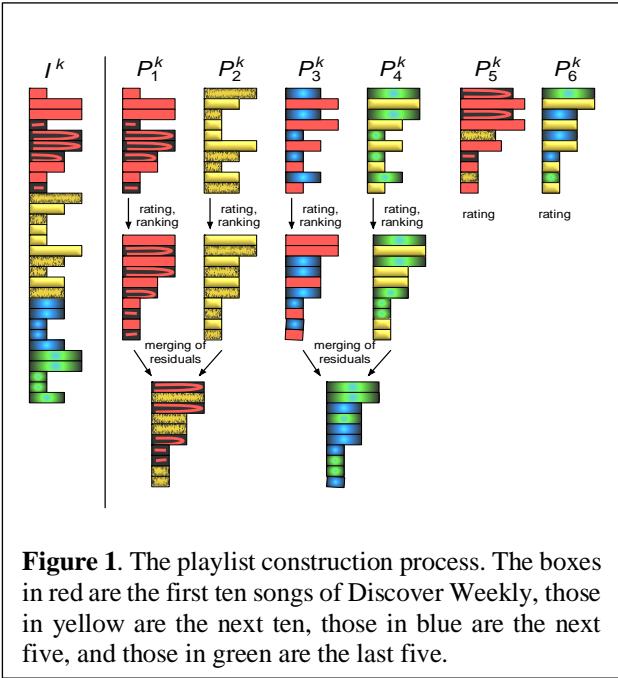
Several studies exist to test the effects of repeated listens on liking and familiarity [7,11,14,15]. All have a result stating that both factors typically increase after the first listen. Hargreaves [7] found that when novel songs were repeated in weekly intervals, rather than in one continuous setting, familiarity ratings plateaued as in the inverted-U shape, but the "like" ratings remained constant in both cases. In [14], Szpunar et al. tested the effects of repeated listens during focused versus unfocused listening on memory and liking. They found that inattentive (or passive) listeners exhibited a slow and steady increase in liking and memory with repeated listens, whereas attentive listeners showed an inverted-U shape for both.

Similarly, in [11], Madison and Schiolde utilized a series of user listening experiments to conclude that familiarity increases liking regardless of the complexity of the music, and that familiarity is the most critical indicator of enjoyment. Van den Bosch et. al [15] conducted experiments involving psychophysiological scans that measured electrodermal activity. They connected self-reports on liking of music with these emotional responses. They found that as the unfamiliar songs were repeated, the emotional measurements became more closely related to the self-reported liking ratings.

Ward et al. [18] tested the effects of familiarity on music choice by asking subjects to rate songs in terms of familiarity and preference. After the initial rating, the songs were paired the songs up, one familiar and one unfamiliar, and asked the subjects which they would rather hear. They found that the subjects tended to select the songs with which they were more familiar, regardless of their liking ratings, leading them to conclude that people prefer to hear familiar music despite claiming they would like to hear more novel songs. Alternatively, their results indicate a greater need to boost familiarity during the discovery process, so that novel songs are more likely to be revisited.

3. PROBLEM OVERVIEW

Automatic personalized music discovery systems can make serendipitous recommendations to their users based on their listening histories. One example of these systems is Spotify's Discover Weekly, which provides users with a 30-song playlist of new and interesting songs based on their listening histories. Since the playlist is refreshed every Monday, the users will need to remember to revisit the songs they enjoy, usually by saving them or adding them to a playlist. If the users forget to do this, or if they prefer to listen to new music in the background [4], there will be missed opportunities to broaden the listener's repertoire, since songs the users enjoyed or may have

Figure 1. The playlist construction process. The boxes in red are the first ten songs of Discover Weekly, those in yellow are the next ten, those in blue are the next five, and those in green are the last five.

grown to enjoy never entered their listening rotation. We are proposing an improvement to the personalized music discovery approach, where the novel recommendations are repeated at least once to boost familiarity and increase the likelihood of the users revisiting the songs in the future. The first and main question of this research is as follows:

RQ1. Is the likelihood for a user to revisit a liked song proportional to the number of times that the song is presented to the user?

To answer this question, we introduce *retention rate*, R, as an evaluation metric. Let N be the set of songs a person listens to for the first time, and let r be the number of songs in N that person listens to more than once, $R = r / |N|$.

As shown in previous research [7,11,14,15], repeated listens of a novel song will typically lead to an initial increase in preference, followed by a decrease once the listener is satiated. The phenomenon is in line with the inverted-U theory for familiarity and preference [1,2]. Although the number of listens will need to be sufficiently large to see this pattern, we should still expect an increase in preference over the initial listening experiences. Therefore, the second research question is as follows:

RQ2. Does the preference toward the unfamiliar songs increase with repeated presentations?

Properly answering these questions will require a real-world listening simulation, where subjects are as free as possible to listen to the music and report on their behaviors. Since this approach utilizes actual listening activities for evaluation, it will yield more practical results than the usual feelings-based self-reports.

4. EXPERIMENT SETUP

To answer the two research questions, 19 Spotify users (15 female) were recruited to participate in a 6-week music listening study. This study was designed to simulate real-

world listening behavior, allowing the subjects to listen freely and report on their activity. To begin, the subjects were each asked to provide their most recent 30-song Discover Weekly playlist (570 total songs) without first listening to it. Let I^k represent this playlist for user k, such that I_n^k represents the nth song from the initial discovery playlist for user k. Every week, on Monday, subject k is provided with a link to a 10-song playlist, P_w^k, where w is the current week number. They were asked to listen to the playlist once, then fill out a survey before any further listens. No restrictions or requirements were placed on the setting or device of the listening. The only suggestion given to the subjects was to treat the listening as they would their normal music discovery. The survey asked the subjects to evaluate the entire playlist in terms of both their enjoyment and its effectiveness for discovery, both on a 5-point Likert scale. Additionally, they were asked to place each of the individual songs into one of three categories: "Like it," "Not sure yet/Neutral," or "Don't like it."

4.1 Playlist Construction

Let us describe in detail the playlist construction process (see Figure 1). Since the duration of the experiment is six weeks and we make weekly playlists, there are six playlists for subject k: P_1^k, ..., P_6^k. The six playlists contain ten songs each. Recall that the subject k disclosed the 30 songs in her Discover Weekly list, I^k, without listening to any of the songs in it. appearing in the Discovery Weekly. Since the selection in the Discover Weekly playlists ensures that the user has never listened to the song before using Spotify, we generated the playlists under the assumption that each subject had not heard any of the 30 songs in her list before. We constructed the six ten-song lists dynamically as follows:

- P_1^k and P_2^k contained songs $[I_1^k : I_{10}^k]$ and $[I_{11}^k : I_{20}^k]$, respectively, in the same order they appeared in I.
- P_3^k and P_4^k contained songs $[I_{21}^k : I_{25}^k]$ and $[I_{26}^k : I_{30}^k]$, respectively, as well as five songs repeated from P_1^k and P_2^k, respectively. It is necessary to control for the subjects' song ratings when deciding which songs to repeat. Practical implementations would simply repeat the songs the users seemed to enjoy most (saved, liked, did not skip, etc.), but the retention rate must be independent of preference in this experiment; otherwise it would be impossible to claim that it is correlated with repetition. We chose the five repeats in P_3^k and those in P_4^k to preserve the proportion of the initial ratings (Like = Not sure/Neutral = Dislike). These ratings are unknown ahead of time due to the nature of the experiment, and so we used the subject's ratings from weeks 1 and 2 for balancing the ratings. We first sorted the ten songs in P_1^k and the ten in P_2^k in the decreasing order of rating (in the case of ties, the order from I^k was preserved). We then assigned the songs at odd numbered positions in the ranking (i.e., 1, 3, 5, 7, and 9) from the first list to P_3^k. Similarly, we assigned the five songs from the second list at odd numbered positions to P_4^k. After determining the ten-song sets from which to build P_3^k and P_4^k, we fixed the order in

which the ten songs appeared. We increased the perception of diversity by placing the repeated songs and new songs alternatingly in these playlists, following the idea from [8].

- In P_5^k and P_6^k, we kept the repeats from P_1^k and P_2^k, that we used in P_3^k and P_4^k, respectively. We selected the remaining songs as follows:
  - For P_5^k, the five remaining songs are those from P_1^k and P_2^k that we did not use in P_3^k and P_4^k (i.e., those with even numbered ranks after sorting in the decreasing order of rating). We merged the two ranked lists of five remaining songs and selected five at those at odd numbered positions in the merged list.
  - P_6^k is constructed in the same manner, but the source of the five repeats are the five non-repeats each from P_3^k and P_4^k. Again, we sorted the ten songs in the decreasing order of rating and then selected those at odd numbered positions.
- In the end, 10 songs were presented 3 times, 10 were presented twice, and the remaining 10 were only presented once and were used as a baseline.

4.2 Surveys and Evaluation Process

After the 6-week music listening portion of the experiment, the subjects were asked to complete a survey containing general questions about their music listening habits, as well as a final evaluation of the songs in I, which was simply a 5-point Likert scale rating of their likelihood to revisit the songs in the future. They were asked to fill out this portion without listening to the songs again, relying solely on their memory of the song based on name and artist. If the subject could not remember the song, they were asked to respond with an asterisk instead of a rating. Since the weekly surveys asked the subjects to place the song name and artist into one of the three fields, they were required to think critically about the song, and therefore, should be expected to remember this information. The analysis will refer to the *forgetting rate*, which refers to the percentage of forgotten songs with respect to some characteristic, such as initial rating or the number of presentations of the song. Finally, if the subjects had heard the song prior to the experiment, they were asked to leave the field blank, and that song would be omitted from the results.
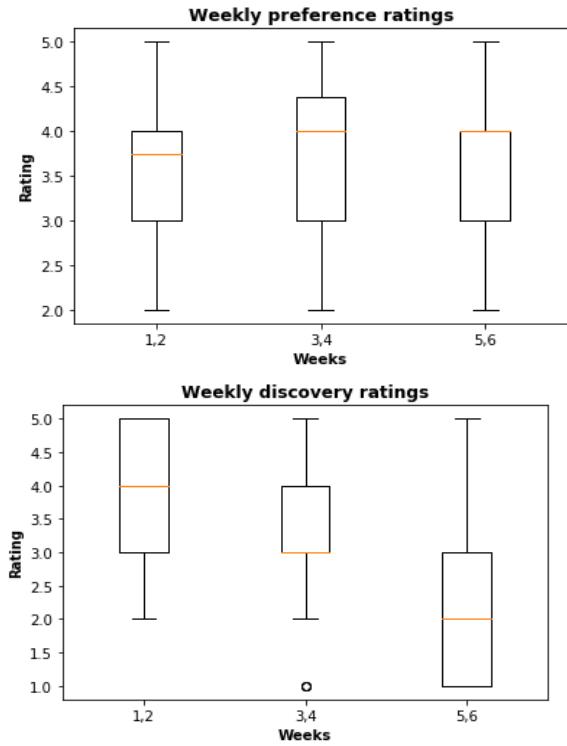
One month after the end of the experiment, the subjects received a follow-up survey asking them to indicate whether they had chosen to listen to each song in I during the month since the experiment. We evaluate the *retention rate*, R, with respect to the number of times the songs were presented, R^x as well as with respect to the subjects' ratings of the songs.

5. RESULTS AND DISCUSSION

The study utilized Spotify's Discovery Weekly playlist to provide serendipitous recommendations to the subjects based on their listening histories on the system, which meant some of the songs were not entirely new to the subjects. This is because Spotify is not the only music listening platform the subjects use, and we rectify it by omitting previously heard songs from the results analysis (26 out of 570 – less than 0.5%). It is also worth noting that the limited number of subjects restricts the generalizability of these results, but this being a multi-week study made finding subjects difficult.
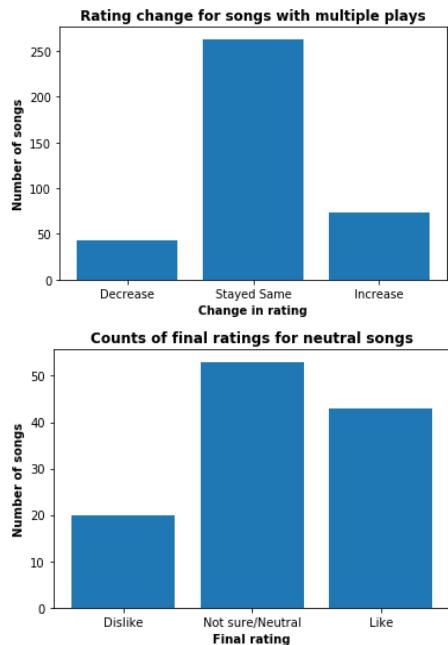
The weekly preference and discovery ratings were grouped by their composition as mentioned in Section 4, and their results are shown in Figure 2. The preference ratings were basically constant across each group, with means as follows: weeks 1-2 = 3.57, weeks 3-4 = 3.75, and weeks 5-6 = 3.67. The discovery ratings unsurprisingly declined in weeks where fewer new songs were presented, with means as follows: weeks 1-2 = 3.88, weeks 3-4 = 3.12, and weeks 5-6 = 2.41. It is worth noting, however, that an even split between new and repeated songs is likely not ideal in a real-world setting, but determining the optimal split was outside of the scope of this research.

Figure 2. Boxplots of the weekly preference (top) and discovery (bottom) ratings for all subjects, grouped based on the composition of the playlists (unfamiliar vs. familiar).

Figure 3 shows the number of songs whose rating changed, either positively or negatively, after the first listen, as well as the counts of the final ratings for "neutral" songs with more than one presentation. Though it was most common for the rating to remain the same, when it changed, it increased almost twice as often as it decreased. Of all songs with more than one play, 43 of them showed a decrease in rating and 73 showed an increase, though 263 retained their initial rating. Isolating the 116 songs with an initial rating of "neutral," we see that 43 of them (37%) had a final rating of "like," versus 20 (17%) whose rating decreased to "dislike" and 53 (46%) that remained at "neutral." Therefore, there is a potential for missed
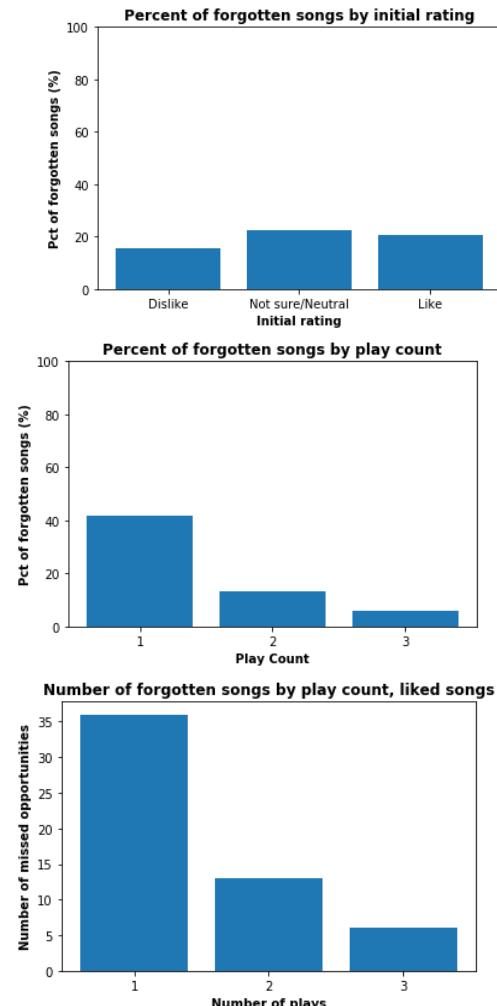
opportunities with a "neutral" song, since the initial listen may not entice the listener to revisit the song, whereas subsequent listens are likely to yield a more favorable response. We can answer RQ2 by saying that it is more likely for a person's preference toward an unfamiliar song will increase with repeated listens, but most of the time their feelings will remain constant. A rating scale with finer granularity would have answered this question more effectively.

Figure 3. The rating changes of all songs (top) and neutral songs (bottom) with more than one presentation during the study.

Figure 4 illustrates the forgetting rate with respect to play count and initial rating. It also takes an isolated look at the songs with an initial "like" rating. In total, 105 of the 544 songs (19.3%) were forgotten. Two one-tailed paired samples z-tests were performed, comparing the forgetting rate between songs with 1 and 2 plays, as well as 2 and 3 plays, $z(343) = 5.88$, $p < 0.00001$ and $z(342) = 2.38$, $p < 0.01$, respectively. The forgetting rate decreases significantly as the number of plays increases, which should be reciprocated with a higher retention rate, though any subjects who saved the songs for later would be less reliant on their memory. In terms of liking, two additional one-tailed paired samples z-tests were conducted to compare the difference between the forgetting rates for songs with an initial rating of "dislike" versus "neutral," $z(250) = 1.38$, $p < 0.1$ and "neutral" versus "like," $z(423) = 0.44$, $p < 0.33$. In both cases, we found no significant differences on the forgetting rates, which indicates a lack of connection between preference and memory. Looking at the forgetting rate of the liked songs, we can see another potential for missed opportunities, as the songs with only one presentation are significantly more likely to be forgotten, $z(182) = 4.21$, $p < 0.00002$ for 1 versus 2 plays and $z(177) = 5.94$, $p < 0.00001$ for 2 versus 3 plays. These missed opportunities occur when a person enjoys a song
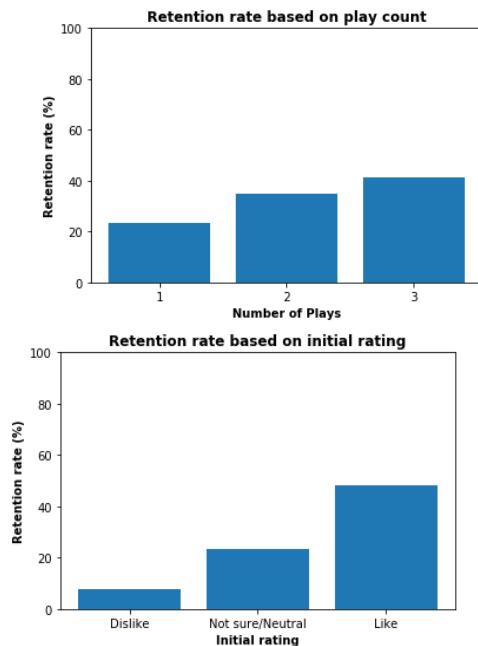
but does not save it and forgets its name or the artist name and is unable to search for it later.
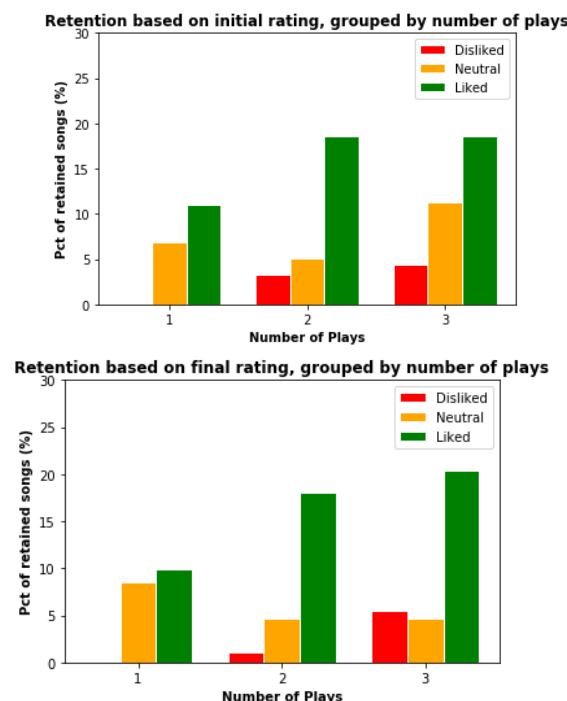
Figure 4. The percentage of forgotten songs by initial rating (top) and play count (middle), as well as the number of forgotten "liked" songs by play count (bottom)

One month after the conclusion of the listening experiment, the subjects received a survey asking them to state whether they listened to each of the songs from the experiment on their own accord (i.e. in a playlist they created or by searching for the song). Of the 19 subjects, 18 of them provided responses, and only one of those did not revisit any songs. Figure 5 shows the percentage of songs which were revisited with respect to both their play counts as well as the initial ratings. Figure 6 groups the songs by play count and shows the percentage of retained songs with respect to both initial and final rating.

A series of paired samples one-tailed z-tests were performed to answer RQ1. First, the songs were grouped by the number of times they were presented, requiring a test comparing R^1 and R^2 and another comparing R^2 and R^3, $z(343) = 2.41$, $p < 0.01$ and $z(342) = 1.22$, $p < 0.2$, respectively. Clearly, there is no significant difference between the retention rate of songs with 2 and 3 plays.

Figure 5. The retention rate with respect to play count (top), initial rating (bottom).

Figure 6. The number of retained songs, grouped by number of plays, with respect to initial (top) and final (bottom) ratings.

However, if we assume that all additional plays have the same effect on retention, we can combine the songs with 2 and 3 plays. Comparing the retention rate of these songs with R^1, we get $z(514) = 3.42$, $p < 0.0005$. Therefore, we can answer RQ1 affirmatively on the premise that additional presentations of novel songs beyond the first are essentially the same. It is worth noting that this experiment did not test more than 3 presentations of a novel song, and it is likely there are diminishing returns beyond the third presentation. We conducted additional z-tests by grouping

the songs by their initial rating, then comparing the songs with 1 play and the songs with more than 1 play, as follows: for songs rated "dislike," $z(91) = 1.98$, $p < 0.025$; rating = "neutral," $z(159) = 0.62$, $p < 0.3$; rating = "like," $z(264) = 3.37$, $p < 0.0004$. The performance of the "neutral" songs is interesting and could be related to the change in rating for these songs with more than one play. In general, only the "liked" songs had a reasonably high retention rate, so it is possible that a neutral or worse feeling towards a song is insufficient to persuade a person to revisit it. As previously seen, the "neutral" songs increased in rating at a 37% rate, and when looking at the graph in Figure 6 which shows final ratings, the "liked" songs still show an increase with play count, but the remaining neutral songs do not.

We performed similar z-tests to evaluate the relationship between the initial rating of the songs and their retention, one for "dislike" vs "neutral/not sure" and another for "neutral/not sure" vs "like," $z(250) = 3.11$, $p < 0.001$ and $z(423) = 5.08$, $p < 0.00001$, respectively. Clearly, initial rating is a strong predictor of the retention of a song, though this should be obvious and does not deter from the results with respect to play count. In practice, a personalized discovery system can infer liking via user interaction (i.e. button clicks, skips, page visits, etc.), then use that to select which songs to repeat.

6. CONCLUSION AND FUTURE WORK

Previous research has concluded that repeated listens of novel music will increase both memory and liking, but the evaluation has typically involved the subjects self-reporting on their feelings. This study implemented a real-world listening simulation and evaluated the effects of repeated listens of novel songs with respect to the rate at which the songs were revisited by the subjects. We found that when songs were played more than once, in general, their retention rate significantly increased, and the rate at which the songs could be recalled from name and artist alone also increased. Additionally, if the ratings of the songs changed after the first listen, it was significantly more likely to be an increase.

We explored the concept of missed opportunities when assuming a music discovery process which recommends songs to users once and expects them to remember to revisit the songs. By only presenting songs once, liked songs are less likely to be remembered or revisited, and songs users feel neutral or unsure about will not have a chance to improve their favorability. Repetition of novel recommendations clearly decreases the potential for missed opportunities in both cases, giving users a greater chance to broaden their musical tastes.

One aspect we did not evaluate was the proper split between new and repeated songs, which may be a user-specific parameter and likely varies from week to week. In our future work, we intend to explore whether there is a predictable pattern to the amount of new music a person consumes on a weekly basis. In addition, we are planning several studies involving electroencephalography, where we will test memory and attention when listening to new music over a sustained period.

7. REFERENCES

[1] D. E. Berlyne: *Aesthetics and psychobiology*, Appleton-Century-Crofts, New York, NY, 1971.

[2] A. Chmiel and E. Schubert: "Back to the inverted-U for music preference: A review of the literature," *Psychology of Music*, Vol. 45, No. 6, pp. 886-909, 2017.

[3] H. Datta, G. Knox, and B. J. Bronnenberg: "Changing their tune: How consumers' adoption of online streaming affects music consumption and discovery," *Marketing Science*, Vol. 37, No. 1, pp. 5-21, 2017.

[4] J. Garcia-Gathright, B. St Thomas, C. Hosey, Z. Nazari, and F. Diaz: "Understanding and evaluating user satisfaction with music discovery," *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 55-64, 2018.

[5] M. Glantz: "Internet radio adopts a human touch: a study of 12 streaming music services," *Journal of Radio & Audio Media*, Vol. 23, No. 1, pp. 36-49, 2016.

[6] E. M. Hamedani and M. Kaedi: "Recommending the long tail items through personalized diversification," *Knowledge-Based Systems*, Vol. 164, pp. 348-357, 2019.

[7] D. J. Hargreaves: "The Effects of Repetition on Liking for Music," *Journal of Research in Music Education*, Vol. 32, No. 1, pp. 35-47, 1984.

[8] M. Ö. Karakaya and T. Aytekin: "Effective methods for increasing aggregate diversity in recommender systems," *Knowledge and Information Systems*, Vol. 56, No. 2, pp. 355-372, 2018.

[9] D. Kotkov, S. Wang, and J. Veijalainen: "A survey of serendipity in recommender systems," *Knowledge-Based Systems*, Vol. 111, pp. 180-192, 2016.

[10] A. Luke, J. Johnson, and Y. K. Ng: "Recommending Long-Tail Items Using Extended Tripartite Graphs," *IEEE International Conference on Big Knowledge (ICBK)*, pp. 123-130, 2018.

[11] G. Madison, and G. Schiolde: "Repeated Listening Increases the Liking for Music Regardless of Its Complexity: Implications for the Appreciation and Aesthetics of Music," *Frontiers in Neuroscience*, Vol. 11, Article 147, 2017.

[12] Y. J. Park: "The adaptive clustering method for the long tail problem of recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, No. 8, pp. 1904-1915, 2012.

[13] Y. J. Park and A. Tuzhilin: "The long tail of recommender systems and how to leverage it," *Proceedings of the 2008 ACM Conference on Recommender Systems*, pp. 11-18, 2008.

[14] K. K. Szpunar, G. E. Schellenberg, and P. Pliner: "Liking and Memory for Musical Stimuli as a Function of Exposure," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, Vol. 30, No. 2, pp. 370-381, 2004.

[15] I. N. Van den Bosch, V. J. Salimpoor, and R. Zatorre: "Familiarity mediates the relationship between emotional arousal and pleasure during music listening," *Frontiers in Human Neuroscience*, Vol. 7, Article 534, 2013.

[16] S. Vargas and P. Castells: "Rank and relevance in novelty and diversity metrics for recommender systems" *Proceedings of the fifth ACM Conference on Recommender Systems*, pp. 109-116, 2011.

[17] Y. Wang, J. Wang, and L. Li: "Enhancing Long Tail Recommendation Based on User's Experience Evolution," *IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 25-30, 2018.

[18] M. Ward, K. Goodman, and J. Irwin: "The same old song: The power of familiarity in music choice," *Marketing Letters*, Vol. 25, No. 1, pp. 1-11, 2014.

AUTOMATIC FIGURED BASS ANNOTATION USING THE NEW BACH CHORALES FIGURED BASS DATASET

Yaolong Ju[1]   **Sylvain Margot**[1]   **Cory McKay**[2]   **Luke Dahn**[3]   **Ichiro Fujinaga**[1]

[1] Schulich School of Music, McGill University, Canada

[2] Department of Liberal and Creative Arts, Marianopolis College, Canada

[3] School of Music, The University of Utah, USA

yaolong.ju@mail.mcgill.ca, sylvain.margot@mail.mcgill.ca, cory.mckay@mail.mcgill.ca, luke.dahn@utah.edu, ichiro.fujinaga@mcgill.ca

ABSTRACT

This paper focuses on the computational study of *figured bass*, which remains an under-researched topic in MIR, likely due to a lack of machine-readable datasets. First, we introduce the Bach Chorales Figured Bass dataset (BCFB), a collection of 139 chorales composed by Johann Sebastian Bach that includes both the original music and figured bass annotations encoded in MusicXML, **kern, and MEI formats. We also present a comparative study on automatic figured bass annotation using both rule-based and machine learning approaches, which respectively achieved classification accuracies of 85.3% and 85.9% on BCFB. Finally, we discuss promising areas for MIR research involving figured bass, including automatic harmonic analysis.

1. INTRODUCTION

Figured bass is a type of music notation that uses numerals and other symbols to indicate intervals to be played above a bass note, and which can provide insight on underlying harmonies [1]. It was commonly used in Baroque music, and served as a guide for performance, especially for the instruments improvising the *basso continuo* accompaniment (e.g., harpsichord, organ, lute, etc.). Fig. 1 shows an example of figured bass, as well as how a harpsichordist might realize the figured bass as an improvised accompaniment. Such realizations are not typically explicitly included in scores, as the musical tradition of the time left them to be improvised based on the skills and taste of the continuo player. There are three aspects of figured bass annotations (FBAs) that should be highlighted:

(1) The Neue Bach Ausgabe edition [2] uses FBAs consisting of numbers with backslashes through them to indicate raised intervals (e.g., m. 3.3 and m. 3.4). Forward slashes indicate lowered intervals (e.g., 5̸).

(2) FBAs followed by continuation lines indicate that the harmony of the preceding figure is prolonged (e.g., m. 1.4, m. 4.2, and m. 4.4).

(3) Multiple FBAs over a stationary bass (e.g., 4–3 in m. 5) usually indicate a suspension being resolved.

Figured bass also serves pedagogical and theoretical purposes: not only does it provide contrapuntal information on how to conduct the resolution of dissonances, it also offers insights into the chords and harmonic rhythm intended by composers. Figured bass can therefore provide a preliminary description of harmonic structure, and serves as a promising basis for approaching harmonic analysis.

As a useful analytical tool for studying Baroque compositional and performance practices, figured bass has been an important topic in music pedagogy [3], music theory, and musicology [4]. The computational study of figured bass, however, has drawn little attention over the years. We have only found two papers on automatic figured bass annotation, both using a rule-based approach: Barthélemy and Bonardi treated figured bass as a harmonic reduction and devised rules to identify and remove ornamental notes, permitting them to cluster the remaining chord tones as figures [5]; Wead and Knopke, in contrast, manually designed a decision tree to determine the figured bass for a given bass line [6]. Unfortunately, with no open-source code and a lack of quantitative results, it is impossible to objectively evaluate or compare the performances of these models. Furthermore, we are not aware of any previous applications of machine learning to figured bass, nor of any existing digital dataset with figured bass annotations (FBAs). These limitations have likely limited the computational study of figured bass to date.

There are four main contributions of this paper:

(1) We introduce the new Bach Chorales Figured Bass (BCFB) dataset, which consists of 139 chorales composed by Johann Sebastian Bach (Section 2). These chorales came from larger choral works composed by Bach: the cantatas, passions, motets, and the Christmas Oratorio. We chose this repertoire due to its key role in modern music pedagogy and its general historical importance.

(2) In order to facilitate the future creation of more figured bass datasets, we include our methodology for digitizing FBAs in an efficient and effective way.

(3) We present a comparative study of automatic figured bass annotations of BCFB, using both rule-based and machine learning approaches (Section 3). The results are discussed with reference to specific musical examples (Section 4).

(4) We highlight possible applications of figured bass annotation, especially in connection with converting figured bass to chord labels, which could benefit research on automatic harmonic analysis (Section 5).

Figure 1. A sample musical passage we composed, where figured bass annotations (FBAs) are shown below the continuo line, and where we added the harpsichord line as an example of what a continuo player might improvise based on the figured bass. Figures indicate intervals above the continuo line that could be played in the improvisation. For example, the "6" in the first measure corresponds to the pitch class "G", which is a 6th above the bass "B♭". An actual improvisation would likely also typically contain the pitch class "D" (a 3rd above the bass "B♭") in this slice, but this is not explicitly indicated in the figures. This is an example of how FBAs do not always specify all the notes to be played by the continuo player, and usually omit some obvious figures (see Section 3.2.2 for details).

2. BACH CHORALES FIGURED BASS DATASET

To the best of our knowledge, there is no prior publicly available digital figured bass dataset. We therefore present the Bach Chorales Figured Bass dataset (BCFB), a corpus we constructed containing FBAs in MusicXML, **kern, and MEI (Music Encoding Initiative) formats. It consists of all 139 J. S. Bach four-voice chorales that include his own figured bass, based on the Neue Bach Ausgabe (NBA) critical edition [2]. NBA is chosen as the source of BCFB because it is the most up-to-date scholarly critical edition, prepared with exacting methods of source criticism.

2.1. Finding Chorales with FBAs

To find all the chorales attributed to Bach, we constructed a reference table (https://bit.ly/303jzfS) with all 420 chorales indexed by BWV catalogue numbers, and cross-referenced them with the NBA. We checked whether original FBAs are accessible for each of these chorales, and found 139 settings meeting this criterion. We then made an expanded reference table, consisting of the: BWV number,[1] Breitkopf number (when relevant),[2] title of the work of origin (e.g. cantata, passion, etc.), date of the first performance, text setting, location of the score in the NBA edition, and other musicological metadata for each cho-

rale. This table is designed to facilitate musicological research, which, along with the BCFB dataset, is available at: https://bit.ly/2OoWC16.

2.2. Digitization

We began the creation of BCFB by assembling existing symbolic encodings of the relevant Bach chorales from the KernScores repository, which contained 109 of the 139 NBA chorales with FBAs.[3] We automatically translated these 109 **kern files into MusicXML using *music21* (v. 5.1.0),[4] and made changes to match the musical content[5] of the NBA edition before adding Bach's figured bass. We manually encoded the remaining 30 figured chorales found in the NBA edition. We chose MusicXML as our master file format since it is widely supported by music notation software. We used the *MuseScore* 3 score editor[6] for both editing musical content and adding FBAs.[7]

2.3. Converting to Other Symbolic File Formats

BCFB includes encodings in two other symbolic file formats[8] beyond MusicXML: **kern, and MEI.[9] Existing software was used to automatically convert the original MusicXML to the other two formats. Through a series of experiments with a variety of alternatives, we found that converting figured bass from MusicXML to **kern using *musicxml2hum*[10] worked well, except for the continuation lines, and the conversion from **kern to MEI using *Verovio*[11] was perfect. However, direct conversion of figured

---

[1] Bach-Werke-Verzeichnis (BWV) catalogue number, which indexes all the compositions attributed to J. S. Bach.

[2] The Breitkopf edition contains 371 four-voice J. S. Bach chorales, and indexes them differently from BWV.

[3] KernScores (kern.ccarh.org) is maintained by Stanford's Center for Computer Assisted Research in the Humanities, and includes 371 four-part chorales encoded in the Humdrum **kern representation (www.humdrum.org).

[4] https://web.mit.edu/music21

[5] Including: adding a continuo line and/or instrumental voices; transposing; changing the meter, pitch, and duration of certain notes; etc. We did not encode the textual content specified in the NBA.

[6] https://musescore.org

[7] https://musescore.org/en/handbook/figured-bass

[8] This diversity of symbolic formats offers researchers the opportunity to use the format most convenient to their preferred software, because if only one format were offered, which might not be supported by a given piece of preferred research software, then it would need to be converted to the format supported by the software. This could lead to a potential loss of figured bass information or to other conversion errors [7].

[9] We also used *music21* (v. 5.1.0) to generate MIDI files from the master MusicXML, but they do not include FBAs.

[10] https://github.com/craigsapp/humlib

[11] https://github.com/rism-ch/verovio

Figure 2. Measures 3 and 4 from BWV 117.04 *Sei Lob und Ehr dem höchsten Gut*. The original FBAs are annotated underneath the bass voice part. Note that not all slices are necessarily figured and not all the intervals in a sonority are necessarily specified in FBAs. We artificially added the final bottom staff, which collapses all sonorities into one octave so as to more directly reveal the pitch-class content. The number of semitones above the bass implied by the original FBAs have also been added underneath this bottom staff. We can also translate the number of semitones back to FBAs by examining the actual notes in the score and then calculating and labelling the intervals from the bass note.

bass from MusicXML to MEI (using *Verovio*) was problematic, as accidentals, slashes, and continuation lines were not converted. We therefore first converted from MusicXML to **kern, and then manually added the continuation lines to the **kern files using a text editor. The resulting **kern files were then converted to MEI files.

3. AUTOMATIC FIGURED BASS ANNOTATION

We automatically generated[12] FBAs that closely resembles those of Bach for two main reasons: to learn about Bach's figured bass habits, which are of musicological interest, and to provide figured bass for those Bach chorales for which no FBAs exist. We used both rule-based and machine learning algorithms to perform this automatic figured bass annotation:[13] the rule-based approach has the potential to model Bach's style of writing figures in ways that are easily human-interpretable, and machine learning has the potential to model patterns in Bach's style that might be difficult to codify into precise, direct rules. We therefore explored the efficacy of both approaches.

3.1. Data

We used 120 chorales out of the full 139 chorales in BCFB to train and test our models. We excluded 12 interlude chorales[14] because they are significantly different from the

other largely homorhythmic chorales, and we excluded five other chorales[15] that are barely figured. Finally, we excluded BWV 8.06[16] and BWV 161.06 because they feature irregular textures, such as having an obbligato continuo and/or instrumental part.

3.2. Rule-base algorithms

3.2.1. Initial Simple Rule-based Algorithm

We began by implementing a simple rule-based algorithm that labels all the intervals above the bass in the generated FBAs. First, the music is segmented into a series of n*ote onset slices* [8,9]. A new slice is formed whenever a new note onset occurs in any musical voice, and each slice consists of the vertical set of notes sounding at that moment. Take the first slice of Fig. 2 as an example: since the pitch classes above the bass "G" are "G", "D", and "B," the FBA generated is 8/5/3.

We then compared the generated FBAs against Bach's original FBAs, and found that the percentage of exact matches was only 3%. This is partly because Bach did not explicitly label all the intervals above the bass in his FBAs; it is often assumed that both he and other Baroque composers employed FBAs that include what are in effect abbreviations that omit obvious intervals [1,10]. For example, consider m. 3.2.5[17] of Fig. 2: although the pitch classes "A" and "D" are present in this slice above the bass "F♯", only "D" is explicitly specified[18] by the figure "6"; "3" is not explicitly indicated, but is nonetheless implied.

3.2.2. Evaluation Metric

To allow for the equivalence in musical content of different figured bass notation conventions, as discussed above, we created an evaluation metric that treats figures that are musically equivalent as notationally equivalent. The purpose of this metric is to realistically evaluate the generated figured bass when it does not match Bach's figured bass exactly. The equivalence rules are inspired by Arnold [11]:

- A "3" can be omitted (Fig. 3a, 3b, 3d, and 3e) unless there is a 4th in the sonority,[19] or unless the 3rd is the resolution of a 4–3 suspension (Fig. 3c).
- A "5" (Fig. 3a, 3c, and 3d) can be omitted, unless one of the following conditions is true: there is a 6th (Fig. 3b) in the sonority, the 5th is the resolution of a 6–5 suspension, or the 5th has an accidental (Fig. 3e).
- An "8" (Figs. 3c and 3d) can be omitted, unless one of the following conditions is true: there is a 9th (Fig. 3b) in the sonority, the 8th is the resolution of a 9–8 suspension, or the 8th has an accidental.
- A "6" can be omitted if the sonority forms a "6/4/3" or a "6/4/2" chord, as shown in Fig. 3f.

---

[12] The generated FBAs use flats and sharps to respectively indicate lowered and raised intervals, and do not contain continuation lines.

[13] The code is available at: `https://bit.ly/2P8Qbju`.

[14] These chorales have elaborate instrumental interludes between phrases (BWV 24.06, 76.07, 100.06, 105.06, 113.01, 129.05, 167.05, 171.06, 248.09, 248.23, 248.42, and 248.64).
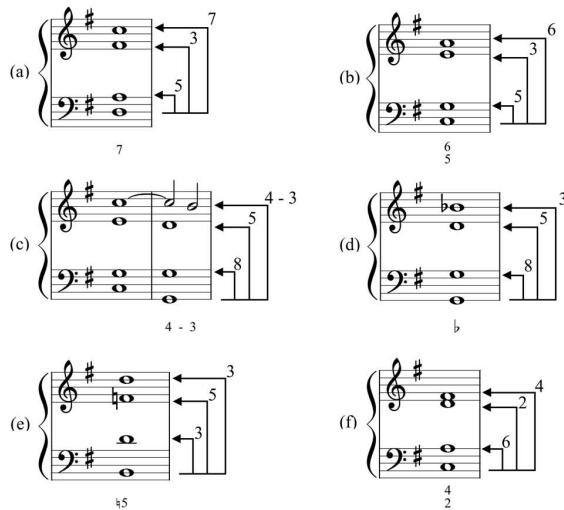
[15] BWV 16.06, 48.07, 149.07, 195.06, and 447.

[16] ".06" in "BWV 8.06" means this chorale is the sixth movement of the "BWV 8" cantata.

[17] "m. 3.2.5" means the third measure, the second and half beat.

[18] Since "D" forms a 6th interval above the bass "F♯".

[19] "Sonority" means the set of pitch classes present in a note onset slice. For example, the added bottom staff of Fig. 2 shows the sonorities of the four voices for each slice. "4th" means that a wrapped interval of a 4th can be found between the bass and an upper voice, regardless of whether it is labelled in the figured bass.

Figure 3. Common examples of standard figured bass abbreviations taken into account by the evaluation metric explained in Section 3.2.2. In each of the six examples (a)-(f), all the intervals above the bass are shown to the right of the notes connected with arrows, and typical abbreviated FBAs for the chords are shown below the notes. For example, (a) consists of a dominant 7th chord in root position, and includes notes that are a 3rd, 5th, and 7th above the bass: the figured bass consists of only a "7", with the "3" and "5" omitted, as is often the practice in FBA.

In order to see how the evaluation metric based on these rules operates in practice, consider Fig. 3a as an example: we can see that "5" and "3" can be omitted, which means "7", "7/3", "7/5", and "7/5/3" are all considered equivalent. Therefore, if the ground truth and the generated figured bass respectively consist of any pairing of "7", "7/3", "7/5", or "7/5/3", then the generated figured bass will be considered correct by the metric.

3.2.3. Improved Rule-based Algorithm

Using this evaluation metric, the simple ruled-based algorithm described in Section 3.2.1 has an effective agreement of 64.5% with Bach's FBAs. We found that when they disagreed, the generated FBAs tend to have more figures than those of Bach. To improve this, we manually developed additional rules for omitting certain figures, permitting us to better predict Bach's style of annotation.[20]

First, we examine each note onset slice and omit the figure for a given note in an upper voice if both of the following two conditions are met: (1) the note is labelled in the previous slice, and (2) the pitch class of the bass in the current slice remains the same as in the previous slice.

Then, we consider slices on fractional beats (e.g., beat 2.5 and 3.5), looking for ornamental notes, such as passing tones, neighbour tones, escape tones, and anticipations, which are all approached or departed by step. If such a note

is in an upper voice, its corresponding number is removed from the figure; if such a note is in the bass, the slice is left entirely unfigured.

After the addition of these rules, the model was able to achieve 85.3% agreement with Bach's figures, a large improvement over the 64.5% agreement achieved with the simple method and equivalency rules.

Although it would have been possible to invest more time manually analyzing Bach's figured bass to develop still more rules to improve agreement, none were readily obvious from the perspective of music theory or performance practice, and we wanted to avoid overfitting our rules. So, we turned our attention to machine learning, to see if it could be employed to model Bach's FBA style with equal or better results.

3.3. Machine learning algorithms

In order to efficiently perform automatic figured bass annotation with machine learning methods—something that has never been explored in the literature—we transformed the FBAs into interval-class vectors.

3.3.1. Transformation from Figured Bass to Interval Classes

Recall that figured bass indicates intervals above the bass note. Thus, for each slice, we convert the figures to an interval-class vector.[21] An interval class, similar to a pitch class, is a set of intervals wrapped by octaves. For example, an interval class of a major second includes a major ninth and all other octave expansions of a major second. As with a pitch-class vector, an interval-class vector contains 12 elements, representing intervals in semitone increments. In our case, each FBA is converted to an interval-class vector that includes all the notes above the bass that are sounding in the current slice. In cases where the figured bass does not specify the exact interval in semitones, such as the "6", which could be either a major 6th or a minor 6th, we rely on the score to determine the exact interval using heuristics-based post-processing. We similarly rely on the score to later convert interval-class vectors back to figures: for example, an interval of three semitones can be interpreted as either a minor third (figure "♭3") or an augmented second (figure "♯2"). We can decide which is appropriate by considering the pitch spelling in the original score.[22] This representation of the figured bass is used for both input and output of the machine learning algorithms.

3.3.2. Input Features

The three feature vectors used as input to the machine learning algorithms are: (1) interval classes (see Section 3.3.1); (2) onsets, which specify which notes above the bass have onsets within the slice, as opposed to being held from a previous slice; and (3) metrical context, which specifies whether a slice occurs on the downbeat of a measure, on another beat (e.g., beat 2, 3, or 4 in 4/4), or on

---

[20] The rules were proposed by observing the generated FBAs and were evaluated against the ground truth FBAs from BCFB. We selected the rules that yielded higher accuracy.

[21] We are using the term "interval class" here to refer to *ordered* interval class. Since we wish to calculate the intervallic relationship *from* the bass *to* and upper voice (in that order), we wish to distinguish between intervals and their inversional equivalents (e.g. minor 7ths are distinguished

from major 2nds). Thus, ordered interval classes range from 0 to 11, while unordered interval classes range from 0 to 6 only.

[22] For example, to distinguish "2" and "9" when unwrapping interval-class vectors, we need to find the actual note above the bass and compare its pitch to the pitch of the bass. If they are one octave apart, the generated figure will be "9", and "2" otherwise.

a fractional beat (e.g., beat 3.5). These binary vectors are specified for each slice. The following example demonstrates each of these feature vectors for m. 4.2.5 of Fig. 2 (the bit-length of each feature is indicated in parentheses):

- *Interval classes (12):* The bass note is "A" (held), with pitch classes of "C♯", "G", and "E" (held) above it, which are respectively four, ten, and seven semitones away from it. The feature vector is thus: [0,0,0,0,1,0,0,1,0,0,1,0].[23]
- *Onsets (12):* "C♯" and "G", which are respectively four and ten semitones above the bass, are the pitch classes with onsets on this slice, so the feature vector will be [0,0,0,0,1,0,0,0,0,0,1,0].
- *Metrical context (3):* Because the slice is on beat 2.5 of a 4/4 measure, the feature vector will be [0,0,1] (i.e., it is a fractional beat).

Each slice, therefore, is represented as a 27-dimensional (12+12+3=27) binary vector. To provide a context for each slice, the machine learning algorithms are also provided with the two 27-dimensional vectors for the previous and following slices (zero-padded for first and the last slices). Thus, the total length of the input vector for each slice is 81 (27×3=81).

3.3.3. Machine Learning Algorithms

We experimented with two machine learning algorithms: Decision Trees (DT) [24] and Deep Neural Networks (DNN).[25] Both algorithms used the input features specified above. Their output each consisted of a 12-dimensional binary vector specifying the number of semitones above the bass,[26] as discussed in Section 3.3.1.

3.3.4. Experimental Setup

Ten-fold cross-validation was used for evaluation. For the DNN experiments, we divided the data into training (80%), validation (10%),[27] and testing (10%) folds. For the DT experiments, the data was divided into training (90%, the union of the DNN training and validation sets) and testing (10%, matching the DNN test sets) partitions.

3.3.5. Results

The Decision Trees and Deep Neural Networks respectively achieved classification accuracies of 84.3±0.5% and 85.9±0.6% on BCFB.[28] These accuracies are calculated based on the evaluation metrics proposed in Section 3.2.2.

4. DISCUSSION

It is useful to examine the types of errors that our model made, in order to better understand its performance and how it can be improved. We will focus this discussion on the two musical examples shown in Fig. 4, as they are representative of the kinds of errors our model made. One

common error made by our model was to miss figures that indicate the resolution of a suspension, such as the 9–8 shown in Fig. 4(a), m. 8.4. This may be because the features we used did not contain sufficient voice-leading information to detect such suspensions.

Two further types of disagreement between our model and Bach's figures are shown in Fig. 4(b). At m. 2.3 our model generated "♯", but the ground truth had no label. In fact, the generated "♯" is technically correct, as the D is explicitly sharpened in the soprano. Turning to m. 3.2, our model's prediction included a "♯7," unlike the ground truth. Perhaps this suggests that Bach might have considered the corresponding "D♯" to be a passing tone? Or perhaps the D♯ was understood as a "diatonic" note in this Dorian chorale tune? At any rate, the "♯7" in the generated figures should not necessarily be considered wrong. Both these figures are in fact theoretically acceptable answers.

Such differences between the ground truth and the predicted figures are intriguing, as they hint at contrapuntally or harmonically meaningful information present in Bach's figures that is not explicit in the four vocal lines. Or, perhaps they are of negligible meaning? It is impossible to know with the information we have now, but future comparisons with models trained not just on Bach but on the figures of many Baroque composers could potentially reveal fascinating insights on compositional style.

We also observed interesting variability in the types of figures Bach used under seemingly similar musical contexts. Three examples are shown in Fig. 4:

- *Accidentals:* in Fig. 4(b), Bach did not label the first "♯" at m.2.3, but did label the second one at m. 3.3.
- *Suspensions:* Bach sometimes labelled suspensions (e.g., m. 8.4 of Fig. 4(a)), and sometimes omitted them (e.g., m. 1.4 of BWV 194.06 [not shown]).
- *The same chord:* Bach sometimes labelled a 6/4/2 chord as a 4/2 chord (e.g., m. 2.1 of Fig. 4(b)), and sometimes as a 6/4/2 chord (e.g., m. 10.4 of BWV 13.06 [not shown]).

So, one cannot expect 100% agreement to be achievable, given such variability. Bach, like everyone, was sometimes inconsistent with himself, which imposes an artificial performance ceiling on our models [12]. Also, these types of variabilities can be of great interest to music theorists and musicologists, and offer significant potential for future research.

---

[23] The first dimension indicates a unison (or collapsed octaves).

[24] We used the "DecisionTreeClassifier" function from the "scikit-learn" library, under default settings. This function is an optimized version of CART (Classification And Regression Tree).

[25] We used a feedforward network with three hidden layers, each with 300 hidden units. Adaptive Moment Estimation was used as an optimizer, with a binary cross-entropy-based loss function. These hyperparameters were tuned using the validation set.

[26] For example, the output vector for m. 4.2.5 of Fig. 2 will be [0,0,0,0,1,0,0,0,0,0,0,0], considering only pitch class "C♯" is indicated by the FBA "♯" and is four semitones above the bass.

[27] The validation set was also used for the selection of the best DNN model using early stopping.

[28] Uncertainty values show standard error across cross-validation folds.

Figure 4. An illustration of figured bass generated by our best-performing model for measure 8 of BWV 108.06 *Es ist euch gut, daß ich hingehe*, and measures 2 and 3 of BWV 145.05 *Ich lebe, mein Herze, zu deinem Ergötzen*, which are labelled (a) and (b) here, respectively. We artificially added the fifth (bottom) staff, which collapses all sonorities into one octave so as to more directly reveal the pitch-class content. As discussed in Section 3.3.1, our model predicts interval classes, and the figured bass is generated based on the intervals between the bass note and each predicted interval class. The agreement of each prediction with Bach's FBAs are shown as well: "✓" means that the generated figured bass exactly matches Bach's FBAs (the ground truth), "✓" in red means they are considered correct by our evaluation metric that treats musically equivalent figures as equivalent (see Section 3.2.2). An example of the latter can be found at m. 2.1 of (b) where the generated figures can be reduced to "2/4" from "2/4/6" (since the "6" can be omitted, as discussed in Section 3.2.2). "✗" means the generated figures are considered to be errors in our evaluations.

5. CONCLUSION AND FUTURE RESEARCH

This paper presents the Bach Chorales Figured Bass dataset (BCFB), which consists of 139 four-voice Bach Chorales with figured bass annotations encoded in MusicXML, **kern, and MEI. This dataset, and others that can be constructed using methodologies similar to those we propose, offer important potential for use in future computational studies in domains such as music theory, musicology, pedagogy, and performance practice.

This paper further shows how BCFB can be used as the basis for developing and evaluating both rule-based and machine learning models for predicting figured bass; our models achieved classification accuracies of 85.3% and 85.9% on BCFB, respectively. A potential reason the machine learning models did not outperform the rule-based model may be the relatively small size of BCFB.

Such automatically generated figured bass could help performers improvise *basso continuo* accompaniment for the remaining unfigured Bach chorales, or inform the design of pedagogical software for teaching Baroque theory or composition. Of particular interest, figured bass can potentially benefit automatic harmonic analysis research. Existing methods tend to either identify chords directly from the music [13–15], or identify and remove non-chord tones from the score and then generate chord labels from the remaining chord tones [16,17]. A new approach would be to first generate figured bass automatically from the music and then convert the figures to chord labels; this would allow a chord classifier to take advantage of knowledge implicitly learned from Bach's ground truth FBAs by a figured bass annotator during its training.

There are several refinements that could potentially improve the quality of the figured bass our approaches generate. The first is to add voice-leading information (how one voice moves horizontally), which may reduce some of the errors discussed in Section 4. The second would be to improve our rule-based model (e.g., by analyzing automatically trained decision trees), which in turn could provide further insights into Bach's approach to figuring bass, and perhaps provide musicological insight on how his methods changed over time or by the context of performance.

Another potential extension to this research would be to incorporate FBAs from other pieces by Bach, such as his chamber music, or from pieces by other Baroque composers, which are usually figured throughout the Baroque period. Once we have a variety of figured bass datasets for different genres and composers, we may then be able to train models that generalize better. Also, by comparing Bach's FBAs to FBAs by other composers, we may gain meaningful insights into Bach's unique compositional style and discover a sense of the degree of stylistic variability with which composers approached figured bass.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] P. Williams and D. Ledbetter, "Figured Bass," *Oxford Music Online*, Oxford University Press, 2001.

[2] J. S. Bach, A. Dürr, and W. Neumann, *Neue Ausgabe Sämtlicher Werke*. Bärenreiter, 1954–2007.

[3] C. P. E. Bach, *Essay on the True Art of Playing Keyboard Instruments,* 2 vols. (Berlin, 1753 and 1762). Translated by W. J. Mitchell, New York: W. W. Norton, 1949.

[4] D. Remeš, *Realizing Thoroughbass Chorales in the Circle of J. S. Bach (2 vols.)*. Wayne Leupold Editions, 2019.

[5] J. Barthélemy and A. Bonardi, "Figured Bass and Tonality Recognition.," in *Proc. of the 2nd International Society for Music Information Retrieval Conference,* 2001.

[6] A. Wead and I. Knopke, "A Computer-Based Implementation of Basso Continuo Rules for Figured Bass Realizations," in *Proc. of International Computer Music Conference*, pp. 188–191, 2007.

[7] N. Nápoles, G. Vigliensoni, and I. Fujinaga, "Encoding matters," in *Proc. of the 5th International Conference on Digital Libraries for Musicology*, pp. 69–73, 2018.

[8] P. Kröger, A. Passos, M. Sampaio, and G. De Cidra, "Rameau: A System for Automatic harmonic Analysis," in *Proc. of International Computer Music Conference*, pp. 273–281, 2008.

[9] Y. Ju, N. Condit-Schultz, C. Arthur, and I. Fujinaga, "Non-chord Tone Identification Using Deep Neural Networks," in *Proc. of the 4th International Workshop on Digital Libraries for Musicology*, pp. 13–16, 2017.

[10] G. Chew and R. Rastall, "Notation. 4. Mensural Notation from 1500. (viii) Scores; Harmonic and Descriptive Notations," *Grove Music Online*, 2001.

[11] F. T. Arnold, *The Art of Accompaniment from a Thorough-bass: As Practised in the XVIIth & XVIIIth Centuries*. Oxford University Press, 1931.

[12] A. Flexer and T. Lallai, "Can We Increase Inter- and Intra-Rater Agreement in Modeling General Music Similarity?," in *Proc. of the 20th International Society for Music Information Retrieval Conference*, pp. 494–500, 2019.

[13] T.-P. Chen and L. Su, "Harmony Transformer: Incorporating Chord Segmentation into Harmony Recognition," in *Proc. of the 20th International Society for Music Information Retrieval Conference*, pp. 259–267, 2019.

[14] K. Masada and R. Bunescu, "Chord Recognition in Symbolic Music: A Segmental CRF Model, Segment-Level Features, and Comparative Evaluations on Classical and Popular Music," *Trans. Int. Soc. Music Inf. Retr.*, Vol. 2, No. 1, pp. 1–13, 2019.

[15] M. T. Granroth-Wilding, "Harmonic Analysis of Music Using Combinatory Categorial Grammar," PhD thesis, *The University of Edinburgh*, 2013.

[16] N. Condit-Schultz, Y. Ju, and I. Fujinaga, "A Flexible Approach to Automated Harmonic Analysis: Multiple Annotations of Chorales by Bach and Prætorius," in *Proc. of the 19th International Society for Music Information Retrieval Conference*, pp. 66–73, 2018.

[17] Y. Ju, S. Howes, C. McKay, N. Condit-Schultz, J. Calvo-Zaragoza, and I. Fujinaga, "An Interactive Workflow for Generating Chord Labels for Homorhythmic Music in Symbolic Formats," in *Proc. of the 20th International Society for Music Information Retrieval Conference*, pp. 862–869, 2019.

A CORPUS-BASED ANALYSIS OF
SYNCOPATED PATTERNS IN RAGTIME

Phillip B. Kirlin

Department of Mathematics and Computer Science, Rhodes College

`kirlinp@rhodes.edu`

ABSTRACT

In this paper, we build on and extend a number of previous studies of rhythmic patterns that occur in ragtime music. All of these studies have used the RAG-C dataset of approximately 11,000 symbolically-encoded ragtime pieces to identify salient rhythmic patterns in the corpus and qualify how they are used. Ragtime music is distinguished from other musical genres by frequent use of syncopation, and previous computational studies have confirmed a number of musicological hypotheses regarding the use of syncopated patterns in ragtime compositions. In this work, we extend these studies to investigate further questions involving the use of syncopation. Specifically, we introduce a new methodological framework for processing the RAG-C dataset and confirm that experiments from previous studies obtain similar results using the new methodology. We investigate the use of the common "short-long-short" syncopated pattern in different time periods and present new results detailing its use by three well-known ragtime composers. We describe how the use of other syncopated patterns has evolved over time and the different distributions of patterns that result from those changes. Lastly, we present novel results identifying statistically significant patterns in the way composers varied the amount of syncopation in consecutive measures in compositions.

1. INTRODUCTION

In this work, we present an analysis of the salient rhythmic patterns that occur in ragtime piano music and quantify how the use of these patterns has changed over time and varies between composers. In particular, we illustrate that the specific sequences of syncopated patterns found in ragtime music and the ways in which they are ordered are not due to chance, but due to deliberate choices made by the composers. We argue that understanding and quantifying the musical choices made by composers is crucial to creating and improving the performance of various music information algorithms, including those for genre classification and algorithmic composition.

Recently, a number of corpus-based studies of ragtime

music have been published [1–4] that use a dataset known as the RAG-collection (RAG-C), a corpus of over 11,000 MIDI files first introduced by Volk and De Haas [1]. This dataset presents particular challenges to use due to its heterogeneous nature: it contains compositions from a wide variety of eras and composers, includes numerous ragtime styles (some of which could be argued are not ragtime at all), and has no standard method for encoding the music in MIDI format: some files are derived from live performances, while others are sequenced from sheet music.

Previous studies using the RAG-C corpus have identified the common usage of certain rhythmic patterns in ragtime, but have varied in their techniques in processing the corpus and interpreting the rhythmic patterns located. We present a new methodology for analyzing the corpus while taking care to confirm that our results align with previously-published studies.

Our contributions are as follows. First, we illustrate the feasibility of using automated algorithmic techniques to extract rhythmic patterns from a collection of MIDI files. We also argue for why our particular techniques work given the heterogeneous nature of the RAG-C dataset, especially involving the different time signatures present in the collection. Second, we extend previous corpus-based studies of ragtime music to illustrate the importance of specific kinds of syncopated patterns across the entire corpus, and also in subsets segmented by era and by composer. Because our methodological techniques are slightly different than those used in previous work, we confirm a number of earlier results and then extend them with new experiments and findings. Third, we show that additional patterns emerge when we study the way composers choose to order the amount of syncopation in successive measures: we statistically illustrate that this is done in a particular, deliberate manner. All the code for the work described here is publicly available. [1]

2. RAGTIME AND SYNCOPATION

In music, *syncopation* occurs when notes that a listener would expect to occur on strong beats in a measure are shifted to weak beats. Syncopation is particularly identified with *ragtime* music; while various definitions of ragtime exist, the unifying characteristic is the presence of certain varieties of syncopated rhythms [5]. Though in the modern era ragtime is often thought of as a form of music

---

[1] https://github.com/pkirlin/ragtime-ismir-2020

Figure 1. Different versions of the 121 pattern. Variations (a) and (b) are untied; (c) and (d) are tied.

restricted to the piano, during ragtime's heyday of roughly 1890–1920, this style of music was composed for all kinds of instrumental ensembles as well as in song form [6]. After 1920, ragtime fell out of compositional favor, though one can still find many rags composed during the modern era.

As syncopation is the defining characteristic of ragtime, it is natural to study how the use of syncopated patterns has changed over time. Ragtime scholars argue that a number of specific syncopated rhythmic patterns that composers gravitated towards changed in their frequency of use during the original ragtime era, with particularly drastic shifts occurring around the turn of the century. In particular, musicologists often focus on the importance of the "short-long-short" or "121" pattern. This pattern occurs with various note durations, but is usually found in the form ♪ ♩ ♪ in $\frac{4}{4}$ or $\frac{2}{2}$ time signatures and ♫♫ in $\frac{2}{4}$. It may occur at various locations within the measure; musicologists focus on how its position within the bar changed as ragtime evolved over time. In an *untied* syncopation, the pattern starts on either the metrical downbeat or halfway through a measure, therefore occurring entirely in the first or second half of a measure, as in Figure 1(a) and (b). In a *tied* syncopation, the pattern begins either one-quarter or three-quarters of the way through a measure, and therefore either crosses the midpoint of a measure or extends into the following measure, leading to a tie displayed in the notation, as in Figure 1(c) and (d). Music historians and previous studies of ragtime have noted that the untied syncopation was more typical of the early ragtime period of approximately 1890–1901, while the tied syncopation picked up in popularity after the turn of the century [5–7].

3. METHODOLOGY

Our methodology is similar to that used in previous studies [1–4], but different enough to warrant some explanation and confirmation that our results align with those of previous work. We begin by preprocessing the RAG-C dataset, with our goal being to identify a set of ragtime pieces sharing a set of basic, consistent properties. We do this by using an established MIR toolkit to identify the time signature and number of parts in each composition. We then use a notation program to automatically quantize the MIDI files and separate the melody from the accompaniment. Because MIDI files do not consistently represent pickup measures (anacruses, or fractional measures at the beginning of a composition), we conduct an experiment to illustrate that they are correctly identified in the dataset. We conclude the preprocessing stage with matching the MIDI files with entries in the RAG-C compendium, which pro-

vides useful metadata for each composition, and extracting the rhythms of all the melodic voices.

The RAG-C Dataset. The RAG-C dataset is a corpus of approximately 11,000 ragtime compositions in MIDI format, compiled over time by many enthusiasts of the ragtime genre. In addition to the compositions, the dataset includes a compendium spreadsheet providing metadata for each piece, including title, composer, year (or approximate year) of composition or publication, subgenre within ragtime such as march or two-step, and information about the source of the MIDI file such as the person who played the recording or sequenced the sheet music.

The MIDI file format, having been designed to support communication between electronic music devices, only contains low-level information about the timing of notes in a composition, and therefore MIDI files are more akin to transcriptions of a piece of music rather than a perfect representation of a printed score. A MIDI file is organized into tracks, with each track containing a sequence of *events* specifying when certain notes should be played. Each track also specifies the musical instrument that should play the notes in that track. While it is possible to specify higher-level information such as time or key signatures in a MIDI file, they are not required and are often omitted. Therefore, it is a non-trivial task to extract music-theoretic features from such files [8], especially metrical information. Notes in a MIDI file are only specified as starting and ending at a timestamp given in "ticks" measured from the beginning of the file. It is easiest to infer note durations when a time signature is present in the file and the file is created from a software sequencer, which will ensure that the notes within a MIDI file correspond to a logical metrical grid. When MIDI files are derived from human performances, however, notes that occur simultaneously on the printed page may not match up exactly in terms of ticks due to natural timing variances in performance. *Quantization*, the process of aligning the notes within a MIDI file to a metrical grid, must therefore occur to derive metrical information in such cases.

Melody and Accompaniment Extraction. To address these issues with the RAG-C dataset, we first decided to study only ragtime pieces for solo piano, a decision made in a number of previous studies using the dataset. Within the ragtime piano repertoire, it is common for most of the syncopated action to occur in the right-hand melody part, while the left-hand plays a steady accompaniment. Therefore, isolating the right-hand melody is a clear prerequisite for investigating the syncopated rhythms of ragtime. To accomplish this, we used the `pretty_midi` library [9] to identify MIDI files from the dataset containing exactly two piano tracks in the file. Normally such files contain the melody in one track and the accompaniment in the other; we verified this by calculating the average pitch of the notes in each track and comparing them. The track with the higher average pitch was labeled as the melody, and the other as the accompaniment. To account for files having two piano tracks due to other circumstances (such as two different compositions in one file or a single piano track duplicated twice), we hand-examined all MIDI files where

the difference in the average pitches of the two tracks was smaller than one octave to ensure that the melody identification algorithm functioned correctly. Furthermore, at this point we verified that all files remaining had time signatures of either $\frac{2}{4}$, $\frac{4}{4}$, or $\frac{2}{2}$. While ragtime is occasionally found in other time signatures, the vast majority of ragtime music occurs in these meters.

Quantization. Quantizing a MIDI file means aligning the notes in the file to a metrical grid in order to assign natural note durations to each note. To quantize our data, we ran each two-piano-track MIDI file through the MuseScore music notation program [10] and converted each file to its MusicXML representation. MusicXML is a richer format than MIDI that supports more features of common music notation; we use it here specifically so MuseScore can deduce standard note durations and measure boundaries for the MIDI files. We analyzed the resulting MusicXML output files using the `music21` library [11] and discarded any files for which more than 5% of the note onsets in the piece did not align with a 16th note grid. Most ragtime compositions rarely go beyond the 16th note level; we determined that any file with an overabundance of 32nd notes or notes at other onsets in the metrical grid probably was quantized incorrectly.

Title Matching. After all previous steps were completed, we were left with 1991 MIDI files. However, some of these files corresponded to the same ragtime composition, but encoded by different contributors to the RAG-C dataset. To ensure we only had one instance of each composition in our analyses, we used the Levenshtein edit distance to compare each MIDI filename — usually a combination of the composition title and MIDI encoder — against the set of composition titles in the RAG-C metadata spreadsheet. Any situation where a filename matched more than one title with an edit distance of 5 or less triggered an inspection by hand to assign the proper title. If two or more files matched with a single composition, we kept only the file with the highest quantization percentage; that is, the version with the highest percentage of notes that matched perfectly to the quantization metrical grid. This left us with a final total of 1058 MIDI files in our corpus, each one corresponding to a unique composition.

Accounting for Pickup Measures. Because of the lack of sophisticated metrical information in MIDI files, inconsistencies may arise when processing MIDI files derived from compositions containing a *pickup measure*, that is, an incomplete measure at the beginning of the music. Such music requires special handling as MIDI files do not explicitly store the locations of measure boundaries, and software that assumes such boundaries occur at regular intervals throughout the file will likely incorrectly process a MIDI file containing an incomplete measure. Because we will be investigating syncopation at different points within a measure, it is important that we correctly identify the measure boundaries in such cases.
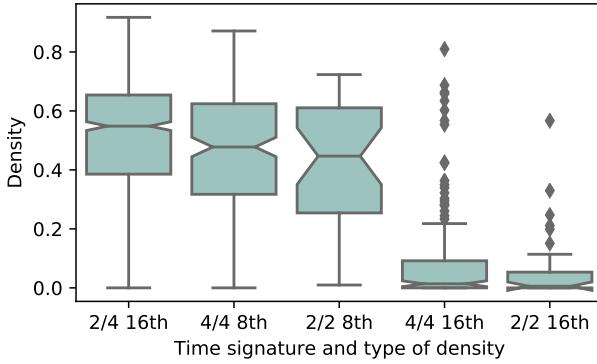
A common convention is to "pad" a pickup measure with silence at the beginning of a MIDI file, thereby lengthening the first incomplete measure into a complete one. Sometimes this convention is extended to MIDI files that do not have a pickup measure: such files begin with a complete measure of silence. Since any file with silence at the beginning clearly has been padded, we are left with what to do with any unpadded files — we made the decision to treat these files as having full measures throughout, without a pickup measure.

We justify this decision with the following experiment. We identified one particular contributor to the RAG-C dataset who chose to always encode MIDI files with silence at the beginning: a padded partial measure of silence for compositions beginning with a pickup, or a complete measure of silence for compositions without a pickup. This individual encoded 104 compositions in the dataset, and 26 of them began with a partial measure of silence, versus 78 with a complete measure of silence. Because 25% of this particular contributor's files begin with a pickup measure, we would expect that this proportion would hold in the remainder of the corpus as well. Of the 1991 MIDI files remaining after the quantization step, 1887 of them do not come from the contributor in question. Of the 1887, 539 begin with a fractional measure of silence and 1348 begin with a complete measure of silence or no silence. Because 539 out of 1887 is approximately 28.6%, it is reasonable to assume that the files with no silence at the beginning correspond to compositions with no pickup measure.

Binary Onset Patterns. The last remaining step in preprocessing the RAG-C dataset is to identify the rhythms of the melody part. Previous studies of ragtime syncopation used the convention of *binary onset patterns* to represent rhythms. These patterns are sequences of ones and zeros where a one represents the onset of a note and a zero represents a continuation of a note or a rest. These patterns can be computed at different levels of metrical granularity from a score. For instance, the binary onset pattern for a $\frac{2}{4}$ measure of four eighth notes would be "`10101010`" computed at the 16th-note level, but "`1111`" computed at the eighth note level.

Our dataset contains ragtime compositions in three different time signatures, namely $\frac{2}{4}$ (810 pieces), $\frac{4}{4}$ (214 pieces), and $\frac{2}{2}$ (34 pieces). We chose to compute binary onset patterns at the sixteenth note granularity for $\frac{2}{4}$ compositions, and at the eighth note granularity for $\frac{4}{4}$ and $\frac{2}{2}$ pieces. The basis for this decision was the observation that the use of sixteenth notes differed between pieces notated in the three time signatures, verified by the following experiment. We define the *sixteenth note density* for a measure of music as the proportion of the *weak* sixteenth note beats in a measure (that is, onsets 2, 4, 6, and 8 in $\frac{2}{4}$) that contain at least one note onset. We define the *eighth note density* for a measure similarly. We then computed the distribution of densities of sixteenth notes in all three time signatures, and eighth note densities in $\frac{4}{4}$ and $\frac{2}{2}$. We observed that our initial choices of appropriate granularities for binary onset patterns — corresponding to the first three plots in Figure 2 — fall in similar ranges, while the last two plots do not, justifying our mixed use of sixteenth and eighth note binary onset patterns. However, Figure 2 does show a number of outliers in the $\frac{4}{4}$ and $\frac{2}{2}$ sixteenth note density plots, falling more in the range of the first three plots, indicating

Figure 2. Box plots illustrating the distribution of 8th and 16th note densities in the various time signatures.

that in future experiments, we should consider analyzing those particular pieces at the sixteenth note level.

4. EXPERIMENTS
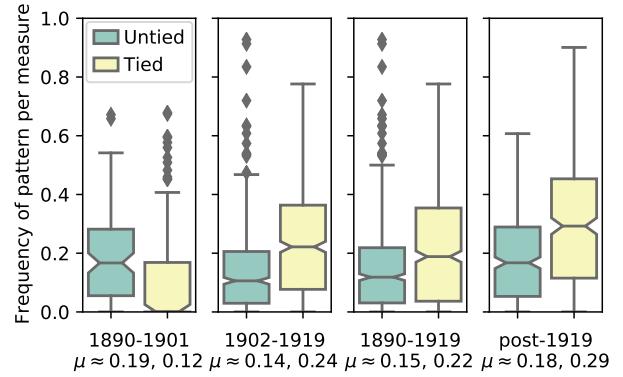
In this section we describe a number of analytical experiments that we conducted to extract information from our corpus about the way syncopation is used in ragtime music.

4.1 Exploring the 121 Pattern

Ragtime scholars hypothesized *untied* syncopations were the predominant form of syncopated pattern found in early ragtime compositions from approximately 1890 to the turn of the century, while *tied* syncopations did not become common until around 1902 in the late ragtime period [6,7]. This hypothesis was confirmed by Volk and De Haas [1]; we replicate their experiment due to differing methodologies for processing the RAG-C dataset. These differences in selecting and quantizing MIDI files naturally produce a slightly different corpus with which we are working, and therefore replicating earlier work shows that these results are invariant with a well-rounded corpus and lends credence to our extended results that build on earlier studies.

In this experiment, we compared the frequency of use of the 121 tied and untied patterns in ragtime compositions from three eras: first, we compared the early ragtime period of 1890–1901 (110 pieces) with the late ragtime period of 1902–1919 (582 pieces), and then compared the entire ragtime period of 1890–1919 (692 pieces) with the modern period of 1920 to the present (362 pieces). The 121 patterns were found by looking at the binary onset patterns that were collected earlier and counting the number of times each variety of syncopation appeared in a composition. It is possible for multiple 121 syncopations to appear in a single measure, as each syncopation covers only part of a measure. To account for differing lengths of pieces, we divided the total tallies by the number of measures in each composition, resulting in *frequency per measure*.

Overall, these results are similar to those of Volk and De Haas, taking into account some variation for differing pieces selected from the RAG-C dataset. In particular, we confirm that the number of tied patterns doubled between the early and late ragtime eras, and the number of untied patterns decreased between the eras as well. The left two

Figure 3. Box plots illustrating the distribution of frequencies of 121 patterns per measure, comparing different eras of ragtime. Means (μ) are shown below each plot.

plots in Figure 3 illustrate this. Wilcoxon rank-sum tests confirm (untied: $p < 0.001$, tied: $p \ll 0.001$) that the differences between the eras are statistically significant given the null hypothesis that the distributions are identical.
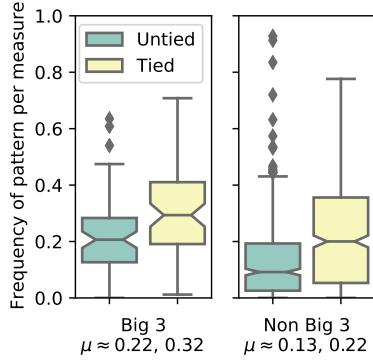
The right two plots in Figure 3 can be analyzed in a similar fashion, and illustrate that the use of both types of syncopation climbed after the end of the ragtime era. Wilcoxon rank-sum tests again confirm (untied: $p \ll 0.001$, tied: $p \ll 0.001$) that these increases are statistically significant.

The Big Three. Ragtime scholars agree that three ragtime composers stand out from the rest in terms of best exemplifying the ragtime genre: Scott Joplin (1867 or 1868–1917), James Scott (1885–1938), and Joseph Lamb (1887–1960) [12–14]. These composers are well-represented in the RAG-C dataset, and it is instructive to examine if they used syncopation patterns differently than each other or as compared to other ragtime composers.

When comparing the big three among themselves, small differences in usage of the 121 pattern emerge but nothing that cannot be attributed to chance. However, statistically significant differences are evident when comparing the output of the big three against other ragtime composers. In particular, even when controlling for era, the big three composers used more 121 patterns than other composers. In this experiment, we isolated the output of the big three composers during the late ragtime era of 1902–1919 (70 pieces), and compared their compositions against those of the remaining composers from the same era (512 pieces). Wilcoxon rank-sum tests confirm that the differences in the frequency of use of both untied ($p < 0.0001$) and tied ($p < 0.0001$) syncopations are different between the big three and the remaining composers. Figure 4 illustrates how the big three used, on average, almost 70% more untied 121 syncopations and almost 50% more tied 121 syncopations.

4.2 Analyzing other syncopated patterns

Analyzing frequencies of the 121 pattern is instructive to verify certain hypotheses put forth by musicologists. However, it is not a given that this pattern or its variants are necessarily the most prevalent patterns found in ragtime. In
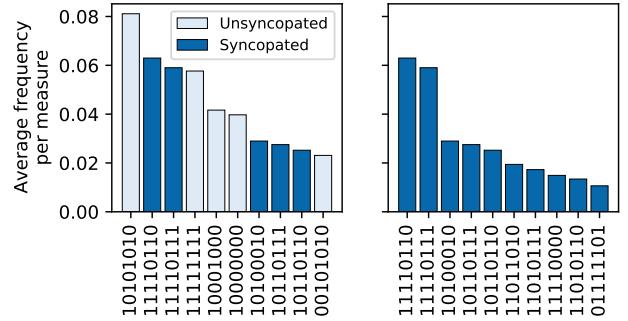
Figure 4. Box plots illustrating the distribution of frequencies of 121 patterns per measure, comparing compositions by the "big three" ragtime composers in the late ragtime era versus all other compositions from that same era.

Figure 5. Left: The ten most frequent binary onset patterns overall, differentiating between unsyncopated patterns (LHL > 0) and syncopated patterns (LHL > 0). Right: Then ten most frequent syncopated patterns.

this section, we expand our analysis to explore all possible binary onset patterns and measure the amount of syncopation present in each pattern.

We follow the model of Koops, et. al. [2] and use the Longuet-Higgins and Lee [15] metric, grounded in aural perception of rhythm [16], to quantify the amount of syncopation present in a measure based on its binary onset pattern. The LHL metric is zero for a measure with no syncopation and increases with each instance of a note onset (1) occurring on a weak beat followed by no onset (0) on the immediately-following (strong) beat. The increases are larger for syncopations crossing more significant divisions of the measure. For example, the binary onset pattern 01010101 contains three syncopations (instances of 10). The syncopation in the middle that crosses the midpoint of the measure has an LHL value of 2, and the two syncopations on either side have values of 1 because they cross weaker divisions of the measure. Therefore, this measure by itself has an LHL score of 4. If this measure were followed by no onset on the downbeat of the next measure, the LHL value would increase by 3 for the additional syncopation crossing the barline, for a total LHL value of 7.

In performing the following experiments, we calculated the LHL values for each measure of the melody in the corpus using the binary onset patterns computed earlier. Our corpus of 1058 pieces contained 140,856 measures of music, with the average LHL value for a measure being 1.17, with a standard deviation of 1.35. However, only 77,012 of the measures ($\approx$ 55%) contained any syncopation at all (LHL > 0). If we only consider measures with LHL > 0, the average LHL value becomes 2.14, with a standard deviation of 1.12.

We note that our methodology for computing and interpreting binary onset patterns differs enough here from Koops, et. al. [2] to warrant explanation. In their work, binary onset patterns may be up to 16 bits in length, corresponding to measures in $\frac{4}{4}$ or $\frac{2}{2}$ analyzed at the 16th note level, allowing for the possibility of having LHL values for a single measure of music as high as 15. Here, all our binary onset patterns are of length 8, due to always analyzing measures in $\frac{4}{4}$ or $\frac{2}{2}$ at the eighth note level and measures in $\frac{2}{4}$ at the 16th note level. Therefore, it is difficult to draw

direct numerical equivalences between our studies.
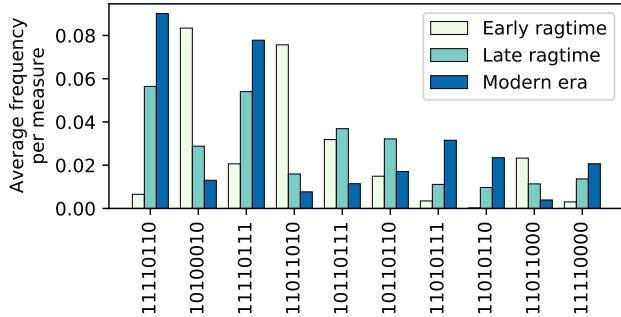
Overall patterns. We first examine the frequencies of all possible binary onset patterns in the corpus. To ensure that the varying lengths of the compositions in the corpus did not affect our results, we divided the number of times each binary onset pattern appeared in a composition by the number of measures in that composition, thereby obtaining the *frequency per measure* for each pattern. We then averaged the frequencies across all pieces in the corpus. The top ten patterns overall, and the top ten with an LHL score greater than 0 are shown in Figure 5. It is noteworthy that for a genre identified with such high levels of syncopation, there are many common non-syncopated patterns. In particular, patterns 1, 4, 5, 6, and 10 do not contain any syncopation.

In analyzing the right side of Figure 5, we note the presence of a tied 121 pattern (1101) in the middle of patterns 1, 2, 4, and 5, along with untied 121 patterns at the beginning of patterns 6, 7, and 9, and at the end of pattern 10. Additionally, pattern 3 is the 121 pattern in augmented form. Only pattern 8 is not connected with the 121 figure.
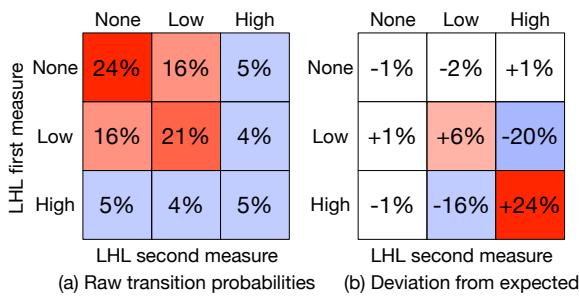
Patterns by era. We conducted an experiment to determine whether composers of different eras used certain types of binary onset patterns differently. Using our earlier grouping of compositions into the early ragtime era, the late ragtime era, and the modern era, we computed the most popular rhythms in each group, which can be seen in Figure 6; we note that popular patterns in some eras become unpopular in others. We used three Wilcoxon signed-rank tests to compare pairs of eras, using all 256 possible binary onset patterns in each test. The results give us weak statistical significance at the $\alpha = 0.05$ level using the Šidák correction to account for the multiple comparisons, suggesting that composers chose rhythmic patterns differently in the three eras ($p \approx 0.01$ for the early versus late eras, $p < 0.001$ for the early versus modern eras, and $p \approx 0.01$ for the late versus modern eras).

4.3 Transitions Between Patterns

The overall sound of a piece of music depends not only on the contents of the individual measures of music but also on the choice of which measures follow other measures. Ragtime is no exception, and we hypothesize that there are

Figure 6. The most frequent syncopated (LHL > 0) rhythmic patterns, segmented by era.

Figure 7. (a) The joint transition probabilities for observing of the nine possible LHL transition pairs. (b) The deviation from the expected probability if there were no correlation between the LHL value in one measure and the next. The four highlighted deviations in (b) are statistically significant.

relationships between the syncopated patterns in consecutive measures. Specifically, we propose the question of whether the degree of syncopation in a measure of music is related to the degree of syncopation in the surrounding measures.

We chose to answer this question by examining all consecutive pairs of measures of music in the corpus and computing their LHL values separately for the first measure and the second measure of the pair. Recalling that the average LHL value for a syncopated measure of music was approximately 2.14, we binned the LHL values according to having a *high* amount of syncopation (LHL ≥ 3), a *low* amount of syncopation (LHL = 1 or 2), or *no* syncopation (LHL = 0). For each piece of music, we computed the frequencies of each of the nine possible LHL transition pairs, normalizing for the number of measures in each composition, and averaged the frequencies across all compositions. The results are displayed as joint probabilities in a heatmap in Figure 7(a). We observe that transitions between consecutive measures with no syncopation are extremely common, while any transition involving a high amount of syncopation is uncommon.

These joint probabilities, however, do not tell the whole story. Because the distribution of LHL values is highly skewed towards the smaller values, it is worthwhile to test if any of these LHL transition pairs occur with certain tendencies due to chance, or due to deliberate choices on the composer's part. For example, given that almost half of

the measures of music in the corpus do not contain any syncopation, should we be surprised that 24% of the total transitions are between two measures without syncopation? We can answer this question with a final experiment. We compared the LHL transition frequencies obtained from the corpus against corresponding frequencies that would be obtained if one were to randomly reorder the measures in each piece of music. Specifically, for each composition, we generated 1000 random reorderings of the measures, computed the LHL transitions for every pair of consecutive measures, averaged them across the 1000 reorderings, and then proceeded as we did earlier with normalizing and averaging across all compositions. These results, illustrated in Figure 7(b), confirm that a number of the LHL transitions occur significantly more or less frequently than would be expected under a random reordering of measures of the compositions.

We used nine individual binomial tests to compare the the true LHL transition frequencies to the expected frequencies under the null hypothesis that measure transitions resemble those done randomly. At a significance level of $\alpha = 0.05$, taking into account the Šidák correction for multiple comparisons, the four highlighted transitions in Figure 7(b) are statistically significant (all with $p \ll 0.0001$). This tells us, for instance, that even though it is overall rare to find consecutive measures with high amounts of syncopation in a composition, this phenomenon still occurs more often than would be expected if a composer were introducing syncopation at random.

5. CONCLUSION AND FUTURE WORK

In this study, we used new methods to extend a number of previous analyses of ragtime syncopation to confirm existing musicological hypotheses and present new ones. Specifically, we confirmed the earlier finding that the 121 syncopation idiom varies in use between ragtime eras, even when using a different preprocessing scheme for the RAG-C dataset. We demonstrated a new finding that the "big three" ragtime composers also employed this syncopation pattern more often than their contemporaries did. We illustrated how other rhythmic patterns evolved over time and revealed different frequency distributions. Lastly, we displayed novel results showing statistically significant differences in the way the amount of syncopation changes between consecutive measures in a ragtime composition.

In future work, we plan to continue to study the use of syncopation in ragtime, specifically towards uncovering more information about varying musical parameters (rhythmic, melodic, or harmonic) between measures. We believe it will be useful to expand the ideas in this paper to other musical genres as well. Previous research has successfully used information about rhythmic patterns to assist in genre classification [17–19], and we imagine the data presented here could be useful in such circumstances. We also hypothesize that algorithmic composition techniques that rely on probabilistic techniques for rhythm generation [4, 20] might be improved by considering how musical parameters like rhythm change from measure to measure as well as within a single measure.

6. REFERENCES

[1] A. Volk and W. B. de Haas, "A corpus-based study on ragtime syncopation," in *Proc. of the 14th International Society for Music Information Retrieval Conference*, 2013, pp. 163–168.

[2] H. V. Koops, A. Volk, and W. B. de Haas, "Corpus-based rhythmic pattern analysis of ragtime syncopation," in *Proc. of the 16th International Society for Music Information Retrieval Conference*, 2015, pp. 483–489.

[3] D. Odekerken, A. Volk, and H. V. Koops, "Rhythmic patterns in ragtime and jazz," in *Proc. of the 7th International Workshop on Folk Music Analysis*, 2017, pp. 44–49.

[4] J. Michelson, H. Xu, and P. B. Kirlin, "Probabilistic generation of ragtime music from classical melodies," in *Proc. of the 6th International Conference on Mathematics and Computation in Music*. Springer International Publishing, 2017, pp. 350–360.

[5] I. Harer, "Defining ragtime music: Historical and typological research," *Studia Musicologica Academiae Scientiarum Hungaricae*, vol. 38, no. 3–4, pp. 409–415, 1997.

[6] E. A. Berlin, "Ragtime," in *Grove Music Online*. Oxford University Press, 2013. [Online]. Available: https://doi.org/10.1093/gmo/9781561592630. article.A2252241

[7] J. E. Hasse, "Ragtime: From the top," in *Ragtime: Its History, Composers, and Music*, J. E. Hasse, Ed. New York: Shirmer Books, 1985, pp. 1–39.

[8] C. Raffel and D. P. W. Ellis, "Extracting ground-truth information from MIDI files: A MIDIfesto," in *Proc. of the 17th International Society for Music Information Retrieval Conference*, 2016, pp. 796–802.

[9] ——, "Intuitive analysis, creation and manipulation of MIDI data with `pretty_midi`," in *Proc. of the 15th International Conference on Music Information Retrieval, Late Breaking and Demo Papers*, 2014.

[10] MuseScore BVBA, "MuseScore," version 3.4.2. Software. http://www.musescore.org.

[11] M. S. Cuthbert and C. Ariza, "`music21`: A toolkit for computer-aided musicology and symbolic music data," in *Proc. of the 11th International Society for Music Information Retrieval Conference*, 2010, pp. 637–642.

[12] R. Blesh and H. Janis, *They All Played Ragtime*, 4th ed. New York: Oak Publications, 1971.

[13] W. M. McNally, "Ragtime then and now: Composers and audiences from the ragtime era to the ragtime revival," Ph.D. dissertation, The City University of New York, 2015.

[14] J. Magee, "Ragtime and early jazz," in *The Cambridge History of American Music*, D. Nicholls, Ed. Cambridge University Press, 1998, pp. 388–417.

[15] H. C. Longuet-Higgins and C. S. Lee, "The rhythmic interpretation of monophonic music," *Music Perception*, vol. 1, no. 4, pp. 424–441, 1984.

[16] ——, "The perception of musical rhythms," *Perception*, vol. 11, no. 2, pp. 115–128, 1982.

[17] S. Dixon, F. Gouyon, and G. Widmer, "Towards characterisation of music via rhythmic patterns," in *Proc. of the 5th International Conference on Music Information Retrieval*, 2004, pp. 509–516.

[18] D. C. Correa, L. d. F. Costa, and J. H. Saito, "Tracking the beat: Classification of music genres and synthesis of rhythms," in *Proc. of the 17th International Conference on Systems, Signals and Image Processing*, 2010, pp. 280–283.

[19] T. M. Esparza, J. P. Bello, and E. J. Humphrey, "From genre classification to rhythm similarity: Computational and musicological insights," *Journal of New Music Research*, vol. 44, no. 1, pp. 39–57, 2015.

[20] J. D. Fernández and F. Vico, "AI methods in algorithmic composition: A comprehensive survey," *Journal of Artificial Intelligence Research*, vol. 48, pp. 513–582, 2013.

"PLAY MUSIC": USER MOTIVATIONS AND EXPECTATIONS FOR NON-SPECIFIC VOICE QUERIES

Jennifer Thom[1]   **Angela Nazarian**[2]   **Ruth Brillman**[1]
Henriette Cramer[1]   **Sarah Mennicken**[1]
[1] Spotify, USA
[2] University of California, Davis
jennthom@spotify.com

ABSTRACT

The growing market of voice-enabled devices introduces new types of music search requests. As voice assistants can potentially support conversational requests, music requests can be more ambiguous than requests in typed search interfaces. However, these systems may not be able to fulfill ambiguous requests in a manner that matches the user need. In this work, we study an example of ambiguous requests which we term as non-specific queries (NSQs), such as "play music," where users ask to stream content using a single utterance that does not specify what content they want to hear. To better understand user motivations for making NSQs, we conducted semi-structured qualitative interviews with voice users. We observed four themes that structure user perceptions of the benefits and shortcomings of making NSQs: the tradeoff between control and convenience, varying expectations for personalization, the effects of context on expectations, and learned user behaviors. We conclude with implications for how these themes can inform the interaction design of voice search systems in handling non-specific music requests in voice search systems.

1. INTRODUCTION

Voice assistants and smart speakers are rapidly becoming ubiquitous. Globally, an estimated 600 million people use voice assistants at least once a week [7]. In the U.S., roughly a quarter of adults own a smart speaker, such as an Amazon Echo or Google Home [16, 28]. One of the most popular use cases for smart speakers is music listening [1, 5]. If a user approaches a music search with a specific piece of content in mind and requests it by artist and track name [14], current voice assistants are typically able to fulfill such requests.

However, voice assistants also provide users with the opportunity to search for music in a conversational and open-ended manner without mentioning specific entities. We focus on non-specific queries (NSQs) in this research, which are requests to play music but which lack any specifications in the user's language about which music to return. An example of a non-specific voice query would be "Play music" (since no specifics are provided) but not "Play me some hip hop," since "hip hop" specifies a genre.

Due to the current limitations of voice search systems, interactions that do not exactly specify what music to request are less likely to be successful. When faced with a potentially unbounded space of conversational interactions with a voice assistant, users may simply not know what to say [9, 25]. As a result, users can end up falling back on habits they developed when they first obtained their voice assistants [5] and resort to more simple interactions. In addition, users' mental models of the voice assistant's capabilities do not always match its actual capabilities [22]. This makes it difficult for users to know what types of voice searches will end up providing the desired results.

Our research questions are as follows: (1) what are user motivations for making non-specific queries (NSQ) and (2) what do users expect from the system when they make NSQs? To study these questions, we conducted a qualitative interview study with users who make such requests. Finally, we conclude with implications for how voice assistants can better meet these user needs and propose avenues for future work.

2. RELATED WORK

2.1 Voice Assistants and Music Search

Voice has become a common interaction modality for users to search for information. In particular, music has emerged as a popular domain for voice search. This has been observed across device types, ranging from mobile phones to smart speakers. Guy's [11] analysis revealed that music videos were the top triggered results on voice search on mobile phones. In the case of smart speakers, music listening is not only the most commonly requested functionality but requests for information about music (e.g., "Who sings this song") also emerged as a prevalent search [1]. However, while voice interactions can potentially enable quick

---

[1] The third author is now at Google and has not contributed to the paper after joining Google.

music playback on a smart speaker, issues such as difficult to pronounce artist names can pose a challenge for users, making these interactions more effortful [32].

2.2 Music Search and Discovery

To observe how users would use natural language to search for music and to better understand users' information needs, researchers have turned to general domain repositories such as Google Answers. Bainbridge et al. [3] observed that users typically employ a variety of metadata to form music searches, such as bibliographic information including performer, title of work, or date of recording. Lee [19] also observed that music searches in Google Answers are typically known-item queries.

Music discovery on music streaming services also occurs through personalized recommendations, which can be potentially supported through natural language voice queries. Such discovery often requires a user to be receptive to novelty, suggesting that listeners are selective about the situations where this would be a positive experience [18]. Lee and Price [20] observe that music listeners with more 'adventurous' music habits are more positive about novelty in recommendations while more 'discerning' listeners expect recommendations to not be novel enough to meet their tastes. Because users of voice assistants can use natural language for ambiguous queries for a variety of reasons, these systems will have to account for a listener's appetite for discovery.

We extend the research on music search and discovery by studying a common query employing non-specific language that may not necessarily indicate a tolerance for exploration. In addition, we note that the prior research covered in this section has predominantly focused on typed search and modalities that offer visual feedback.

3. METHOD

3.1 INTERVIEWS

To elicit a rich descriptive user-centric dataset about non-specific queries that may carry meaning to users despite their simple format, we chose to employ semi-structured interviews. The data collection took place over a one month period in 2018 and consisted of 17 in-person 60 minute interviews. We sent a recruitment email to a random sample of Spotify users in a city in the Northeastern United States. The user selection criteria were: (1) owning a voice-enabled device and (2) making at least one NSQ on that device within the previous 30 days. The recruitment email also asked users to indicate what voice-enabled devices they owned and which music streaming services they used.

We sampled participants who used smart speakers, such as Google Home, Amazon smart devices as well as mobile voice assistants to cover a broader range. To better reflect the diversity of the listener population, we selected participants from a range of ages, occupation, and gender. Participant age ranged from 21 to 52 (mean = 31) with a wide range of occupational backgrounds. Table 3.1 summarizes

P#	Age	Gender	Job or Industry	Used Device(s)
1	29	f	Health care	AA, MVA, Other
2	23	m	Student	GH, MVA
3	39	m	Technology	GH, MVA
4	26	f	Asst Director	MVA
5	21	f	Student	MVA
6	25	f	Operations	GH, MVA
7	26	f	Clerk	MVA
8	35	f	Director	AA, AA
9	52	m	Education	AA
10	39	m	Dispatcher	GH, MVA
11	26	m	Advisor	MVA
12	29	m	Analyst	AA, MVA
13	33	f	Legal	AA, MVA, Other
14	34	m	Education	MVA
15	25	m	Student	GH, MVA
16	32	m	Manager	MVA, Other
17	38	m	Design	MVA, Other

Table 1. Demographic information of our study participants and their voice assistant usage. Google Home (GH), Amazon Alexa (AA), Mobile Voice Assistant (MVA)

the demographic characteristics of the participants. Participants were paid $100 for their time, consistent with the compensation level for industry user research.

Our interviews consisted of two parts. The first part focused on presenting scenarios to elicit user motivations for making NSQs. The second part employed specific examples of common NSQs as probes to investigate user expectations for the results of NSQs.

3.1.1 Part 1: Scenarios

The interviewer began by asking questions related to user's daily listening habits to develop an understanding of each user's unique music listening style. The participant would then answer questions about their voice device habits regarding general domain voice interactions as well as those specific to music streaming and recommendation. We probed users' motivations for requesting music through voice, and any factors that might discourage them. Following this discussion, we prompted participants with eight different scenarios in randomized order to describe whether and how they would choose to issue NSQs to their voice devices. We included the following scenarios: listening to music right now, during a workout, with friends at party, in the morning, after work or school, during chores, during the commute, while starting the day.

3.1.2 Part 2: Utterances as probes

Next, the interviewer presented five different wordings of NSQs selected from a pool of the 50 most common NSQs appearing in logged data in the 30 days preceding the month of the study. The interviewer then asked the participants to provide the utterances verbatim to the voice assistant in the room, which matched the assistant that participants reported owning and interacting with most frequently in our recruitment email. The five utterances used dur-

ing this part of the interview were: "Play Spotify playlist", "Turn on the music","Play music from Spotify","Play music","Play my Spotify."

3.2 Analysis

To identify broader themes encompassing the original annotation codes, researchers analyzed the interview transcripts using Braun & Clarke's 6-step framework [6]. Thematic analysis [24] was used to develop annotation codes to identify common themes and pervasive concepts. Three coders verified the codes by annotating the transcripts independently. The annotation tags were subsequently grouped into larger conceptual categories to provide insight into participants' behaviors, motivations, expectations and desired experiences.

4. FINDINGS

Our findings can be categorized along four themes that characterize user perceptions of the benefits and shortcomings of NSQs.

4.1 Trade-offs Between Effort and Control

We observed that our participants perceived NSQs as a convenient way to start listening to music with little effort by ceding control to the voice assistant. On the other hand, participants refrained from making NSQs when they believed that the returned content would be unpredictable. In this section, we describe how participants actively weigh the tradeoffs between user control and effort when deciding whether to make an NSQ.

4.1.1 Starting music effortlessly is more important than a specific outcome

Participants reported that they made non-specific queries as a lower effort way to request music. For instance, they described wanting to easily start a 'lean back' (or 'hands off') music listening session where the music comprised a soundtrack or background effect to the user's activity. In these situations, participants prioritized the convenience of making a lower effort NSQs rather than requesting something specific.

> I just need something to play in the background and accompany my morning. - P6, 25

Users also reported a desire to not add an additional level of effort to a current activity, especially when that activity required the use of the user's hands or eyes. In these cases, any keyboard or typed search would require an extra level of effort, as it would involve interrupting their current activity. Below, a participant enumerates the situations where they would choose voice search and NSQs over text-based search.

> Yeah, I would definitely say cooking, cleaning, housework type stuff, where I probably have my hands full. - P8, 36

Some users identified NSQs as an effortless way to discover novel music. Here, a user describes using NSQs as a fallback to specific queries when they needed an accessible and low-effort way to seek music that is new to them. For this user, a willingness to make an NSQ for low-effort music discovery is closely tied to their understanding of Spotify's reputation. When this user does not see one service as allowing for new music discovery, they switch to an alternative service when experiencing this NSQ intent.

> I've been using [music service X] longer than [music service Y]. And I consider [X] my go-to, let's say...But [Y] I like when I don't want to be so hands on...so, I would say if I can find new music, it's normally through [Y], but I create lists – playlists on [X]. And so, a little music that I've basically cultivated for at least a decade...I typically go to [Y] when I'm trying to be mindless about what I'm listening to - P16, 32

Participants reported providing NSQs when they did not have something specific in mind that they wanted to hear but still wanted to begin a listening session without having to expend decision-making effort.

> I don't want to make a decision necessarily, or I can't really think of what I want to listen to at that point. - P1, 29

This effect was particularly strong when the act of choosing a more specific piece of content felt like an obstacle or burden.

> [An NSQ] also sort of takes the edge off of having to make your own playlist, which is something I like doing, but if I didn't feel like putting the effort then. - P11, 26

This is consistent with findings observed by Hosey et al. [14] in which users reported being open to various results when they did not have anything specific in mind when searching for music. In the current study, we observed this as well, in particular when participants also wanted to initiate listening to music with as little effort and mental load as possible.

4.1.2 Desire for predictable outcome and associated feedback

Users consistently expressed the belief that they could not exercise as much control through voice search, compared to keyboard or touch searches. This was partly because users felt more context was available through typed queries.

> I'm a visual person. I guess I kind of like to read all the songs that might come up in a playlist or something like that. Whereas if I'm [providing a voice request] I can't really do that, it's just whatever [music streaming service] picks out. - P2, 23

This perceived lack of control is at least partially tied to real-world limitations of voice assistants, which can break down at multiple points. For instance, ASR systems can struggle to correctly transcribe the non-standard spellings (e.g., for artist "6lack"), difficult or ambiguous pronunciations (e.g., for the hip hop duo "Rae Sremmurd"), foreign languages and non-Latinate alphabets (e.g., Hebrew, Arabic, Greek) [32]. Named entity-recognition systems can struggle with ambiguous requests, e.g., "Play Changes." Here, the voice system might still return an unintended result, such as playing "Changes" by David Bowie rather than "Changes" by 2Pac.

In an attempt to avoid these errors, users often make voice requests that they perceive as *safe bets*, such as artists they expect the system to understand. Below, a participant discusses not requesting a particular album because ASR systems often incorrectly transcribe it.

> I definitely default to playing an artist rather than a specific album...I try to have it play like 'Citsuoka' [an album by My Morning Jacket]– there've been times when I've struggled and it hasn't been able to recognize that. And so some of it is definitely a habit that I've developed where it's more reliable to just go with the artist. - P3, 39

These experiences with voice systems can influence how a user will engage with NSQs. Users who trust the recommender system may start to consider NSQs as *safe bets*. Users who experience a sense of distrust due in part to NLU errors may refrain from making NSQs, believing that they would be too difficult for the system to fulfill.

4.2 Expectations for Personalization

Participants' perceptions of NSQs are tied to their expectations of whether or not the requests would yield a personalized result. In particular, we observed that participants had varying levels of trust in the quality of recommendations that they would receive in the first turn of their voice interactions.

4.2.1 Trust that generic queries would lead to personalized results

Participants who were open to musical exploration or felt comfortable relinquishing control over their listening session often used NSQs. By making such open-ended requests, users were aware of the possibility of hearing something new or unexpected by requesting music in such an open-ended manner. This was particularly marked with participants who expressed high degrees of confidence in the service's recommendation algorithms.

> I think the algorithms on here are super sharp and I think that it does a really great job of condensing what I'm into, what I have saved, and pulling something up. - P4, 26

We observed that participants created their own theories about how they would receive personalized results from NSQs, consistent with Eslami et al. [8]. For instance, participants hypothesized that they received personalized recommendations based on prior music listening behavior.

> I'm assuming it bases it off of whatever kind of music I like. I mean, it started playing a song that I was okay with...Or, I assume, based on whatever the app thinks that I would like based on the kind of music that I currently like and have favorited and stuff. - P10, 39

Participants also expressed a varying level of comfort with proactive recommendations that could be fulfilled with an NSQ. For instance, a participant describes a desire for an NSQ result that could be personalized to meet a user need for music discovery and exploration to return something novel.

> I would know at this point 'play my [music streaming service]' is my default where I'm at at that time, where I usually and if I want something different in the usual time and space, I have to look for something specific. Because it doesn't know that I don't want the usual. - P17, 38

In addition, some participants expected that personalization could be sufficiently proactive to be predictive of user need without explicit input from the user.

> I think the ideal situation is I want [music streaming service] to know what I want to hear before I know what I want to hear. - P4, 26

When participants had positive expectations about the personalization capabilities of the voice assistant, they were more optimistic about making NSQs and receiving a successful result.

4.2.2 Fear that generic requests might lead to unfamiliar content

In contrast, participants who were less open to musical exploration refrained from making NSQs out of a reported fear that the returned content will be outside the range of music they normally listen to. As a result, familiar content would be more likely to result in an enjoyable listening experience to these participants.

> I'm usually really specific because I like certain things. Well, as you get older, you realize this is what you like, and there's enough music for me to go back to and things I like. I don't discover too much anymore. Occasionally, but not as much as I used to. - P9, 52

> I'm not too adventurous with my music, so I opt and like to listen to songs that are similar to songs that I have heard before. - P6, 25

These findings are consistent with Li et al.'s [21] observations that users who conduct non-focused music

searches consume more novel songs. When users want to hear familiar content, they will not use non-specific language to request it.

Finally, our qualitative interview responses indicate that users do not perceive all NSQs identically and had different expectations for outcomes depending on unique meaningful linguistic markers in the NSQ. For instance, there was a consensus among users that the presence of a personal pronoun in a non-specific request was tied to an elevated desire to hear personalized content that is suited specifically to their musical taste.

> I just assumed that 'play my [music streaming service]' would yield a different thing than just turning on the music because the 'my', to me, indicates the music that I've already liked or downloaded. - P11, 26

4.3 Context Affects User Motivation to Make NSQs

Our participants reported on different contextual factors that affected their motivation to make a non-specific request.

4.3.1 More openness to discovery at certain times of day

Our participants reported that certain times of day were more conducive to openness to new music and searching for music in an exploratory mindset in the manner described by Hosey et al. [14].

> I can see myself doing this in the mornings when I just need something to play in the background and accompany my morning. When I'm working out or when I'm sleeping or when I'm reading, I kind of want a certain mood, but in the mornings, I'm more open to exploration. - P6, 25

Participants further noted that there were certain activities that would occur at certain times of the day, which may make time of day a potentially good proxy for sensing context. This is consistent with other research suggesting that listening preferences change throughout the day [27].

4.3.2 Less individual control needed in a group setting

Social context can influence how participants perceive the utility of making NSQs. Below, a participant expressed her willingness to make NSQs with others in a group.

> If I'm with a bunch of other people, my friends, they're like, 'Oh, you're still listening to this playlist? Put something else on.' Just having – it's like a nice neutral third party, where it's like, 'Oh, I'm not dominating the radio, and neither are you.' - P8, 36

Here we see social context play a role in how participants would make NSQs. In social settings, participants reported using non-specific queries to purposely relinquish control over their music listening experience. In this specific context, being able to abdicate control allowed the music streaming service to step in as a DJ and provide music for their social listening session. This was motivated not only by the aforementioned desire for an effortless experience, but also for users to avoid being judged for their personal taste in music.

4.3.3 Desire for specific results to fit moods and activities

Because participants perceived the results of NSQs to be unpredictable, they were less willing to make NSQs when they wanted to hear music that would fit their current mood or emotional state. This user motivation aligns with previous work on how music serves to regulate affect [15,27,29,33]. We distinguish this motivation from the observation that participants make NSQs as a low-effort way to start a background music session. Here, the stakes felt higher to participants when the resulting music did not match one's mood, potentially even altering into an unwanted state.

Some users do not expect that music recommendation systems can accomplish this level of emotional congruence in the context of NSQs. Below, a participant describes how unexpectedly hearing an album called "Planetarium" as the content returned for an NSQ–music which she describes as having a deeply emotional quality and unpleasant associations–would have a negative impact on her if she wasn't prepared to hear it.

> I would probably be more specific than that because the last thing I want is to be like, 'Play music' and it puts on 'Planetarium' and now I'm on my back on my bed with a jug of ice cream like, 'How did this happen?' - P4, 26

To avoid such situations, she exercises control by asking for specific artists or albums that fit her current mood rather than handing control to the music streaming service.

4.4 Learned User Behaviors

As users become more familiar with the capabilities of voice assistants, they fall fairly quickly into settled routines and habits [5]. We observed similar behaviors with our participants in how they adopted NSQs over time.

4.4.1 Tool to learn about boundaries of the voice assistant

Participants reported making NSQs out of initial curiosity. Consistent with Mennicken et al. [23], we observed that participants perceived NSQs as an opportunity to test the limits of the system and playfully learn the rules for what they can accomplish through their voice-enabled device.

> I just said 'Hey Google play some good music' or something like that just to see what it would do, or I've also just [said] 'play music' before just because I was curious what it would play. - P2, 23

Participants felt that if they received a result that they deemed enjoyable, this would increase the likelihood that they would use these types of queries in the future again. This aligns with prior research in general web search that suggests that users will change their request strategies if their results are unsatisfactory [2, 13].

4.4.2 Simple requests facilitate habit formation

Participants who reported making NSQs frequently developed their mental models of how the system would respond and then learned to integrate NSQs into their daily routine.

> I think I've kind of just made it a habit. Go up the stairs, set your keys down, set the backpack down, talk to [music streaming service], take the shoes off, so I think it's kind of ingrained into that coming home routine. - P8, 36

Similar to developing hypotheses about how the voice assistant would return personalized content, participants also created folk theories of how the system would interact with them depending on prior behavior.

> I expect it to be some default behavior that happens all the time, whether it's resuming whether it's starting from like alphabetical like the stuff I've tagged in my library [...] different stuff like over time that I've gotten used to. - P3, 39

For those participants who were more familiar with possible results of NSQs, a final motivation for issuing NSQs was to resume a previous listening session.

> When I come home from work [and provide an NSQ] it picks up [...] whatever thing I was listening to on my phone, so that's kind of nice. It picks up where you left off. - P8, 36

> For days like that, again, where I would prefer to use [an NSQ], it's when I'm in and out of the Jeep constantly. That way I could just jump in the Jeep, I throw the phone up on the dashboard, I tell the Jeep to start playing the music and I'm already rolling out of the parking lot. If I'm gonna be in the Jeep for an extended period of time like when I'm on my way to work, then I just pick it manually. But, [...] Running errands and stuff where you're constantly in and out, it'd be nice to actually just say, [an NSQ] and it just picks up right where I left off. - P10, 39

5. DISCUSSION

Based on our findings, we discuss (1) how the user experience of NSQs could be improved through a better integration of feedback; and (2) what additional linguistic cues could be considered in the NSQs to better address user needs.

5.1 Design for Feedback

When participants perceived a lack of control over their results, they were less inclined to make NSQs. One possible way to address that concern through design is to provide contextualized results when appropriate. For example, voice output, such as text-to-speech, can be used to give additional information about how the content is personalized. Metadata, such as an artist's collaborators or genre, can be used to feed into text-to-speech contextualization of the results of an NSQ [4].

Our results revealed that participants were willing to make NSQs when they wanted to start a low-effort listening session. However, this was balanced with a reluctance to receive overly surprising results. For users who make NSQs on a device with text-to-speech output, conversational search can help guide users by offering personalized options in a dialogue. This guidance can help balance a listener's need to retain control while minimizing cognitive load [12]. To generate candidate responses in a conversational agent for music discovery and exploration, systems can employ various NLP techniques to take advantage of the semantic relationships between entities found in music corpora and catalogs [26]. This can be helpful for the 'discerning' users, observed by [20], who are not satisfied with their recommendations or users who are willing to put in effort for exploration, as observed by [14, 17].

5.2 Leverage Linguistic Cues

Our findings can also be applied to queries slightly more specific to the ones currently studied. Our observations about NSQs can provide insight into descriptive music queries, such as "Play hip hop" or "Play something calming." While there are indeed categorically 'wrong' answers for descriptive queries such as returning a classic rock song for "Play hip hop," we suggest that expectations of personalization and the trade-off between effort and control remain important motivators for users who make descriptive queries.

Another future research direction related to non-specific language would be requests such as "recommend me something". While this type of request does not specify the type of content, the intent expressed by "recommend" suggests that a user is open to new content or to personalized results. Prior research [10] has uncovered different user goals for discovery-related content, suggesting that fulfillment of these voice requests should be aware of the different user needs behind these recommendation-related requests. Additionally, if the assistant has knowledge of user context such as location or time of day, the fulfillment of parsimonious "recommend me something" utterances can potentially be highly personalized [30, 31].

In summary, our research suggests that user intent when making ambiguous and non-specific requests can vary depending on user tolerance for effort and novelty, and that this tolerance can impact the level of user trust with the system. Future directions include iterating on system design to better support NSQs and further investigating utterance language to better understand user intent.

6. REFERENCES

[1] T. Ammari, J. Kaye, J. Tsai, and F. Bentley. Music, search, and IoT: How people (really) use voice assistants. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 26(3, Article 17):1–28, 2019.

[2] A. Aula, R.M. Khan, and Z. Guan. How does search behavior change as search becomes more difficult? In *Proc. of the 2010 CHI Conference on Human Factors in Computing Systems*, pages 35–44, Atlanta, Georgia, USA, 2010.

[3] D. Bainbridge, S.J. Cunningham, and J.S. Downie. Analysis of queries to a Wizard-of-Oz MIR system: Challenging assumptions about what people really want. In *Proc. of the 4th Int. Society for Music Information Retrieval Conf.*, Baltimore, Maryland, USA, 2003.

[4] M. Behrooz, S. Mennicken, J. Thom, R. Kumar, and H. Cramer. Augmenting music listening experiences on voice assistants. In *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, pages 303–310, Delft, The Netherlands, 2019.

[5] F. Bentley, C. Luvogt, M. Silverman, R. Wirasinghe, B. White, and D. Lottridge. Understanding the long-term use of smart speaker assistants. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3,Article 91), 2018.

[6] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006.

[7] E. Cherian and J. Pounder. Speak easy mindshare report. Technical report, J. Walter Thompson, London, UK, 2017.

[8] M. Eslami, K. Karahalios, C. Sandvig, K. Vaccaro, A. Rickman, K. Hamilton, and A. Kirlik. First I "like" it, then I hide it: Folk theories of social feeds. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2371–2382, San Jose, California, USA, 2016.

[9] A. Furqan, C. Myers, and J. Zhu. Learnability through adaptive discovery tools in voice user interfaces. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1617–1623, Denver, CO, USA, 2017.

[10] J. Garcia-Gathright, B. St. Thomas, C. Hosey, Z. Nazari, and F. Diaz. Understanding and evaluating user satisfaction with music discovery. In *Proc. of the 41st Int. ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 55–64, Ann Arbor, MI, USA, 2018.

[11] I. Guy. Searching by talking: Analysis of voice queries on mobile web search. In *In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 35–44, Pisa, Italy, 2016.

[12] A. N. Hagen. The playlist experience: Personal playlists in music streaming services. *Popular Music and Society*, 38(5):625–645, 2015.

[13] A. Hassan, R.W. White, S.T. Dumais, and Y. Wang. Struggling or exploring?: Disambiguating long search sessions. In *Proceedings of the 7th ACM Int. Conference on Web Search and Data Mining*, pages 53–62, New York, NY, USA, 2014.

[14] C. Hosey, L. Vujović, B. St. Thomas, J. Garcia-Gathright, and J. Thom. Just give me what I want: How people use and evaluate music search. In *Proc. of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 299:1–299:12, Glasgow, Scotland UK, 2019.

[15] P.N. Juslin and J. Sloboda, editors. *Handbook of Music and Emotion: Theory, Research, Applications*. Oxford University Press, Oxford, UK, 2011.

[16] B. Kinsella and A. Mutchler. Smart speaker consumer adoption report. Technical report, 2019.

[17] J. Kiseleva, K. Williams, A. Hassan Awadallah, A.C. Crook, I. Zitouni, and T. Anastasakos. Predicting user satisfaction with intelligent assistants. In *Proc. of the 39th Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 45–54, Pisa, Italy, 2016.

[18] A. Laplante. Users' relevance criteria in music retrieval in everyday life: An exploratory study. In *Proc. of the 11th Int. Society for Music Information Retrieval Conference*, pages 601–606, Utrecht, Netherlands, 2010.

[19] J.H. Lee. Analysis of user needs and information features in natural language queries seeking music information. *Journal of the American Society for Information Science and Technology*, 61(5):1025–1045, 2010.

[20] J.H. Lee and R. Price. User experience with commercial music services: An empirical exploration. *Journal of the Association for Information Science and Technology*, 67(4):800–811, 2016.

[21] A. Li, J. Thom, P. Chandar, C. Hosey, B. St. Thomas, and J. Garcia-Gathright. Search mindsets: Understanding focused and non-focused information seeking in music search. In *Proc. of the World Wide Web Conference*, pages 2971–2977, San Francisco, CA, USA, 2019.

[22] E. Luger and A. Sellen. Like having a really bad PA: the gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5286–5297, San Jose, CA, USA, 2016.

[23] S. Mennicken, R. Brillman, J. Thom, and H. Cramer. Challenges and methods in design of domain-specific voice assistants. In *2018 AAAI Spring Symposia*, Palo Alto, CA USA, 2018.

[24] S.B. Merriam. *Qualitative Research in Practice: Examples for Discussion and Analysis*. Jossey-Bass Inc, Hoboken, NJ, 2002.

[25] C. Myers, A. Furqan, and J. Zhu. The impact of user characteristics and preferences on performance with an unfamiliar voice user interface. In *Proc. of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 47:1–47:9, Glasgow, Scotland UK, 2019.

[26] S. Oramas, L.E. Anke, F. Gómez, and X. Serra. Natural language processing for music knowledge discovery. *CoRR*, abs/1807.02200, 2018.

[27] M. Park, J. Thom, S. Mennicken, H. Cramer, and M. Macy. Global music streaming data reveal diurnal and seasonal patterns of affective preference. *Nature Human Behaviour*, 3:230–236, January 2019.

[28] Edison Research and NPR. The smart audio report. Technical report, Somerville, NJ, 2019.

[29] S. Saarikallio and J. Erkkilä. The role of music in adolescents' mood regulation. *Psychology of Music*, 35(1):88–109, January 2007.

[30] M. Schedl and A. Flexer. Putting the user in the center of music information retrieval. In *Proc. of the 13th Int. Society for Music Information Retrieval Conference*, pages 385–390, Porto, Portugal.

[31] M. Schedl, A. Flexer, and J. Urbano. The neglected user in music information retrieval research. *Journal of Intelligent Information Systems*, 41(3):523–539, December 2013.

[32] A. Springer and H. Cramer. "Play PRBLMS": Identifying and correcting less accessible content in voice interfaces. In *Proc. of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*, pages 296:1–296:13, Montreal, Quebec, 2018. ACM.

[33] M. Zentner, D. Grandjean, and K.R. Scherer. Emotions evoked by the sound of music: Characterization, classification, and measurement. *Emotion*, 8(4):494–521, August 2008.

LEARNING INTERPRETABLE REPRESENTATION FOR CONTROLLABLE POLYPHONIC MUSIC GENERATION

Ziyu Wang   **Dingsu Wang**   **Yixiao Zhang**   **Gus Xia**

Music X Lab, Computer Science Department, NYU Shanghai

`{ziyu.wang, dingsu.wang, yixiao.zhang, gxia}@nyu.edu`

ABSTRACT

While deep generative models have become the leading methods for algorithmic composition, it remains a challenging problem to *control* the generation process because the latent variables of most deep-learning models lack good interpretability. Inspired by the content-style disentanglement idea, we design a novel architecture, under the VAE framework, that effectively learns two interpretable latent factors of polyphonic music: chord and texture. The current model focuses on learning 8-beat long piano composition segments. We show that such chord-texture disentanglement provides a controllable generation pathway leading to a wide spectrum of applications, including compositional style transfer, texture variation, and accompaniment arrangement. Both objective and subjective evaluations show that our method achieves a successful disentanglement and high quality controlled music generation.[1]

1. INTRODUCTION

With the development of artificial neural networks, deep learning has become one of the most popular techniques for automated music generation. In particular, we see recurrent and attention-based models being able to generate creative and human-like music without heavily hand-crafted rules [1–3]. However, the main drawback of these deep generative models is that they behave like "black boxes", and it is difficult to interpret the musical meaning of their internal latent variables [4]. Consequently, it remains a challenging task to control the generation process (i.e., to guide the music flow by manipulating the high-level compositional factors such as melody contour, accompaniment texture, style, etc.). This limitation restricts the application scenario of the powerful deep generative models.

In this paper, we improve the model interpretability for music generation via constrained representation learning. Inspired by the content-style disentanglement idea [5],

we enforce the model to learn two fundamental factors of polyphonic music: *chord* (content) and *texture* (style). The former refers to the representation of the underlying chord progression, and the latter includes chord arrangement, rhythmic pattern, and melody contour. The current design focuses on learning 8-beat long piano composition segments under a variational autoencoder (VAE) framework.

The core of the model design lies in the encoder. We incorporate the encoder with two **inductive biases** for a successful **chord-texture disentanglement**. The former applies a rule-based chord recognizer and embeds the information into the first half of the latent representation. The latter regards music as 2-D images and uses a chord-invariant convolutional network to extract the texture information, storing it into the second half of the latent representation. As for the decoder, we adopt the design from PianoTree VAE [6], an architecture that can reconstruct polyphonic music from the latent representation in a hierarchical manner.

We further show that the interpretable representations are **general-purpose**, empowering a wide spectrum of controllable music generation. In this study, we explore the following three scenarios:

Task 1: Compositional style transfer by swapping the chord and texture factors of different pieces of music, which can help us re-harmonize or re-arrange a music piece following the style of another piece.

Task 2: Texture variation by sampling the texture factor while keeping the chord factor, which is analogous to the creation of "Theme and Variations" form of composition.

Task 3: Accompaniment arrangement by predicting the texture factor given the melody using a downstream encoder-decoder generative model.

In sum, the contributions of our paper are as follows:

- We design a representation disentanglement method for polyphonic music, which learns two interpretable factors: chord and texture.

- We show that the interpretable factors are general-purpose features for controllable music generation, which reduces the necessity to design heavily-engineered control-specific model architectures. As far as we know, this is the first attempt to explicitly con-

---

[1] Code and demos can be accessed via `https://github.com/ZZWaang/polyphonic-chord-texture-disentanglement`

trol the compositional texture feature for symbolic polyphonic music generation.

- We demonstrate that control methods are effective and the quality of generated music is high. Some style transferred pieces are rated even higher than the original ones composed by humans.

2. RELATED WORK

We review two techniques of automated music generation related to our paper: controlled generation (in Section 2.1) and representation disentanglement (in Section 2.2). For a more general review of deep music generation, we refer readers to [7, 8].

2.1 Controlled Music Generation

Most existing learning-based methods regard controlled music generation a *conditional estimation* problem. That is, to model $p(\text{music}|\text{control})$, in which both music and control are usually time-series features. Another approach that is closely related to conditional estimation is to first learn the joint distribution $p(\text{music}, \text{control})$ and later on *force* the value of control during the generation process.

The above two methods have been used in various tasks, including generating chords based on the melody [9], creating the melody based on the chords [10, 11], completing the counterparts or accompaniment based on the melody or chord [3, 12–16], and producing the audio waveform based on timbre features [17, 18].

However, many abstract music factors, such as texture and melody contour, could hardly be explicitly coded by labels. Even if such labels are provided, the control still does not allow continuous manipulation, such as sampling and interpolation. Consequently, it remains a challenging task to control music by more abstract factors without complex heuristics [19].
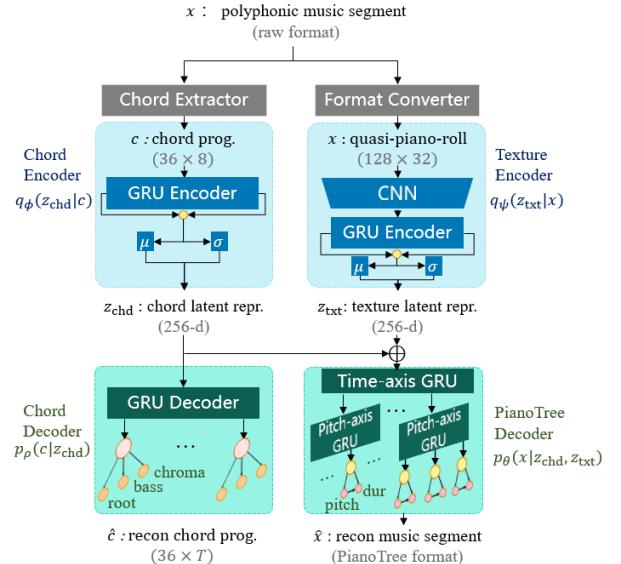
2.2 Music Representation Disentanglement

Learning disentangled representations is an ideal solution to the problem above, since: 1) representation learning embeds discrete music and control sequences into a continuous latent space, and 2) disentanglement techniques can further decompose the latent space into interpretable subparts that correspond to abstract music factors. Recent studies show that VAEs [20, 21] are in general an effective framework to learn the representations of discrete music sequences, and the key to a successful disentanglement is to incorporate proper inductive biases into the representation learning models [22].

Under a VAE framework, an inductive bias can be realized in various forms, including constraining the encoder [23–25], constraining the decoder [26], imposing multitask loss functions [27, 28], and enforcing transformation invariant results during the learning process [29, 30]. This study is based on our previous work Deep Music Analogy [27] in which we disentangle pitch and rhythm factors for monophonic segments. We extend this idea to polyphonic composition while the model design is more similar to [24].

3. MODEL

In this section, we introduce the model design and data representation in detail. The goal is to learn the representations of 8-beat long piano compositions (with $\frac{1}{4}$ beat as the shortest unit) and disentangle the representations into two interpretable factors: chord and texture.

Figure 1: The model diagram.

Figure 1 shows the overall architecture of the model. It adopts a VAE framework and contains four parts: 1) a chord encoder, 2) a chord decoder, 3) a texture encoder, and 4) a PianoTree decoder. The chord encoder and chord decoder can be seen as a standalone VAE which extracts the latent chord representation z_{chd}. On the other hand, the texture encoder aims to extract the texture representation z_{txt} using a chord-invariant convolutional mapping. Finally, the PianoTree decoder takes in both z_{chd} and z_{txt} and outputs the original music in a tree-structured data format.

3.1 Chord Encoder

The chord encoder first applies rule-based methods [31,32] to extract the chord progression under one-beat resolution. Each extracted chord progression is a 36 by 8 matrix, where each column denotes a chord of one beat. Each chord is a 36-D vector consisting of three parts: a 12-D one-hot vector for the pitch class of the *root*, a 12-D one-hot vector for the *bass*, and a 12-D multi-hot *chroma* vector.

The chord progression is then fed into a bi-directional GRU encoder [21], and the last hidden states on both ends of the GRU are concatenated and used to approximate the posterior distribution of z_{chd}. Following the assumption of a standard VAE, z_{chd} has a standard Gaussian prior and follows an isotropic Gaussian posterior.

Note that although the chord progression here is extracted using algorithms, it can also be provided by external labels, in which case the whole model becomes a conditional VAE [33].

3.2 Chord Decoder

The chord decoder reconstructs the chord progression from z_{chd} using another bi-directional GRU. The reconstruction loss of a chord progression is computed as a summation of 8 beat-wise chord loss using cross entropy functions [34]. For each beat, the chord loss is defined as the product of three parts: 1) the root loss, 2) the bass loss, and 3) the chroma loss. The root and bass are both considered 12-way categorical distributions and the chroma is regarded as 12 independent Bernoulli distributions.

3.3 Texture Encoder

The input of the texture encoder is an 8-beat segment of polyphonic piece represented by an image-like data format slightly modified from the piano-roll [14]. Each 8-beat segment is represented by a 128 by 32 matrix, where each row corresponds to a MIDI pitch and each column corresponds to $\frac{1}{4}$ beat. The data entry at (p, t) records the duration of the note if there is a note onset, and zero otherwise.

The texture encoder aims to learn a chord-invariant representation of texture by leveraging both the translation invariance property of convolution and the blurry effect of max-pooling layers [35]. We use a convolutional layer with kernel size 12×4 and stride 1×4, which is followed by a ReLU activation [36] and max-pooling with kernel size 4×1 and stride 4×1. The convolutional layer has one input channel and 10 output channels. The convolutional layer design aims at extracting a blurry "concept sketch" of the polyphonic texture which contains minimum information of the underlying chord. Ideally, when such blurry sketch is combined with specific chord representation, the decoder can identify its concrete pitches in a musical way.

The output of the convolutional layer is then fed into a bi-directional GRU encoder to extract the texture representation z_{txt}, similar to how we encode z_{chd} introduced in Section 3.1.

3.4 PianoTree Decoder

The PianoTree decoder takes the concatenation of z_{chd} and z_{txt} as input and decodes the music segment using the same decoder structure invented in PianoTree VAE [6], a hierarchical model structure for polyphonic representation learning. The decoder works as follows. First, it generates 32 frame-wise hidden states (one for each $\frac{1}{4}$ beat) using a GRU layer. Then, each frame-wise hidden state is further decoded into the embeddings of individual notes using another GRU layer. Finally, the pitch and duration for each note are reconstructed from the note embedding using a fully-connected layer and a GRU layer, respectively. For more detailed derivation and model design, we refer the readers to [6].

3.5 Training Objective

Let x denote the input music piece and $c = f(x)$ denote the chord progression extracted by algorithm $f(\cdot)$. We assume standard Gaussian priors of $p(z_{\text{chd}})$ and $p(z_{\text{txt}})$, and denote the output posteriors of chord encoder and texture encoder by $q_\phi(z_{\text{chd}}|c)$, $q_\psi(z_{\text{txt}}|x)$, the output distributions of chord decoder and PianoTree decoder by $p_\rho(c|z_{\text{chd}})$ and $p_\theta(x|z_{\text{chd}}, z_{\text{txt}})$. The objective of the model is:

$$
\begin{aligned}
\mathcal{L}(\phi, \psi, \rho, \theta; x) = \\
- \mathbb{E}_{\substack{z_{\text{chd}} \sim q_\phi \\ z_{\text{txt}} \sim q_\psi}} \big[\log p_\rho(c|z_{\text{chd}}) + \log p_\theta(x|z_{\text{chd}}, z_{\text{txt}}) \big] \\
+ \text{KL}(q_\phi || p(z_{\text{chd}})) + \text{KL}(q_\psi || p(z_{\text{txt}})).
\end{aligned} \tag{1}
$$

4. CONTROLLED MUSIC GENERATION

In this section, we show some controlled generation examples of the three tasks mentioned in the introduction.

4.1 Compositional Style Transfer

By regarding chord progression *content* and texture *style*, we can achieve compositional style transfer by swapping the texture representations of different pieces. Figure 2 shows the transferred results ((c) & (d)) based on two 16-bar samples ((a) & (b)) in the test set by swapping z_{txt} every 2 bars (without overlap).[2]

We see that such long-term style transfer is successful: The generated segment (c) follows the chord progression of (b) while mimicking the texture of (a), while (d) follows the chord progression of (a) while mimicking the texture of (b). As shown in the marked scores, the style transfer is effective. E.g., the cut-offs, melody contours, and the shape of the left-hand accompaniment are all preserved.

4.2 Texture Variation by Sampling

We can make variations of texture by sampling from z_{txt} while keeping z_{chd}. Here, we investigate two sampling strategies: sampling from the posterior $q_\psi(z_{\text{txt}}|x)$, and sampling from the prior $p(z_{\text{txt}})$.

Sampling from the posterior distribution $q_\psi(z_{\text{txt}}|x)$ yields reasonable variations as shown in Figure 3a. The variations of the right-hand melody can be seen as an improvisation following the chord progression and the melody. On the contrary, there is only small variation in the left-hand part, showing that the model regards the left-hand accompaniment as the dominant feature of texture.

Sampling from the prior distribution $p(z_{\text{txt}})$ changes the texture completely. Figure 3b shows a series of examples of prior sampling under the same chord progression C-Am-F-G. The resulting generations follow exactly the chord progression but with new textures.

4.3 Accompaniment Arrangement

We use a downstream predictive model to achieve accompaniment arrangement. For this task, we provide extra vocal melody tracks paired with the piano samples, and the model learns to generate 16-bar piano accompaniment *conditioned* on melody in a supervised fashion.

We encode the music every 2 bars (without overlap) into latent representations. For the accompaniment, we use the proposed model to compute the latent chord and texture representation, denoted by $\mathbf{z}_{\text{chd}} = [z_{\text{chd}}^{(1)}, ..., z_{\text{chd}}^{(4)}]$ and $\mathbf{z}_{\text{txt}} = [z_{\text{txt}}^{(1)}, ..., z_{\text{txt}}^{(4)}]$. For the melody, we use the

---

[2] The presented excerpts are converted from MIDI by the authors. The chord labels are inferred from the original/generated samples.

(a) A real piece.

(b) The other real piece.

(c) The generated piece by combining z_{txt} from (a) and z_{chd} from (b).
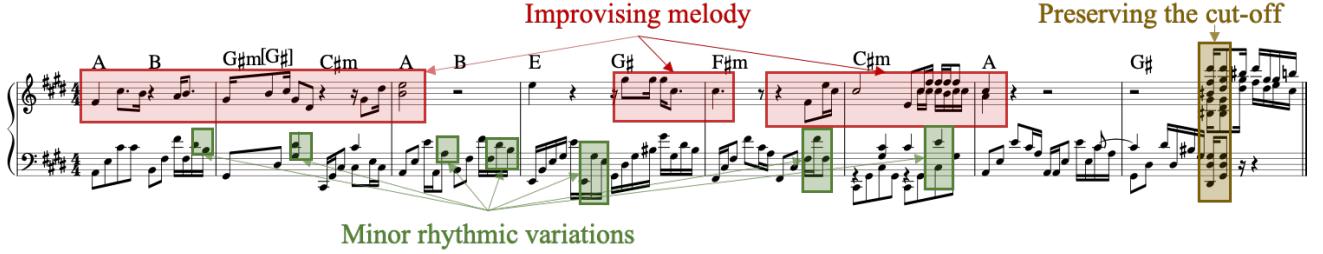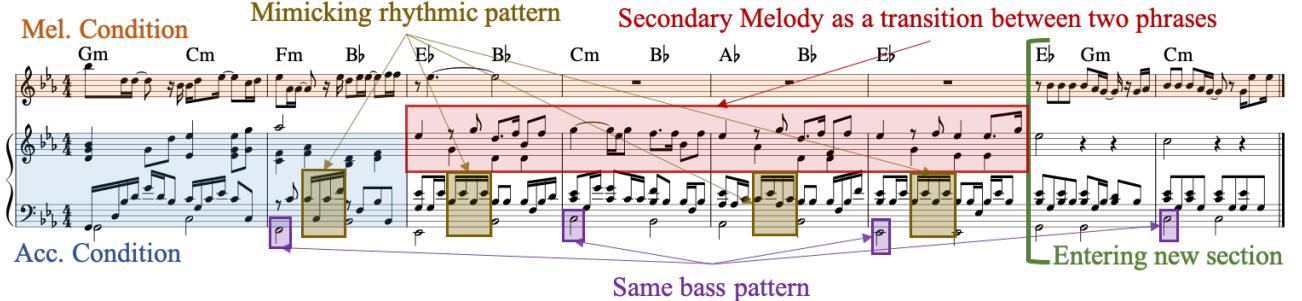
(d) The generated piece by combining z_{txt} from (b) and z_{chd} from (a).

Figure 2: An example of compositional style transfer of 16-bar-long samples when $k = 2$.

EC^2-VAE [27] to compute the latent pitch and rhythm representations, denoted by $\mathbf{z}_{\text{p}} = [z_{\text{p}}^{(1)}, ..., z_{\text{p}}^{(4)}]$ and $\mathbf{z}_{\text{r}} = [z_{\text{r}}^{(1)}, ..., z_{\text{r}}^{(4)}]$. Then, we adopt a vanilla Transformer [37] to model $p(\mathbf{z}_{\text{txt}}, \mathbf{z}_{\text{chd}} | \mathbf{z}_{\text{p}}, \mathbf{z}_{\text{r}})$, in which the encoder takes in the condition and the decoder's input is a shifted right version $[\mathbf{z}_{\text{chd}}, \mathbf{z}_{\text{txt}}]$. Both encoder and decoder inputs are incorporated with a *positional encoding* indicating the time po-

sitions and a learned *factor embedding* indicating the representation type (i.e., pitch, rhythm, chord or texture).

Figure 4 shows an example of accompaniment arrangement, where the first staff shows the melody and the second staff shows the piano accompaniment. In this case, the whole melody, together with the complete chord progression and the first 2 bars of accompaniment are given. The chord conditioning is done by forcing the decoded chord

(a) An example of posterior sampling of z_{txt} of the first 8 bars of the segment (a) in Figure 2

(b) An example of prior sampling of z_{txt} under given chord progression C-Am-F-G. Each two-bar segment is independently sampled, having different texture.

Figure 3: Examples of texture variations via posterior sampling and prior sampling.

Figure 4: An example of accompaniment arrangement conditioned on melody, chord progression, and first 2 bars of accompaniment.

representation to match the given input during inference time. (A similar trick is used in [15].) From Figure 4, we see that the model predicts a similar texture to the given accompaniment. Moreover, it fills in a secondary melody line as a transition when the lead melody is rest.

Note that the arrangement can be generated in a flexible way by conditioning on different sets of latent factors. Much longer examples and more conditioning settings are available on our github page.

5. EXPERIMENTS

5.1 Dataset and Training

We train our model on the POP909 dataset [38], which contains about 1K MIDI files of pop songs (including paired vocal melody and piano accompaniment). We further extract the chord annotations using [31, 32]. We only keep the pieces with $\frac{2}{4}$ and $\frac{4}{4}$ meters and cut them into 8-beat music segments (so that each data sample in our experiment contains 32 time steps under 16th note resolution). In all, we have 66K samples. We randomly split the dataset (at song-level) into training set (90%) and test set (10%). All training samples are further augmented by transposing to all 12 keys.

In our experiment, the VAE model uses 256, 512, and 512 hidden dimensions for the GRUs in chord encoder, chord decoder and texture encoder respectively. The latent dimension of z_{chd} and z_{txt} are both 256. The model size of the PianoTree decoder is the same as the implementation

in the original paper [6]. The transformer model has the following size: hidden dimension $= 256$, number of layers $= 4$ and number of heads $= 8$.

For both models, we use Adam optimizer [39] with a scheduled learning rate from 1e-3 to 1e-5. Moreover, for the VAE model, we use KL-annealing [40], i.e. setting a weight parameter for the KL-divergence loss starting from 0 to 0.1. We set batch size to be 128 and the training converges within 6 epochs. For the downstream transformer model, we use 12K warm-up steps for learning rate update [41]. We use the same batch size and the model converges within 40 epochs.
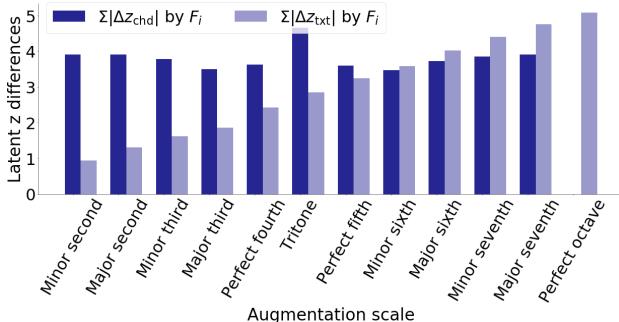
5.2 Objective Measurement

When z_{chd} and z_{txt} are well disentangled, small variations over the note pitches of the original music should lead to a larger change on z_{chd}, while variations of rhythm will influence more on z_{txt}. Following this assumption, we adopt a *disentanglement evaluation via data augmentation* method used in [42] and further developed in [27].
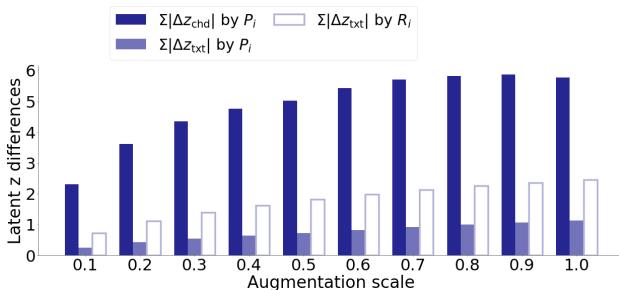
We define F_i as the operation of transposing all the notes by i semitones, and use the L_1-norm to measure the change of latent z after augmentation. Figure 5a shows a comparison between $\Sigma|\Delta z_{chd}|$ and $\Sigma|\Delta z_{txt}|$ when we apply F_i to all the music pieces in the test set (where $i \in [1, 12]$).

It is conspicuous that when augmenting pitch in a small range, the change of z_{chd} is much larger than the change of z_{txt}. At the same time, the change of z_{txt} gets higher as the

augmentation scale increases. Similar to the result in [27], the change of z_{chd} reflects human pitch perception as z_{chd} is very sensitive to a tritone transposition, and least sensitive for a perfect octave.

(a) A comparison between Δz_{chd}, Δz_{txt} after pitch transposition on all notes.

(b) A comparison among Δz_{chd}, Δz_{txt} after beat-wise pitch transposition and texture augmentation with different probabilities.

Figure 5: Results of objective measurement.
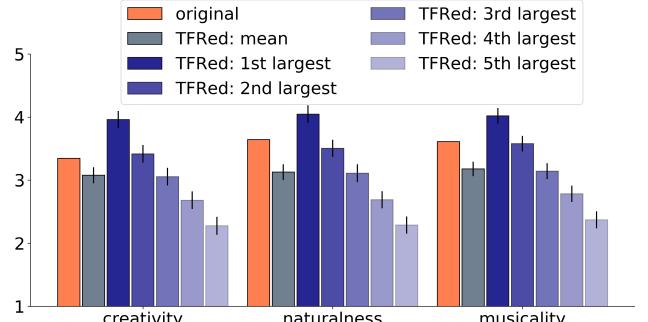
We further define P_i as the function to randomly transpose all the notes in one beat either up or down one semitone under a certain probability i, and R_i as the function to randomly reduce the note duration by half. Figure 5b shows a comparison between $\Sigma|\Delta z_{\text{chd}}|$ and $\Sigma|\Delta z_{\text{txt}}|$ when we apply P_i and R_i to all the music pieces in our test set (where $i \in [0.1, 1.0]$).

For each value of i in the figure 5b, the first and second bars demonstrate $\Sigma|\Delta z_{\text{chd}}|$ and $\Sigma|\Delta z_{\text{txt}}|$ caused by P_i function, while the third bar indicates $\Sigma|\Delta z_{\text{txt}}|$ caused by R_i function. (We did not show $\Sigma|\Delta z_{\text{chd}}|$ caused by R_i since they are all zero.) It again proves that the chord representation is more sensitive than texture representation under pitch variations, and conversely, texture representation is more sensitive than chord representation under rhythm variations.

5.3 Subjective Evaluation

Besides objective measurement, we conduct a survey to evaluate the musical quality of compositional style transfer (see Section 4.1). Each subject listens to ten 2-bar pieces with different chord progressions, each paired with 5 style-transfer versions generated by swapping the texture representation with a random sample from the test set. In other words, each subject evaluates 10 groups of samples, each of which contains 6 versions of textures (1 from the original piece and 5 from other pieces) under the same chord progression. Both the order of groups and the sample order

within each group are randomized. After listening to each sample, the subjects rate them based on a 5-point scale from 1 (very low) to 5 (very high) according to three criteria: *creativity*, *naturalness* and *musicality*.

Figure 6: Subjective evaluation results. Here "TFRed: xth largest" denotes the xth (largest) order statistic of the transferred segments.

A total of 36 subjects (26 females and 10 males) participated in the survey. Figure 6 shows the comparison result among the original pieces (indicated by the orange bars) and the transferred pieces in terms of their mean and *order* statistics. The heights of bars represent averaged ratings across the subjects and the error bars represent the confidence intervals computed via paired t-test [43]. The result shows *if we randomly transfer a piece's texture 5 times, the best result is significantly better than the original version (with p-value < 0.005)*, and there are only marginal differences between the second-largest statistics and the original (with p-value > 0.05) in terms of creativity and musicality. We also see that on average the transferred results are still rated lower than the original ones. How to automatically decide the quality of a transferred result is considered a future work.

6. CONCLUSION AND FUTURE WORK

In conclusion, we contributed an effective algorithm to disentangle polyphonic music representation into two interpretable factors, chord and texture, under a VAE framework. Such interpretable representations serve as an intuitive human-computer co-creation interface, by which we can precisely manipulate individual factors to control the flow of the generated music. In this paper, we demonstrated three ways to interact with the model, including compositional style transfer via swapping the latent codes, texture variation by sampling from the latent distribution, accompaniment arrangement using downstream conditional prediction, and there are potentially many more. We hope this work can shed light on the field of controllable algorithmic composition in general, especially on the paradox between model complexity and model interpretability.

We acknowledge that the learned music factors are still very basic. In the future, we plan to extract more abstract and longer-range features using hierarchical models. We also plan to explore more ways to control the music generation for practical usage.

7. REFERENCES

[1] K. Chen, W. Zhang, S. Dubnov, G. Xia, and W. Li, "The effect of explicit structure encoding of deep neural networks for symbolic music generation," in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP).* IEEE, 2019, pp. 77–84.

[2] C. A. H. et al., "Music transformer: Generating music with long-term structure," in *7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA*, 2019.

[3] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Generating music with rhythm and harmony," *arXiv preprint arXiv:2002.00212*, 2020.

[4] J.-P. Briot and F. Pachet, "Deep learning for music generation: challenges and directions," *Neural Computing and Applications*, vol. 32, no. 4, pp. 981–993, 2020.

[5] S. Dai, Z. Zhang, and G. G. Xia, "Music style transfer: A position paper," *arXiv preprint arXiv:1803.06841*, 2018.

[6] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, and G. Xia, "Pianotree vae: Structured representation learning for polyphonic music," in *Proceedings of 21st International Conference on Music Information Retrieval (ISMIR), virtual conference*, 2020.

[7] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep learning techniques for music generation–a survey," *arXiv preprint arXiv:1709.01620*, 2017.

[8] J.-P. Briot, "From artificial neural networks to deep learning for music generation–history, concepts and trends," *arXiv preprint arXiv:2004.03586*, 2020.

[9] I. Simon, D. Morris, and S. Basu, "Mysong: automatic accompaniment generation for vocal melodies," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2008, pp. 725–734.

[10] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.

[11] K. Chen, W. Zhang, S. Dubnov, G. Xia, and W. Li, "The effect of explicit structure encoding of deep neural networks for symbolic music generation," in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP).* IEEE, 2019, pp. 77–84.

[12] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR. org, 2017, pp. 1362–1371.

[13] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2837–2846.

[14] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[15] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," *arXiv preprint arXiv:1907.04868*, 2019.

[16] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, "Learning a latent space of multitrack measures," *arXiv preprint arXiv:1806.00195*, 2018.

[17] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer," *arXiv preprint arXiv:1811.09620*, 2018.

[18] O. Kwon, I. Jang, C. Ahn, and H.-G. Kang, "Emotional speech synthesis based on style embedded tacotron2 framework," in *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC).* IEEE, 2019, pp. 1–4.

[19] S. Lattner, M. Grachten, and G. Widmer, "Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints," *arXiv preprint arXiv:1612.04742*, 2016.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," *arXiv preprint arXiv:1803.05428*, 2018.

[22] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," *arXiv preprint arXiv:1811.12359*, 2018.

[23] Y.-J. Luo, K. Agres, and D. Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders," *arXiv preprint arXiv:1906.08152*, 2019.

[24] Y. Wu, T. Carsault, E. Nakamura, and K. Yoshii, "Semi-supervised neural chord estimation based on a variational autoencoder with discrete labels and continuous textures of chords," *arXiv preprint arXiv:2005.07091*, 2020.

[25] T. Akama, "Controlling symbolic music generation based on concept learning from domain knowledge." in *ISMIR*, 2019, pp. 816–823.

[26] K. Choi and K. Cho, "Deep unsupervised drum transcription," *arXiv preprint arXiv:1906.03697*, 2019.

[27] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," *arXiv preprint arXiv:1906.03626*, 2019.

[28] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer," *arXiv preprint arXiv:1809.07600*, 2018.

[29] S. Lattner, M. Dörfler, and A. Arzt, "Learning complex basis functions for invariant representations of audio," *arXiv preprint arXiv:1907.05982*, 2019.

[30] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun, "Disentangling factors of variation in deep representation using adversarial training," in *Advances in neural information processing systems*, 2016, pp. 5040–5048.

[31] B. Pardo and W. P. Birmingham, "Algorithms for chordal analysis," *Computer Music Journal*, vol. 26, no. 2, pp. 27–49, 2002.

[32] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.

[33] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European Conference on Computer Vision*. Springer, 2016, pp. 776–791.

[34] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[36] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[38] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proceedings of 21st International Conference on Music Information Retrieval (ISMIR), virtual conference*, 2020.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[41] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T.-Y. Liu, "On layer normalization in the transformer architecture," *arXiv preprint arXiv:2002.04745*, 2020.

[42] H. Kim and A. Mnih, "Disentangling by factorising," *arXiv preprint arXiv:1802.05983*, 2018.

[43] H. Hsu and P. A. Lachenbruch, "Paired t test," *Encyclopedia of Biostatistics*, vol. 6, 2005.

ATTRIBUTES-AWARE DEEP MUSIC TRANSFORMATION

Lisa Kawai[1] **Philippe Esling**[2] **Tatsuya Harada**[1,3]

[1] The University of Tokyo, Japan [2] IRCAM, France [3] RIKEN, Japan

`{kawai, harada}@mi.t.u-tokyo.ac.jp, philippe.esling@ircam.fr`

ABSTRACT

Recent machine learning techniques have enabled a large variety of novel music generation processes. However, most approaches do not provide any form of interpretable control over musical attributes, such as pitch and rhythm. Obtaining control over the generation process is critically important for its use in real-life creative setups. Nevertheless, this problem remains arduous, as there are no known functions nor differentiable approximations to transform symbolic music with control of musical attributes.

In this work, we propose a novel method that enables attributes-aware music transformation from any set of musical annotations, without requiring complicated derivative implementation. By relying on an adversarial confusion criterion on given musical annotations, we force the latent space of a generative model to abstract from these features. Then, reintroducing these features as conditioning to the generative function, we obtain a continuous control over them. To demonstrate our approach, we rely on sets of musical attributes computed by the jSymbolic library as annotations and conduct experiments that show that our method outperforms previous methods in control. Finally, comparing correlations between attributes and the transformed results show that our method can provide explicit control over any continuous or discrete annotation.

1. INTRODUCTION

For a long time, music composition has been considered a skill reserved for highly-trained experts. However, since the emergence of new technologies, it appears possible for non-expert and untrained users to indulge in this task. One such way is to rely on machine learning models, which train on existing sets of music to produce pieces with similar characteristics. The importance of music generation is growing in the industrial world as well, for instance, in the generation of soundtracks for video games and movies [1, 2].

In this context, models that provide control over the attributes of the generated music are of critical importance [1, 3–5]. Indeed, the simple generation of "self-contained" music through a single button rapidly becomes dull. Instead, users are rather looking to transform aspects of the music to make it fit their intent in terms of musical attributes such as pitch, rhythm, and melody. Ideally, these controls should be continuous, providing flexibility akin to the *knobs* of musical synthesizers. For instance, if a user wants to generate music with longer notes, the system should provide a way to decide *how much* the notes are extended, not just whether extend them or not.

Nevertheless, obtaining explicit controls over music generation is still an open and complex problem [5]. Although recent generative models can produce a large variety of music genres, they usually do not provide any mechanism on *how to modify* the generation with musical attributes in a controllable way. However, acquiring such controls seems to require the implementation of complicated derivatives or approximation for each attribute [6].

In this paper, we introduce a novel method to generate symbolic music similar to a given dataset, while being able to control its musical attributes continuously. Our method can be trained using raw annotations, without requiring access to their computational functions nor derivative implementations. This allows us to rely on any form of musical annotations, even high-level semantic or subjective characteristics. To demonstrate this aspect, we rely on attribute vectors computed by jSymbolic [7] as annotations to make the model learn a continuous control for each. This produces numerical descriptions of music, such as pitch ranges and mean note durations, which enables us to learn understandable control over the generation.

To empower our generative model with continuous control over non-differentiable musical attributes, we rely on adversarial learning. Our model is based on a latent encoder-decoder model, where the decoding function takes continuous musical attributes as conditioning information. As the model is trained to reconstruct input data, the latent encoding potentially contains all the information necessary for reconstruction. Hence, this would lead to a decoding function that can generate the output without reflecting changes in the conditioning. Aiming to prevent this situation, we introduce an adversarial discriminator on the latent space of the model. The role of this discriminator is to drive the encoder to remove any attribute information from the latent vector. This adversarial framework with an encoder-decoder model is similar to Fader Networks [8]. However, one major difference is that we handle continuous values instead of binary ones. Hence, a major contribution of this paper is to define an approach to train an adversarial discriminator on continuous values. Indeed, discriminators across the literature [8–15] predict

binary values such as real/fake or with/without attributes, whereas we aim to deal with continuous values. We solve this problem in two steps. First, we quantize attribute values and compute balanced class labels. Second, we extend the discriminator so that it relies on multivariate class vectors. As usual in adversarial training, the encoder is trained to prevent the discriminator from predicting the labels correctly. This produces continuous controls over the musical characteristics by shifting the condition to the decoder. This behavior stems from the need for the model to correctly utilize the decoder condition so as to reconstruct the input data. That way, we force the generated data to reflect the musical control values, for any attribute annotation.

For the purpose of evaluation, we demonstrate the ability to gain interpretable control over musical attributes by conducting extensive experiments. We show that our method outperforms previous proposals [6, 16, 17] and also allows us to rely on unconstrained sets of attributes. We evaluate these results by computing the correlation between control attributes and generation results. This enables us to evaluate quantitatively how the generation reflects the control. Our contributions are: (1) We propose a novel training framework to learn from ordered annotation values. (2) We solve the above problem by quantizing annotations and adversarial training to obtain controllability of musical attributes without their derivatives.

2. RELATED WORK

2.1 Generative Models

Generative models allow production of new data samples $\tilde{x}$, by training on a collection of examples x, with the most popular approaches being Generative Adversarial Networks [9] and Variational Auto-Encoder (VAE) [18, 19].

2.1.1 Variational Auto-Encoder

The VAE is based on an encoder-decoder architecture. The encoder takes input data $\mathbf{x} \in \mathbb{R}^{d_{\mathbf{x}}}$ and outputs a compressed latent vector $\mathbf{z} \in \mathbb{R}^{d_{\mathbf{z}}}$. The decoder takes this latent vector $\mathbf{z}$ as input and tries to reconstruct the input data $\mathbf{x}$. Hence, this approach models two distributions: $\mathbf{z} \sim p_{enc}(\mathbf{z} \mid \mathbf{x}, \theta_{enc})$ and $\tilde{\mathbf{x}} \sim p_{dec}(\mathbf{x} \mid \mathbf{z}, \theta_{dec})$, where θ_{enc} and θ_{dec} are parameters of the encoder and the decoder. The original objective function for training a VAE approximates the distribution $p(\mathbf{x})$, that we wish to model by a lower bound (Evidence Lower Bound (ELBO)) [18, 19].

$$\text{ELBO}_{\{\theta_{enc}, \theta_{dec}\}} = -(\mathcal{L}_R + \mathcal{L}_{KL}) \leq \log p(\mathbf{x}) \quad (1)$$

Here, $\mathcal{L}_R$ and $\mathcal{L}_{KL}$ are mean reconstruction error and Kullback-Leibler divergence, respectively.

$$\mathcal{L}_R = -\mathbb{E}_{p_{enc}(\mathbf{z}|\mathbf{x})}\big[\log(p_{dec}(\mathbf{x} \mid \mathbf{z}))\big] \quad (2)$$
$$\mathcal{L}_{KL} = D_{KL}\big(p_{enc}(\mathbf{z} \mid \mathbf{x}) \,||\, p(\mathbf{z})\big) \quad (3)$$

Therefore, the model optimizes its parameters by minimizing the reconstruction error $\mathcal{L}_R$, while regularizing the latent space through $\mathcal{L}_{KL}$, so that the encoded latent variables match with a Gaussian prior.

2.1.2 Attribute Control in Generative Models

Several works have focused on attribute control for generative models in the image domain [8, 20–23], which enable changing properties of the generation, such as facial attributes. In some cases [21], learning is not performed on the attributes themselves, but rather through a rich pre-trained model. This allows computing the mean directions of interpolation, which maximize attribute change in feature space, in order to re-apply the obtained direction.

In Fader Networks [8], an encoder-decoder-based model is trained to generate facial images given binary visual attributes through adversarial learning. As the decoder is expected to rely on control attributes for generation, this requires that the other latent vector input does not contain any information about the attributes. Thus, they introduce a discriminator on those latent vectors, which tries to classify the input latent vectors based on target binary attributes, such as wearing glasses or not. The encoder is simultaneously trained to prevent the discriminator from predicting target attributes correctly. Therefore, this adversarial criterion enables the model to learn an attributes-invariant representation by forcing the encoder to remove any attribute information from the latent vector. In a similar way, [20, 22] also proposed to gain attributes control by removing attribute information from a latent vector.

Given the success of these methods for facial image-specific components, our proposed model explores a similar approach. However, our model needs to train on continuous ordered values instead of binary attributes.
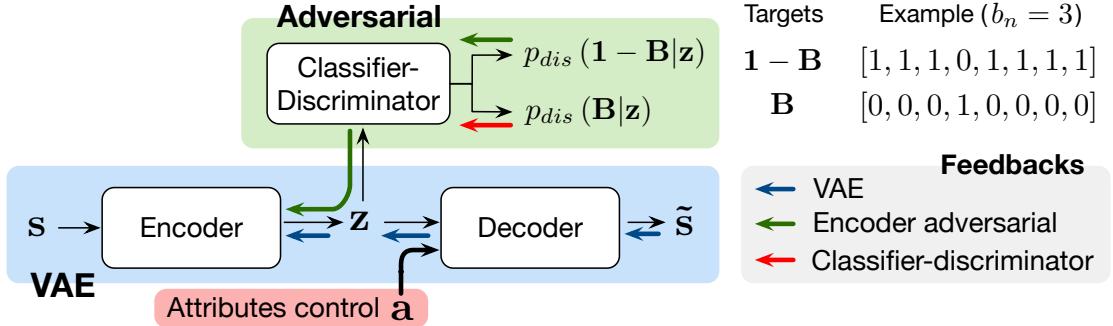
2.2 Controllable Music Generation

Several works have been focusing on music generation conditioned on composer styles [1, 24] or chord progression [10]. In contrast, our work focuses on the direct and continuous transformation of music.

2.2.1 Music Interpolation

In most music interpolation approaches, the models do not rely on annotations while training [16, 17]. Instead, they interpolate musical data by moving the latent vector in different directions. These directions are usually defined by reference points from the training set. In some cases, musical annotations are also used to learn the characteristics or certain styles of music, or even sequences of features [4, 25], which mean the models have annotation values in every time step. For instance, [4] relies on contour and fragmentation & consolidation sequences in addition to rhythmic ones, and they split the latent vector between a target attribute and the other unsupervised dimensions. Although these models are able to account for target attributes, their precise control remains unclear. Moreover, only a single attribute can be learned in each model, requiring a full model per attribute. In [26], the authors train separate models to explore the latent space based on gradient descent using binary annotations or user-defined functions.

Similarly, [6] explicitly maps the number of notes to a dimension of the latent vector with geometrical regularization. However, it requires access to a differentiable compu-

Figure 1. Overview of our proposed model. It is based on the Variational Auto-Encoder, and the decoder is conditioned on an attribute vector $\mathbf{a}$. To force the output to reflect the conditioning, our model has a classifier-discriminator in addition, which induces the encoder to remove the attribute information from the latent vector $\mathbf{z}$. The classifier-discriminator predicts the class labels $\mathbf{B}$, which are calculated by quantizing musical attributes $\mathbf{a}$, and trains the encoder in an adversarial way.

tational function of attributes. Thus, there is no guarantee that this model can deal with more complex attributes nor multiple attributes at the same time. Recent works [27, 28] also map the musical attributes to certain dimensions.

2.2.2 Music Style Transfer

A field related to our research is music style transfer. In this task, models aim to apply a *target style* while preserving the *content* of a given music piece. This task appears more complex than music interpolation as the precise definition of *style* is somewhat elusive. Depending on papers, style is either defined as genre, composer, or other undefined factors. The wide array of researches on music style transfer [15, 29–31] can be broadly separated between supervised and unsupervised approaches.

In the supervised realm, [31] relies on synthesized data of the same content in a variety of target styles, which can be considered as a ground-truth for transfer. Although this work appears to perform transfer successfully, its problem setting appears unrealistic, as we usually do not have access to paired ground truth data for every style.

Unsupervised music style transfer [15, 29, 30], provide a more realistic approach, by simply collecting random data with style labels and aiming to learn a model of the different musical styles. Therefore, these approaches avoid the pitfall of providing a clear definition of *content* or *style*. However, this leads to models which are extremely difficult to evaluate and does not provide controls on the precise characteristics of the generated material.

In our work, we aim to provide explicit control over a set of interpretable musical attributes, which can be defined as *musical style* collectively. Hence, we believe that our research can provide the first step towards a more grounded, interpretable music style transfer.

3. METHODOLOGY

In this paper, our goal is to devise a method for symbolic music generation, providing understandable controls over musical attributes of the generation. Furthermore, these controls should provide continuous modifications to the output, akin to the parameters of a modern synthesizer.

This implies that we need to control the extent of different transformations, ensuring the quality of the generation.

3.1 Model Overview

To achieve our goal, we rely on a VAE architecture with an adversarial classifier-discriminator on the latent vector, as depicted in Figure 1. The encoder-decoder model is trained to reconstruct the input data, as in usual VAE frameworks. Moreover, as we aim to address the explicit control of musical attributes, the decoder takes these musical attributes as additional conditioning information to the latent vector. However, this latent vector potentially already contains all the information required to reconstruct the input data. This would lead to a decoder that simply does not use the information of conditioned attributes. In this degenerate situation, the model would not account for control modifications in the generation. The classifier-discriminator solves this problem, by driving the encoder to remove any information on the attribute from the latent space. Hence, the decoder is forced to use the attribute information to reconstruct the input data adequately, as this information is missing from the latent vector.

Although this architecture is similar to Fader Networks [8], our model is based on the VAE, and the attributes in our work are continuous ordered values, instead of binary indicators. This means that we cannot make direct use of existing discriminators [8–15]. To overcome this problem, we extend the discriminator mechanism so that it acts as a classifier as well. To do so, we quantize the attributes into K balanced classes and use these as targets to a multivariate discriminator. As a result of adversarial training, the encoder prevents the discriminator from predicting the class labels correctly. Hence, the resulting latent vectors should not contain any musical attributes information.

3.2 Model Architecture

We consider a monophonic pitch sequence $\mathbf{s}_{1:T}$ as input, where T represents time steps, encoded in a piano-roll representation, as a sequence of its one-hot vectors (Eq. 4).

Encoder: The encoder is simply defined as a one-layer bidirectional Gated Recurrent Unit (GRU), followed by

linear layers (denoted as MLP) to generate the mean and variance of the latent vector $\mathbf{z}$.

$$\mathbf{E}_t = \text{onehot}(s_t) \tag{4}$$
$$(\mathbf{O}, \mathbf{H}) = \text{GRU}(\mathbf{E}_{1:T}) \tag{5}$$
$$p_{enc}(\mathbf{z} \mid \mathbf{s}) = \mathcal{N}\Big(\mathbf{z}\Big|\text{MLP}(\mathbf{O}_T), \text{diag}\big(\exp(\text{MLP}(\mathbf{O}_T))\big)\Big) \tag{6}$$

where $\mathbf{O}$ is the output feature, and $\mathbf{H}$ the hidden state.

Decoder: The decoder reconstructs the input $\mathbf{E}_t$ at time t, based on the latent vector $\mathbf{z}$, the one-hot vector of the previous step $\mathbf{E}_{t-1}$, and the vector of musical attributes $\mathbf{a}$. The difference between our decoder and a conditional VAE is that this musical attributes vector is used along with the latent information, based on the premise that the latent vector does not have information about these attributes. This decoder is composed of a two-layer GRU.

$$p_{dec}(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid 0, \mathcal{I}) \tag{7}$$
$$(\mathbf{O}_1, \mathbf{H}_1) = \text{GRU}\big([\mathbf{E}_0; \mathbf{z}; \mathbf{a}], \text{MLP}(\mathbf{z})\big) \tag{8}$$
$$(\mathbf{O}_t, \mathbf{H}_t) = \text{GRU}\big([\mathbf{E}_{t-1}; \mathbf{z}; \mathbf{a}], \mathbf{H}_{t-1}\big) \tag{9}$$
$$p_{dec}(s_t \mid \mathbf{s}_{1:t-1}, \mathbf{z}, \mathbf{a}) = \text{Cat}\big(s_t \mid \sigma(\text{MLP}(\mathbf{H}_t))\big) \tag{10}$$

where $[;]$ is a concatenation of vectors, Cat the categorical distribution, and σ the softmax function.

3.3 Attribute Control

In this section, we detail our method to provide understandable control over musical attributes in the music generation process.

3.3.1 Musical Attributes

We rely on the jSymbolic [7] software to calculate statistical musical attributes from symbolic music data. It provides 246 kinds of features, such as pitch statistics, melodies, intervals, rhythm, instrumentation, texture, and dynamics. Out of these, we picked twelve features based on their interpretability such as the total number of notes, pitch variability, and rhythmic value variability [1]. Each feature is normalized to have zero mean and unit variance. We obtain a set $\mathbf{a} = (a_1, \ldots, a_N)$ of N attributes, where each attribute is averaged across a given input example.

3.3.2 Quantize Attributes
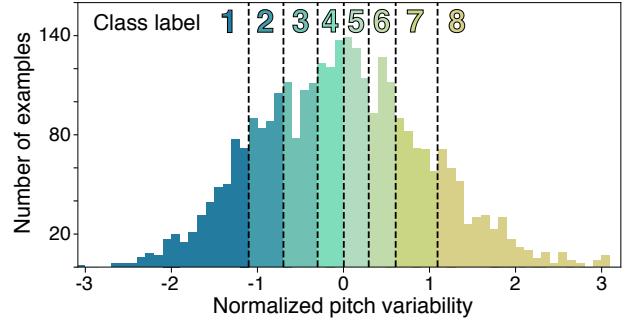
To train our discriminator, we first quantize the attributes into class labels $b_n \in \{1, 2, 3, \ldots, K\}$, where $K = 8$ in this work. The quantization operation $b_n = \text{quantize}(a_n)$, which is a μ-law compression, leading to an equal number of training data in each class, as depicted in Figure 2.

3.3.3 Classifier-Discriminator

The classifier-discriminator takes a latent vector $\mathbf{z}$ and predicts the class which the data belongs to. Hence, the target of the classifier-discriminator can be described as follows

$$gt(n, k) = \begin{cases} 1 & (k = b_n) \\ 0 & (\text{else}) \end{cases} \tag{11}$$

---

[1] The detailed set of features along with their meaning is available in associated webpage of this paper (https://lisakawai.github.io/music_transformation/).

Figure 2. Data distribution of pitch variability, showing how the data is split.

where $k \in [1, K]$ is the index of the quantized class and $n \in [1, N]$ is the attribute index. In order to define the adversarial criterion, the target of the encoder is $1 - gt(n, k)$.

The classifier-discriminator is defined as linear layers followed by *tanh* activation functions for all layers, except the last layer, which uses a *sigmoid* function. Thus, this network outputs a matrix $\mathbf{B} \in \mathbb{R}^{N \times K}$. Hence, this is a major difference of our classifier-discriminator, which relies on a multivariate output, instead of a scalar one [8].

3.4 Loss Function

In this work, three separate loss functions are used to optimize the model, namely, the reconstruction loss $\mathcal{L}_R$, the Kullback-Leibler divergence $\mathcal{L}_{KL}$, and the adversarial loss $\mathcal{L}_D$. The first two functions are used to optimize the VAE, as detailed in Section 2.1.1. $\mathcal{L}_D$ is used to train the classifier-discriminator and influences the encoded latent vectors, as detailed in the following section.

3.4.1 Adversarial Loss

To compute $\mathcal{L}_D$, we rely on the vectors of attribute class labels $\mathbf{b} = (b_1, \ldots, b_N)$, as detailed in Section 3.3.2. The classifier-discriminator aims to predict the class labels, by optimizing its probability distribution p_{dis}. On the other hand, the encoder aims to prevent the classifier-discriminator from predicting the class labels correctly. Hence, the targets of the encoder for the classifier-discriminator prediction is the complement vector of the class labels instead of a scalar in [8], defined as $\mathbf{1} - \mathbf{B} \in \mathbb{R}^{N \times K}$, where all the elements of $\mathbf{1}$ are equal to 1.

$$\mathbf{B} = \text{onehot}(\mathbf{b}) \tag{12}$$
$$\mathcal{L}_D(\theta_{dis}|\theta_{enc}) = -\mathbb{E}_{p_{enc}(\mathbf{z}|\mathbf{s})}[\log(p_{dis}(\mathbf{B}|\mathbf{z}))] \tag{13}$$
$$\mathcal{L}_D(\theta_{enc}|\theta_{dis}) = -\mathbb{E}_{p_{enc}(\mathbf{z}|\mathbf{s})}[\log(p_{dis}(\mathbf{1} - \mathbf{B}|\mathbf{z}))] \tag{14}$$

where θ_{dis} is parameters of the discriminator. Note that each element of the one-hot vectors $B_{n,k}$ is equal to $gt(n, k)$ in Eq (11).

3.4.2 Total Loss Function

The complete loss function of our method is defined as:

$$\mathcal{L}(\theta_{enc}, \theta_{dec} \mid \theta_{dis}) = \mathcal{L}_R + \alpha\mathcal{L}_{KL} + \beta\mathcal{L}_D(\theta_{enc} \mid \theta_{dis}) \tag{15}$$
$$\mathcal{L}(\theta_{dis} \mid \theta_{enc}) = \mathcal{L}_D(\theta_{dis} \mid \theta_{enc}) \tag{16}$$

where α and β are hyperparameters for controlling the impact of each loss in the optimization of the model.

4. EXPERIMENTS

4.1 Dataset

In our experiments, we rely on Nottingham dataset [32], a collection of monophonic British and American folk tunes, including both melodies and chords information. The examples are divided between 694 train, 170 test, and 173 validation instances. We filtered the dataset to retain only examples with 4/4 signature and used only the melody information. We split the data to obtain every sequence of four bars and performed pitch augmentation by shifting pitches from -5 to 6 for the training set. In the final dataset, the pitch ranges from 50 (D3) to 95 (B6).

4.2 Input Representation

In this work, we use a piano-roll-like input representation of monophonic music, which is a sequence of one-hot vectors, representing fixed-bar melodies as a matrix of dimension $time \times (pitch+2)$. The difference from a piano-roll is that we add two dimensions to represent *continue* (holding the previous note) and *rest* (no pitch is on) information. We choose this representation as it allows distinguishing short repeated notes more precisely. We compute the final matrix from the input $\mathbf{s} = (s_1, s_2, \ldots, s_T)$, where $T = 64$.

4.3 Baselines

We compare our proposal with two baselines: *naive VAE* and *GLSR-VAE* [6]. The implementation of the naive VAE is the same as ours without the attribute conditioning and the classifier-discriminator, and we calculate mean latent vectors with/without an attribute to decide an interpolation direction for a given latent vector following [16,17]. To apply binary attributes information, in this case, we split the data into two classes so that the instance is considered *with* an attribute if $a_n \geq 0$ and *without* the attribute if $a_n < 0$. We define the mean latent vector of the attribute presence as $\mathbf{z_w}$ and absence as $\mathbf{z_{wo}}$ and perform the interpolation by moving a latent vector as $\mathbf{z} + \delta \times (\mathbf{z_w} - \mathbf{z_{wo}})$, where δ ranges from -0.5 to 0.5 by steps of 0.1, which is within the interquartile range of $\mathcal{N}(0, \mathcal{I})$.

In the implementation of the GLSR-VAE, we simply add its regularization term to the naive VAE model. In this model, one dimension of the latent vector z_0 corresponds to the *number of notes* attribute. The interpolation is performed by computing $z_0 + \delta$, where δ is as defined previously. Note that this model is only able to interpolate the number of notes with existing implementation, while ours is applicable to any attributes.

4.4 Implementation Details

We train all of the models by using a batch size of 64, the learning rate is set to 1e-4, and we use the ADAM optimizer [33] for 50K iterations. The GRU layer has a hidden size of 1024 for both the encoder and the decoder, and the latent vector has 128 dimensions. In the VAE loss function (see Eq (15)), we use $\alpha = 0.1$ and $\beta = 0.1$.

attribute	Naive	GLSR	Ours
total number of notes	0.973	0.975	**0.981**
pitch variability	0.807	-	**0.938**
rhythmic value variability	0.830	-	**0.938**
pitch kurtosis	0.528	-	**0.723**
pitch skewness	0.366	-	**0.492**
most common rhythm val.	0.795	-	**0.851**
average note duration	0.968	-	**0.983**
note density variability	0.677	-	**0.855**
amount of arpeggiation	0.126	-	**0.386**
chromatic motion	0.284	-	**0.622**
direction of melodic motion	0.428	-	**0.702**
melodic arcs interval span	0.262	-	**0.523**
total number of notes	0.949	0.279	**0.950**
pitch variability	0.797	-	**0.918**
rhythmic value variability	0.809	-	**0.849**

Table 1. Correlation coefficient comparison of musical attributes. **Top**: Results of transformed outputs by changing δ. **Bottom**: Results of *cycle* transformation reverting original attributes as condition to already transformed outputs.

4.5 Evaluation Metrics
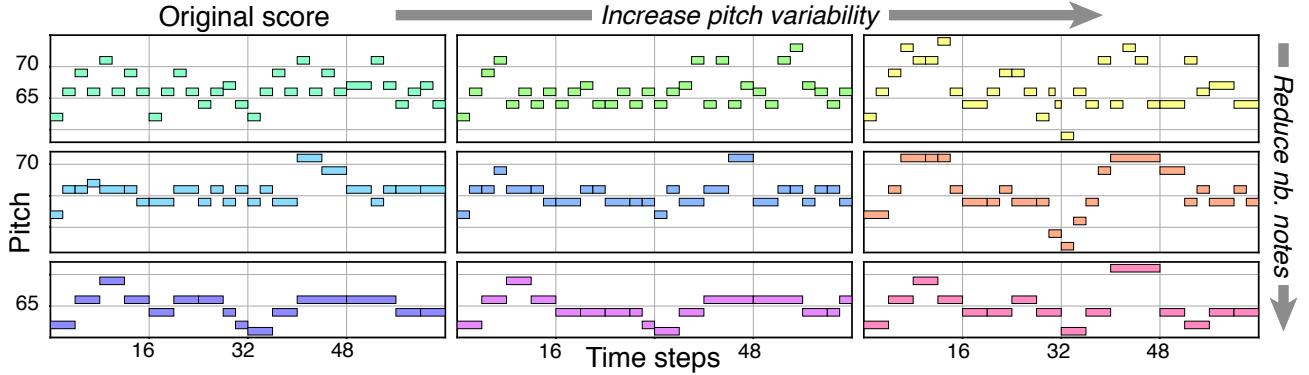
To evaluate our model performance, we need to assess both the reconstruction accuracy and the efficiency of the attributes control. Hence, the outputs generated by the model should adequately reflect the changes in the attributes. To evaluate this aspect, we rely on Spearman's rank-order correlation coefficient between conditioning attribute a_{in} and the resulting attribute a_{out} calculated from the output. The conditioning attribute a_{in} is calculated from the original attribute a_{org} depending on the change made in the control, so that $a_{in} = a_{org} + \delta$, where δ is the same as Section 4.3.

5. RESULTS

5.1 Results with Single Attribute

We compute the correlation coefficient between the target attributes and the corresponding jSymbolic features calculated on the interpolated outputs. All the interpolation results can be found in Table 1 (Top). Although some attributes such as the *total number of notes* and the *pitch variability* are easily handled by a naive VAE, some others such as *chromatic motion* and *amount of arpeggiation* are largely more difficult to detect. Our model allows us to obtain control over all the features, while outperforming others on the interpolated results.

Cycle consistency check. We also perform an experiment to check the *consistency* of attribute control. Ideally, based on a transformed output, if we revert the attribute to its original value, and use it as a condition to regenerate the interpolated results, the model should be expected to reproduce the original score. According to Table 1 (Bottom), our model is able to maintain high correlation coefficients even after successive interpolations. Oppositely, GLSR-VAE is unable to maintain this correlation, as the model cannot control the degree of attribute change.

Figure 3. Generated samples by changing the *total number of notes* and *pitch variability* attributes. Top left shows the original score and the others are outputs from our model, which is able to transform multiple musical attributes simultaneously.

	Naive	GLSR	Ours
NLL	2.269	**1.002**	1.679
accuracy	0.759	**0.808**	0.790

Table 2. Negative log-likelihood (NLL) and accuracy comparison when models are trained on an attribute, the *total number of notes*.

Figure 4. Cosine similarity of chroma features changing the *total number of notes* to compare chord consistency.

Reconstruction quality. We compute the negative log-likelihood (NLL) and reconstruction quality (accuracy) for unchanged attribute values and display results in Table 2. It seems that the GLSR-VAE approach provides slightly better results in the pure reconstruction scenario. However, our proposed model does not deteriorate the reconstruction quality as seen with the naive VAE.

Chord consistency check. Previous works in the field of music style transfer [30, 31] rely on chord consistency analysis to evaluate the capacity of the model to transfer the style in a given music piece while keeping its content. In [14], the authors use chroma feature as one of the criteria for the evaluation of music similarity. We follow this evaluation metrics to observe how different models can preserve the chord structures. This feature possesses 12 dimensions, each of them representing the intensity of a given pitch across octaves in one bar. To evaluate consistency, we compute the cosine similarity between chroma features of the original data and the interpolated one. The results for the *total number of notes* attribute are displayed

N	attribute	corr
2	total number of notes	0.983
	pitch variability	0.944
3	total number of notes	0.978
	pitch variability	0.936
	rhythmic value variability	0.922

Table 3. Correlation coefficient with multiple attributes used in experiments for our model.

in Figure 4 for varying values of δ. As a result, our method performs better than baselines for preserving chord consistency. Although GLSR-VAE provides better results for $\delta \approx 0$, its cosine similarity significantly drops for a positive δ, where its chord consistency quality rapidly degrades as $|\delta|$ becomes larger.
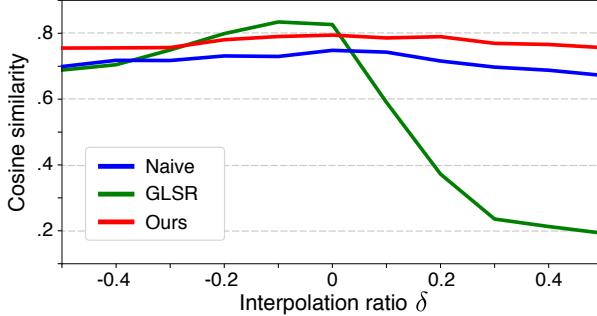
5.2 Results with Multiple Attributes

We display in Figure 3 the samples generated by our model trained on multiple attributes simultaneously (here the *total number of notes* and *pitch variability*). It shows that we can obtain interesting transformations on a given original score, while using interpretable controls. This emphasizes the ability of our model to control multiple attributes.

To further analyze this behavior, we compute the correlation coefficient of multiple attributes by shifting only one attribute at the same time, while inputting the original values for the others, as displayed in Table 3. These results indicate that the correlation coefficient remains stable, even when successively adding new controls. This shows that our model is able to produce *independent* control of multiple musical attributes, which is of prime importance for precise and intuitive music creation.

6. CONCLUSION

In our work, we proposed a new model for deep music transformation. We relied on musical attributes and introduced a model able to learn how to generate music based on these attributes. This was done by quantizing the attributes and introducing an adversarial classifier-discriminator on latent features. The experimental results showed that our model leads to independent and robust controls of musical attributes for monophonic music.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] H. H. Mao, T. Shin, and G. W. Cottrell, "Deepj: Style-specific music generation," in *Proc. of International Conference on Semantic Computing, ICSC*, 2018, pp. 377–382.

[2] D. Williams, A. Kirke, J. Eaton, E. Miranda, I. Daly, J. Hallowell, E. Roesch, F. Hwang, and S. J. Nasuto, "Dynamic game soundtrack generation in response to a continuously varying emotional trajectory," in *Audio Engineering Society Conference: Audio for Games*, 2015.

[3] F. Pachet, A. Papadopoulos, and P. Roy, "Sampling variations of sequences for structured music generation," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2017, pp. 167–173.

[4] T. Akama, "Controlling symbolic music generation based on concept learning from domain knowledge," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 816–823.

[5] L. Ferreira and J. Whitehead, "Learning to generate music with sentiment," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 384–390.

[6] G. Hadjeres, F. Nielsen, and F. Pachet, "Glsr-vae: Geodesic latent space regularization for variational autoencoder architectures," in *IEEE Symp. Series on Computational Intelligence, SSCI*, 2017, pp. 1–7.

[7] C. McKay and I. Fujinaga, "jsymbolic: A feature extractor for midi files," in *Proc. of the International Computer Music Conference, ICMC*, 2006, pp. 302–305.

[8] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. A. Ranzato, "Fader networks:manipulating images by sliding attributes," in *Advances in Neural Information Processing Systems, NeurIPS*, 2017, pp. 5967–5976.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems, NeurIPS*, 2014, pp. 2672–2680.

[10] L. Yang, S. Chou, and Y. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2017, pp. 324–331.

[11] H.-M. Liu and Y.-H. Yang, "Lead sheet generation and arrangement by conditional generative adversarial network," in *Proc. of Machine Learning and Applications, ICMLA*, 2018, pp. 722–727.

[12] H. Dong, W. Hsiao, L. Yang, and Y. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proc. of Conference on Artificial Intelligence, AAAI*, 2018, pp. 34–41.

[13] H. Dong and Y. Yang, "Convolutional generative adversarial networks with binary neurons for polyphonic music generation," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2018, pp. 190–196.

[14] M. Bretan and L. P. Heck, "Self-supervised methods for learning semantic similarity in music," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 446–453.

[15] G. Brunner, Y. Wang, R. Wattenhofer, and S. Zhao, "Symbolic music genre transfer with cyclegan," in *Proc. of Tools with Artificial Intelligence, ICTAI*, 2018, pp. 786–793.

[16] A. Roberts, J. H. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. of International Conference on Machine Learning, ICML*, 2018, pp. 4361–4370.

[17] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, "Learning a latent space of multitrack measures," *CoRR*, vol. abs/1806.00195, 2018.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. of International Conference on Learning Representations, ICLR*, 2014.

[19] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. of International Conference on Machine Learning, ICML*, 2014, pp. 1278–1286.

[20] N. Hadad, L. Wolf, and M. Shahar, "A two-step disentanglement method," in *Proc. of Computer Vision and Pattern Recognition, CVPR*, 2018, pp. 772–780.

[21] P. Upchurch, J. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Weinberger, "Deep feature interpolation for image content changes," in *Proc. of Computer Vision and Pattern Recognition, CVPR*, 2017, pp. 6090–6099.

[22] M. Li, W. Zuo, and D. Zhang, "Deep identity-aware transfer of facial attributes," *CoRR*, vol. abs/1610.05586, 2016.

[23] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras, "Neural face editing with intrinsic image disentangling," in *Proc. of Computer Vision and Pattern Recognition, CVPR*, 2017, pp. 5444–5453.

[24] C. M. Payne, *MuseNet*, 2019, https://openai.com/blog/musenet/.

[25] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 596–603.

[26] J. H. Engel, M. Hoffman, and A. Roberts, "Latent constraints: Learning to generate conditionally from unconditional generative models," in *Proc. of International Conference on Learning Representations, ICLR*, 2018.

[27] A. Pati and A. Lerch, "Latent space regularization for explicit control of musical attributes," in *ICML Workshop on Machine Learning for Music Discovery Workshop (ML4MD)*, 2019.

[28] A. Pati and A. Lerch, "Attribute-based regularization of vae latent spaces," *CoRR*, vol. abs/2004.05485, 2020.

[29] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2018, pp. 747–754.

[30] W. T. Lu and L. Su, "Transferring the style of homophonic music using recurrent neural networks and autoregressive model," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2018, pp. 740–746.

[31] O. Cífka, U. Simsekli, and G. Richard, "Supervised symbolic music style translation using synthetic data," in *Proc. of International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 588–595.

[32] E. Foxley, *Nottingham Database*, 2011, http://abc.sourceforge.net/NMD/.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of International Conference on Learning Representations, ICLR*, 2015.

STRUCTURAL SEGMENTATION OF DHRUPAD VOCAL BANDISH AUDIO BASED ON TEMPO

Rohit M A   **Vinutha T P**   **Preeti Rao**

Department of Electrical Engineering
Indian Institute of Technology Bombay, India

`{rohitma, vinutha, prao}@ee.iitb.ac.in`

ABSTRACT

A Dhrupad vocal concert comprises a composition section that is interspersed with improvised episodes of increased rhythmic activity involving the interaction between the vocals and the percussion. Tracking the changing rhythmic density, in relation to the underlying metric tempo of the piece, thus facilitates the detection and labeling of the improvised sections in the concert structure. This work concerns the automatic detection of the musically relevant rhythmic densities as they change in time across the *bandish* (composition) performance. An annotated dataset of Dhrupad bandish concert sections is presented. We investigate a CNN-based system, trained to detect local tempo relationships, and follow it with temporal smoothing. We also employ audio source separation as a pre-processing step to the detection of the individual surface densities of the vocals and the percussion. This helps us obtain the complete musical description of the concert sections in terms of capturing the changing rhythmic interaction of the two performers.

1. INTRODUCTION

Dhrupad is one of the oldest forms of North Indian classical vocal music. A typical Dhrupad concert setting comprises a solo vocalist or vocalist duo as the lead and a pakhawaj player for the percussion accompaniment, with a tanpura in the background for the harmonic drone [1]. A Dhrupad performance lasts for over an hour and consists of an elaborate, unaccompanied *raga alap* followed by a composed piece, the *bandish*, performed along with the percussion instrument [2]. The bandish is not only presented as composed but also used as a means for further rhythmic improvisation (*laykari*), where the vocalist sings the syllables of the bandish text at various rhythmic densities and in different patterns [3, Chapter 10]. All the while, the pakhawaj accompaniment is either playing a basic pattern (*theka*) of the metric cycle (*tala*), a rhythmic improvisation to match the vocalist's improvisation, or a free solo

improvisation while the vocalist presents the lines of fixed composition. The simultaneous rhythmic improvisation by both players is peculiar to the Dhrupad genre.

Figure 1 depicts the structure of a bandish performance from the vocalist's perspective. The intermediate refrain portions are the un-improvised sections where the artist sings a portion of the bandish before diving back into another spell of improvisation. A complete segmentation of

Figure 1: The structure of a bandish performance - vocalist's perspective [3]

a Dhrupad bandish performance would thus involve providing rhythmic descriptions of un-improvised and improvised sections pertaining to each - the vocals and the pakhawaj.

The goal of this work is to develop automatic methods for the structural segmentation of the Dhrupad bandish concert. With tempo and the relationships of the rhythmic densities of the individual instruments defining the distinct sections of a Dhrupad bandish concert, we explore new approaches to the reliable detection of these musical attributes as they vary across the concert. Given that vocal onsets are difficult to detect (even in isolated vocals due to the diversity inherent to singing), we turn to alternate methods for the direct estimation of the local rhythmic density. Advances in deep learning have led to the development of methods that treat the estimation of the predominant tempo from the raw audio spectral representation as a classification task [4–6]. We explore a similar approach for our task of estimating the changing surface tempo or rhythmic density across a concert audio. In view of the significant improvements reported in audio source separation in recent years, we also consider the use of source separation followed by tempo estimation for the constituent instruments in order to give a more complete description of each section.

The chief new contributions of our work are as follows: (i) a dataset of tempo markings and rhythmic density based structural segmentation annotations for Dhrupad bandish concerts, (ii) adapting a state-of-the-art tempo estimation method to the task of estimating local rhythmic density of the polyphonic mix, and (iii) the use of source separa-

tion to extend this to each instrument(vocals and pakhawaj) to eventually obtain a musically relevant segmentation of bandish concerts with section labels defined in terms of the rhythmic density inter-relationships.

2. BACKGROUND

Compositions in Hindustani music are sung at a tempo in one of roughly three broad ranges - *vilambit* (10.4-60 BPM), *madhya* (40-175 BPM) or *drut* (170-500 BPM) [3, p. 86]. This tempo is determined by the interval between the *matras* of the tala (a cyclic pattern of beats) that the composition is set to, and is referred to as the metric tempo. The metric tempo is fairly stable with only a gradual upward drift across the performance. However there are local variations in the rhythmic density of the singing or playing during what can be called episodes of improvisation that constitute the surface rhythmic density or surface tempo. For the voice, this is calculated using the number of syllables or distinct notes uttered in a unit interval and for the pakhawaj, the number of strokes played in a unit interval [3, p. 86], [7]. The surface tempo is found to generally be an integer multiple (ranging between 2 and 16) of the underlying metric tempo and we use the term 'surface tempo multiple'(*lay* ratio) to refer to this integer. The metric and surface tempi in this form of music thus have fairly objective definitions in terms of the performers' intentions and do not necessarily coincide with 'perceptual tempo'. And indeed as stated in [3, p. 85], the perceived tempo at extreme values of the metric or surface tempo may be quite different due to subdivisions at the lower end and grouping and accenting at the higher.

Related work on structural segmentation for Hindustani classical music can be found in [8–11]. The work in [8] relates to the segmentation of the initial unaccompanied alap portion of a Dhrupad vocal concert into the alap, jod and jhala sections. The methods exploit the changing nature of the energy, pulse clarity (salience), speed, and timbre of the vocals. In [10, 11], the task of segmenting the unaccompanied, and in [9] the accompanied portion of instrumental concert audios consisting of a lead melodic instrument(sitar, sarod) and a tabla accompaniment, was addressed. Signal processing methods based on finding onsets followed by periodicity detection were made use of for tempo and rhythmic density estimation. Section boundaries were obtained with the help of a similarity detection matrix, using frame-level ACF vectors of the detected onsets in [9], and using additional acoustic features and feature transformations in [11]. Faced with the problem of two instruments playing together, differences in the instrument timbres were exploited to separate the plucked string and tabla onsets in [9] to determine separately the metric and the surface tempo. Other source separation methods like HPSS [12, 13] and PLCA [14] have also been used to obtain tempo estimates for individual sources, which are then combined together to refine the overall tempo estimate.

In this work we address the structural segmentation of the bandish section in Dhrupad vocal performances, which

has not yet been attempted. We propose to achieve this by first estimating the surface tempo using the CNN-based approach of [4] with a modified architecture to predict it directly as a multiple of the metric tempo. To obtain the surface tempo of each instrument, we make use of a pre-trained model provided by spleeter [15] that separates vocals from the accompaniment. We then detect section boundaries in a concert audio using changes in the estimated local surface tempi.

3. DATASET DESCRIPTION

To the best of our knowledge there is no existing dataset of tempo and segmentation related annotations for Dhrupad bandish performances. The dataset chosen for this work contains 14 concert audios in the *vilambit* and *madhya laya* - 8 from the Dunya corpus [16] and the rest from publicly available, good quality recordings. 9 of the 14 are by the vocalist duo Gundecha brothers, and the others by Uday Bhawalkar. Each recording is of a single bandish performance by the vocals, accompanied by pakhawaj, with a tanpura in the background. The recordings are 8-15 minutes long and the total duration of the dataset is about 3 hours. The performances are not all in the same raga or tala with at least one composition in each of 4 distinct talas commonly found in Dhrupad. 7 more publicly available audios were partially annotated to balance the cross-validation dataset described in Section 3.2.

3.1 Annotations
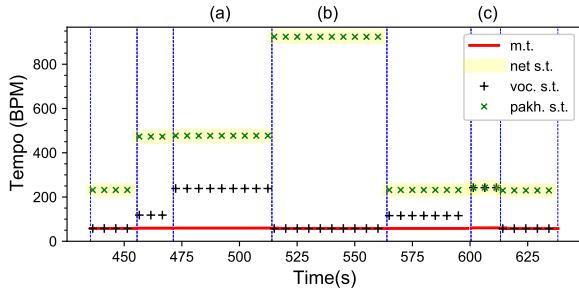
Annotations are of (i) the *sam* positions of the tala, i.e., the cycle boundaries, across the concert (ii) boundaries marking changes in the surface tempo multiple of each instrument and (iii) a label for each section in terms of the surface tempo multiple of each instrument. The annotations were marked by one of the authors, who is a trained musician, using the relatively objective criteria described here.

Information about the tala was obtained from the metadata accompanying the recording. With this, the sam positions were inferred either from the particular stroke of the pakhawaj or the syllable of the bandish refrain that appears on the sam in performance [17], or the number of matras elapsed since the previous sam. Although slight deviations are commonly observed in the metric tempo, large abrupt jumps do not occur. Hence, once a pass was made over the entire audio, the annotations were corrected at points of ambiguity to ensure coherence with adjacent sam markings. The metric tempo was then calculated versus time, once for every cycle, by dividing the cycle duration by the number of matras in the tala.

A section boundary was marked whenever the rhythmic density of either instrument changed and the new density was maintained for at least a duration of 5s. As mentioned earlier, the surface tempo is typically related to the metric tempo as an integer multiple. Therefore every section was labelled with the surface tempo multiple of each instrument, determined by calculating the rate of events (syllables for the vocals and strokes for the pakhawaj) as a mul-

tiple of the metric tempo in the section. Pauses at the pulse level occurring between syllables or strokes were considered as musical events contributing to the surface tempo, while pauses longer than 5s were labeled as having no surface tempo. A more detailed discussion on this appears in [7]. The maximum of the vocal and pakhawaj surface tempo multiples was then added to the section label as the net surface tempo multiple denoting the overall level of rhythmic density. Henceforth, we use the abbreviations m.t., s.t. and s.t.m. to refer to the metric tempo, surface tempo and surface tempo multiple.

Figure 2 is a visualisation of the annotations for a portion of a bandish audio in the dataset [1]. This roughly 4 minute long snippet captures a few sections - (a) vocal s.t. at 4 times the m.t.($\sim$60 BPM) and pakhawaj at 8, (b) vocals at the m.t. and pakhawaj at 16 times - in each of these the net is due to the pakhawaj, and (c) both at 4 times, where the net is due to both.
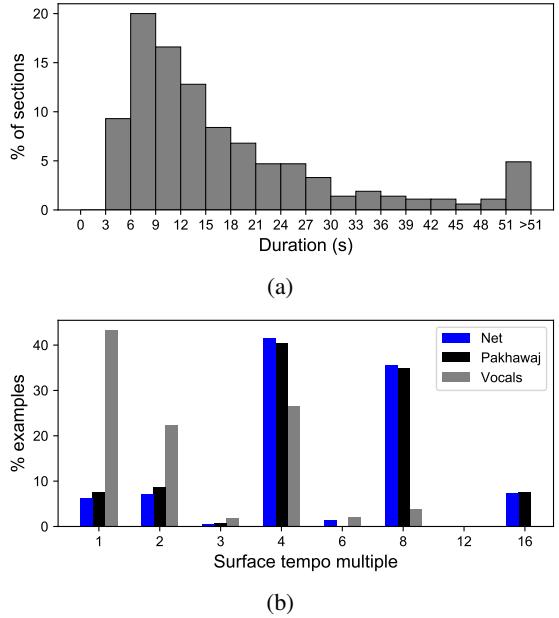
Figure 2: The ground truth metric tempo (m.t.) and surface tempo (s.t.) annotations for a portion of an audio in the dataset. Vertical dashed lines indicate section boundaries.

3.2 Dataset Statistics and Train-test Split

Every annotated section is homogenous in the sense that the s.t. of each instrument remains the same throughout its duration. We therefore pool the sections from all the concert audios into a dataset for training and testing our methods, treating each section as an independent entity. The total number of sections comes up to 634 (593 from the completely annotated and the rest from the partially annotated audios), but they are not all of similar durations. Figure 3 (a) shows the distribution of section durations with a single bar at the end for values more than 51s. We see that a section is mostly between 6 and 20s long. With the goal of tracking the s.t. as it is changing across a performance, we need to perform tempo estimation on shorter examples from each section. The duration of these examples is set to be 8s since a higher value would give us no examples from the large number of sections that are only 6-9s long. Further, for the slowest tempo in the dataset of about 30 BPM, an 8s duration would contain at most 4 beats, fewer than which may not be sufficient for accurate tempo estimation.

The distribution of s.t.m. in the dataset for each instrument and the net is shown in Figure 3 (b) in terms of the

(a)

(b)

Figure 3: Distributions of (a) section duration and (b) net, pakhawaj and vocal s.t.m. across our dataset of 1127 examples.

relative number of non-overlapping 8s examples (extracted from sections) available at each integer multiple, out of a total of 1127 examples. The dataset has a narrow m.t. range of 30 - 85 BPM, but the observed range of s.t. extends upto a large 960 BPM, due to the nature of the *lay* ratios. For the pakhawaj, we find that the multiples 4 and 8 are more abundant than 1, 2 and 16, while the multiples 3, 6 and 12 are nearly absent. For the vocals, 1, 2 and 4 are most represented and even though the multiples 3, 6 and 8 have a similar share, the sections for 8 were found to come from several concerts, while 3 and 6 were only found in a couple. We thus retain only the sections with s.t.m. values from the set {1, 2, 4, 8, 16}.

To manage the data imbalance, while generating the 8s training examples, the hop between consecutive examples is kept shorter for sections belonging to the less populous s.t.m values. We also augment the dataset by time-scaling the audio of each section [18] using one or more factors in the range {0.8, 0.84, 0.88, ... 1.2} (the s.t.m. label remains the same), generating more time-scaled versions for the less populous classes. The whole pool of examples is divided into three folds such that all the examples from a single audio section are assigned to the same fold, and each fold has a similar distribution of the s.t.m. values.

4. METHODS

We consider the recent CNN-based tempo estimation method from [4] (denoted as tempo-cnn) for our work. After examining the viability of the pre-trained model, we first attempt to train new models with the same architecture on our dataset, and then propose some suitable modifications.

4.1 Metric Tempo Estimation

The m.t. of a Dhrupad bandish performance gradually drifts across a performance. Hence, we are interested in estimating it locally and tracking it versus time. With the m.t. range of our dataset being a subset of the tempo-cnn output range, the pre-trained model can be used as it is to observe the nature of its predictions. Upon obtaining estimates frame-wise at 0.5s hops and picking the output class with the highest confidence in each frame, it is found that the model almost always makes octave errors, which is to be expected since the m.t. in our case is not always the perceptual tempo that the model was trained to estimate. We fix these errors by constraining the predicted tempo to lie in the range of m.t. values in the dataset.

We do not attempt to train a new model for m.t. estimation and instead compare the above with a non-learning based approach from [9]. A spectral flux based method is used to obtain the onsets and the autocorrelation function is calculated on 12s long windows at 0.5s hops for values of lag upto 2s. The tempo candidates are constrained to be in the required range and an additional Viterbi smoothing step is used to penalise jumps and obtain a consistent estimate across a concert. We refer to this as the odf-acf method. We also note that the metrical cycle tracking work of [19] offers an alternative that can be investigated for m.t. estimation in future work.

4.2 Surface Tempo Estimation

The s.t. values in our dataset fall outside the tempo-cnn output range. And since the task requires correct identification of tempo without octave errors, using the pre-trained tempo-cnn is not possible. If we are to re-train tempo-cnn on our dataset by increasing the output range, the huge size of the range presents a problem due to the resulting target class imbalance. Therefore, given that the s.t.m. is one of a small set of integer values, we modify the task to predicting this multiple instead of the actual s.t. value.

An attempt to train new models using the tempo-cnn architecture on our dataset by reducing the final softmax layer dimensions does not turn out to be fruitful as the model overfits due to its high capacity and the small size of our dataset. The main issues seem to be the high number of dense layers at the end and the large filter lengths in the multi-filter modules. After a series of simplifications with some inspiration from [5], the architecture summarised in Table 1 is found to be promising (details in [7]). The reduction of dense layers and the addition of dropout layers is found to be crucial in overcoming overfitting. To prevent too much information from getting cut-off due to the dropout, the p value is set to 0.1 in the first three conv. layers, and 0.5 in the later ones. As for the multi-filter conv. layer, fewer filters in parallel and smaller filter lengths are found to make the network easier to train. However, to ensure adequate capacity, the number of filters in each layer is kept moderately high.

Every 8s training example is transformed to a log-scaled mel-filtered magnitude spectrogram, using the following parameters - 40ms windows, 20ms hops and 40 mel filters

Layer	Dimensions
Input	40 x 400
(BN, Conv, ELU, DO) x3	16 x 1 x 5
AvgPool	5 x 1
BN, MF Conv, DO	12x {1x16, 1x32, 1x64, 1x96}
Concat, Conv	16 x 1 x 1
AvgPool	1 x 400
BN, DO, FC, Softmax	# output classes

Table 1: Proposed model architecture, adapted from [4] & [5]

over the band 20-8000 Hz, at a sampling rate of 16kHz. The input to the network is a spectrogram of size 40 x 400 with the values normalized to lie in the range 0 - 1, and the target is one of 5 classes corresponding to the 5 s.t.m. values - 1,2,4,8,16. The network is trained using CCE loss on examples from two folds, with the other fold as the validation set, for a maximum of 500 epochs. Training is carried out using the Adam optimizer with a learning rate of 1e-4 and a batch size of 32, and is halted early if the validation loss does not decrease for 50 epochs.

4.3 Extension to Separated Sources

Given our interest in estimating the s.t. of each instrument to obtain a more complete rhythmic description and the section boundaries in a concert, the pre-trained 2-stems model by spleeter [15] is used to separate the mixture audios into vocals and accompaniment, and new models with the same architecture as proposed above are trained to predict the s.t.m. for each. The dataset of sections remains the same but the input examples are of the separated sources and the training and validation folds are generated again for each source to balance the number of examples across the corresponding classes. The target classes for the pakhawaj are the same as earlier but those for vocals do not include the s.t.m. 16.

4.4 Boundary Detection and Section Labelling

We aim to automatically identify sections in a concert by looking for abrupt changes in the s.t.m. values of each instrument across the concert duration. For this task only the completely annotated 14 concert audios are used. Estimates of s.t.m. are obtained once every 0.5s using 8s long excerpts over the entire duration of each audio. While doing so, each excerpt is presented to that saved model out of the three from the 3-fold CV procedure, to which no portion of the section that this excerpt lies in was presented as a training example, thus preventing any train-test leak. The output class with the highest confidence is taken as the s.t.m. estimate. This procedure is applied to the mixture and the source separated audios. A boundary is marked wherever the s.t.m. of either instrument changes, and the section label is the tuple of the three s.t.m. estimates.

We experiment with two methods for obtaining the three s.t.m. estimates. One, the three values are estimated

Method	Accuracy 1	Accuracy 2
tempo-cnn	5.2	73.8
tempo-cnn with range constraint	71.6	74.7
odf-acf	72.0	72.0

Table 2: Metric tempo estimation accuracies (%) at 4% tolerance using tempo-cnn [4] and the odf-acf method [9].

independently, and we refer to this method as seg1. Here the net s.t.m. may not be equal to the higher of the other two (which should be true by definition). We thus report results using the model output for the net s.t.m. as well as by simply taking the maximum of the other two as the net s.t.m. value. Two, to investigate whether using the expected relationship between the three s.t.m. values helps improve performance, instead of obtaining them independently, we pick that tuple of the three estimates in every frame which has the highest average classifier confidence value and in which the net s.t.m. is the maximum of the other two. We refer to this method as seg2. To reduce the number of false alarms, a post-processing step is used with each method to smooth the outputs by constraining the duration of a detected section to be at least 5s. This is implemented by removing the boundaries of any section that is shorter and replacing the label by that of the previous section.

5. EXPERIMENTS AND RESULTS

5.1 Metric Tempo Estimation

To evaluate m.t. estimation we calculate *accuracy1* and *accuracy2* (allowing for octave errors) with a tolerance of 4% across each audio at a 0.5s frame-level and then average it across the dataset. We find that both the methods fare equally well (Table 2) and the simple fix of including a range constraint significantly improves *accuracy1* for tempo-cnn (except in cases where the prediction is an octave off but already in the m.t. range).

A closer look at concert-wise scores revealed that the accuracy was below 70% in the same 4 (out of 14) concerts in both the methods, where most of the errors were due to the predicted value being either 1.5 or 0.75 times the actual m.t. value. The tempo-cnn makes errors only in small portions of such concerts, but in the odf-acf method, due to the imposed penalty on jumps, the predicted tempo was found to be incorrect over longer durations. Even so, what we take away from the overall results is that for most of the concerts, m.t. is estimated well across the entire duration despite the presence of sections where both the instruments are improvising and playing at different multiples of the m.t.

5.2 Surface Tempo Estimation

Here, we first report the average 3-fold cross-validation accuracy values. This accuracy measures the proportion of 8s

Case	Net s.t.m	Vocal s.t.m	Pakhawaj s.t.m
Accuracy	75.2	69.1	76.9

Table 3: Average 3-fold cross-validation accuracies (%) for surface tempo multiple estimation

examples for which the s.t.m. was correctly identified. Table 3 shows the results for all three cases - estimation of net s.t.m. from the original mixture, and that of the individual instruments from separated audios.

The results are poorer for separated vocals and better for pakhawaj, which reflects also in the net score, given that the net s.t.m is dominated by that of the pakhawaj. The class-wise performance is shown using a confusion matrix for each case in Table 4. In the case of vocals, classes 1 and 8 are estimated more accurately. For class 8, this could be due to the distinct nature of vocalisation and the limited diversity of examples due to fewer available sections. For class 1, most examples come from sections where the bandish is sung at a steady rate without improvisation thus making tempo estimation easier. For class 2, sections often come from the earlier stages of improvisation in a concert where the singing is not fully rhythmic and is characterized by pauses, melismatic singing and changes to other s.t.m. levels, making the estimation harder. The confusions between classes 2 and 4 could also be due to some bleed of pakahwaj into the vocals during source separation.

In the case of net and pakhawaj s.t.m., classes 1 and 2 are estimated quite accurately, while the other classes are confused with their immediate neighbours. The class 16 being confused with 8 is most likely because of the presence of accents on every other stroke. We also notice a drop in the performance of this class in the case of separated pakhawaj when compared to the mixture audios, possibly due to a further loss of weak onsets after separation.

5.3 Boundary Detection and Section Labelling

We evaluate boundary retrieval performance using precision, recall and F-score (Table 5a). A predicted boundary is declared a hit if it falls within a certain duration of an unmatched ground truth boundary, and a false alarm otherwise. Results are reported at two values of temporal tolerance: ±1.5s and ±3s. The latter value is as used in [20] and the former is included with the reason that since a large number of sections are 6-9s long, even if both the detected boundaries are off by 1.5s, the detected section still captures at least half of the ground truth section.
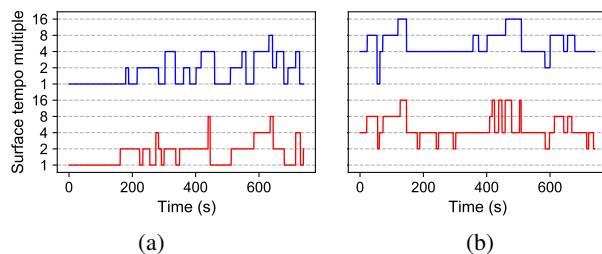
To evaluate section labelling, we report labelling accuracy (Table 5b) as the fraction of the duration of each concert that is correctly labelled (excluding regions where the ground truth is not one of {1,2,4,8,16}), averaged across the dataset, as defined in [21]. Each of the three s.t.m. labels are first evaluated individually and also when taken together (i.e., a frame is said to be correctly labelled only if all three labels are correct). We expect these scores to be different from the cross-validation accuracies reported in Table 3 as the test set is now no longer balanced, with

		Predicted				
		1	2	4	8	16
(a)	1	**90.1**	2.2	6.4	0.0	1.3
	2	5.8	**82.0**	10.8	0.8	0.5
	4	4.5	13.4	**66.9**	11.9	3.3
	8	2.4	1.8	14.4	**65.7**	15.7
	16	1.8	1.0	6.5	15.1	**75.5**
(b)	1	**77.3**	15.5	5.3	2.0	
	2	**21.0**	50.8	26.2	2.0	
	4	5.8	20.1	**64.6**	9.4	
	8	1.8	0.0	13.2	**84.9**	
(c)	1	**93.0**	0.9	5.0	0.8	0.2
	2	0.2	**83.3**	14.4	1.7	0.3
	4	5.8	15.1	**65.4**	11.1	2.6
	8	2.9	1.1	**11.5**	69.9	14.5
	16	1.4	1.1	6.1	**24.8**	66.6

(Ground truth — row labels, left vertical axis)

Table 4: Confusion matrix of (a) net, (b) vocal, and (c) pakhawaj s.t.m. predictions (values in %)

the confused classes being the more common ones.

The individual labelling accuracies are quite similar for the pakhawaj and net tempo labels, slightly lower for the vocals, but much lower for getting all the labels right in every frame. With `seg1`, we see that the vocal and pakhawaj estimates are reliable enough that taking their maximum as the net s.t.m. instead of using the model estimate improves the net s.t.m. labelling accuracy. Hence, for the evaluation in the last column, the net is taken as the maximum of the other two. Although this seemingly renders the model trained to predict the net s.t.m. not very useful, we see in `seg2` that using it to obtain all the estimates together improves all the accuracies, proving its utility.

Figure 4: The ground truth (above) and estimated (below) s.t.m. labels of the (a) vocals and (b) pakhawaj across the concert *GB_AhirBhrv_Choutal*.

Although better tempo estimation should result in better boundary detection since the boundaries are based entirely on tempo changes, the boundary detection results using `seg2` are only slightly better than `seg1`. In both the cases, the smoothing step was found to improve the results (detailed in [7]). Looking at the vocal and pakhawaj s.t.m. estimates obtained using `seg2` in Figure 4, we see that for both the instruments, at a coarse level, the various sur-

	±1.5s tolerance			±3s tolerance		
	Prec.	Rec.	F-sc.	Prec.	Rec.	F-sc.
seg1	0.27	0.38	0.32	0.39	0.54	0.45
seg2	0.29	0.38	0.33	0.40	0.53	0.45

(a)

	Vocals	Pakhawaj	Net from model	Net as max.	All 3 labels
seg1	67.2	68.7	66.9	67.5	45.9
seg2	67.7	71.0	70.4	-	48.6

(b)

Table 5: (a) Boundary detection performance and (b) s.t.m. labelling accuracies (in %).

face tempo regions are captured well. And while for the pakhawaj, finer section changes are also estimated accurately, such changes are not tracked well in the case of vocals, thus reducing the overall boundary detection scores.

6. CONCLUSIONS

We have presented a system that provides a complete rhythmic description of a Dhrupad bandish performance, enabling its segmentation into musicologically relevant sections based on the rhythmic interaction between the instruments. The metric tempo is estimated by adapting existing methods whereas the surface tempo, with its much larger dynamic range, is estimated in a novel manner by predicting its relationship with the m.t. to directly obtain the musically significant *lay* ratio. Because of the challenges presented by imperfect source separation, we benefit from using a model trained also on the mixture audios. We find that s.t.m. values at the lower and higher extremes are estimated better than the intermediate values. This, despite the intermediate values being the more represented classes in the dataset, points to the diversity in the acoustic realisations of the different surface densities. Future work could involve extending the dataset to encompass more singers and compositions in the *drut* lay, where we might see the same s.t.m. manifesting completely different acoustic properties. In such a scenario, estimating m.t. could help provide useful 'conditioning', and ways to jointly estimate the metric and surface tempi could be explored. Source separation can be improved by introducing new loss functions that preserve onsets better and hence allow better tempo estimation on separated audios. Finally this work provides an example of adapting available MIR methods to music genre specific problems.

Supplementary material
All the dataset details, annotations, code and pre-trained models are available here: `https://github.com/ DAP-Lab/dhrupad-bandish-segmentation`.

7. REFERENCES

[1] B. C. Wade, *Music in India: The classical traditions.* Englewood Cliffs, New Jersey: Prentice-Hall, 1979.

[2] R. Widdess, "Involving the performers in transcription and analysis: a collaborative approach to Dhrupad," *Ethnomusicology*, vol. 38, no. 1, pp. 59–79, Winter 1994.

[3] M. Clayton, *Time in Indian Music: Rhythm, Metre, and Form in North Indian Rāg Performance.* Oxford, England: Oxford University Press, 2000.

[4] H. Schreiber and M. Müller, "A single-step approach to musical tempo estimation using a convolutional neural network," in *Proc. of the 19th Int. Society for Music Information Retrieval Conf.*, Paris, France, 2018, pp. 98–105.

[5] H. Schreiber and M. Müller, "Musical tempo and key estimation using convolutional neural networks with directional filters," in *Proc. of the 16th Sound and Music Computing Conf.*, Malaga, Spain, 2019, pp. 47–54.

[6] S. Böck, M. E. P. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, Delft, The Netherlands, 2019, pp. 486–493.

[7] M. A. Rohit and P. Rao, "Structure and automatic segmentation of Dhrupad vocal bandish audio," Unpublished technical report, arXiv:2008.00756 [eess.AS], 2020.

[8] P. Rao, T. P. Vinutha, and M. A. Rohit, "Structural segmentation of alap in Dhrupad vocal concerts," *Transactions of the Int. Society for Music Information Retrieval, Under review*, 2020.

[9] T. P. Vinutha, S. Sankagiri, K. K. Ganguli, and P. Rao, "Structural segmentation and visualization of sitar and sarod concert audio," in *Proc. of the 17th Int. Society for Music Information Retrieval Conf.*, New York City, USA, 2016, pp. 232–238.

[10] T. P. Vinutha, S. Sankagiri, and P. Rao, "Reliable tempo detection for structural segmentation in sarod concerts," in *Proc. of the 22nd National Conf. on Communications*, Guwahati, India, 2016, pp. 1–6.

[11] P. Verma, T. Vinutha, P. Pandit, and P. Rao, "Structural segmentation of hindustani concert audio with posterior features," in *Proc. of the 40th IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Brisbane, Australia, 2015, pp. 136–140.

[12] A. Gkiokas, V. Katsouros, G. Carayannis, and T. Stajylakis, "Music tempo estimation and beat tracking by applying source separation and metrical relations," in *Proc. of the 37th IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Kyoto, Japan, 2012, pp. 421–424.

[13] A. Elowsson, A. Friberg, G. Madison, and J. Paulin, "Modelling the speed of music using features from harmonic/percussive separated audio," in *Proc. of the 14th Int. Society for Music Information Retrieval Conf.*, Curitiba, Brazil, 2013, pp. 481–486.

[14] P. Chordia and A. Rae, "Using source separation to improve tempo detection," in *Proc. of the 10th Int. Society for Music Information Retrieval Conf.*, Utrecht, The Netherlands, 2009, pp. 183–188.

[15] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020, Deezer Research.

[16] A. Srinivasamurthy, G. K. Koduri, S. Gulati, V. Ishwar, and X. Serra, "Corpora for music information research in Indian art music," in *Proc. of the 40th Int. Computer Music Conf. / 11th Sound and Music Computing Conf.*, Athens, Greece, 2014, pp. 1029–1036.

[17] J. C. Ross, T. P. Vinutha, and P. Rao, "Detecting melodic motifs from audio for Hindustani classical music," in *Proc. of the 13th Int. Society for Music Information Retrieval Conf.*, Porto, Portugal, 2012, pp. 193–198.

[18] Rubber Band Library, "Rubber band library v1.8.2," https://breakfastquay.com/rubberband/, 2018.

[19] A. Srinivasamurthy and X. Serra, "A supervised approach to hierarchical metrical cycle tracking from audio music recordings," in *Proc. of the 39th IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Florence, Italy, 2014, pp. 5217–5221.

[20] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks," in *Proc. of the 15th Int. Society for Music Information Retrieval Conf.*, Taipei, Taiwan, 2014, pp. 417–422.

[21] J. Paulus and A. Klapuri, "Music structure analysis using a probabilistic fitness measure and a greedy search algorithm," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1159–1170, August 2009.

EXPLORING ALIGNED LYRICS-INFORMED SINGING VOICE SEPARATION

Chang-Bin Jeon, Hyeong-Seok Choi and Kyogu Lee

Department of Intelligence and Information
Music and Audio Research Group (MARG)
Center for Superintelligence
Seoul National University
`{vinyne, kekepa15, kglee}@snu.ac.kr`

ABSTRACT

In this paper, we propose a method of utilizing aligned lyrics as additional information to improve the performance of singing voice separation. We have combined the highway network-based lyrics encoder into *Open-unmix* separation network and show that the model trained with the aligned lyrics indeed results in a better performance than the model that was not informed. The question now remains whether the increase of performance is actually due to the phonetic contents that lie in the informed aligned lyrics or not. To this end, we investigated the source of performance increase in multifaceted ways by observing the change of performance when incorrect lyrics were given to the model. Experiment results show that the model can use not only just vocal activity information but also the phonetic contents from the aligned lyrics.

1. INTRODUCTION

Singing voice separation is one of the most widely studied areas in the field of audio signal processing. In particular, the importance of research is greatly emphasized because it can contribute to the pre-processing step of research in various fields of Music Information Retrieval (MIR), such as automatic music transcriptions and automatic lyrics alignments. With the recent development of deep neural networks, a number of music source separation studies have been published and showed excellent performance. These studies have a common feature of separating music source by using only information from the sound source itself, such as a 1-dimensional waveform [1,2] or a 2-dimensional spectrogram [3–5].

One of the distinguishing characteristics that differentiate music signals from other audio signals is that usually there exists the corresponding music scores or lyrics. Therefore, several studies attempted to separate the sources by utilizing additional information other than the informa-

tion of the sound source itself. For example, the additional information such as pitch [6] or even the whole score [7] can be used as a prior to help separate the source of interest from the mixture. In general, however, music scores for certain songs are not readily available, while lyrics can be easily collected on the web.
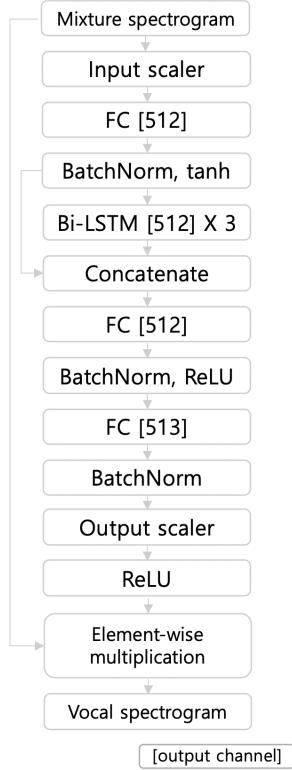
The lyrics are particularly closely related information to singing voice; thus, have promising possibility to be used as additional information for singing voice separation. Recent studies proposed the ways of using linguistic features extracted from the end-to-end automatic speech recognition model [8] or voice conversion model [9] to the singing voice separation framework. However, the way of using explicit lyrics information has not been studied enough so far, which motivates us to study the possibility of lyrics-informed singing voice separation. We expect that singing voice separation systems can benefit from lyrics information because of the rich information contained in the phonetic features such as formant frequencies.

To utilize the lyrics, we combined the highway network-based lyrics encoder [10] into the current state-of-the-art music source separation network, *Open-unmix* [5]. In addition, we tried two conditioning methods — 1. local conditioning, 2. concatenation — and compare the performance. Note that the alignment between the lyrics and songs itself is another separate line of research [11, 12]. For our study, we assume that the alignment is already done and only focus on the use of the *aligned* lyrics.

The information in the aligned lyrics can be seen from two perspectives: 1. The timing information of vocal activity, 2. phonetic information. Therefore, it is important to check if the network is using the phonetic information other than the vocal activity information. Various evaluations were conducted to examine whether the phonetic information of the aligned lyrics actually contribute to improving the performance. We found that the proposed model trained with the aligned lyrics clearly show better performance than the baseline model trained without any additional information. Furthermore, the experiment results show that the performance of the proposed model even exceeds the model trained only with additional vocal activity information. To the best of our knowledge, this is the first research to directly use the lyrics information for a singing voice separation task.

Figure 1: The structure of the baseline *Open-unmix* network.

Figure 2: The structure of the lyrics encoder.

tures of *Open-unmix*, differentiating from other studies that use decibel scales.

We used *Open-unmix* network as our baseline model because we wanted to check whether the aligned lyrics information could improve the performance of the current state-of-the-art model. The channel inputs and outputs are mono in our study although the original study used stereo. This is because our singing dataset is made up of a clean singing voice without any reverberation, chorus, or doubling, as opposed to the singing tracks in MUSDB18's singing voice dataset, which are already processed for the stereo. Details and the full structure of the baseline *Open-unmix* are illustrated in Figure 1.

2. RELATED WORK

2.1 Informed Source Separation

Several studies using machine learning algorithms other than deep neural networks attempted to separate singing voice with side information. For example, robust Principal Component Analysis (rPCA) was used with additional vocal activity information [13]. Also, rPCA and Non-negative Matrix Factorization (NMF) were used with pronounced lyrics [14].

Although only few studies have tried to use additional information for singing voice separation using deep neural networks, it was reported that vocal activity information can be used as an input to the network along with spectrogram to enhance the performance of the singing voice separation network [15]. Very recently, in the speech enhancement field, attempts have been made to utilize text information to increase the separation performance [16].

2.2 Open-unmix

Open-unmix [5] is the state-of-the-art music source separation network using the MUSDB18 dataset [17]. In particular, it consists of 3 bi-directional Long Short-Term Memory (LSTM) layers for source separation with 3 additional fully-connected layers. Batch normalization [18] was used after every fully-connected layer and skip connection [19] was used between the inputs and outputs of 3 consecutive bi-directional LSTM layers. Trainable input and output scalers through frequency-axis are also the special fea-

2.3 Singing Voice Synthesis

Singing voice synthesis and lyrics-informed singing voice separation tasks share a similar framework in that the input and output are the same as lyrics and singing voice spectrograms, respectively. Recently, [10] proposed the singing voice synthesis network that succeeds in creating high-quality singing based on 60 Korean songs sung by a single singer. It is based on the Text-to-Speech model [20], which consists of 1-dimensional convolutional neural networks and highway networks [21]. Therefore, we borrowed the idea of using the highway network-based lyrics encoder and integrated it into the source separation network.
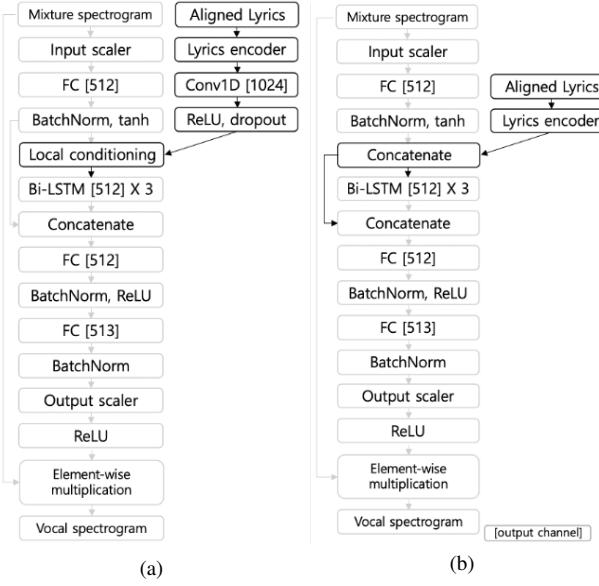
3. PROPOSED METHOD

3.1 Lyrics Encoder

The detailed structure of the lyrics encoder is shown in Figure 2 and the structure of the highway networks used in the lyrics encoder is defined as follows,

$$y = ReLU(x*W_H)\cdot\sigma(x*W_T)+x\cdot(1-\sigma(x*W_T)), \quad (1)$$

, where x and y are input and output of the network and $\cdot$ refers to the element-wise multiplication. $x*W_H$ and $x*W_T$ are the 1-dimensional convolution layers which

Figure 3: The structure of the *Open-unmix* networks combined with the lyrics encoder using (a) local conditioning and (b) concatenation method.

have the same input and output channel sizes. Biases are omitted in Eqn (1). Zero-padding was applied to keep the input and output length the same in every convolution layer. Dropouts [22] with 0.05 dropout rate, were applied after the activation functions. Dilated convolution [23] were used to expand the receptive field to 165 frames. It is about seven times larger than the model without the dilation. In our experimental settings, 165 frames are equal to about 1.915 seconds.

3.2 Local Conditioning of Lyrics Encoding

The local conditioning method [10] was used to insert the encoded lyrics information into the singing voice separation network. The local conditioning is defined as follows,

$$y = ReLU(x * W_f + L_1) \cdot \sigma(x * W_g + L_2), \quad (2)$$

where, $x * W_f$ and $x * W_g$ are the 1-dimensional convolution layers which have same input and output channel sizes. L_1 and L_2 are equally separated features through the channel axis from the output of the consecutive lyrics encoder and the 1-dimensional convolutional layer. The 1-dimensional convolution layer with filter size 1 was added on the output of the lyrics encoder so that the channel size of each L_1 and L_2 could be 512. σ refers to the sigmoid activation function. Details of the full structure are in Figure 3a.

3.3 Concatenation of Lyrics Encoding

A concatenation method, which is a simple but powerful conditioning method, was also used in our study to pass the encoded lyrics information into the singing voice separation network. By concatenating the output of the first fully-connected layer and the lyrics encoder output, the

Singer	Gender	Train	Validation	Test	Total
1	Female	79	5	8	92
2	Male	8	2	0	10
3	Female	8	2	0	10
4	Female	9	0	1	10
5	Female	8	0	1	9
6	Male	10	0	0	10
7	Female	8	0	2	10
8	Female	9	1	0	10
9	Male	9	0	1	10
10	Male	7	1	1	9
11	Female	7	3	0	10
12	Female	0	5	5	10
13	Male	0	0	1	1
		162	19	20	201

Table 1: The composition of our singing dataset.

channel size of LSTM layers input becomes 1024. Also, the channel size of the second fully-connected layer input becomes 1536 by the skip-connection of LSTM input and output. Details of the full structure is in Figure 3b.
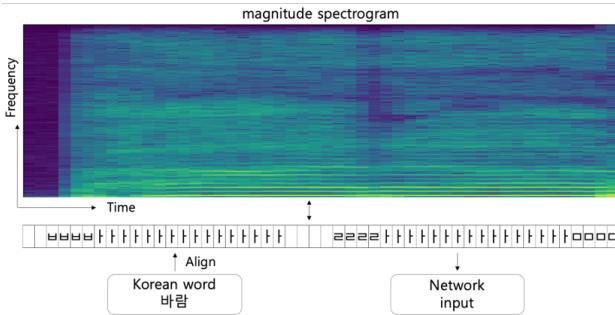
4. EXPERIMENTS

4.1 Dataset

Here we used a total of 201 Korean pop songs sung by 13 amateur singers as target clean singing sources. This dataset has a total length of 11 hours and 44 minutes. Of these, we used 162 songs (9h 3m) for training, 19 songs (1h 7m) for validation, and 20 songs (1h 7m) for test dataset. The detailed composition of the dataset is described in Table 1.

We aligned Korean syllable following [10]; one Korean syllable is made up of onset (consonant), nucleus (vowel) and coda (consonant), we aligned onset and coda for 4 frames, and nucleus for other frames. The example of the alignment applied to the spectrogram is shown in Figure 6.

A total of 19,113 instrumental songs were used as accompaniment for training networks because we did not have real accompaniment tracks corresponding to the singing voice dataset. Since various studies using MUSDB18 [17] or DSD100 [24] dataset also used random mixing techniques, i.e. creating random accompaniment for each iteration that was not related to the original singing, we decided that using arbitrary accompaniments would not be a problem for training. In addition, if we used the same specific accompaniments for the test singing voice dataset, we assumed that it is reasonable for identifying how the information containing the phonetic features of the aligned lyrics has changed the performance of the network, the fact we wanted to identify. Therefore, for the validation and test dataset, we randomly chose each 19 and 20 instrumental songs which have a longer length than the singing data, and shorten the length to the same with the singing.

Figure 4: The example of inserting the aligned lyrics to the networks.

4.2 Training

In our singing dataset, the number of songs recorded by the first singer is outnumbered compared to the others, accounting for about 46 percent of the total. In order to prevent bias to a particular singer when training the networks, a singer was first selected with the same probability when constructing a batch to be used for each iteration, and the vocal source to be used for training was sampled only for the songs recorded by the selected singer.

A mono sound source with a sample rate of 22050 Hz was used in the experiment. FFT point size and window size were set to 1024 samples (0.0464 seconds) to convert them into a spectrogram, and Short-time Fourier transform (STFT) hop size to 256 samples (0.0116 seconds). Adam optimization method [25] was used for training with a learning rate of 0.001, β_1 for 0.9, β_2 for 0.999. Mean squared error (MSE) loss function between the ground-truth and the outputs of the models were used in our study. We trained the models for 500 epochs with calculating validation loss for every epoch. The learning rate was reduced to 30 percent if there was no decrease in validation loss during 25 epochs. Early stopping was applied after 50 epochs without a decrease in validation loss.

4.3 Evaluation Methods

Here we briefly summarize the various experiments that will be shown in the following Section 5. The experiments will be conducted in three following ways.

First, in Section 5.1, we compare and analyze how much performance improvement there are between the baseline model trained without the lyrics and the model trained with the aligned lyrics.

Second, in Section 5.2, we check if the network exploits the vocal activity information included in the aligned lyrics. The lyrics include both the vocal activity information and phonetic information, and thus expected to use the vocal activity information correctly.

Third, in Section 5.3, we check if the given input is correctly being used by the network trained with aligned lyrics. This experiment was done by giving incorrect inputs in the evaluation stage. It is expected that the incorrect inputs will significantly reduce performance.

For network performance evaluations, Signal-to-

Model name	Inputs to the lyrics encoder
model 1	None
model 2	Meaningless inputs (all 0)
model 3	Vocal activity information
model 4	Aligned lyrics

Table 2: The description of each models in our experiments.

Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), Signal-to-Artifact Ratio (SAR) scores [26] were computed by museval python library [27].

5. RESULTS
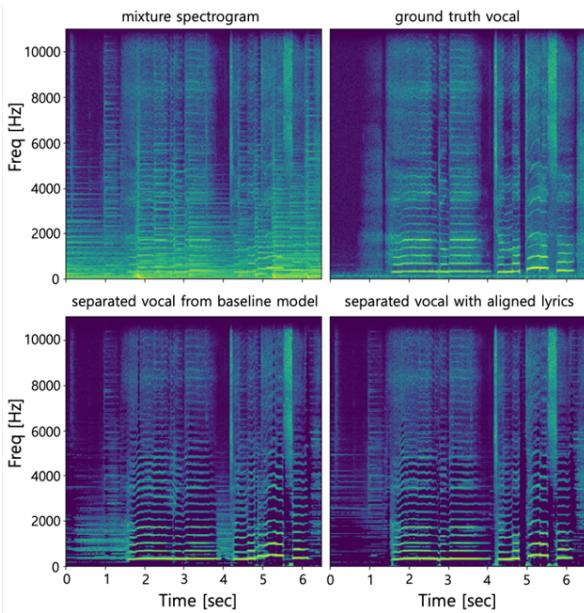
The configuration of the models we trained in our experiments is in Table 2. We trained four models each using local conditioning and concatenation method. *model 1* is the baseline model that trained without the lyrics encoder. *model 2* is the model that trained with meaningless 0 value inputs to the lyrics encoder. This model is only for checking the performance change of the networks caused by the extended network capacity. *model 3* is the model that trained with only vocal activity information to the lyrics encoder. We simply used 0 value as unvoiced sections and 1 as voiced sections so that the 128-dimensional embedding can train useful meaning from it. *model 4* is that trained with aligned lyrics information. For both *model 3* and *model 4*, note that 0 value has clear meaning, unvoiced sections, unlike 0 value in *model 2* is meaningless. Since the baseline model is the same for each local conditioning and concatenation method, we trained a total of 7 models for the experiments. Except *model 1*, we will use the abbreviation of local conditioning and concatenation methods, each *LC* and *CC*, in front of the model names for convenience. For example, the model trained with aligned lyrics and the local conditioning method is *LC-model 4*.

5.1 Performance Evaluation

The quantitative performance evaluation scores of the models are shown in Table 3. Median scores were taken from the median values of 20 songs, which were calculated for every frame (a median of frames, a median of tracks). Mean scores represent the scores taken by a mean of frames, a mean of tracks. Each frame was set to 1 second.

It was confirmed that the separation performance of both *LC-model 4* and *CC-model 4* improved from the *model 1*. This implies that aligned lyrics information can be used as helpful features for singing voice separation networks. Comparing to *LC-model 3* and *CC-model 3*, we could verify that there were clear performance gains not only from the lyrics alignment information but also the phonetic features of the lyrics itself. It was also confirmed that the performance gains do not come from just network capacity growth, given that there are no significant differences in the performance of *model 1* and *model 2*. The spectrograms of the separated sample are in Figure 5.

Models	Median			Mean		
	SDR	SIR	SAR	SDR	SIR	SAR
model 1	9.956	18.674	9.847	8.595	16.062	9.145
LC-model 2	10.140	18.465	9.766	8.589	16.001	9.093
LC-model 3	10.090	18.713	9.763	9.250	16.298	9.153
LC-model 4	**10.767**	19.505	10.223	9.723	17.116	9.699
CC-model 2	10.110	18.434	9.909	8.691	16.164	9.207
CC-model 3	10.444	19.328	10.169	9.718	17.031	9.609
CC-model 4	10.757	**19.623**	**10.371**	**9.752**	**17.250**	**9.803**
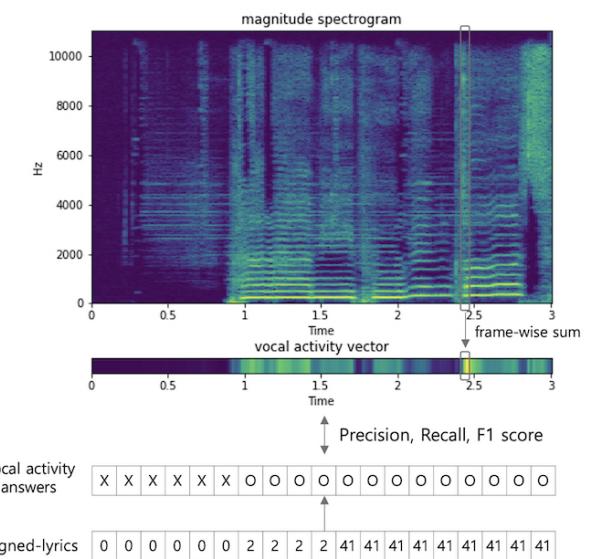
Table 3: Evaluation scores of our singing voice separation models. All scores are in [dB] scale.

Figure 5: The examples of the mixture, ground truth vocal, separated vocal spectrograms of baseline *model 1* and *CC-model 4*.

Figure 6: The example of making the vocal activity vector from the separated vocal spectrogram.

Despite the expectation that the vocal activity information is powerful information to the networks, performance gains observed in *LC-model 3* were very slight. It was much smaller than the improvements achieved from *CC-model 3*. By these, we analyzed that the concatenation method is slightly better for making the networks to reflect the vocal activity information. Nevertheless, we considered that both conditioning methods were effective when giving the networks aligned lyrics information.

5.2 Analysis of Vocal Activity Information Usage

In this section, we quantitatively assessed how well the networks leverage vocal activity information of aligned lyrics. The purpose of this is to see if the models have not been trained by focusing only on either one of the vocal activity information or the phonetic information of aligned lyrics, which are both critical for the separation performance.

To this end, the separated spectrogram values were divided by the largest values of each source for normalization, so that the minimum and maximum values become 0 and 1. Then, the energy of each time axis was summed to

create a vector that contains vocal activity information. It was decided whether vocal activity exists or not based on whether the values were larger or smaller than 0.1 as was done in [15]. Precision, recall, and F1 scores were calculated with the created vocal activity vectors by taking the place where the lyric exists as the ground-truth voiced sections. Scores are contained in Table 4.

From the results of Table 4, we have confirmed that *model 4* can separate the vocal source by reflecting the vocal's timing information more accurately than *model 1* for both lyrics conditioning methods. Also, it was confirmed that *model 3* achieved higher scores for all measures than *model 4*. This is a reasonable result because *model 3* were trained with vocal activity information only, while *model4* needed to learn how to leverage both vocal activity information and phonetic information appropriately while training. Nevertheless, F1 score differences were negligible, which means *model 4* was also capable of reflecting timing information as well as *model 3*.

5.3 Analysis with Using Incorrect Lyrics

To check if *model 4* effectively uses the information in lyrics, we observed the performance change when incor-

Models	Precision	Recall	F1 score
model 1	0.807	0.853	0.828
LC-model 2	0.810	0.852	0.830
LC-model 3	0.887	**0.857**	0.872
LC-model 4	0.876	0.854	0.865
CC-model 2	0.814	0.853	0.833
CC-model 3	**0.896**	0.855	**0.875**
CC-model 4	0.879	0.855	0.867

Table 4: The precision, recall, and F1 scores to evaluate how well the networks used the vocal activity information from aligned lyrics.

Models	Inputs	SDR	SIR	SAR
LC-model 4	Zero	0.001	8.040	-4.286
	Random	5.317	15.802	4.845
	VA+Random	7.899	19.270	6.403
	AlignedLyrics	**10.767**	19.505	10.223
CC-model 4	Zero	0.002	7.246	-3.671
	Random	0.946	14.837	0.341
	VA+Random	7.164	19.545	6.290
	AlignedLyrics	10.757	**19.623**	**10.371**

Table 5: Performance comparisons when different inputs are given in the evaluation stage. `Zero` : 0 value inputs. `Random` : Random value inputs. `VA+Random` : Replace voiced sections with random value. `AlignedLyrics` : Aligned lyrics (The proposed method). All scores are in [dB] scale.
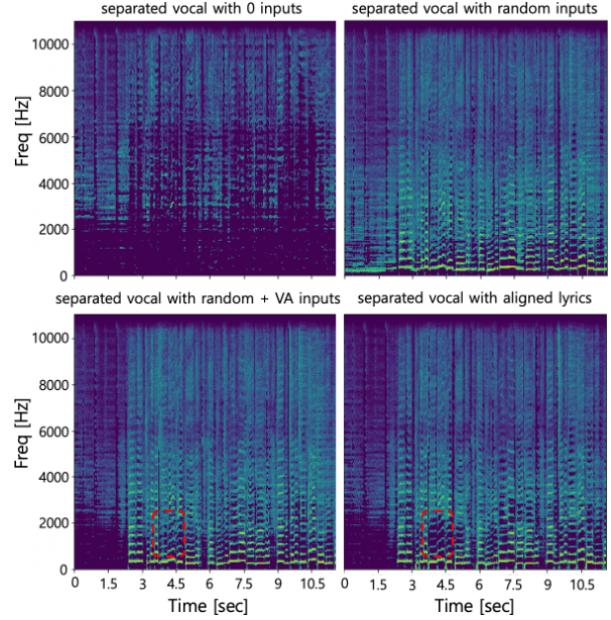
rect lyrics were given as input during the evaluation stage. The results are shown in Table 5.

If the networks had learned to effectively use the information in lyrics it is expected to output silence when the lyrics meaning unvoiced sections are given in the evaluation step. To validate this assumption, we inserted 0 values (`Zero`) to the lyrics encoder in to *model 4* in the evaluation step. As expected, almost every sound has been erased from the mixture with only a little noise left as seen in Figure 7 and critical performance degradation, over 10 dB in SDR score, has occurred.

Furthermore, performance degradation was observed when all the lyrics were replaced with random values (`Random`). This also shows that the network is significantly dependent on the encoded lyrics information and the proposed conditioning method is effectively applied.

Next, we experimented to see if the network is able to use the phonetic information included in the lyrics. We show this by removing all the phonetic information from the aligned lyrics. In other words, the changed lyrics still contain the vocal activity information. More specifically, it was done by replacing all the voiced sections with random values and leaving the unvoiced sections intact (`VA+Random`). Interestingly, the performance was still far below than the model trained tested on aligned lyrics, which indicates that the network can reflect the phonetic information into the separation process.

It is noteworthy that the SIR scores of `VA+Random`

Figure 7: The examples of the separated vocal spectrograms with incorrect inputs and correct aligned lyrics inputs were given to *LC-model 4*. The dashed line shows the enhanced parts when the aligned lyrics are used. Note that the region is closely related to the formant frequencies.

are not much different from the aligned lyrics input (`AlignedLyrics`). Since SIR scores are heavily related to the remained accompaniment sources of the separated singing voice, we expected that the networks were still capable of removing the accompaniments only using the vocal activity information. On the other hand, the impact on SDR and SAR scores were significant. This implies that while the network was able to erase the accompaniment well by using vocal activity information only in unvoiced sections, it was able to remove the accompaniment better by using phonetic information in voiced sections.

In the experiments of using (`Random`) and (`VA+Random`) inputs, median values of 5 different experimental tries with different random seeds were taken.

6. CONCLUSION

In this study, we proposed an integrated framework of combining the lyrics encoder into the state-of-the-art *Open-unmix* separation network. Local conditioning and concatenation methods were shown to be able to effectively condition the aligned lyrics into the singing voice separation networks. Through various experiments, it was confirmed that the phonetic information of aligned lyrics can contribute to the performance improvements as well as the vocal activity information. We plan to use the unaligned lyrics for the singing voice separation for the future works.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," *arXiv preprint arXiv:1806.03185*, 2018.

[2] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," *arXiv preprint arXiv:1911.13254*, 2019.

[3] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," 2017.

[4] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 106–110.

[5] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix - a reference implementation for music source separation," *Journal of Open Source Software*, 2019. [Online]. Available: https://doi.org/10.21105/joss.01667

[6] E. Cano, G. Schuller, and C. Dittmar, "Pitch-informed solo and accompaniment separation towards its use in music education applications," *EURASIP Journal on Advances in Signal Processing*, vol. 2014, no. 1, p. 23, 2014.

[7] J. F. Woodruff, B. Pardo, and R. B. Dannenberg, "Remixing stereo music with score-informed source separation." in *ISMIR*, 2006, pp. 314–319.

[8] N. Takahashi, M. K. Singh, S. Basak, P. Sudarsanam, S. Ganapathy, and Y. Mitsufuji, "Improving voice separation by incorporating end-to-end speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 41–45.

[9] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "Content based singing voice extraction from a musical mixture," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 781–785.

[10] J. Lee, H.-S. Choi, C.-B. Jeon, J. Koo, and K. Lee, "Adversarially trained end-to-end korean singing voice synthesis system," *arXiv preprint arXiv:1908.01919*, 2019.

[11] C. Gupta, E. Yılmaz, and H. Li, "Acoustic modeling for automatic lyrics-to-audio alignment," *arXiv preprint arXiv:1906.10369*, 2019.

[12] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 181–185.

[13] T.-S. Chan, T.-C. Yeh, Z.-C. Fan, H.-W. Chen, L. Su, Y.-H. Yang, and R. Jang, "Vocal activity informed singing voice separation with the ikala dataset," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 718–722.

[14] Z. Chen, P. Huang, and Y. Yang, "Spoken lyrics informed singing voice separation," in *Proc. HAMR*, 2013.

[15] K. Schulze-Forster, C. Doire, G. Richard, and R. Badeau, "Weakly informed audio source separation," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 273–277.

[16] K. Schulze-Forster, C. S. Doire, G. Richard, and R. Badeau, "Joint phoneme alignment and text-informed speech separation on highly corrupted speech," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7274–7278.

[17] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[20] H. Tachibana, K. Uenoyama, and S. Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4784–4788.

[21] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of*

machine learning research, vol. 15, no. 1, pp. 1929–1958, 2014.

[23] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[24] A. Liutkus, F.-R. Stöter, Z. Rafii, D. Kitamura, B. Rivet, N. Ito, N. Ono, and J. Fontecave, "The 2016 signal separation evaluation campaign," in *International conference on latent variable analysis and signal separation*. Springer, 2017, pp. 323–332.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[27] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation: 14th International Conference, LVA/ICA 2018, Surrey, UK*, 2018, pp. 293–305.

SCORE FOLLOWING WITH HIDDEN TEMPO USING A SWITCHING STATE-SPACE MODEL

Yucong Jiang
University of Richmond
`yjiang3@richmond.edu`

Christopher Raphael
Indiana University Bloomington
`craphael@indiana.edu`

ABSTRACT

A score-following program traces the notes in a musical score during a performance. This capability is essential to many meaningful applications that synchronize audio with a score in an on-line fashion. Existing algorithms often stumble on certain difficult cases, one of which is piano music. This paper presents a new method to tackle such cases. The method treats tempo as a variable rather than a constant (with constraints), allowing the program to adapt to live performance variations. This is first expressed by a Kalman filter model at the note level, and then by an almost equivalent switching state-space model at the audio frame level. The latter contains both discrete and continuous hidden variables, and is computationally intractable. We show how certain reasonable approximations make the computation manageable. This new method is tested on a dataset of 50 piano excerpts. Compared with a previously established state-of-the-art algorithm, the new method shows more stable and accurate results: it reduces fatal score-following errors, and improves accuracy from 65.0% to 69.1%.

1. INTRODUCTION

The score-following problem involves building a computer program that can trace musical events in a given musical score during a live performance. This is called "on-line audio-to-score alignment", which constantly figures out the current position in the score while the performance is going on; the program can only access the audio received before the current moment. In contrast, *off*line audio-to-score alignments start the task *after* the performance is done, allowing the program to access the entire recording.

Score following enables a number of useful applications: a musical score page turner [1], automatic accompaniment systems [2], virtual scores designed to react to a live performance [3], real-time audio enhancement during a music performance, and even a computer tutor [4].

A typical score-following algorithm infers the score positions by evaluating the hypothesis paths through a state graph, each path representing a possible performance. From the Bayesian point of view, the evaluation criteria

for a path include two aspects: how much a hypothesis is consistent with the *prior model*, which represents the anticipation of the musical performance *before* observing any data, and how much the data support a hypothesis, which is called the *data model*. This paper focuses on improving the former aspect, which is essentially about timing—when a note appears and how long it lasts.

Many researchers have considered timing in their prior models. [5, 6] used a hidden semi-Markov model where the duration of each state represents a note length. Others chose to directly model the tempo as a latent variable: [7, 8] treated the tempo as a discrete variable, while [9–12] adopted continuous state-space models, and used particle filter to approximate the results. [13] developed a hybrid graphical model, and tested it by aligning orchestra music offline. [13] provides a jumping-off point for the proposed method here. Note that feature-based methods (e.g., DTW or DNN) are beyond the scope of this discussion.

Unfortunately, existing score-following algorithms can still stumble on some challenging cases, especially when the data model is not reliable; e.g., shared notes among neighboring chords, extended sound from previous chords by pedaling, and blurring effects caused by fast playing, as in piano music. This paper presents a new method aiming to improve the timing model—this aspect is especially meaningful in those challenging cases. In practice, we can assume that the tempo tends to be smooth: the tempo is steady most of the time, sometimes floating around slowly, but rarely jumps up and down abruptly. Therefore, the new method models the tempo as a continuous variable, and it is smooth. This allows the note lengths (or the local tempo) to adapt to the performance data. One of the most popular state-space models, the Kalman filter model, is suitable for tracking such a tempo variable (Section 3.1). It becomes a *switching* state-space model after changing the scope of time (Section 3.2). Section 4 shows how the increased computational complexity is manageable with approximations. This new method was tested on real piano performance data presented in Section 5.

2. REPRESENTING SCORE AND AUDIO

We can view the musical score in a "homophonic" way, representing the score as a sequence of chords, as in Figure 1. It enables polyphonic music to be linearly represented as the same fashion as in monophonic music—a sequence of chords, each chord associated with a score position.

Figure 1. "Homophonic" view of polyphonic music [14]. The left bar is the original score with two voices. The right bar is its "homophonic" view.

Figure 2. Dependency graph for the tempo and the onset.

The audio is sampled, and evenly segmented into frames, with some overlap between adjacent frames. Each frame is transformed into a Fourier spectrum [15]. The *data model* here defines the likelihood of observing the data spectrum given the chord index. Denote y as the spectrum of a frame, a vector $\{y_w\}$, $1 \leq w \leq W$. Denote the template (see [13]) of the kth chord as $q^k = \{q_w^k\}$, which sums to 1. The likelihood of observing the data y given the index k is:

$$p(y|k) = \prod_{w=1}^{W} (q_w^k)^{y_w'} \tag{1}$$

where $y_w' = y_w / \sum_{w=1}^{W} y_w$.

3. THE MODEL

3.1 Kalman Filter Model for Tempo

The polyphonic score is represented as a sequence of chords, with a new chord appearing whenever any note is added, ended, or changed in the current chord. Let's assume there are K such chords in the musical score, each chord with a nominal musical length (e.g., 1/4 for a quarter note and 1 for a whole note) represented by l_k, $k = 1, \ldots, K$. We assume every chord has its own tempo value, and it doesn't change within a chord's lifetime. Let t_k be the tempo of the kth chord (seconds per whole note), and o_k be this chord's onset time (in seconds). The joint evolution of the tempo and the onset is modeled as a linear dynamical system. If we treat the onsets as observable data, the formula is the same as a Kalman filter model:

$$o_{k+1} = o_k + l_k t_k + \varepsilon_{k+1} \tag{2}$$
$$t_{k+1} = t_k + \eta_{k+1} \tag{3}$$
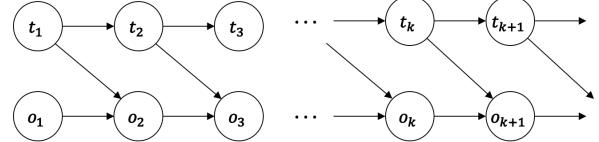
where all random variables have normal distributions:

$$o_1 \sim N(\mu_{o,1}, \sigma_{o,1}^2)$$
$$t_1 \sim N(\mu_{t,1}, \sigma_{t,1}^2)$$
$$\varepsilon_k \sim N(0, \sigma_{\varepsilon,k}^2), \ k = 2, \ldots, K$$
$$\eta_k \sim N(0, \sigma_{\eta,k}^2), \ k = 2, \ldots, K$$

The ε_k's and η_k's are all mutually independent, and they are independent from o_1 and t_1 as well. (In the experiments, however, $\sigma_{\varepsilon,k}$ was modified to be proportional to t_k, and $\sigma_{\eta,k}$ proportional to $l_k t_k$, with manually set scales.) The dependency graph is in Figure 2. In the rest of this paper, we refer to a "chord" as a "note" for simplicity's sake.

3.1.1 Marginal Likelihood of Onsets

This linear dynamical system can be viewed as a Markov process of generating the onsets, $o_1^K = (o_1, o_2, \ldots, o_K)$,

demonstrated as follows. According to the chain rule and the described model, we can write

$$p(o_1^K) = p(o_1) \prod_{k=1}^{K-1} p(o_{k+1}|o_1^k)$$

$$= p(o_1) \prod_{k=1}^{K-1} \int_{t_k} p(o_{k+1}|t_k, o_k) p(t_k|o_1^k) \, \mathrm{d}t_k$$

Because $o_{k+1} = o_k + l_k t_k + \varepsilon_{k+1}$, the integral factors can be further simplified as

$$\int_{t_k} N(o_{k+1}; o_k + l_k t_k, \sigma_{\varepsilon,k+1}^2) N(t_k; \mu_t(o_1^k), \sigma_t^2(o_1^k)) \, \mathrm{d}t_k$$

$$= N(o_{k+1}; o_k + l_k \mu_t(o_1^k), \sigma_{\varepsilon,k+1}^2 + l_k^2 \sigma_t^2(o_1^k))$$

where $\mu_t(o_1^k) = E(t_k|o_1^k)$ and $\sigma_t^2(o_1^k) = Var(t_k|o_1^k)$, which can be iteratively calculated using a Kalman filter as the system receives $o_1, o_2, \ldots, o_k$ [16]. Thus, we have

$$p(o_1^K) = \tag{4}$$
$$p(o_1) \prod_{k=1}^{K-1} N(o_{k+1}; o_k + l_k \mu_t(o_1^k), \sigma_{\varepsilon,k+1}^2 + l_k^2 \sigma_t^2(o_1^k))$$

which can be computed iteratively as k increases.
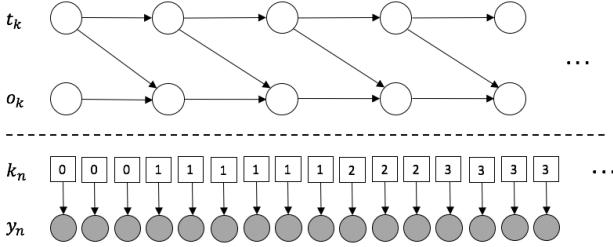
3.2 Frame-wise Representation

The discussions in Section 3.1 is based on the linear dynamical system at the note level, as in Figure 2 where the discretized "time step" is the note index. However, in real-world applications, the audio is received frame by frame (every 16 milliseconds in the experiments). In order to incorporate such audio frames as observed data, we have to change the scope and view the model in Figure 2 at the *frame* level, with each audio frame as a time step. Let's use n to denote the index of a frame, $n = 1, \ldots, N$, where N is the total number of frames in the audio. Any frame, n, has a label variable, $k_n \in \{0, \ldots, K\}$, which is the index of the sounding note at that frame. Frames before the first note being played are labeled as 0, i.e., $\{n : k_n = 0\}$. Denote y_n as the observed audio at the nth frame. We can assume the distribution of the audio frame data only depends on this frame's label—which note is being played. Figure 3 shows the model at both levels in the same graph.

The note onsets and the frame labels are nearly interchangeable. Let Δ be the time difference between adjacent frames (in milliseconds), a sequence of frame labels, k_1^n, can be recovered from a sequence of onsets, o_1^k, or vice versa, like this:

$$k_n = \min \left\{ k \in \{0, \ldots, K\} : n\Delta < o_{k+1} \right\} \tag{5}$$
$$o_k \approx \Delta \cdot \min \left\{ n \in \{1, \ldots, N\} : k_n = k \right\} \tag{6}$$

694

Figure 3. Dependency graph. Upper panel is note level; lower panel is frame level. Circles are continuous variables; squares are discrete. Observed variables are shaded.

The note-wise representation and the frame-wise representation are almost equivalent, except that there are two additional assumptions in the latter. First, the note onsets are now discrete because they have to be multiples of Δ, as in Equation 6. Second, a note must last at least one frame long, so $o_{k+1} - o_k \geq \Delta$. Because the labels and the onsets can be derived deterministically from each other as in the above two equations, the onsets can actually be viewed as "discrete" variables, and the tempo variables are the only real continuous (hidden) variables. Such state-space models that involve both discrete and continuous hidden variables are called *switching* state-space models [17].

3.2.1 Marginal Likelihood of Labels

This section describes a generative model for the frame labels, k_1^N, (almost) equivalent to the generative model in Section 3.1.1, but at the frame level. Using the chain rule, we have

$$p(k_1^N) = p(k_1) \prod_{n=1}^{N-1} p(k_{n+1}|k_1^n) \tag{7}$$

Following the assumptions in the frame-wise representation, there could be only two possible values for k_{n+1} in each factor $p(k_{n+1}|k_1^n)$: the same as k_n if it "decides" to stay at the current note, or $k_n + 1$ if it "decides" to move on to the next note. From Equation 4, we know that the onset of the pending note $k_n + 1$, given all previous onsets $o_1^{k_n}$ (equivalent to k_1^n), has a density function

$$p(o_{k_n+1}|o_1^{k_n}) = \tag{8}$$
$$N(o_{k_n+1}; o_{k_n} + l_{k_n}\mu_t(o_1^{k_n}), \sigma_{\varepsilon,k_n+1}^2 + l_{k_n}^2 \sigma_t^2(o_1^{k_n}))$$

Let's write ϕ as the standard normal density, and define

$$f(x) = \phi(\frac{x-\mu}{\sigma})$$

Then, the density in Equation 8 is $f(o_{k_n+1})$, where

$$\mu = o_{k_n} + l_{k_n}\mu_t(o_1^{k_n})$$
$$\sigma = \sqrt{\sigma_{\varepsilon,k_n+1}^2 + l_{k_n}^2 \sigma_t^2(o_1^{k_n})}$$

We can use this density function to compute $p(k_{n+1}|k_1^n)$ with two cases:

$$p(k_{n+1}|k_1^n) = \begin{cases} p_1, & k_{n+1} = k_n \ (same \ note) \\ 1 - p_1, & k_{n+1} = k_n + 1 \ (new \ note) \end{cases} \tag{9}$$

We can focus on calculating the first case, and the second case has the complimentary probability. In the first case where it stays in the same note at frame $n+1$, the information we know is that the onset of the next note will be *after* frame $n+1$, given we already know that the onset must be after frame n. Therefore, p_1 is a conditional probability:

$$p_1 = P(o_{k_n+1} > (n+1)\Delta \mid o_{k_n+1} > n\Delta, o_1^{k_n}) \tag{10}$$

Writing Φ as the cumulative distribution function of ϕ, we have

$$p_1 = \frac{1 - \Phi(\frac{c+\Delta-\mu}{\sigma})}{1 - \Phi(\frac{c-\mu}{\sigma})} \tag{11}$$

where $c = (n+1)\Delta$. Therefore, we can calculate the probability of any label sequence $p(k_1^N)$ iteratively as in Equation 7, by using Equation 9.

3.2.2 Note Duration and Note Age

In Equation 9, $p(k_{n+1}|k_1^n)$ is calculated based on $f(o_{k_n+1})$, the distribution of the onset for the pending note. In this section, we introduce two variables—note duration and note age—and calculate $p(k_{n+1}|k_1^n)$ from a slightly different perspective.

The duration of the kth note is the difference between its two adjacent onsets:

$$L_k = o_{k+1} - o_k = l_k t_k + \varepsilon_{k+1}$$

Given the previous onsets, L_k also has a Gaussian distribution with its mean as $l_k\mu_t(o_1^k)$ and its variance as $\sigma_{\varepsilon,k+1}^2 + l_k^2\sigma_t^2(o_1^k)$. In the frame-wise representation, a note's duration, L_{k_n}, has additional requirements that it should be multiples of Δ, and be at least Δ (milliseconds) long. Let's define a note's age as the number of frames this note has been through so far at the nth frame, denoted as a_n. The age of the note k_n can be calculated by

$$a_n = n - o_{k_n}/\Delta + 1$$

To get p_1 in Equation 9, we can rewite Equation 11 from the angle of the note length: given that this note has lasted a_n frames, what's the probability of it lasting for at least $a_n + 1$ frames. It can be expressed as

$$p_1 = P(L_{k_n} \geq (a_n+1)\Delta \mid L_{k_n} \geq a_n\Delta, o_1^{k_n})$$
$$= \frac{1 - \Phi(\frac{(a_n+1)\Delta-\mu}{\sigma})}{1 - \Phi(\frac{a_n\Delta-\mu}{\sigma})} \tag{12}$$
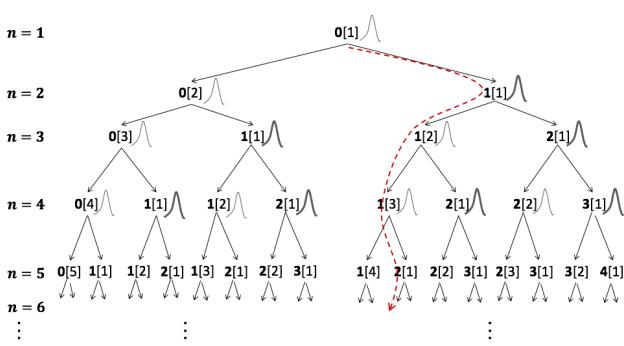
where $\mu = l_{k_n}\mu_t(o_1^{k_n})$ and $\sigma = \sqrt{\sigma_{\varepsilon,k_n+1}^2 + l_{k_n}^2\sigma_t^2(o_1^{k_n})}$.

4. COMPUTATION

The number of possible label sequences grows exponentially with n, as in the tree structure in Figure 4. This section discusses the computational aspects of the model: what is the filtered probability of the hidden variables, given all observed audio data up to the current frame; how to reasonably approximate the calculation so it is tractable.

4.1 Tree Representation

The tree in Figure 4 represents all possible label sequences, $\{k_1^N\}$. At any frame n, each node has a label for its note index, k_n, and also includes the age information of the note—the number of frames it has been through so far—denoted by a_n. From a label sequence k_1^n, we can thus determine the age sequence a_1^n, and vice versa. The tree includes the age variable because we need it in the generative model (see Equation 12). A node has two children: left means it stays in the same note and thus the age increases by one frame, and right means it moves on to the next note and thus the age resets to 1. Any node in the tree actually represents a label sequence by the path from the root to the node. For example, the dotted line in Figure 4 represents the label sequence of $k_1^6 = 011122$ (or the age sequence of $a_1^6 = 112312$). A path can be viewed as a sequence of decisions of choosing the left or right branch, from the root node at frame 1 to the end node in the path.

Figure 4. Exponential growth of label sequences. Each node has three aspects: note index label, age of this note (in square brackets), and distribution of the tempo. Tempo distributions at *onset* nodes are drawn with thicker lines.

For every node in the tree, denoted by its path of k_1^n, we can calculate two probabilities: the discrete probability of arriving at this node, $p(k_1^n)$, and the continuous probability distribution of the tempo at this node, $p(t_{k_n}|k_1^n)$. As discussed before, the tempo has a Gaussian distribution (drawn besides the nodes in the first four frames in Figure 4), given the path leading to this node. A Kalman filter keeps track of the tempo down the path, and updates its distribution when and only when the path chooses a right branch (drawn with thicker lines), i.e., whenever it observes a note onset (discussed in Section 3.1).

The probability of arriving at a node, $p(k_1^n)$, according to the frame-wise generative model, can be iteratively calculated by $p(k_1^n) = p(k_1^{n-1})p(k_n|k_1^{n-1})$. Thus, $p(k_1^n)$ can be calculated from two parts: the probability of arriving at its parent node, $p(k_1^{n-1})$, and the probability of choosing the left or the right branch when transitioning from frame $n-1$ to frame n, $p(k_n|k_1^{n-1})$. We can use either Equation 11 or Equation 12 to calculate the latter, and both equations require the distribution of the tempo at the parent node, $p(t_{k_{n-1}}|k_1^{n-1})$—or equivalently, $p(t_{k_{n-1}}|o_1^{k_{n-1}})$.

Adopting the iterative nature of the calculation, we can compute the probability of (arriving at) every node and its

tempo distribution, frame by frame starting from the root. Since the tree includes every possible label sequence, those probabilities give us $p(k_1^n)$ for all $\{k_1^n\}$, $n = 1, \ldots, N$.

4.2 Conditioning on data

This section continues to focus on calculating the probability of arriving at a node in the tree, but now *conditioned* on the observed audio data up to the current frame—$p(k_1^n|y_1^n)$. Considering y_1^n ensures that sequences more consistent with the observed data would receive higher probabilities than the rest, helping identify more likely sequences. On the other hand, the tempo will not be affected by the observed data since the tempo distribution at a node depends only on the corresponding label sequence, i.e., $p(t_{k_n}|k_1^n, y_1^n) = p(t_{k_n}|k_1^n)$, as used later in Equation 13.

The audio frame data $y_1, \ldots, y_n$ are assumed to be conditional independent from each other given the frame labels $k_1, \ldots, k_n$, so we can have

$$p(k_1^n|y_1^n) = \frac{1}{Z_n}p(k_1^n, y_1^n)$$
$$= \frac{1}{Z_n}p(k_1^n)p(y_1^n|k_1^n)$$
$$= \frac{1}{Z_n}p(k_1^n)\prod_{i=1}^{n}p(y_i|k_i)$$
$$Z_n = p(y_1^n) = \sum_{\{k_1^n\}}p(k_1^n, y_1^n)$$

The joint probability $p(k_1^n, y_1^n)$ can be calculated iteratively from $p(k_1^{n-1}, y_1^{n-1})$ by

$$p(k_1^n, y_1^n) = p(k_1^{n-1}, y_1^{n-1})p(k_n|k_1^{n-1})p(y_n|k_n)$$

The calculation of the middle factor $p(k_n|k_1^{n-1})$ is discussed in Equations 9, 11, and 12, which involve using the tempo distribution of $p(t_{k_{n-1}}|k_1^{n-1})$. The third factor $p(y_n|k_n)$ is the data likelihood of the nth frame, discussed in Section 2. Therefore, we can calculate $p(k_1^n, y_1^n)$ for every node in the tree iteratively from frame 1 to frame N. To get the filtered probability of $p(k_1^n|y_1^n)$, we simply normalize $p(k_1^n, y_1^n)$ with Z_n at every frame. In sum, with the help of the data model factors $\prod_{i=1}^{n}p(y_i|k_i)$, we should be able to distinguish the sequences better by using $p(k_1^n|y_1^n)$ instead of $p(k_1^n)$: hypothesized sequences that are closer to the true sequence should have larger values of $p(k_1^n|y_1^n)$ than those of sequences further from the truth.

4.3 Filtering with Approximation

The task of filtering is to compute $p(k_n, t_{k_n}|y_1^n)$: the joint distribution of the last hidden states in the sequence, given the sequence of observed data so far. Writing $\mathcal{K}(k_n)$ as the set of all label sequences in Figure 4 that are n-label long and end with k_n, this filtered probability is:

$$p(k_n, t_{k_n}|y_1^n) = \sum_{k_1^n \in \mathcal{K}(k_n)}p(k_1^n, t_{k_n}|y_1^n)$$
$$= \sum_{k_1^n \in \mathcal{K}(k_n)}p(k_1^n|y_1^n)p(t_{k_n}|k_1^n) \quad (13)$$

This is a Gaussian mixture distribution with $|\mathcal{K}(k_n)|$ components. Without approximation, the computation is intractable because $|\mathcal{K}(k_n)|$ grows exponentially with n.
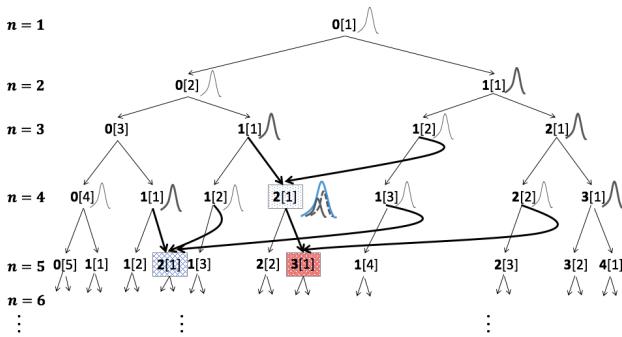
In Figure 4, there are duplicated nodes in terms of "label[age]" at every level (except for the first three levels), and these duplicates have the same subtree. The idea for simplifying the computation is to merge any duplicates at the same level into one node, as in Figure 5. The merged node then has the summed probability:

$$p(\text{merged node}) = \sum\nolimits_{\text{nodes}} p(\text{node}_i)$$

Each node in the frame-wise model also carries a Gaussian distribution for the tempo. The merged node would then carry a Gaussian mixture distribution:

$$p(t_{\text{merged}}) = \sum\nolimits_{\text{nodes}} \frac{p(\text{node}_i)}{p(\text{merged node})} p(t \mid \text{node}_i)$$

As more and more merging operations happen over time, the number of components in a tempo distribution grows exponentially, with each component corresponding with a possible label sequence, and the problem remains intractable. We can solve this by using *moment matching*: approximating a Gaussian mixture with a single Gaussian that has the same mean and variance as the mixture. This is proven to be the optimal approximation in the sense of Kullback-Leibler distance [18]. This approximation is reasonable if the components with larger weights are close to each other (agreeing with each other) in terms of mean and variance, and if those further away have smaller weights, thus could be ignored anyway. It's reasonable to believe that reality more often reflects this case, because unlikely nodes tend to have smaller probabilities and thus smaller weights in the mixture (thus should be ignored anyway), while more likely nodes tend to have more similar opinions about the tempo because they lean towards the truth.

Figure 5. Limiting tree growth by merging nodes with the same label and age. The first merge happens at frame #4, when the right child of 1[1] and the right child of 1[2] (at frame #3) merges into one node. This merged node's tempo has a Gaussian mixture distribution with two components from the two copies of the 2[1], and is approximated by a single Gaussian (thicker blue line). The two merges at frame #5 have similar approximations.

Under this strategy, it's guaranteed that a left child has no duplicates at a frame. All left children's ages are

at least two (frames), because they represent continuing notes. Merging nodes are always the right children of their parent nodes, and these nodes always have the age of 1. For nodes identified as $k_n[1]$ at frame n, their parent nodes must have the label of $k_n - 1$ and could have the age of 1, 2, $\ldots$, $n - k_n$ from frame $n - 1$. Therefore, there are $n - k_n$ copies of nodes $k_n[1]$ being merged together at frame n. It's worth noting that the size of the merging group, $n - k_n$, could be 1, which means there is no merging happening.

Distinguishing those two cases, we can iteratively approximate the discrete filtered probability of the label and the age, $p(k_n, a_n | y_1^n)$, and the continuous filtered probability of the tempo, $p(t_{k_n} | k_n, a_n, y_1^n)$. From previous discussions, we know that $p(t_{k_n} | k_n, a_n, y_1^n)$ is always a single Gaussian (after approximation), and we write its mean and variance as $\mu_n = \mu(k_n, a_n, y_1^n)$ and $\sigma_n^2 = \sigma^2(k_n, a_n, y_1^n)$.

The discrete and the continuous filtered probabilities can be iteratively calculated as follows in two cases. In the case where it stays in the same note, so $k_n = k_{n-1}$ and $a_n > 1$, there is only one parent node (without merging):

$$p(k_n, a_n | y_1^n) = \frac{1}{W_n} p(k_n, a_n, y_n | y_1^{n-1})$$
$$= \frac{1}{W_n} g(k_{n-1}, a_{n-1}, k_n, a_n, y_1^n) \quad (14)$$

$$g(k_{n-1}, a_{n-1}, k_n, a_n, y_1^n) = \quad (15)$$
$$p(k_{n-1}, a_{n-1} | y_1^{n-1}) p(k_n, a_n | k_{n-1}, a_{n-1}, y_1^{n-1}) p(y_n | k_n)$$
$$W_n = p(y_n | y_1^{n-1}) = \sum_{\{k_n, a_n\}} p(k_n, a_n, y_n | y_1^{n-1})$$

The middle factor in Equation 15 has been discussed in Equations 9 and 12, and the third factor is the data model. The filtered tempo in this case doesn't change, i.e.,

$$p(t_{k_n} | k_n, a_n, y_1^n) = p(t_{k_{n-1}} | k_{n-1}, a_{n-1}, y_1^{n-1}) \quad (16)$$

In the other case where it transitions to the next note, so $k_n = k_{n-1} + 1$ and $a_n = 1$, the only difference for the discrete filtered probability is that there is a merging process represented by the summation operation:

$$p(k_n, a_n | y_1^n) = \frac{1}{W_n} \sum_{\substack{1 \le a_{n-1} \\ \le n - k_n}} g(k_{n-1}, a_{n-1}, k_n, a_n, y_1^n) \quad (17)$$

The filtered tempo becomes a Gaussian mixture:

$$p(t_{k_n} | k_n, a_n, y_1^n) \quad (18)$$
$$= \frac{1}{p(k_n, a_n | y_1^n) W_n} \sum_{\substack{1 \le a_{n-1} \\ \le n - k_n}} g(k_{n-1}, a_{n-1}, k_n, a_n, y_1^n) \cdot$$
$$p(t_{k_n} | o_{k_n} = n, o_{k_{n-1}} = n - a_{n-1}, \mu_{n-1}, \sigma_{n-1}^2)$$

The last term can be calculated by the Kalman filter in Section 3.1. We further approximate the above Gaussian mixture using a single Gaussian with the mean as μ_n and the variance as σ_n^2. Putting Equations 14-18 together, we can iteratively calculate $p(k_n, a_n, t_{k_n} | y_1^n)$ by

$$p(k_n, a_n, t_{k_n} | y_1^n) = p(k_n, a_n | y_1^n) \, p(t_{k_n} | k_n, a_n, y_1^n)$$

5. PRELIMINARY EXPERIMENTS

5.1 Data Set

In score-following , the ground truth is the note onset times in the performance audio. This is usually difficult to obtain without tedious work of hand labeling or correction. One idea is to record the performances on pianos that capture movements of the keys, hammers and pedals, and store the information in MIDI files (e.g., *Disklavier* pianos). We can infer the note onsets from a MIDI file fairly easily. The MAESTRO data set by [19] contains such audio and MIDI data from 172 hours of piano performances from the International Piano-e-Competition [20].

The preliminary experiments contained 50 excerpts of real performances from 14 solo-piano pieces, as shown in Table 1. A typical excerpt lasted from 40 sec. to 90 sec., making the entire data set 48 min. of music. About 33 min. of it was from the MAESTRO, and the rest from the publicly available music recordings on the Internet. In the former case, we matched the performance MIDI data with the digital score according to the minimum-edit-distances criterion, with some minor manual corrections. For the latter case, we ran an offline audio-to-score algorithm which generated "close to perfect" results, and then manually corrected them. The audio data, along with a detailed description of the measures in these pieces are available at http://music.informatics.indiana.edu/papers/ismir20/.

Composer	Piece	#Excerpts
Mozart	Piano Concerto No. 17 in G major, mvmt1	3
Schumann	Piano Concerto in A minor, mvmt1	3
Chopin	Barcarolle, Op. 60	2
Chopin	Prelude, Op. 28 No. 4	2
Chopin	Ballade No. 1	8
Liszt	*La campanella*	5
Rachmaninoff	Prelude, Op. 3, No. 2	5
Schubert	Six Moments, D. 780 No. 2	1
Schubert	Ständchen, D 957 No. 4 from Schwanengesang	4
Debussy	Prelude, No. 2 (Voiles)	1
Debussy	*La fille aux cheveux de lin*	3
Beethoven	Piano Sonata No. 8 (Sonata Pathétique)	1
Beethoven	Piano Sonata No. 31	8
Haydn	Piano Sonata No. 24 in D major, mvmt1	1
Haydn	Piano Sonata No. 24 in D major, mvmt2&3	3

Table 1. Piano music used in the experiments.

5.2 Evaluation Method

Write $\kappa_1, \ldots, \kappa_N$ as the ground truth index labels of all audio frames. At any frame, the probability of recognizing the truth note is then the sum of the filtered probabilities with the correct label (regardless of age), as in Equation 19. The overall accuracy on an excerpt is the average across all frames. This is called the *frame-wise accuracy* [14], accounting for the accuracy of every frame.

$$Acc_n = \sum_{\substack{k_n = \kappa_n \\ 1 \le a_n \le n - \kappa_n + 1}} p(k_n, a_n | y_1^n) \quad (19)$$

$$Acc = \sum_n Acc_n / N \quad (20)$$

5.3 Results

The baseline algorithm for comparison was Music Plus One [21], a state-of-the-art score-following systems, based on a hidden Markov model. Both algorithms used $8kHz$ sampling, 512-sample frame size (64 ms), and 128-sample hop size. Both algorithms also deployed the *beam search* technique to limit hypotheses at each frame to $\le$200.

Out of the 50 excerpts, 12 excerpts had "very low" accuracies ($< 40\%$) by at least one of the two algorithms. Very low accuracy can either exemplify a fatal error, in which the program got lost at certain frames, and never found its way back, or it can mean high uncertainty among neighboring chords. As shown in Table 2, the proposed method failed exactly two times fewer than the baseline, and it also had higher average accuracy among the failed excerpts.

	baseline	proposed
# failed excerpts	11	9
average accuracy	15.1%	22.1%

Table 2. Counts of low-accuracy excerpts.

Excluding those 12 excerpts, we calculated the average overall accuracy across all of the other 38 excerpts, as shown in Table 3. The proposed method achieved 4.1% higher accuracy than the baseline. A p-value of .0096 ($\alpha <$ .01) on a paired t-test indicates that the proposed method is measurably better than the baseline.

	baseline	proposed
average accuracy	65.0%	69.1%

Table 3. Average accuracies of 38 excerpts.

6. DISCUSSION AND CONCLUSION

Piano music is one of the most challenging cases in the score-following realm, as the preliminary experiments indicate: the baseline algorithm failed on 22% of excerpts. Further investigation suggests that the proposed method is more robust than the baseline: fewer fatal errors, easier recovery from mistakes, and successful following even when the performance tempo was far from the default tempo. With this evidence and significantly improved accuracy on successfully followed excerpts, we can speculate that treating the tempo as a variable helps the program adapt to unpredictable performance variations, and that modeling the tempo as smooth helps discriminate among hypotheses.

Although the dataset is not large enough to draw general conclusions, the preliminary experiments showed strong promise in the direction of tracking tempo while following a score. The presented method here is general and can be applied to a variety of instruments, monophonic or polyphonic. The tracked tempo information is also meaningful for anticipating the next note in automatic accompaniment systems, and is scalable to analyze the timing aspects of large numbers of performances.

In conclusion, this paper presents an interesting new method for improved score-following, and suggests a promising direction for future research endeavors.

7. REFERENCES

[1] A. Arzt, G. Widmer, and S. Dixon, "Automatic page turning for musicians via real-time machine listening." in *ECAI*, 2008, pp. 241–245.

[2] R. B. Dannenberg and C. Raphael, "Music score alignment and computer accompaniment," *Communications of the ACM*, vol. 49, no. 8, pp. 38–43, 2006.

[3] A. Cont, "On the creative use of score following and its impact on research," in *SMC*, 2011.

[4] E. Schoonderwaldt, A. Askenfelt, and K. F. Hansen, "Design and implementation of automatic evaluation of recorder performance in IMUTUS," in *In Proceedings of the International Computer Music Conference (ICMC)*, 2005.

[5] A. Cont, "A coupled duration-focused architecture for real-time music-to-score alignment," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 6, pp. 974–987, 2010.

[6] P. Cuvillier, "On temporal coherency of probabilistic models for audio-to-score alignment," Ph.D. dissertation, Université Pierre et Marie Curie-Paris VI, 2016.

[7] C. Joder, S. Essid, and G. Richard, "A conditional random field viewpoint of symbolic audio-to-score matching," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 871–874.

[8] ——, "Hidden discrete tempo model: A tempo-aware timing model for audio-to-score alignment," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 397–400.

[9] N. Montecchio and A. Cont, "A unified approach to real time audio-to-score and audio-to-audio alignment using sequential Montecarlo inference techniques," in *ICASSP 2011: Proceedings of International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2011, pp. 193–196.

[10] T. Otsuka, K. Nakadai, T. Takahashi, T. Ogata, and H. Okuno, "Real-time audio-to-score alignment using particle filter for coplayer music robots," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 1, p. 384651, 2011.

[11] F. Korzeniowski, F. Krebs, A. Arzt, and G. Widmer, "Tracking rests and tempo changes: Improved score following with particle filters," in *ICMC*, 2013.

[12] Z. Duan and B. Pardo, "A state space model for online polyphonic audio-score alignment," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 197–200.

[13] C. Raphael, "Aligning music audio with symbolic scores using a hybrid graphical model," *Machine learning*, vol. 65, no. 2-3, pp. 389–409, 2006.

[14] Y. Jiang and C. Raphael, "Piano score-following by tracking note evolution." in *SMC*, 2019.

[15] K. Gröchenig, *Foundations of time-frequency analysis*. Springer Science & Business Media, 2013.

[16] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*. Oxford university press, 2012.

[17] Z. Ghahramani and G. E. Hinton, "Variational learning for switching state-space models," *Neural computation*, vol. 12, no. 4, pp. 831–864, 2000.

[18] A. R. Runnalls, "Kullback-Leibler approach to Gaussian mixture reduction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 989–999, 2007.

[19] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," 2018.

[20] "International piano-e-competition," http://piano-e-competition.com.

[21] C. Raphael, "Music Plus One and Machine Learning." in *ICML*, 2010, pp. 21–28.

UNSUPERVISED DISENTANGLEMENT OF PITCH AND TIMBRE FOR ISOLATED MUSICAL INSTRUMENT SOUNDS

Yin-Jyun Luo[1,2]   **Kin Wai Cheuk**[1,2]   **Tomoyasu Nakano**[3]
Masataka Goto[3]   **Dorien Herremans**[1,2]

[1] Singapore University of Technology and Design (SUTD), Singapore
[2] Institute of High Performance Computing, A*STAR, Singapore
[3] National Institute of Advanced Industrial Science and Technology (AIST), Japan

`{yinjyun_luo, kinwai_cheuk}@mymail.sutd.edu.sg`

ABSTRACT

Disentangling factors of variation aims to uncover latent variables that underlie the process of data generation. In this paper, we propose a framework that achieves unsupervised pitch and timbre disentanglement for isolated musical instrument sounds without relying on data annotations or pre-trained neural networks. Our framework, based on variational auto-encoders, takes as input a spectral frame, and encodes pitch and timbre as categorical and continuous variables, respectively. The input is then reconstructed by combining those variables. Under an unsupervised training setting, a major challenge is that encoders are tasked to capture factors of interest with distinct latent representations, without access to the corresponding ground-truth labels. We therefore introduce auxiliary tasks and objectives which leverage pitch shifting as a strategy to create surrogate labels, thereby encouraging the disentanglement of pitch and timbre. Through an ablation study we analyze the impact of the proposed objectives. The evaluation shows the efficacy of the proposed framework for learning disentangled representations, and verifies its applicability to unsupervised pitch classification and conditional spectral synthesis.

1. INTRODUCTION

The generative process from observed data can be described as having multiple latent factors of variation to explain the observations. For example, we may consider that a musical instrument sound consists of its pitch and timbre characteristic as the major underlying factors of variation. The concern of representation learning is to learn a model that captures such explanatory factors which are expected to be transferable to downstream tasks [1].

Disentanglement is said to be crucial for a good representation [1]. A disentangled representation allocates distinct factors of variation into separate dimensions of the representation, which facilitates an interpretable structure. Interventions along certain dimensions thereby only affect the corresponding latent factors, leading to a sparse change to the observation. In this paper, we propose a framework for learning disentangled representations of pitch and timbre, the two dominant factors of an isolated musical instrument sound. Unlike the supervised frameworks that address similar tasks [2, 3], we do not rely on data annotations or networks pre-trained on any form of supervision.

Many of the recent endeavors to achieve disentangled representation learning in an unsupervised setting are based on variational auto-encoders (VAEs) [4]. VAEs depict a data-generating process $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, where a multivariate latent variable $\mathbf{z}$ is first sampled from a prior distribution $p(\mathbf{z})$, and the observation $\mathbf{x}$ is sampled from the conditional distribution $p(\mathbf{x}|\mathbf{z})$ parameterized by a neural network; a variational distribution $q(\mathbf{z}|\mathbf{x})$, also parameterized using a neural network, is introduced to approximate the true posterior $p(\mathbf{z}|\mathbf{x})$.

In order to achieve disentanglement without access to data annotations, recent studies have proposed to impose regularizations on the latent space to promote a factorized aggregated posterior distribution $q(\mathbf{z})$ [5–7]. These works, however, demand further probes (e.g., traversal of latent space) to identify the semantics of the learned representations. One can also leverage prior knowledge of data structure and inject specific constraints [8–11]. For example, factors of variation of speech or video data are categorized as sequence-level (e.g., speakers) and segment-level (e.g., phonetic contents) latent variables [9, 10]. The mentioned prior knowledge, however, is not trivially applicable to factors of interest lacking of structural hierarchy (e.g., an isolated musical instrument sound with a constant pitch has both timbre and pitch as the sequence-level variable).

Given the challenge of disentangled representation learning in the unsupervised setting, literature has also assumed the accessibility to implicit or weak supervision in the form of grouped or paired data [12–14]. Our proposed framework, in contrast, does not require such a form of supervision; instead, we leverage pitch-shifting to create paired data, thereby introducing auxiliary objective functions to enhance feature disentanglement.

The underlying assumption of the proposed framework

is that a moderate shift of pitch does not alter the timbre of the original musical instrument sound; we can thereby consider the original and its pitch-shifted version as a pair, and introduce several constraints to promote disentanglement of pitch and timbre. In particular, we adapt the contrastive learning method [15] to our framework, and maximize the similarity measure of the paired data. We also employ cycle-consistency loss [16, 17] to further improve the disentanglement. Moreover, we propose an objective function that explicitly accommodates the information of pitch difference that arises from pitch-shifting [18], which plays a key role for performance improvement. An ablation study is conducted to evaluate the efficacy of the introduced objective functions.

We consider a generative process that samples a categorical and a continuous latent variable, referred to as pitch and timbre, respectively, and samples the data conditioned on both the variables. This manifests the discrete nature of pitch and introduces a strong inductive bias crucial to the success of unsupervised disentanglement [19], which is made feasible as each sample in this study corresponds to a pitch class in the equal tempered scale.
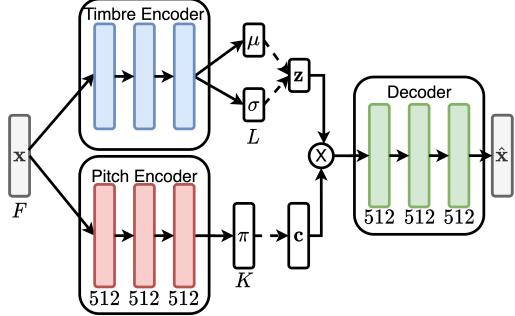
For evaluation, classifiers are built to predict ground-truth pitch and instrument labels, which take as input the learned timbre representation. The low accuracy for pitch, and the high accuracy for instrument indicate a disentangled timbre representation. We also evaluate the pitch latent variable in terms of the metrics used for clustering tasks, which demonstrates the model's capability of unsupervised pitch classification. Attributed to the interpretability of the disentangled representation, we can achieve pitch-conditioning spectral synthesis whereby disentanglement is evaluated through the lens of conditional generation. We also propose a metric that accounts for consistency and diversity of pitch of the generated data. Our main contributions can be summarized as follows:

- Propose a framework based on VAEs to tackle unsupervised disentanglement of pitch and timbre.

- Leverage pitch-shifting which enables the auxiliary objectives that further introduce inductive biases to improve disentanglement.

- Design a metric that accounts for pitch consistency and diversity which quantifies the performance of disentanglement.

We present the proposed framework and the auxiliary objective functions in Section 2, and detail the implementation along with the experimental setup in Section 3. The evaluation methods and the proposed metric are elaborated in Section 4, followed by experimental results and discussions in Section 5. The paper is concluded in Section 6.

2. METHOD

In this section, we describe the proposed framework, and present the auxiliary objective functions that are introduced to further enhance the model.

Figure 1: The proposed framework. The dashed lines denote sampling, and the cross denotes concatenation.

2.1 Overview

Figure 1 illustrates the proposed framework, which depicts a data-generating process of $\mathbf{x} \in \mathbb{R}^F$ being sampled from a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{c})$, referred to as a decoder, where $\mathbf{c} \in \mathbb{R}^K$ is a categorical latent variable for pitch, and $\mathbf{z} \in \mathbb{R}^L$ is a continuous latent variable for timbre. θ denotes the parameters of the decoder. Variational distributions $q_\phi(\mathbf{c}|\mathbf{x})$ and $q_\phi(\mathbf{z}|\mathbf{x})$, referred to as the pitch and timbre encoder, are introduced to approximate the true posterior distributions. The parameters of the two encoders are collectively denoted as ϕ. Under the framework of variational inference, the generative model is optimized through the evidence lower bound (ELBO) of $p_\theta(\mathbf{x})$:

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})q_\phi(\mathbf{c}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{c})] \\ - \mathcal{D}_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\big) - \mathcal{D}_{\mathrm{KL}}\big(q_\phi(\mathbf{c}|\mathbf{x})\|p(\mathbf{c})\big). \tag{1}$$

For the continuous latent variable $\mathbf{z}$, we follow the literature [4] assuming $p(\mathbf{z}) = \mathcal{N}(0, I)$ and $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_\phi(\mathbf{x}), diag(\sigma_\phi^2(\mathbf{x})))$. For the categorical latent variable $\mathbf{c}$, we let $p(\mathbf{c}) = U(0, 1)$, a standard uniform distribution over number of categories K, and $q_\phi(\mathbf{c}|\mathbf{x}) = Cat(\mathbf{c}|\pi_\phi(\mathbf{x}))$. We can treat the pitch encoder as a pitch classifier that can be trained altogether with the entire network without pitch labels.

A major challenge for the unsupervised disentanglement is that the pitch encoder and timbre encoder are tasked to capture pitch and timbre features, respectively, without access to the corresponding labels. The presented model manifests the discrete nature of pitch with the categorical variable, thereby encouraging the pitch encoder to leave timbral information to the timbre encoder.

2.2 Gumbel-Softmax Distribution

In particular, we let $\mathbf{c}_k$ be a one-hot encoding of pitch, indexed at k, that is sampled from the $q_\phi(\mathbf{c}|\mathbf{x})$. To enable back-propagation through sampling of the discrete node, a common technique is to approximate $\arg\max$ with the Gumbel-Softmax distribution [20]. We specifically employ the straight-through estimator, which forward-passes the one-hot vector $\mathbf{c}_k$, and approximates its gradient with that of the Gumbel-Softmax distribution.

2.3 Auxiliary Objective Functions

Based on the underlying assumption that a moderate shift of pitch does not change the timbre of a musical instrument sound, we exploit pitch-shifting, thereby enabling the following auxiliary objective functions to enhance the model. We refer to $\mathbf{x}$ and $\mathbf{x}'$ respectively for the original sample and the pitch-shifted version throughout, and $(\mathbf{z}, \mathbf{c})$ and $(\mathbf{z}', \mathbf{c}')$ are the corresponding latent variables.

2.3.1 Latent Regression

One obvious auxiliary loss function enabled by pitch-shifting would be $\mathcal{L}_{\text{regression}} = \|\mathbf{z} - \mathbf{z}'\|_2^2$, which we include in the ablation study for comparison.

2.3.2 Contrastive Learning

We adapt SimCLR [15], a discriminative approach for representation learning [15, 21, 22], to our generative framework. Particularly, each sample in a minibatch of size N is pitch-shifted randomly upward or downward to a number of semitones, resulting in an augmented minibatch of size $2N$. A positive pair of data is defined as $(\mathbf{x}, \mathbf{x}')$, and the other $2(N-1)$ pairs are treated as negative ones, instead of being explicitly defined. The loss function for a positive pair, indexed as (i, j), is then defined as

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{m=1}^{2N} 1_{[m \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_m)/\tau)}, \quad (2)$$

where $1_{[m \neq i]} \in \{0, 1\}$ is an indicator function evaluated as 1 if and only if $m \neq i$, and τ is a temperature parameter. The final loss $\mathcal{L}_{\text{contrast}}$ is obtained by aggregating $\mathcal{L}_{i,j}$ across all positive pairs. Following SimCLR [15], the cosine similarity is used as the similarity measure $\text{sim}(\cdot, \cdot)$.

Intuitively, the loss function attracts $\mathbf{z}$ and $\mathbf{z}'$ and repels the other negative pairs that are possibly derived from different instruments, which is expected to encourage the timbre encoder to extract pitch-invariant latent variables whereby the disentanglement is improved.

2.3.3 Cycle-consistency Loss

Cycle-consistency has been proposed to address unpaired image-to-image translation between different domains [16], which has been incorporated with VAEs to learn disentangled representations for images [17, 23].

We adopt the approach to further constrain the model and encourage disentanglement. Specifically, let E_p, E_t, and D denote the pitch encoder, the timbre encoder, and the decoder, respectively; whereby the cycle-consistency loss is defined as

$$\begin{aligned} \mathcal{L}_{cycle} =& \|E_t(D(\mathbf{z}, \mathbf{c}'_k)) - \mathbf{z}\|_2^2 + \|E_t(D(\mathbf{z}', \mathbf{c}_k)) - \mathbf{z}'\|_2^2 \\ &+ \text{CE}(E_p(D(\mathbf{z}, \mathbf{c}'_k)), k') + \text{CE}(E_p(D(\mathbf{z}', \mathbf{c}_k)), k), \end{aligned}$$
$$(3)$$

where $k = \arg\max_k q_\phi(\mathbf{c}|\mathbf{x})$ (similar for k'), and $\text{CE}(\cdot, \cdot)$ refers to cross-entropy loss.

Intuitively, given $(\mathbf{z}, \mathbf{c}'_k)$, D should generate a sample embodying timbre $\mathbf{z}$ and pitch category k', such that E_t and E_p can correctly predict $\mathbf{z}$ and k' in order to minimize

the loss (similar for $(\mathbf{z}', \mathbf{c}_k)$). The objective function is expected to enforce D to faithfully render the given conditioning signals, and to further encourage E_t and E_p to encode the respective factors. Empirically, we freeze the weights of E_t and E_p when back-propagating $\mathcal{L}_{cycle}$ as suggested in the literature [24].

2.3.4 Surrogate Label Loss

We also propose to exploit the information of the shifted amount of semitones. Specifically, we minimize $\mathcal{L}_{\text{surrogate}} = \text{CE}(E_p(\mathbf{x}'), y')$. The surrogate label for $\mathbf{x}'$ is $y' = k + \delta$, where $k = \arg\max_k q_\phi(\mathbf{c}|\mathbf{x})$, $\delta \in [-S, S]$ denotes the shifted number of semitones, and S is the maximum amount of shift. This enforces E_p to acknowledge the pitch difference. As shown in Section 5, this loss term plays a key role in reaching the best-performing model.

To sum up, based on the underlying assumption that moderate shift of pitch does not alter timbre characteristics, all the objective functions are made possible thanks to the pitch-shifting strategy. The aggregated objective function to be maximized thereby becomes

$$\begin{aligned} \mathcal{L} = \mathcal{L}(\theta, \phi; \mathbf{x}) - (&\lambda_1 \mathcal{L}_{\text{regression}} + \lambda_2 \mathcal{L}_{\text{contrast}} \\ &+ \lambda_3 \mathcal{L}_{\text{cycle}} + \lambda_4 \mathcal{L}_{\text{surrogate}}), \end{aligned} \quad (4)$$

where λ_1, λ_2, λ_3, and λ_4 denote the weights of each loss term. We conduct an ablation study to investigate the efficacy of each auxiliary objective in terms of the metrics elaborated in Section 4.

3. EXPERIMENTAL SETUP

In this section, we describe the dataset used to evaluate our framework along with the implementation details.

3.1 Dataset

We train the framework using a subset of Studio-On-Line (SOL) [25], which includes 1,885 samples of 12 musical instruments and 82 possible pitches. We resample all the recordings to 22,050Hz, after which they are converted to short-time Fourier transform (STFT) with a 92ms of Hann window and 11ms of hop size. Mel-spectrograms with 256 filterbanks are then derived from power magnitude spectrum of the STFT. The dataset is split into a training set (90%) and validation set (10%), both of which have a same distribution of instruments. The magnitude of the Mel-spectrogram is logarithmically scaled, and min-max normalized within $[-1, 1]$ using the minimum and maximum values in the training set. The normalization is performed corpus-wide to preserve the variety of dynamics.

As a preliminary study, we extract the spectral frame at 200ms from the processed spectrograms, a time instant that usually displays the sustained part of a musical note; a datum is therefore referred to as a spectrum $\mathbf{x} \in \mathbb{R}^{256}$ of a Mel-spectrogram. To facilitate the timbre encoder to extract pitch-invariant features, we further derive 30-dimensional Mel-frequency cepstral coefficients (MFCCs) from the Mel-spectrograms. Therefore, the inputs to the timbre and pitch encoder are $\mathbf{x}_{\text{MFCC}} \in \mathbb{R}^{30}$ and

$x_{\text{Mel-spec}} \in \mathbb{R}^{256}$, respectively. For convenience, we refer x to input data and do not distinguish x_{MFCC} and $x_{\text{Mel-spec}}$ in the text and Figure 1. Note that the reconstruction target for evaluating $p_\theta(x|z, c)$ remains as the Mel-spectrum.

As mentioned in Section 2, pitch-shifting is employed to augment the model by enabling the auxiliary objective functions. This is performed by stretching or shrinking an audio waveform with linear interpolation, which results in pitch-shifting in the frequency domain.

3.2 Implementation Details

Both the pitch and timbre encoders are comprised of three 512-unit fully-connected (FC) layers. They differ in the parametric layer; the pitch encoder outputs a categorical distribution $q_\phi(c|x)$ through a FC layer with number of units equal to $K = 82$, i.e., the number of possible pitches. We henceforth refer c to *pitch category*, which differentiates from the pitch labels y. The timbre encoder, on the other hand, contains a Gaussian parametric layer, which outputs two L-dimensional vectors, $L = 8$, representing mean $\mu_\phi(x)$ and variance $\sigma_\phi^2(x)$.

$c \sim q_\phi(c|x)$ and $z \sim q_\phi(z|x)$ are concatenated as the input to the decoder for reconstructing x. The straight-through Gumbel-Softmax estimator [20] and the reparameterization trick [4] enable gradients to back-propagate through the parametric layers with stochastic gradient descent. The decoder is also composed of three 512-unit FC layers, which finally outputs $\hat{x} \in \mathbb{R}^{256}$. Except for the two parametric layers, we use `tanh` as the activation function, and batch normalization follows thereafter.

All the experiments are conducted with a batch size of 256, and the model parameters are optimized using Adam [26] with a learning rate of 10^{-4}. The model stops training if the objective function (Equation (4)) does not improve over 300 epochs, i.e., we do not use any of the metrics presented in Section 4 as the stopping criteria, which assures absence of leakage of label information. We conduct an ablation study of the loss terms in Equation (4); however, we do not perform an exhaustive search for the corresponding weights, and instead evaluate $\lambda_i \in \{0, 1\}$ to investigate their effects. Fine-tuning the weights is left for future work.

4. EVALUATION METRICS

Our evaluation protocol relies on the properties of disentangled representations. From the synthesis point of view, the pitch of the synthesized spectrum should be invariant to perturbations in the timbre space as much as possible; from the perspective of analysis, the timbre space (pitch space) should mostly accommodate timbre information (pitch information) while minimizing clues for pitch (timbre). Accordingly, we consider the following metrics. Note that ground-truth annotations and pre-trained classifiers are employed only for evaluation purpose.

4.1 Classification Accuracy

We train logistic regression models which take as input the learned timbre latent variable z and predict labels of instrument and pitch. A well disentangled timbre representation should yield high accuracy for instrument, and low accuracy for pitch.

4.2 Clustering Accuracy (ACC)

During testing, the pitch encoder outputs a categorical distribution $q_\phi(c|x)$ from which a pitch category of x can be assigned as $k = \arg\max_k q_\phi(c|x)$. We can thereby consider it as a clustering task and calculate ACC [27] using pitch labels. Furthermore, since we do not train our model with pitch labels, the mapping from the inferred pitch categories to the pitch labels is unknown. For each category, we thus assign a pitch label that occurs the most within that category, and pitch classification accuracy can be calculated accordingly. This approximated pitch mapping is termed PM. Both ACC and PM are served to evaluate the unsupervised pitch classification.

4.3 Fréchet Inception Distance (FID)

We exploit FID [28] to quantitatively measure the quality of the synthesized spectrum. The metric measures the distributional difference between two multivariate Gaussians, which are fit to features derived from the real and generated samples, respectively. In our case, the features are extracted from a pre-trained instrument classifier, using the identical training data, which shares the same architecture with the encoder.

4.4 Consistency-Diversity Score (CDS)

In order to assess the model's capability of pitch-conditional generation, we propose a metric, termed CDS, to account for consistency of $p_k(y|\hat{x})$ and diversity of $\mathbb{E}_k[p_k(y|\hat{x})]$, where $p_k(y|\hat{x}) = p(y|D(z, c_k))$ is the posterior distribution of a pre-trained pitch classifier given the generated samples $\hat{x}$; and $\mathbb{E}_k[\cdot]$ denotes marginalization over k, where $k \in \{1, 2, \ldots, K\}$. Note that we can not simply measure pitch classification accuracy given the generated samples, as the true mapping from categories to pitch labels is unknown under the unsupervised setting.

Intuitively, the pre-trained pitch classifier should consistently output similar posterior distribution $p_k(y|\hat{x})$, if the generated samples $\hat{x}$ are synthesized conditioned on a fixed c_k regardless of z; and the aggregated distribution $\mathbb{E}_k[p_k(y|\hat{x})]$ should be uniformly distributed over y, which indicates that the generated samples $\hat{x}$, when conditioned on different c_k's, are predicted as having different pitches. Formally, CDS combines the two indicators as follows:

$$\begin{aligned} \text{CDS} &= -\mathbb{E}_k[H(p_k(y|\hat{x}))] + H(\mathbb{E}_k[p_k(y|\hat{x})]) \\ &= \mathbb{E}_k[\mathcal{D}_{\text{KL}}(p_k(y|\hat{x})\|\mathbb{E}_k[p_k(y|\hat{x})])], \end{aligned} \tag{5}$$

namely, the marginal KL-divergence of the per-category and the aggregated posterior. A higher CDS thus hinds toward better consistency and diversity of pitch manifested by the generated pitch-conditioning spectrum.

λ_1	λ_2	λ_3	λ_4		Pitch	Instrument	Combine	ACC	PM	$\text{FID}_{\text{recon}}$	FID_{rand}	CDS
				♭	8.81±3.47	87.68±1.09	89.43±1.85	95.14±0.98	96.04±0.71	21.80±1.05	23.78±1.47	24.33±0.71
0	0	0	0	♯	33.78±7.38	80.90±4.41	73.55±5.77	72.65±4.82	74.46±4.06	24.86±2.27	25.27±1.80	8.49±1.96
				M0	16.38±7.65	86.44±2.20	85.02±4.03	78.53±5.68	80.22±6.01	23.93±1.97	26.40±2.39	11.45±2.34
1	0	0	0	M1	17.85±4.52	87.34±1.26	84.74±2.53	77.28±3.47	78.75±3.60	18.86±1.77	21.53±1.10	9.15±1.28
0	1	0	0	M2	20.45±7.98	84.74±2.67	82.14±5.17	77.40±5.01	79.09±6.08	26.00±1.78	26.90±2.28	9.20±1.55
0	0	1	0	M3	32.54±6.28	84.18±1.92	75.81±4.08	**80.45±1.58**	**82.71±1.26**	18.68±2.36	20.82±1.67	10.79±2.37
0	0	0	1	M4	17.06±3.83	84.18±1.38	83.55±1.84	74.35±2.75	75.59±3.32	22.36±2.36	24.74±2.17	11.99±2.67
1	1	1	0	M5	18.19±4.79	**87.90±1.62**	84.85±2.48	78.19±2.35	79.66±2.81	16.73±2.13	21.39±2.49	9.35±2.81
1	1	1	1	M6	**14.57±2.29**	86.44±2.55	**85.93±2.06**	79.88±1.84	80.90±2.18	**13.76±1.07**	**19.18±1.90**	**13.46±1.64**

Table 1: The ablation study. For simplicity, we focus more on examining individual effects and do not exhaust all combinations. Each model (per row) is evaluated over all the evaluation metrics. For *Pitch* and *FID*, lower numbers indicate better performance, while the rest suggest the otherwise. The best-performing *unsupervised* models (♯, M0-M6) are highlighted.

Note that CDS bears resemblance to Inception Score (IS) [29]; the latter, however, was originally proposed to evaluate visual quality of synthetic images, whereas CDS evaluates the extent to which the model faithfully renders the conditional signal and enables correct classification.

5. EXPERIMENTS AND RESULTS

We train the framework with different configurations of the objective function $\mathcal{L}$ (Equation (4)), and quantify the performance of disentanglement with the metrics detailed in Section 4. For each model configuration, we initialize the model parameters with five random seeds, and report an averaged score along with standard deviation for each metric.

The results are summarized in Table 1, where we highlight the best-performing *unsupervised* models for each metric. Each row represents a model configuration; the symbol ♭ denotes the pitch-supervised model, which is trained to minimize an additional cross-entropy loss between the categorical distribution $q_\phi(\mathbf{c}|\mathbf{x})$ and the pitch labels, and is treated as a reference. The symbol ♯ denotes the unsupervised model trained without pitch-shifting. The rest M0-M6 are all unsupervised, utilizing pitch-shifting with maximum two semitones upward or downward.

For convenience, E_p, E_t, and D denote the pitch encoder, the timbre encoder, and the decoder, throughout.

5.1 Timbre Space Classification

Using the learned timbre representation $\mathbf{z}$, which we replace with $\mu_\phi(\mathbf{x})$ (μ in Figure 1) as the input feature to the logistic regression models, we obtain a relatively low accuracy for pitch classification, and a high accuracy for instrument classification, as shown in columns *Pitch* and *Instrument* in Table 1. As mentioned previously, low and high accuracy of pitch and instrument, respectively, indicate disentanglement of the timbre representation; we thereby aggregate the two metrics by $\frac{1}{2}(1-a_{\text{pitch}}+a_{\text{instrument}})$ shown in column *Combine*, where a_{pitch} and $a_{\text{instrument}}$ are the classification accuracy.

The pitch-supervised (reference) model attains the best aggregated score, as E_p is explicitly trained to classify pitch, thereby preventing pitch leak to E_t. Among the proposed models, M6 outperforms in terms of the aggregated

score contributed by low a_{pitch}, which implies that combining all the auxiliary loss terms helps prevent pitch from leaking into the timbre space. Pitch-shifting alone improves the baseline unsupervised model significantly; this might be due to the rather imbalanced pitch distribution of the data. The high score attained without additional losses implies the efficacy of the proposed architectural design on disentangling pitch and timbre, given the augmented data.

Individually adding the auxiliary objective functions does not contribute much to the performance. For example, while M1-M4 degrade the aggregated score, combining M1-M3 (M5) approaches the best-performing model (M6). Notably, we can see that the proposed surrogate label loss further improves the performance of M5, which similarly applies to other metrics that follow.

5.2 Unsupervised Pitch Classification
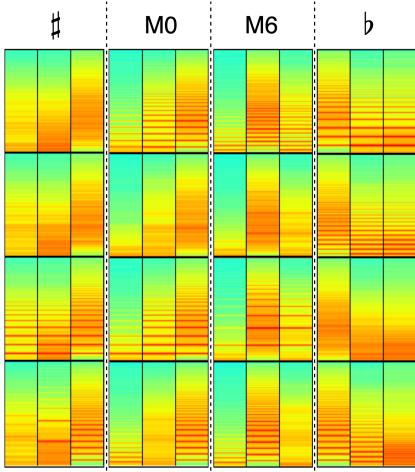
As described in Section 4.2, we can consider E_p as a pitch classifier trained without labels as in our proposed models. We thus evaluate the performance with ACC.

We also report the pitch classification accuracy derived by the approximated mapping from pitch categories to pitch labels, which is the PM described in Section 4.2.

The supervised model can therefore be treated as the upper bound of pitch classification accuracy attained by the unsupervised E_p. M3 is the best model in terms of both ACC and PM, which could be attributed to the cycle-consistency that acknowledges the pitch-swapping scheme during training. This however promotes pitch leak to the timbre space as shown in column $Pitch$, which implies that an accurate E_p does not guarantee the absence of pitch leak, and, without supervision, more constraints are necessary to maintain both the pitch accuracy and timbre disentanglement, as demonstrated by M6.

5.3 Spectral Synthesis

We now turn our attention to the evaluation of generative tasks. In particular, we first evaluate the timbre representation by FID between the synthesized spectrum and the real one. To be more specific, the synthesized data are generated by D which takes as input $\mathbf{z}$ and $\mathbf{c}_k$, where $\mathbf{z} \sim p(\mathbf{z})$ and $k = \arg\max_k q_\phi(\mathbf{c}|\mathbf{x})$; that is, we first infer $\mathbf{c}_k$ of the

Figure 2: Pitch-conditioning spectrum generation. Each column represents a model, the bottom row refers to seed samples, and the top three rows correspond to generated samples conditioned on different pitch categories.

validation set, which is then combined with the randomly sampled **z** for decoding.

FID$_{recon}$ measures between the real and the reconstructed data, while FID$_{rand}$ is for the real and the synthesized data. FID$_{recon}$ can thus be treated as a lower bound of the metric. From Table 1, it is clear that M6 prevails. As discussed earlier, adding the proposed $\mathcal{L}_{surrogate}$ to M5 makes the best model in terms of FID.

Interestingly, the supervised model does not achieve satisfying performance, which implies that the discriminability gain of E_p does not correlate well with the generative quality of timbre features. This similarly applies to the model that employs only the contrastive loss (M2).

5.4 Pitch-Conditioning Synthesis

Next we evaluate the disentanglement through the lens of conditional generation. Particularly, we first infer $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ from the validation set, which we directly take the mean $\mu_\phi(\mathbf{x})$ as the representative latent variable. We then enumerate all possible pitch categories $k \in \{1, 2, ..., K\}$, each of which is converted to a one-hot vector and concatenated with the inferred **z**. D consumes the pitch-conditioned latent vector, and generates samples $\hat{\mathbf{x}}$ which are then classified by a pre-trained pitch classifier. This computes $p(\mathbf{y}|D(\mathbf{z}, \mathbf{c}_k))$, and CDS is derived by Equation (5). We report $\exp(\text{CDS})$ to restrict the value in $\{1, K\}$.

The supervised model performs well, due to the available pitch labels during training. M6 outperforms all the unsupervised models. Notably, $\mathcal{L}_{surrogate}$ alone (M4) outperforms M0, and, once again, the loss term further improves M5 to reach the best model M6.

The proposed $\mathcal{L}_{surrogate}$ synergizes with other loss terms, as evidenced by comparing M5 and M6 in terms of most metrics. This is probably attributed to the extra information from the amount of pitch-shift, which enables the model to explicitly account for the pitch difference [18].

Among all metrics, the t-test only yields a significant difference between the bold and M0 in terms of FID. Apart from the relatively high variances obtained by M0, this could be due to the small sample size (five random seeds) and the suboptimal configuration of values of loss weights, which we will investigate in future work.

5.5 Qualitative Study

We conclude our evaluation with a qualitative study on pitch-conditioning synthesis, as demonstrated in Figure 2. For each model (column), the bottom row refers to three reconstructed samples (with corresponding **z**'s) which are the seed spectrums sampled from the validation set. Each of the rest of the rows corresponds to generated samples conditioned on the same **z**'s but a different **c**.

As a result, for each model (column), a good performance is indicated by a matched harmonic pattern across all three frames in a row (consistency), and diverse harmonic patterns across the top three rows (diversity). Note that the seed samples (bottom row) and the three conditioning pitches are not fixed across the four models, thus a direct comparison is not available. Nonetheless, we can have a rough idea that model ♯ does not perform as well as others, as harmonic patterns do not clearly appear aligned except for the one at the third row. This to some extent verifies the proposed CDS, in terms of which model ♯ attains the worst performance, although a study of larger scale is required for a faithful verification.

We can also observe that the overall timbre, characterized by the spectral energy distribution, stays rather consistent along each frame of each column despite the change of the pitch condition, which verifies the disentanglement.

6. CONCLUSION AND FUTURE WORK

We have proposed a VAE-based framework for unsupervised learning of disentangled pitch and timbre representation. The framework accommodates a categorical and a continuous latent variable, with the former embodying the discrete nature of pitch. We exploit pitch-shifting which enables the auxiliary objective functions, that are shown to potentially enhance the performance in terms of the quantitative evaluation.

A major challenge for future research is to infer pitch values from the categorical assignment, without access to ground-truth annotations. Furthermore, the proposed model imposes a strong inductive bias to the pitch encoder, by restricting degree of freedom through a one-hot encoded categorical pitch variable. This might pose a challenge when a tuning difference among instruments is present in the dataset. Increasing the capacity of the pitch representation while maintaining enough constraints for disentanglement is a direction for future work. We also aim to train the framework on a larger and more structured dataset [30], and to evaluate the method on data with larger time scale, for which we aim to learn dynamical latent factors on top of the global variables that we have studied.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] Y. Bengio, "Deep learning of representations: Looking forward," in *Proc. of the Int. Conf. on Statistical Language and Speech Processing.* Springer, 2013, pp. 1–37.

[2] Y.-J. Luo, K. Agres, and D. Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders," in *Proc. of the Int. Society for Music Information Retrieval Conf.*, 2019, pp. 746–753.

[3] Y.-N. Hung, Y.-A. Chen, and Y.-H. Yang, "Learning disentangled representations for timbre and pitch in music audio," *arXiv preprint arXiv:1811.03271*, 2018.

[4] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. of the Int. Conf. on Learning Representations*, 2014.

[5] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, M. Shakir, and A. Lerchner, "Beta-vae: Learning basic visual concepts with a constrained variational framework," in *Proc. of the Int. Conf. on Learning Representations*, 2017.

[6] H. Kim and A. Mnih, "Disentangling by factorising," in *Proc. of the Int. Conf. on Machine Learning*, 2018.

[7] A. Kumar, P. Sattigeri, and A. Balakrishnan, "Variational inference of disentangled latent concepts from unlabeled observations," in *Proc. of the Int. Conf. on Learning Representations*, 2018.

[8] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," in *Proc. of the Int. Conf. on Neural Information Processing Systems*, 2017, pp. 3601–3610.

[9] E. Denton and V. Birodkar, "Unsupervised learning of disentangled representations from video," in *Proc. of the Int. Conf. on Neural Information Processing Systems*, 2017.

[10] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," in *Proc. of the Int. Conf. on Neural Information Processing Systems*, 2017, pp. 1878–1889.

[11] Y. Li and S. Mandt, "Disentangled sequential autoencoder," in *Proc. of the Int. Conf. on Machine Learning*, 2018.

[12] A. Ruiz, O. Martinez, X. Binefa, and J. Verbeek, "Learning disentangled representations with reference-based variational autoencoders," *arXiv preprint arXiv:1901.08534*, 2019.

[13] D. Bouchacourt, R. Tomioka, and S. Nowozin, "Multi-level variational autoencoder: Learning disentangled representations from grouped observations," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2018.

[14] J. Chen and K. Batmanghelich, "Weakly supervised disentanglement by pairwise similarities," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2019.

[15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. of the Int. Conf. on Machine Learning*, 2020.

[16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. of the IEEE Int. Conf. on Computer Vision*, 2017, pp. 2223–2232.

[17] A. H. Jha, S. Anand, M. Singh, and V. Veeravasarapu, "Disentangling factors of variation with cycle-consistent variational auto-encoders," in *Proc. of the European Conf. on Computer Vision*, 2018, pp. 805–820.

[18] B. Gfeller, C. Frank, D. Roblek, M. Sharifi, M. Tagliasacchi, and M. Velimirović, "SPICE: Self-supervised pitch estimation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1118–1128, 2020.

[19] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *Proc. of the Int. Conf. on Machine Learning*, 2019.

[20] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. of the Int. Conf. on Learning Representations*, 2017.

[21] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in *Proc. of the IEEE Int. Conf. on Computer Vision*, 2017, pp. 2051–2060.

[22] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *Proc. of the Int. Conf. on Learning Representations*, 2018.

[23] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *Proc. of the European Conf. on Computer Vision*, 2018, pp. 35–51.

[24] J. Lezama, "Overcoming the disentanglement vs reconstruction trade-off via Jacobian supervision," in *Proc. of the Int. Conf. on Learning Representations*, 2019.

[25] G. Ballet, R. Borghesi, P. Hoffmann, and F. Levy, "Studio Online 3.0: An Internet "killer application" for remote access to IRCAM sounds and processing tools," *Journee Informatique Musicale*, 1999.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proc. of the Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2003, pp. 267–273.

[28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proc. of the Int. Conf. on Neural Information Processing Systems*, 2017, pp. 6626–6637.

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. of the Int. Conf. on Neural Information Processing Systems*, 2016, pp. 2234–2242.

[30] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

AI SONG CONTEST: HUMAN-AI CO-CREATION IN SONGWRITING

Cheng-Zhi Anna Huang[1]   **Hendrik Vincent Koops**[2]   **Ed Newton-Rex**[3]

Monica Dinculescu[1]   **Carrie J. Cai**[1]

[1] Google Brain   [2] RTL Netherlands   [3] ByteDance

`annahuang,noms,cjcai@google.com, h.v.koops,ednewtonrex@gmail.com`

ABSTRACT

Machine learning is challenging the way we make music. Although research in deep generative models has dramatically improved the capability and fluency of music models, recent work has shown that it can be challenging for humans to partner with this new class of algorithms. In this paper, we present findings on what 13 musician/developer teams, a total of 61 users, needed when co-creating a song with AI, the challenges they faced, and how they leveraged and repurposed existing characteristics of AI to overcome some of these challenges. Many teams adopted modular approaches, such as independently running multiple smaller models that align with the musical building blocks of a song, before re-combining their results. As ML models are not easily steerable, teams also generated massive numbers of samples and curated them post-hoc, or used a range of strategies to direct the generation, or algorithmically ranked the samples. Ultimately, teams not only had to manage the "flare and focus" aspects of the creative process, but also juggle them with a parallel process of exploring and curating multiple ML models and outputs. These findings reflect a need to design machine learning-powered music interfaces that are more decomposable, steerable, interpretable, and adaptive, which in return will enable artists to more effectively explore how AI can extend their personal expression.

1. INTRODUCTION

Songwriters are increasingly experimenting with machine learning as a way to extend their personal expression [12]. For example, in the symbolic domain, the dance punk band Yacht used MusicVAE [59], a variational autoencoder type of neural network, to find melodies hidden in between songs by interpolating between the musical lines in their back catalog, commenting that *"it took risks maybe we aren't willing to"* [46]. In the audio domain, Holly Herndon uses neural networks trained on her own voice to produce a polyphonic choir. [15, 45]. In large part, these human-AI experiences were enabled by major advances in machine learning and deep generative models [18, 65, 67], many of which can now generate coherent pieces of long-form music [17, 29, 38, 54].

Although substantial research has focused on improving the algorithmic performance of these models, much less is known about what musicians actually need when songwriting with these sophisticated models. Even when composing short, two-bar counterpoints, it can be challenging for novice musicians to partner with a deep generative model: users desire greater agency, control, and sense of authorship vis-a-vis the AI during co-creation [44].

Recently, the dramatic diversification and proliferation of these models have opened up the possibility of leveraging a much wider range of model options, for the potential creation of more complex, multi-domain, and longer-form musical pieces. Beyond using a single model trained on music within a single genre, how might humans co-create with an open-ended set of deep generative models, in a complex task setting such as songwriting?

In this paper, we conduct an in-depth study of what people need when partnering with AI to make a song. Aside from the broad appeal and universal nature of songs, songwriting is a particularly compelling lens through which to study human-AI co-creation, because it typically involves creating and interleaving music in multiple mediums, including text (lyrics), music (melody, harmony, etc), and audio. This unique conglomeration of moving parts introduces unique challenges surrounding human-AI co-creation that are worthy of deeper investigation.

As a probe to understand and identify human-AI co-creation needs, we conducted a survey during a large-scale human-AI songwriting contest, in which 13 teams (61 participants) with mixed (musician-developer) skill sets were invited to create a 3-minute song, using whatever AI algorithms they preferred. Through an in-depth analysis of survey results, we present findings on what users needed when co-creating a song using AI, what challenges they faced when songwriting with AI, and what strategies they leveraged to overcome some of these challenges.

We discovered that, rather than using large, end-to-end models, teams almost always resorted to breaking down their musical goals into smaller components, leveraging a wide combination of smaller generative models and recombining them in complex ways to achieve their creative goals. Although some teams engaged in active co-creation with the model, many leveraged a more extreme, multi-

stage approach of first generating a voluminous quantity of musical snippets from models, before painstakingly curating them manually post-hoc. Ultimately, use of AI substantially changed how users iterate during the creative process, imposing a myriad of additional model-centric iteration loops and side tasks that needed to be executed alongside the creative process. Finally, we contribute recommendations for future AI music techniques to better place them in the music co-creativity context.

In sum, this paper makes the following contributions:

- A description of common patterns these teams used when songwriting with a diverse, open-ended set of deep generative models.

- An analysis of the key challenges people faced when attempting to express their songwriting goals through AI, and the strategies they used in an attempt to circumvent these AI limitations.

- Implications and recommendations for how to better design human-AI systems to empower users when songwriting with AI.

2. RELATED WORK

Recent advances in AI, especially in deep generative models, have renewed interested in how AI can support mixed-initiative creative interfaces [16] to fuel human-AI co-creation [27]. *Mixed initiative* [32] means designing interfaces where a human and an AI system can each "take the initiative" in making decisions. *Co-creative* [43] in this context means humans and AI working in partnership to produce novel content. For example, an AI might add to a user's drawing [20, 49], alternate writing sentences of a story with a human [14], or auto-complete missing parts of a user's music composition [4, 33, 37, 44].

Within the music domain, there has been a long history of using AI techniques to model music composition [22, 30, 52, 53], by assisting in composing counterpoint [21, 31], harmonizing melodies [13, 42, 50], more general infilling [28, 34, 40, 60], exploring more adventurous chord progressions [26, 36, 48], semantic controls to music and sound generation [11, 23, 35, 62], building new instruments through custom mappings [24], and enabling experimentation in all facets of symbolic and acoustic manipulation of the musical score and sound [3, 7].

More recently, a proliferation of modern deep learning techniques [8, 9] has enabled models capable of generating full scores [39], or producing music that is coherent to both local and distant regions of music [38, 54]. The popular song form has also been an active area of research to tackle modeling challenges such as hierarchical and multi-track generation [51, 56, 63, 69].

Despite significant progress in deep generative models for music-making, there has been relatively little research examining how humans interact with this new class of algorithms during co-creation. A recent study on this topic [44] found that deep learning model output can feel non-deterministic to end-users, making it difficult for users to steer the AI and express their creative goals. Recent

work has also found that users desire to retain a certain level of creative freedom when composing music with AI [25, 44, 64], and that semantically meaningful controls can significantly increase human sense of agency and creative authorship when co-creating with AI [44]. While much of prior work examines human needs in the context of co-creating with a single tool, we expand on this emerging body of literature by investigating how people assemble a broad and open-ended set of real-world models, data sets, and technology when songwriting with AI.

3. METHOD AND PARTICIPANTS

The research was conducted during the first three months of 2020, at the AI Song Contest organized by VPRO [66]. The contest was announced at ISMIR in November 2019, with an open call for participation. Teams were invited to create songs using any artificial intelligence technology of their choice. The songs were required to be under 3 minutes long, with the final output being an audio file of the song. At the end, participants reflected on their experience by completing a survey. Researchers obtained consent from teams to use the survey data in publications. The survey consisted of questions to probe how teams used AI in their creative process:

- How did teams decide which aspects of the song to use AI and which to be composed by musicians by hand? What were the trade-offs?

- How did teams develop their AI system? How did teams incorporate their AI system into their workflow and generated material into their song?

In total, 13 teams (61 people) participated in the study. The teams ranged in size from 2 to 15 (median=4). Nearly three fourths of the teams had 1 to 2 experienced musicians. A majority of teams had members with a dual background in music and technology: 5 teams had 3 to 6 members each with this background, and 3 teams had 1 to 2 members. We conducted an inductive thematic analysis [2, 5, 6] on the survey results to identify and better understand patterns and themes found in the teams' survey responses. One researcher reviewed the survey data, identifying important sections of text, and two researchers collaboratively examined relationships between these sections, iteratively converging on a set of themes.

4. HOW DID TEAMS CO-CREATE WITH AI?

The vast majority of teams broke down song composition into smaller modules, using multiple smaller models that align with the musical building blocks of a song, before combining their results: *"So my workflow was to build the song, section by section, instrument by instrument, to assemble the generated notes within each section to form a coherent whole"* (T12). A few teams first attempted to use end-to-end models to generate the whole song at once, such as through adversarial learning from a corpus of pop song audio files (T6) or through generating an audio track using SampleRNN [47] (T13). However, they

Music building blocks	Models & techniques
Lyrics	GPT2, LSTM, Transformer
Melody	CharRNN, SampleRNN, LSTM + CNN, WaveNet + LSTM, GAN, Markov model
Harmony	LSTM, RNN autoencoder, GAN, Markov model
Bassline	LSTM + CNN, WaveNet + LSTM, GAN
Drums	DrumRNN, Neural Drum Machine, SampleRNN, Markov model
Multi-part	MusicVAE trio (melody, bass, drums), MiniVAE trio, Coconet/Coucou (4-part counterpoint), MusicAutobot (melody, accompaniment), Transformer (full arrangement)
Structure	Markov model
Vocal synthesis	WaveNet, SampleRNN, Vocaloid, Sinsy, Mellotron, Emvoice, Vocaloid, custom vocal assistant
Instrument synthesis	SampleRNN, WaveGAN, DDSP

Table 1. Overview of musical building blocks used by teams.

quickly learned that they were unable to control the model or produce professional quality songs, and thus turned to the modular approach instead. In the following sections, we summarize how teams used modular building blocks, combined and curated them, and in some cases more actively co-created with AI to iterate on the outcomes.

4.1 Leveraging modular musical building blocks

Overall, teams leveraged a wide range of models for musical components such as lyrics, (vocal) melody, harmony, bassline, drums, arrangement, and vocal and instrument synthesis. Table 1 shows an overview of models used for each song component, and Figure 1 illustrates how the 13 teams co-created with AI along these different components. The number of unique model types used by teams ranged from 1 to 6 (median 4). Some teams used the same type of model for modeling different song components.
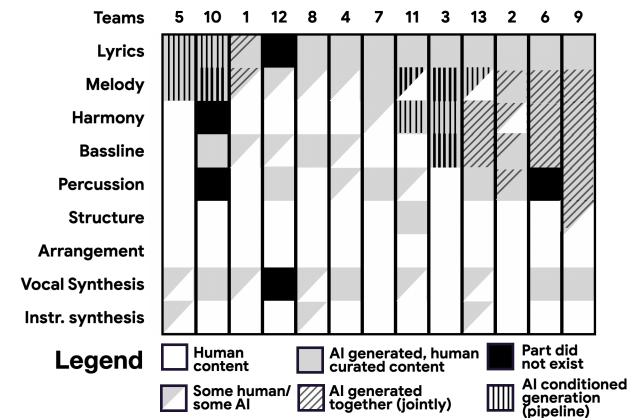
All teams used AI to generate lyrics and melodies, and more than half of the teams synthesized vocals for parts of the song. Some of these decisions were due to model availability and existing levels of model performance. For example, teams almost always generated lyrics using AI because high-performing models like GPT-2 [55] along with a fine-tuning script were readily available.

Teams with professional musicians often chose to only generate lyrics and melodic lines, in order to leave enough creative space to musicians to decide how the various lines can be put together and to arrange the song in their own style (T5, 8). One exception is T3 who generated a lead sheet with lyrics, melody and harmony.

Teams with more ML and less music expertise opt for minimal arrangements (T10, 6, 9), and often used multi-part models because they could generate melody, harmony, bass, drums together in a coherent way, providing teams with larger building blocks to work with. In one extreme case, the team was able to generate all sections of their song by traversing the latent space of MusicVAE (T9) (see "Bridging sections through latents" below for more detail).

4.2 Combining building blocks

Teams leveraged many strategies for combining outputs of smaller models, piecing together the musical building

Figure 1. An overview of how 13 teams co-created with AI in songwriting. Each column shows whether each song's component was musician composed, AI generated then human curated, or both. Nearly all teams manually aligned AI generated *lyrics* and *melody*, except teams in the first three columns. T5 used a two-stage pipeline by first generating *lyrics* and *melody* separately, then algorithmically matching them up using their stress patterns. T10 first generated *lyrics*, and then conditioned on *lyrics* to generate *melody*. T1 jointly modeled *lyrics* and *melody*, generating syllables and notes in an interleaving fashion. T12, 8, 4, 7 all generated *melodic lines* first, and then manually **stitched** them together by layering them vertically as *melody* and *bassline* to yield *harmony*. In contrast, T11, 3 first generated *chords*, then conditioned on *chords* to generate *melody* (and *bassline* separately) in a **pipeline**. T13 iterated between conditioning on *melody* to jointly generate the other parts and vice versa. T2, 6 and 9 focused on models that could **jointly** generate multiple parts at once.

blocks to form a coherent song. These ranged from manually combining individual components, to using heuristics to automatically pair them up, to creating a pipeline between models, to adopting models that jointly model multiple components at once.

Stitching Many teams manually "stitched" together machine generated material, with the result informing the manual composition of other musical components. In

one team, a musician *"selected, tweaked, and weaved AI-generated melody, chords and drum parts into a ballad song form"*, while another musician wrote the bassline *"that seemed more or less implied by the AI materials"* (T7). This is echoed by another team, who composed the accompaniment *"based on chordal movements predicted by the melodic fragments"* (T5). Several teams layered melodic lines to yield harmony (T1, 12, 8, 4).

Pipelines Several teams leveraged model pipe-lining, feeding the output of one model into the input of another model. To generate melody that aligns well with lyrics, one team first used GPT-2 to generate lyrics, then a lyric-conditioned model to generate melody [68] (T10). One team decomposed melody generation into two steps, first using a LSTM to generate rhythm as note onset patterns, and then a CNN to assign a pitch to each onset (T8). While many teams "stitched" together melodic lines to create harmony, two teams first generated chords and then melody (and bassline) (T11 and T13). Pipeline approaches allowed teams to refine the output at each intermediate stage before passing content into the next model.

Joint modeling To generate multiple parts together, several teams adopted models such as MusicVAE trio [56], Coconet [34], MusicAutobot [61] or Transformers that are trained to jointly model multiple instrumental parts (T13, 2, 6, 9). One team experimented with jointly modeling notes and syllables from pairs of melodies and lyrics, but found it *"very hard to concurrently generate semantically meaningful lyrics and a melody that is not aimless"* (T1).

4.3 Generate then curate

A common approach was to generate a large quantity of musical samples, followed by automatically or manually curating them post-hoc. Teams took a range of approaches to curating the large quantity of results, ranging from brute-force manual curation, to a two-stage process of first filtering with AI, then curating manually.

Generation Often, teams used models to generate a large volume of samples to choose from. For instance, one team used their pipeline LSTM + CNN model to generate over 450 melodies and basslines (T8). Another team generated 10K lines of death metal lyrics (T13).

Manual curation While curating, teams were often looking for the key musical themes or motifs to use for their song. For example, one team used MusicVAE to generate several combinations of lead, bass, and drums, and *"handpicked the most appealing"* version to serve as their verse (T2). Another team was looking for a small, catchy snippet or *"earworms"* to flesh out the music (T11).

Two-stage curation A few teams first used automated methods to narrow down the choices, before manually selecting what would fit best in their song. For example, one team used a *"catchiness"* classifier trained on rankings of songs to filter melodies before handing them to an artist (T8). Another team curated their generated material by first algorithmically matching up the stress patterns of lyrics and melodies to make sure the material could be more immediately useful (T5).

Several teams found the process of generating and curating painstaking, or similar to a difficult puzzle (T1). However, one team described this massive generation process as exhilarating, or like *"raging with the machines"* (T5). They most appreciated the unexpected surprises, and actively engaged with this firehose of raw, AI-generated material: *"We couldn't resist including as much of the good and quirky machine output as possible...makes it much less repetitive than much of the music we might produce as humans. We really enjoyed having this firehose of generative capability...its constantly moving nature"* (T5).

4.4 Active co-creation

Some teams co-created music with AI in a more blended manner, where the model outputs influenced human composing and vice versa, similar to how human musicians might work together to write a song.

A few teams used AI-generated output as an underlying foundation for composing on top of, such as improvising a melody over AI-generated chords: *"we played the chords, and all of us around the table hummed along until we got to a simple and catchy melody"* (T11). Others took the AI output as raw material generated in a predefined structure, and manually composed an underlying beat (T8).

Others took AI output as an initial seed for inspiration and further elaboration. For example, one team trained SampleRNN on acappellas, which generated nonsensical output similar to babbling. A musician then tried to "transcribe" the words and the melody, and sang along with it. *"She found sections that sounded like lyrics. She wrote those lyrics down and sang them. She spent a day riffing on those lyrics, building a dialogue"*. These riffs and words fueled the formation of the larger story and song (T13).

For one participant, working with the AI was like jamming with another musician, an active back and forth process of priming the AI with chord progressions, hearing AI output, riffing on that output, then using those new ideas to seed the AI again (T12).

One team described making some deliberate decisions about how much agency to provide the AI vs. themselves as artists. To preserve the AI's content, an artist tried to only transpose and not "mute" any of the notes in two-bar AI-generated sequences, as he chose which ones to stitch together to align with lyrics. However, to bring the artist's own signature as a rapper, he decided to compose his own beat, and also improvise on top of the given melodies freely with a two-syllable word that was made up by ML (T8).

5. TEAMS OVERCOMING AI LIMITATIONS

In the previous section, we described the ways in which participants co-created with AI. Although teams made some headway by breaking down the composition process into smaller models and building blocks, we observed a wide range of deeper challenges when they attempted to control the co-creation process using this plethora of models. In this section, we describe participants' creative coping strategies given these challenges, and the strategies

they used to better direct the co-creation process.

5.1 ML is not directly steerable

Due to the stochastic nature of ML models, their output can be unpredictable. During song creation, teams leveraged a wide range of strategies in an attempt to influence model output, such as through data during fine-tuning, or through input or conditioning signals during generation. Below, we describe the most common patterns observed:

Fine-tuning   After experimenting with a model, many teams tried to influence the mood or style of the generated content by fine-tuning models on a smaller dataset. For example, teams fine-tuned GPT-2 with lyrics from uplifting pop songs to German death metal (T13), in order to steer the generation towards phrase structures more common in lyrics and also sub-phrases that reflect the desired sentiment and style.

Priming   While co-creating, teams often desired to create musical content in a particular key, chord, contour, or pitch range. To do so, many attempted to reverse engineer the process by priming the model's input sequence using music containing the desired property, in the hopes that it would produce a continuation with similar characteristics: *"I found that I could control the generation process by using different kinds of input. So if I wanted a sequence that is going upward, I would give that kind of input. If I wanted the model to play certain notes, I would add those notes in the input pattern"* (T12). This seemingly simple way of requesting for continuations led to a wide range of controls when used creatively.

To further direct content in lyrics, a team used specific words to start off each sentence (T5). Another team wondered *"can pop songs convey insightful messages with only two words?"*, and put together a verse with frequent bigrams from that dataset, such as *"my love"*, *"your heart"*. They entered this verse through the TalkToTransformer [41] web-app interface as context for GPT-2 to generate the next verses (T11).

Interpolating   Models such as MusicVAE provide a continuous latent space that supports interpolating between existing sequences to generate new sequences that bear features of both [58]. Some teams leveraged this latent space as a way to explore. For example, one participant found by chance that *"interpolating between really simple sequences at high temperatures would end up giving me these really cool baseline sequences"* (T12).

5.2 ML is not structure-aware

Because large, end-to-end models were not easily decomposable into meaningful musical structures, most teams used multiple smaller, independent models. Yet, these smaller, independent ML models are not able to take into account the holistic context of the song when generating local snippets. To address this, users created their own workarounds to fill in this contextual gap, by arranging and weaving these independent pieces into a coherent whole.

Creating an overall song structure   To create a backbone for the song, some teams used their musical knowledge to first curate chord progressions that can serve well for each song section (i.e. verse, chorus). One team then used a conditioned model (pretrained to be conditioned on chord progressions) to generate melody and basslines that would go well with those chords (T3).

Creating contrast between sections   The verse and chorus sections of a song often carry contrast. However, participants did not have a direct way to express this desire for structured contrast while preserving overall coherence between verse and chorus. To address this, one team used their verse as a starting point to generate various continuations to the melody and bass lines, and finally chose a variation for the chorus section (T2). Another team used similar priming approaches but used different temperatures to generate the verse and chorus in order to *"add some randomness into the generation"* (T12). These approaches gave users a way to manually create structured variety.

Rewriting to add variation   Rewriting allows one to generate new material while borrowing some structure from the original material. For example, one team was able to generate a *"darker version"* of the chorus of another song by rewriting it multiple times, alternating between regenerating the melody conditioned on the accompaniment, and then re-generating the accompaniment conditioned on the melody. To create a coherent song structure, the team initially attempted to *"repeat the same rave section twice"*, but later *"realized that was boring"*. The team then decided to vary how the second section ended by reharmonizing the melody with a new flavor: *"Coconet is great at reharmonizing melodies with a baroque flavor. We entered in the notes from our rave chorus. After a few tries, it made this awesome extended cadence"* (T13).

Bridging sections through latents   One team devised an unusual strategy for connecting different sections of a song in a meaningful way (T9). They first trained multiple miniVAEs [19], one for each section of the song (e.g. intro, verse, chorus or genre such as rock and pop). They then composed the song by computing the *"shortest path"* through these latent spaces, allowing the sections to share elements in common with each other, despite each being distinctive. The genre miniVAEs also made style transfer possible by interpolating an existing trio towards their latent areas, allowing them to tweak the style of each section.

5.3 ML setup can interfere with musical goals

A logistical hurdle faced by teams was the significant setup and customization issues encountered to even start composing with a model. Aside from musical considerations, many teams chose or abandoned models based on what was available out-of-the-box, such as pre-trained checkpoints, fine-tuning scripts, scripts for retraining on a different dataset, data pre-processing scripts. In addition, different models expected different types of music representation as input (e.g. MIDI, piano roll, custom), adding additional pre-processing overhead to begin experimenting with them. To decrease time spent model wrangling, large teams sometimes divide-and-conquered, exploring several models in parallel to see which worked best. Ultimately,

teams' co-creation process involved navigating not only their musical goals, but also the logistical costs and benefits of using one model or another.

6. DISCUSSION

6.1 Decomposeable and context-aware modeling

Writing a song involves composing multiple sections, and multiple vocal and instrumental parts that all need to work well together as a whole. Because end-to-end models lacked meaningful hierarchical structures, and because smaller models lacked global awareness, participants often needed to reverse engineer a multitude of models, applying heuristics and domain knowledge to approximate high-level structure or to achieve desired musical effects. To ameliorate this process, one approach could be to infuse smaller models with more context-awareness, and exposing the common ways that they can be customized through user-facing controls. For example, a melody model could allow users to express whether they are creating a verse as opposed to a chorus, or whether they would like it to contrast with the next section. Another possibility is to design end-to-end models to have intermediate representations that align with the musical objects that musicians already know how to work with. The sweet spot is probably a hybrid that combines the flexibility of smaller models with the benefits of global context in end-to-end modeling.
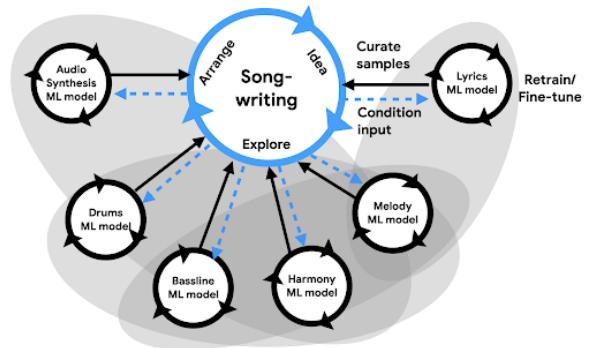
6.2 AI-defined vs. User-defined building blocks

The design of ML models for music involves a series of upstream decisions that can have a large impact on how musicians think about music when they co-create with these models downstream. Whereas regular songwriting often starts with an initial spark of a human idea, in this work we found that the practical availability and limitations of AI tools were instead key drivers of the creative process, defining the scope of what's musically legitimate or possible. For example, most teams broke down songwriting into lyrics, melody, chords, etc., in part because these models were readily available. Yet, there are also other music building blocks that do not have corresponding, ready-made generative modsels (e.g. motif, verse, chorus, coda, etc.) or that currently are not treated as separate, first-class building blocks in deep models (e.g. rhythm, pitch). Likewise, a musician's creative thought process can be unintentionally influenced by the order of steps taken to fine-tune, condition, prime, and apply a model. In the future, the design of ML models should be coupled with a more careful consideration of what workflows and building blocks end-users already use in their existing practice, and perhaps start with those as first-class principles in guiding the design of AI systems.

6.3 Support for parallel music and ML exploration

A central aspect of the creative process involves a "flare and focus" [10] cycle of ideating, exploring those ideas, selecting ideas, then rapidly iterating. We found that a key

challenge of human-AI co-creation was the need to juggle not only this *creative* process, but also the *technological* processes imposed by the idiosyncrasies and lack of steerability of learning algorithms. For instance, while ideating motifs for a song, participants needed to carry out a large additional burden of sub-tasks, such as selecting which combination of models to use, re-training or conditioning them as necessary, chaining them together, and "gluing" their outputs together. In essence, the typical "flare and focus" cycles of creativity were compounded with a parallel cycle of having to first explore and curate a wide range of models and model outputs (Figure 2). While some of these model-wrangling tasks led to new inspiration, many interfered with the rapid-iteration cycle of creativity.

Figure 2. Parallel music and ML feedback loops in human-AI co-creative songwriting.

These issues raise important questions around how best to support users in juggling the dual processes of creative and technological iteration cycles. One approach is to have ML models readily available to musicians in their natural workflows. For example, Magenta Studio [57] makes available a suite of model plug-ins to music production software such as Ableton Live [1], and Cococo [44] allows users to semantically steer a model directly in its user interface. Beyond this, human-AI interfaces could scaffold the *strategic* part of the model exploration and selection process by surfacing effective model combinations (e.g. using general infilling models for rewriting or to reharmonize another generated melody) or fruitful workflows (e.g. matching lyric and melody stress patterns), so that new users can benefit from past users' experiences. Reducing this overhead of model-based decisions could empower users to more easily prototype their creative ideas, accelerating the feedback and ideation cycle.

7. CONCLUSION

We conducted an in-depth examination of how people leverage modern-day deep generative models to co-create songs with AI. We found that participants leveraged a wide range of workarounds and strategies to steer and assemble a conglomeration of models towards their creative goals. These findings have important implications for how human-AI systems can be better designed to support complex co-creation tasks like songwriting, paving the way towards more fruitful human-AI partnerships.

8. ACKNOWLEDGEMENTS

9. REFERENCES

[1] Ableton Live. https://www.ableton.com/en/live/. Accessed: 2020-05-04.

[2] Anne Adams, Peter Lunt, and Paul Cairns. A qualititative approach to hci research. 2008.

[3] Carlos Agon, Gérard Assayag, and Jean Bresson. *The OM Composer's Book 1*. Editions Delatour France / Ircam-Centre Pompidou, 2006.

[4] Théis Bazin and Gaëtan Hadjeres. Nonoto: A model-agnostic web interface for interactive music composition by inpainting. *ICCC*, 2019.

[5] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

[6] Virginia Braun and Victoria Clarke. What can "thematic analysis" offer health and wellbeing researchers? *International journal of qualitative studies on health and well-being*, 9, 2014.

[7] Jean Bresson, Carlos Agon, and Gérard Assayag. *The OM Composer's Book 2*. Editions Delatour France / Ircam-Centre Pompidou, 2008.

[8] Jean-Pierre Briot. From artificial neural networks to deep learning for music generation – history, concepts and trends, 2020.

[9] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation-a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[10] Bill Buxton. *Sketching user experiences: getting the design right and the right design*. Morgan kaufmann, 2010.

[11] Mark Brozier Cartwright and Bryan Pardo. Social-eq: Crowdsourcing an equalization descriptor map. In *ISMIR*, pages 395–400, 2013.

[12] Barbican Centre. 12 songs created by ai how musicians are already embracing new technologies. https://artsandculture.google.com/theme/12-songs-created-by-ai/FwJibAD7QslgLA. Accessed: 2020-05-03.

[13] Ching-Hua Chuan and Elaine Chew. A hybrid system for automatic generation of style-specific accompaniment. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*.

[14] Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A Smith. Creative writing with a machine in the loop: Case studies on slogans and stories. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces*, 2018.

[15] Jack Denton. Future 25: Holly herndon, artist and inventor of spawn. https://www.rollingstone.com/music/music-features/future-25-holly-herndon-889072/. Accessed: 2020-05-03.

[16] Sebastian Deterding, Jonathan Hook, Rebecca Fiebrink, Marco Gillies, Jeremy Gow, Memo Akten, Gillian Smith, Antonios Liapis, and Kate Compton. Mixed-initiative creative interfaces. In *CHI 2017 Extended Abstracts*.

[17] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[18] Sander Dieleman, Aaron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In *Advances in Neural Information Processing Systems*.

[19] Monica Dinculescu, Jesse Engel, and Adam Roberts. Midime: Personalizing a MusicVAE model with user data. 2019.

[20] Judith E Fan, Monica Dinculescu, and David Ha. collabdraw: An environment for collaborative sketching with an artificial agent. In *Proceedings of the 2019 on Creativity and Cognition*.

[21] Mary Farbood and Bernd Schöner. Analysis and synthesis of Palestrina-style counterpoint using markov chains. In *Proceedings of the International Computer Music Conference*, 2001.

[22] Jose D Fernández and Francisco Vico. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48.

[23] Lucas Ferreira and Jim Whitehead. Learning to generate music with sentiment. In *ISMIR*, 2019.

[24] Rebecca Anne Fiebrink. Real-time human interaction with supervised learning algorithms for music composition and performance. *PhD dissertation, Princeton University*, 2011.

[25] Emma Frid, Celso Gomes, and Zeyu Jin. Music creation by example. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[26] Satoru Fukayama, Kazuyoshi Yoshii, and Masataka Goto. Chord-sequence-factory: A chord arrangement system modifying factorized chord sequence probabilities. 2013.

[27] Werner Geyer, Lydia B. Chilton, Ranjitha Kumar, and Adam Tauman Kalai. Workshop on human-AI co-creation with generative models. In *Proceedings of the 25th International Conference on Intelligent User Interfaces Companion*, 2020.

[28] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. DeepBach: a steerable model for Bach chorales generation. In *International Conference on Machine Learning*, pages 1362–1371, 2017.

[29] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

[30] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys (CSUR)*, 50(5), 2017.

[31] Dorien Herremans and Kenneth Sörensen. Composing fifth species counterpoint music with a variable neighborhood search algorithm. *Expert systems with applications*, 40(16):6427–6437, 2013.

[32] Eric Horvitz. Principles of mixed-initiative user interfaces. In *SIGCHI Conference on Human Factors in Computing Systems*.

[33] Cheng-Zhi Anna Huang, Sherol Chen, Mark Nelson, and Doug Eck. Mixed-initiative generation of multi-channel sequential structures. In *International Conference on Learning Representations Workshop Track*, 2018.

[34] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Doug Eck. Counterpoint by convolution. In *Proceedings of the International Conference on Music Information Retrieval*, 2017.

[35] Cheng-Zhi Anna Huang, David Duvenaud, Kenneth C. Arnold, Brenton Partridge, Josiah W. Oberholtzer, and Krzysztof Z. Gajos. Active learning of intuitive control knobs for synthesizers using gaussian processes. In *Proceedings of the 19th International Conference on Intelligent User Interfaces*, IUI, 2014.

[36] Cheng-Zhi Anna Huang, David Duvenaud, and Krzysztof Z Gajos. Chordripple: Recommending chords to help novice composers go beyond the ordinary. In *Proceedings of the International Conference on Intelligent User Interfaces*, 2016.

[37] Cheng-Zhi Anna Huang, Curtis Hawthorne, Adam Roberts, Monica Dinculescu, James Wexler, Leon Hong, and Jacob Howcroft. The Bach Doodle: Approachable music composition with machine learning at scale. *ISMIR*, 2019.

[38] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *International Conference on Learning Representations*, 2019.

[39] Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Generating music with rhythm and harmony. *arXiv preprint arXiv:2002.00212*, 2020.

[40] Daphne Ippolito, Anna Huang, Curtis Hawthorne, and Douglas Eck. Infilling piano performances. In *NIPS Workshop on Machine Learning for Creativity and Design*, 2018.

[41] Adam King. Talk to Transformer. `https://talktotransformer.com/`. Accessed: 2020-05-03.

[42] Hendrik Vincent Koops, José Pedro Magalhães, and W. Bas de Haas. A functional approach to automatic melody harmonisation. In *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling Design*. Association for Computing Machinery, 2013.

[43] Antonios Liapis, Georgios N. Yannakakis, Constantine Alexopoulos, and Phil Lopes. Can computers foster human users' creativity? Theory and praxis of mixed-initiative co-creativity. *Digital Culture & Education*, 8(2):136–153, 2016.

[44] Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie J. Cai. Novice-AI music co-creation via AI-steering tools for deep generative models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2020.

[45] Jonathan Maas. Holly herndon: We are ai. `https://www.vprobroadcast.com/titles/ai-songcontest/articles/we-are-ai.html`. Accessed: 2020-05-03.

[46] Nathan Mattise. How yacht used machine learning to create their new album. `https://www.wired.com/story/how-yacht-used-machine-learning-to-create-their-new-album/`. Accessed: 2020-05-03.

[47] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

[48] Eric Nichols, Dan Morris, and Sumit Basu. Data-driven exploration of musical chord sequences. In *Proceedings of the international conference on Intelligent user interfaces*, 2009.

[49] Changhoon Oh, Jungwoo Song, Jinhan Choi, Seonghyeon Kim, Sungwoo Lee, and Bongwon Suh. I lead, you help but only with enough details: Understanding user experience of co-creation with artificial intelligence. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, 2018.

[50] François Pachet and Pierre Roy. Musical harmonization with constraints: A survey. *Constraints*, 6(1):7–19, 2001.

[51] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted lead sheet composition using flowcomposer. In *International Conference on Principles and Practice of Constraint Programming*. Springer, 2016.

[52] George Papadopoulos and Geraint Wiggins. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symposium on Musical Creativity*, volume 124, 1999.

[53] Philippe Pasquier, Arne Eigenfeldt, Oliver Bown, and Shlomo Dubnov. An introduction to musical metacreation. *Computers in Entertainment (CIE)*, 14(2):2, 2016.

[54] Christine Payne. Musenet. `https://openai.com/blog/musenet`, 2019. Accessed: 2020-05-04.

[55] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[56] Adam Roberts and Jesse Engel. Hierarchical variational autoencoders for music. In *Proceedings NIPS machine learning for creativity and design workshop*, 2017.

[57] Adam Roberts, Jesse Engel, Yotam Mann, Jon Gillick, Claire Kayacik, Signe Nørly, Monica Dinculescu, Carey Radebaugh, Curtis Hawthorne, and Douglas Eck. Magenta studio: Augmenting creativity with deep learning in Ableton live. 2019.

[58] Adam Roberts, Jesse Engel, Sageev Oore, and Douglas Eck. Learning latent representations of music to generate interactive musical palettes. In *IUI Workshops*, 2018.

[59] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. *ICML*, 2018.

[60] Andrew Shaw. A multitask music model with bert, transformer-xl and seq2seq. `https://towardsdatascience.com/a-multitask-music-model-with-bert-transformer-xl-and-seq2seq-3d80bd2ea08e`. Accessed: 2020-05-03.

[61] Andrew Shaw. Musicautobot. `https://musicautobot.com/`. Accessed: 2020-05-03.

[62] Ian Simon, Dan Morris, and Sumit Basu. Mysong: automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 725–734. ACM, 2008.

[63] Ian Simon, Adam Roberts, Colin Raffel, Jesse Engel, Curtis Hawthorne, and Douglas Eck. Learning a latent space of multitrack measures. *arXiv preprint arXiv:1806.00195*, 2018.

[64] Bob L Sturm, Oded Ben-Tal, Una Monaghan, Nick Collins, Dorien Herremans, Elaine Chew, Gaëtan Hadjeres, Emmanuel Deruty, and François Pachet. Machine learning research that matters for music creation: A case study. *Journal of New Music Research*, 48(1), 2019.

[65] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[66] Karen van Dijk. AI Song Contest. `https://www.vprobroadcast.com/titles/ai-songcontest.html`. Accessed: 2020-05-03.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[68] Yi Yu and Simon Canales. Conditional LSTM-GAN for melody generation from lyrics. *arXiv preprint arXiv:1908.05551*.

[69] Yichao Zhou, Wei Chu, Sam Young, and Xin Chen. Bandnet: A neural network-based, multi-instrument Beatles-style midi music composition machine. *arXiv preprint arXiv:1812.07126*, 2018.

ANALYSIS OF SONG/ARTIST LATENT FEATURES AND ITS APPLICATION FOR SONG SEARCH

Kosetsu Tsukuda   **Masataka Goto**

National Institute of Advanced Industrial Science and Technology (AIST), Japan

`{k.tsukuda, m.goto}@aist.go.jp`

ABSTRACT
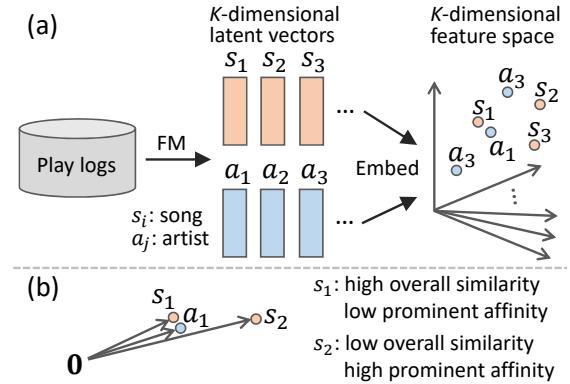
For recommending songs to a user, one effective approach is to represent artists and songs with latent vectors and predict the user's preference toward the songs. Although the latent vectors represent the characteristics of artists and songs well, they have typically been used only for computing the preference score. In this paper, we discuss how we can leverage these vectors for realizing applications that enable users to search for songs from new perspectives. To this end, by embedding song/artist vectors into the same feature space, we first propose two concepts of artist-song relationships: overall similarity and prominent affinity. Overall similarity is the degree to which the characteristics of a song are similar overall to the characteristics of the artist; while prominent affinity is the degree to which a song prominently represents the characteristics of the artist. By using Last.fm play logs for two years, we analyze the characteristics of the concepts. Moreover, based on the analysis results, we propose three applications for song search. Through case studies, we demonstrate that our proposed applications are beneficial for searching for songs according to the users' various search intents.

1. INTRODUCTION

Embedding songs into a feature space is beneficial for realizing various applications for music information retrieval (MIR). For example, by using tags of each song, songs can be embedded into a tag-based feature space [1]; this enables a user to search for similar songs of her favorite song. In another example, by embedding songs into the Arousal-Valence space [2] according to their audio features [3], a user can search for the song with the highest Arousal value from her favorite artist's songs. The same can be said for artists; by embedding artists into a feature space based on the topics of lyrics, artists similar to a user's favorite artist in terms of topic similarity can be retrieved [4].

Embedding heterogeneous data into a feature space is also useful for MIR tasks: song recommendations for playlists by embedding tags and songs [5, 6], computing tag similarity by embedding artists and tags [7], etc.

Figure 1. Overview of our proposed ideas: (a) embedding artists' latent vectors and songs' latent vectors into the same feature space and (b) concepts of overall similarity and prominent affinity.

In spite of the potential to embed heterogeneous data, there have been few studies that have embedded songs and artists [8]. If both songs and artists are embedded into the same feature space, greater variety of MIR applications can be realized. For example, given an artist, we can search for songs that have similar characteristics to those of the artist (i.e., songs whose feature vectors are close to the feature vector of the artist).

To realize this, we use Factorization Machines (FM) [9], which is an item recommendation technique. By using FM, each song, user, and artist can be represented by K-dimensional latent vectors. Although such vectors are usually used to compute a user's preference toward a song, we leverage the latent vectors of songs and artists to embed songs and artists into the same feature space (Fig. 1 (a)).

In the embedded feature space, we propose two concepts of artist-song relationships: overall similarity and prominent affinity. Suppose the vector's direction of artist a_1 in Fig. 1 (b) represents the sadness. The length of the vector indicates the degree of sadness. In this case, song s_1 has almost the same degree of sadness and s_1 is similar overall to a_1. On the other hand, song s_2 has a higher degree of sadness. This means that s_2 prominently represents a_1's characteristics and has a higher prominent affinity with a_1 in terms of sadness. We can recommend s_1 and s_2 to a user as a_1's characteristic songs because of the high overall similarity and the high prominent affinity, respectively. Such a recommendation would be useful when the user listens to one of a_1's songs for the first time because she can decide whether to listen to a's other songs after listening to a characteristic song of a_1. By using these concepts, we can realize not only such song recommendations

but also various applications for MIR, as in Section 5.

Our main contributions in this paper are as follows.

- We propose the concepts of overall similarity and prominent affinity between an artist and a song by leveraging their latent vectors learned through FM (Section 3).

- By using Last.fm play logs for two years, we show various characteristics of overall similarity and prominent affinity. For example, we reveal that an artist's popular songs tend to have high prominent affinity with the artist (Section 4).

- We demonstrate that the concepts of overall similarity and prominent affinity can be used to realize various applications for MIR. Specifically, we show three applications: familiarity-oriented search, typicality-oriented search, and analogy search (Section 5).

2. RELATED WORK

2.1 Matrix Factorization for Song Recommendations

Matrix Factorization (MF) [10] has been widely used for item recommendations. In the context of song recommendations, too, the effectiveness of MF has been reported [11–14]. One of the characteristics of MF for song recommendations is that each user and song are represented by K-dimensional latent vectors. This enables the model to learn a user's latent preference toward songs. Although MF typically considers interactions between users and songs, Factorization Machines (FM) [9] can include side information in addition to information about users and songs. Side information can be an artist, category of a song, and even the weather when a user listens to a song. Because of such flexibility, FM has also been used for song recommendations [15–18]. The idea of using artist information in FM for song recommendations has already been proposed [15, 16]; in this case, each user, song, and artist are represented by K-dimensional latent vectors.

Although such latent vectors in FM represent the characteristics of users, songs, and artists well, they have typically been used for computing a user's preference score toward a song and generating a personalized ranked list of songs for each user. Different from existing studies, we analyze latent vectors of songs and artists based on new relationships between artists and songs (overall similarity and prominent affinity) and show their potential to implement MIR applications.

2.2 Heterogeneous Embedding for MIR

The usefulness of embedding heterogeneous data into a feature space has been reported in various MIR tasks, where data has been embedded by a Markov-model-based method [5, 19], co-occurrence-based method [20], etc. For example, by embedding tags and songs, song recommendations for playlists have been realized [5, 6]. Other examples include computing tag similarity by embedding artists and tags [7], retrieving songs by words by embedding songs and words in playlist titles [20], and visualizing

the time-dependent listening preferences of a population by embedding users and songs [19]. Our study is different from theirs in that we embed artists and songs into a feature space.

The study closest to ours is that by Weston *et al.* [8]. They proposed a method for embedding artists and songs into a feature space and solved tasks such as predicting songs for a given artist and retrieving similar songs for a given song. To embed songs, their method requires audio data for all songs. In contrast, we use users' play logs. Although comparing the embedding accuracy of these two approaches is beyond the scope of this paper, our approach using play logs has an advantage over their approach in terms of the applicability of insights into the MIR community because various kinds of large play logs are easily accessible [21–27] compared to the accessibility of large audio data.

2.3 MIR Applications for Song Search

In the MIR community, various kinds of applications for song search have been proposed. These applications have enabled users to more easily find their desired songs and search for songs from a new perspective. For example, query by humming [28–34] and query by singing [35–38] enable users to search for songs even when they do not know the song title. Similarity-based song search is also beneficial for searching for new songs that are similar to a user's favorite song, where similarity between songs is measured by low-level acoustic features [39, 40], voice timbres of vocals [41], tags [1], and a combination of these characteristics [42]. When a user has a specific search intent, searching for songs using metadata [43,44] and words in lyrics [45] is also helpful to find her desired songs.

In this paper, we propose two concepts of artist-song relationships. These concepts can be used to search for an artist's characteristic songs, as we will show in Section 4.3. In addition, in Section 5, we demonstrate application examples that can be realized by leveraging these concepts and latent vectors of artists and songs. Our proposed applications are also helpful for searching for users' desired songs and new songs. We believe our study is beneficial for other researchers to implement MIR applications based on our proposed concepts.

3. OVERALL SIMILARITY AND PROMINENT AFFINITY

In this section, we first describe how to generate latent vectors of artists and songs through FM. We then propose the concepts of overall similarity and prominent affinity.

3.1 Notation

Let $\mathcal{U}$, $\mathcal{I}$, and $\mathcal{A}$ denote the sets of users, songs, and artists, respectively. $\mathcal{I}_u^+$ represents the set of songs preferred by user $u \in \mathcal{U}$. Following Lim *et al.* [46], we define the songs played $\geq \mu$ times by u as the preferred songs of u. By using the data, we first aim to accurately generate a per-

sonalized ranked list of songs for each user u from $\mathcal{I} \setminus \mathcal{I}_u^+$, which is a set of songs not included in u's preferred songs.

3.2 FM for Song Recommendation

FM [9] is a method for predicting a user's preference toward an item based on MF [10]. A typical MF deals with only interactions between users and items; while in FM, side information (e.g., the artist or category of a song) can also be included in the model. By considering artist information, we can extract new relationships between artists and songs, as we will describe in Section 3.3. Below, we describe FM considering artist information.

In FM, user u, song s, and artist a have K-dimensional latent vectors $\boldsymbol{\nu_u}$, $\boldsymbol{\nu_s}$, and $\boldsymbol{\nu_a}$, respectively. The preference score of user u toward song s based on the second-order estimator of FM is computed with the following model:

$$\widehat{x}_{us} = \alpha + \beta_u + \beta_s + \beta_{a_s} + \langle \boldsymbol{\nu_u}, \boldsymbol{\nu_s} \rangle + \langle \boldsymbol{\nu_u}, \boldsymbol{\nu_{a_s}} \rangle + \langle \boldsymbol{\nu_{a_s}}, \boldsymbol{\nu_s} \rangle,$$
(1)

where α is the global offset, a_s is the artist of s, and β_u, β_s, and β_{a_s} are the user/song/artist bias terms. As can be seen in the model, the preference score is computed by a user's affinity with a song ($\langle \boldsymbol{\nu_u}, \boldsymbol{\nu_s} \rangle$), user's affinity with an artist ($\langle \boldsymbol{\nu_u}, \boldsymbol{\nu_{a_s}} \rangle$), and artist's affinity with a song ($\langle \boldsymbol{\nu_{a_s}}, \boldsymbol{\nu_s} \rangle$). Regarding the last term $\langle \boldsymbol{\nu_{a_s}}, \boldsymbol{\nu_s} \rangle$, if artist a_s tends to sing calm songs and song s is also calm, the value of $\langle \boldsymbol{\nu_{a_s}}, \boldsymbol{\nu_s} \rangle$ becomes high; while if s is exciting music, the value becomes low. Note that because a song's popularity is reflected in β_s (i.e., more popular songs tend to have higher values of β_s), popular songs do not always have high values of $\langle \boldsymbol{\nu_{a_s}}, \boldsymbol{\nu_s} \rangle$. Rather, the value of $\langle \boldsymbol{\nu_{a_s}}, \boldsymbol{\nu_s} \rangle$ is determined purely by the affinity between a and s.

Note that although we use *simple* FM just by adding artist information, our goal in this paper is not to improve recommendation accuracy. Rather, we aim to study how to leverage latent vectors of songs and artists. Although this simple FM can learn reliable latent vectors because it can achieve high enough recommendation accuracy, as we will report in Section 4.2, we can adopt more sophisticated FM (e.g., FM considering song co-occurrence [47] and audio information [11]); we leave this for future work.

We adopt Bayesian Personalized Ranking (BPR) [48] to learn latent vectors. BPR is a pairwise ranking optimization framework and is designed to deal with users' implicit consumption behaviors such as playing a song rather than explicit ones such as rating. In BPR, the training set $\mathcal{D}$ used for optimizing parameters is defined as follows:

$$\mathcal{D} = \{ (u, i, j) \mid u \in \mathcal{U} \wedge i \in \mathcal{I}_u^+ \wedge j \in \mathcal{I} \setminus \mathcal{I}_u^+ \}.$$

That is, a triad (u, i, j) means that user u prefers song i to song j. The optimization criterion for $\mathcal{D}$ is given by:

$$\sum_{(u,i,j) \in \mathcal{D}} \ln \sigma(\widehat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2,$$
(2)

where σ is the sigmoid function, $\Theta = \{\beta_s, \beta_a, \boldsymbol{\nu_u}, \boldsymbol{\nu_s}, \boldsymbol{\nu_a}\}$ represents all model parameters, and λ_Θ is a regularization hyperparameter. $\widehat{x}_{uij}$ represents the difference between u's preference for i and that for j, which is defined as $\widehat{x}_{uij} = \widehat{x}_{ui} - \widehat{x}_{uj}$. Finally, we learn the parameters by using TensorFlow [49] with Adam optimizer [50].

3.3 Overall Similarity and Prominent Affinity

The learned model parameters are usually used for computing a user's preference score toward a song by using Eq. 1. In this paper, we propose using the learned model parameters (i.e., latent vectors) $\boldsymbol{\nu_a}$ and $\boldsymbol{\nu_s}$ for capturing the relationships between artists and their songs. In Eq. 1, because the affinity between $\boldsymbol{\nu_a}$ and $\boldsymbol{\nu_s}$ is considered, the nth ($1 \leq n \leq K$) dimension of $\boldsymbol{\nu_a}$ and that of $\boldsymbol{\nu_s}$ have the same meaning. For example, if the first dimension of $\boldsymbol{\nu_a}$ represents the calmness, the first dimension of $\boldsymbol{\nu_s}$ also represents the calmness of the song. Hence, we can embed $\boldsymbol{\nu_a}$ and $\boldsymbol{\nu_s}$ into the same feature space, as shown in Fig. 1 (a). In the feature space, the angle and length of a vector represents its qualitative and quantitative aspects, respectively. Specifically, the difference of the angle between two vectors represents the difference of their characteristics as a song or artist because each dimension represents a characteristic of songs and artists. While the length of a vector represents the degree of its characteristics: if the value of $\boldsymbol{\nu_s}$'s nth dimension is large, $\boldsymbol{\nu_s}$ represents nth characteristic well. By using the embedded feature space, we propose two concepts of the relationships between artists and songs: overall similarity and prominent affinity.

3.3.1 Overall Similarity
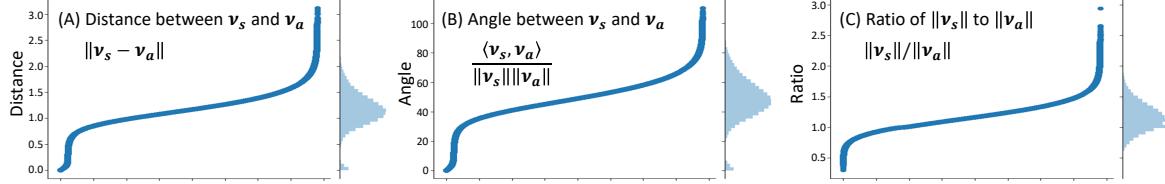
Suppose artist a_1's song s_1 is mapped fairly close to a_1 in the feature space, as shown in Fig. 1 (b). In this case, we can say that s_1 is similar overall to a_1. Thus, we define the closeness between an artist and a song in the feature space as the overall similarity between them. More formally, given artist a and song s, the overall similarity between a and s is computed based on the Euclidean distance between them: $f_{os}(a, s) = \frac{1}{\|\boldsymbol{\nu_a} - \boldsymbol{\nu_s}\| + 1}$.

The concept of overall similarity can be used to search for an artist's song that represents the artist's characteristics well. Searching for such a song and listening to it is useful to quickly understand the artist's characteristic songs. In particular, when a user listens to an artist's song for the first time, it might be helpful for her to listen to the artist's characteristic song and then decide if she wants to listen to the artist's other songs.

3.3.2 Prominent Affinity

Now suppose a_1's song s_2 is mapped fairly close to the extended position of a_1 as shown in Fig. 1 (b). This means that s_2 prominently represents a_1's characteristics because the degree of each characteristic of s_2 is higher than that of a_1. Thus, we define the inner product of an artist vector and a song vector as the prominent affinity between them. More formally, given artist a and a's song s, the prominent affinity between a and s is given by $f_{pa}(a, s) = \langle \boldsymbol{\nu_a}, \boldsymbol{\nu_s} \rangle$.

The concept of prominent affinity can be used to search for an artist's song that strongly represents the artist's characteristics. Similar to the overall similarity, searching for such a song would be helpful when a user listens to an artist's song for the first time.

Figure 2. Distributions of (A) distance between ν_s and ν_a, (B) angle between ν_s and ν_a, and (C) ratio of $\|\nu_s\|$ to $\|\nu_a\|$. Each dot represents a song where songs are sorted in ascending order of the y-values in each graph.

Number of users ($|\mathcal{U}|$)	84,708
Number of songs ($|\mathcal{I}|$)	390,158
Number of artists ($|\mathcal{A}|$)	32,448
Sum of preferred songs ($\sum_{u \in \mathcal{U}} |\mathcal{I}_u^+|$)	18,478,304

Table 1. Dataset statistics.

4. ANALYSIS

In this section, we analyze the characteristics of overall similarity (OS) and prominent affinity (PA).

4.1 Dataset

We use the LFM-1b dataset that includes music listening logs on Last.fm [21]. Each listening log consists of user ID, song ID, artist ID, and timestamp. The dataset also includes data for converting song ID and artist ID to song name and artist name, respectively. We use logs for two years (between 1/1/2012 and 12/31/2013). The μ in Section 3.1 is set to five, where μ is the threshold to determine that song s is included in $\mathcal{I}_u^+$ when u listens to s equal to or more than μ times. To increase the reliability of learned model parameters, songs and artists that have been listened to by less than 10 different users and users who have listened to less than 10 different songs are discarded. Table 1 shows the dataset statistics after the preprocessing.

4.2 Model Development

For each user, we split $\mathcal{I}_u^+$ into training/validation/test sets. To this end, we randomly select one preferred song (i.e., $i \in \mathcal{I}_u^+$) for validation $\mathcal{V}_u$ and another for testing $\mathcal{T}_u$ [51]. All the remaining songs are used for training $\mathcal{R}_u$. The recommendation performance is evaluated by the AUC (Area Under the ROC Curve), which is widely used to evaluate whether model parameters are appropriately learned [52, 53]: $AUC = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{D}_u|} \sum_{(i,j) \in \mathcal{D}_u} \delta(\hat{x}_{ui} > \hat{x}_{uj})$, where $\mathcal{D}_u = \{(i,j) \mid i \in \mathcal{T}_u \land j \in \mathcal{I} \setminus \mathcal{I}_u^+\}$, and $\delta(z)$ is 1 when z is true and 0 otherwise. The AUC value ranges between 0 and 1, and a higher value represents better performance (i.e., the model parameters are appropriately learned). The hyperparameters (i.e., λ_Θ in Eq. 2 and the learning rate) are tuned on a validation set in terms of the AUC, where the hyperparameters are selected from {0.0001, 0.001, 0.01, 0.1, 1}. We set the latent dimensionality K to 50. [1] We emphasize that our goal here is not to evaluate the recommendation accuracy of FM by comparing with other methods. Rather, we aim to evaluate whether the parameters in FM developed by our dataset are

---

[1] We evaluated the AUC on the validation dataset by changing K from 10 to 100 in increments of 10. The AUC saturated when $K = 50$.

Rank	OS	PA
1	I'm So Tired	Something
2	Get Back	All You Need Is Love
3	The End	Come Together
4	Sun King	Hey Jude
5	Here Comes the Sun	I Am the Walrus
6	She's Leaving Home	Lucy in the Sky with Diamonds
7	Glass Onion	Eleanor Rigby
8	You Like Me Too Much	A Hard Day's Night
9	You Never Give Me Your Money	I Want to Hold Your Hand
10	The Night Before	Golden Slumbers

Rank	OS	PA
1	Minor Thing	Californication
2	Police Station	Scar Tissue
3	If	Dani California
4	Can't Stop	Snow (Hey Oh)
5	Fortune Faded	By the Way
6	Annie Wants a Baby	The Adventures of Rain Dance Maggie
7	Happiness Loves Company	The Zephyr Song
8	Brendan's Death Song	Brendan's Death Song
9	Give It Away	Torture Me
10	Emit Remmus	Dosed

Table 2. Ranking results of songs in terms of OS and PA (top: "The Beatles," bottom: "Red Hot Chili Peppers").

appropriately learned by showing that the AUC is close to 1 after the learning process.

The AUC on the test set achieved a high value: 0.973. This result means that the model parameters and the embedded feature space are appropriately learned and the artist-song relations based on OS and PA are reliable. We also computed the Spearman rank correlation between the values of β_s and song popularities to evaluate if β_s correctly reflects song popularity, as mentioned in Section 3.2. The popularity of song s was measured by the number of different users who have played s. The popularity ranking of songs are obtained by sorting them in descending order of their popularities. For each artist, we compute the correlation between the popularities of the artist's all songs and their values of β_s. The average of the correlations over all artists was relatively high: 0.502. When we define the popularity of artist a as the number of different users who have played at least one of a's songs, the average of the correlations became high with increasing artist popularity: artists whose popularities are ≥ 100 and ≥ 300 had correlation values of 0.682 and 0.755 on average, respectively. From these results, we can say that the bias term β_s certainly reflects the song's popularity, and the value of $\langle \nu_{a_s}, \nu_s \rangle$ in Eq. 1 is determined mainly by the affinity between a_s and s especially for popular artists. Hereafter, the parameter values computed in this section are used for artist latent vectors and song latent vectors.

4.3 Analysis Results

Although we showed that the model parameters are appropriately learned, if most of an artist's songs are mapped

Rank	Area (a)	Area (b)	Area (c)	Area (d)
1	Gold on the Ceiling	Have Love Will Travel	Yearnin'	Can't Find My Mind
2	Lonely Boy	Same Old Thing	Keep Your Hands Off Her	Her Eyes Are A Blue Million Miles
3	Tighten Up	I Got Mine	Grown So Ugly	Howlin For You
4	Little Black Submarines	Run Right Back	Till I Get My Way	The Wicked Messenger
5	Everlasting Light	Your Touch	Just Got To Be	The Baddest Man Alive

Table 3. Ranking results of familiarity-oriented search for "The Black Keys" in terms of PA.

very close to the artist, it would be useless to rank songs according to the subtle difference of their positions. To confirm whether artists' songs are well distributed, we evaluate the distributions of (A) distance between ν_s and ν_a, (B) angle between ν_s and ν_a, and (C) ratio of $\|\nu_s\|$ to $\|\nu_a\|$. Fig. 2 shows the results. In all cases, the value distribution is close to the Gaussian distribution. These results indicate that songs are well distributed in the feature space and it is meaningful to rank songs according to OS, which is affected by (A), and PA, which is affected by (B) and (C).
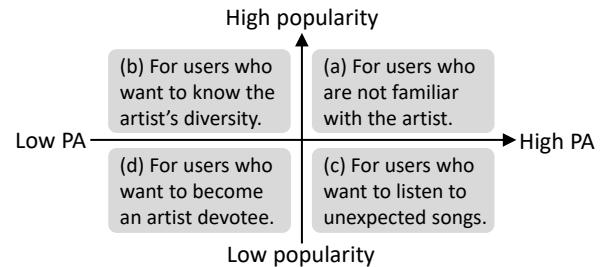
Next, we analyze the ranked results of songs by OS/PA. Table 2 shows the top 10 songs of "The Beatles" and "Red Hot Chili Peppers." Since the top ranked songs are largely different between the two concepts, it would be meaningful to generate two ranked lists so that we can show one of them (or both of them) to a user according to her intent. We can also see that the top ranked songs in PA tend to be more popular than those in OS. To evaluate this, we compute the Spearman rank correlation between OS-based/PA-based song ranking and popularity-based song ranking. As expected, the correlation of PA-based ranking is high (0.577), while that of OS-based ranking is low (-0.147). As mentioned in Section 4.2, the effect of song popularity is eliminated by the bias terms β_s. Therefore, this high correlation of PA-based ranking is caused purely by the characteristics of the songs: songs strongly reflecting the artist's characteristics tend to be popular.

5. APPLICATIONS

In Section 4.3, we showed how to directly use the concepts of OS and PA to rank an artist's songs. In this section, by leveraging the concepts and learned parameters of ν_a and ν_s, we demonstrate three applications: familiarity-oriented search, typicality-oriented search, and analogy search. Through these demonstrations, we show that this paper brings reusable insights in that our proposed concepts can be used for various kinds of MIR applications.

5.1 Familiarity-oriented Search

By considering PA and song popularity, an application to search for songs according to a user's familiarity with an artist can be realized. As we mentioned in Section 4.3, songs with a high PA tend to be popular, but some of them are unpopular. Similarly, some popular songs have a low PA. Hence, according to the degree of PA and the degree of popularity, an artist's songs can be classified into four areas, as shown in Fig. 3. Searching for songs in area (a) would be beneficial especially for a user who listens to the artist's song for the first time because these songs are popular and represent the artist's typical characteristics well; she can decide if she wants to listen to the artist's other

Figure 3. Properties of songs classified by the degree of PA and the degree of popularity.

songs by listening to the searched songs. After listening to such songs, searching for songs in area (b) would be useful to let her understand the diversity of the artist because area (b) includes songs that are popular but different from the artist's typical characteristics. Moreover, searching for songs in area (c) enables her to listen to unexpected songs in terms of the fact that the searched songs match the artist's typical characteristics well but are not known by many people. Finally, searching for songs in area (d) would be helpful for her to become an artist devotee by listening to them.

In light of the above, given artist a, we rank all a's songs for each area in terms of PA as follows. For area (a), the score of song s is computed by $r_{ap}(s, \mathcal{I}_a) + r_{pop}(s, \mathcal{I}_a)$, where $\mathcal{I}_a$ is a set of all songs of a and $r_{ap}(s, \mathcal{I}_a)$ and $r_{pop}(s, \mathcal{I}_a)$ represent the ranks of s among $\mathcal{I}_a$ in terms of AP and popularity, respectively. The songs are then ranked in ascending order of score. For area (b), (c), and (d), the score is given by $r_{pop}(s, \mathcal{I}_a) - r_{ap}(s, \mathcal{I}_a)$, $r_{ap}(s, \mathcal{I}_a) - r_{pop}(s, \mathcal{I}_a)$, and $-r_{ap}(s, \mathcal{I}_a) - r_{pop}(s, \mathcal{I}_a)$, respectively. In these areas, the songs are also ranked in ascending order. Regarding OS, songs in each area has the same meaning as with PA and we can also make a song ranking for each area in the same manner as with PA.

Table 3 shows example results of the top five song rankings in each area for "The Black Keys" in terms of PA. It would be beneficial to show each ranking to a user according to her familiarity with "The Black Keys."

5.2 Typicality-oriented Search

Because the parameters of all artists and all of their songs are learned by using the same optimization criterion (Eq. 2), all artists and all songs can be embedded into the same feature space. This means that given artist a, all songs in the dataset can be ranked in terms of OS or PA. In other words, we can search for songs that represent a's typical characteristics well even if they were not a's songs. By showing such songs to a user who is a fan of a, she may be willing to listen to unfamiliar songs because they are highly related to her favorite artist. This is also benefi-

Rank	OS	PA
1	California Gurls / Katy Perry	Circus / Britney Spears
2	Racy Lacey / Girls Aloud	...Baby One More Time / Britney Spears
3	Gimme More / Britney Spears	I Wanna Go / Britney Spears
4	Cannibal / Ke$ha	Hung Up / Madonna
5	Piece of Me / Britney Spears	I Kissed a Girl / Katy Perry

Rank	OS	PA
1	Cymbal Rush / Thom Yorke	Untitled / Interpol
2	Atoms for Peace / Thom Yorke	The Clock / Thom Yorke
3	And It Rained All Night / Thom Yorke	I've Seen It All / Björk
4	Convergence / Jonny Greenwood	Svefn-g-englar / Sigur Rós
5	Quick Canal / Atlas Sound	Kingdom of Rust / Doves

Table 4. Ranking results of typicality-oriented search (top: "Lady Gaga," bottom: "Radiohead").

cial for online music streaming services because they can expand users' interests and let users listen to more songs.

Given artist a, when we generate a ranked list of all songs in $\mathcal{I}$ in terms of OS, we compute the score of each song by $f_{os}(a, s)$, which was defined in Section 3.3.1, and rank all songs in descending order of scores. Similarly, for PA, we use $f_{pa}(a, s)$, which was defined in Section 3.3.2, and rank all songs in descending order of scores.

Table 4 shows the top five song rankings for "Lady Gaga" and "Radiohead." Although we do not use acoustic features and metadata of songs, in the case of "Lady Gaga," all artists in the table are female artists. Similarly, in the case of "Radiohead," members of the band such as "Thom Yorke" and "Jonny Greenwood" are retrieved. In addition, because "Radiohead" is a rock band, rock bands such as "Interpol," "Sigur Rós," and "Doves" are ranked at higher positions. These results show the potential of this application to enable users to find new attractive songs.

5.3 Analogy Search

When a user searches for an object in an unfamiliar domain and obtains the desired search results, it is helpful to give an example object in her familiar domain to the search system. This kind of search is known as analogy search and its usefulness to search for persons [54] and restaurants [55] has been studied. An analogy search for MIR can also be an attractive application as follows. Suppose a user is a big fan of "Eminem" and likes his song "Not Afraid." She has recently become interested in "Lady Gaga." However, because she has little knowledge on "Lady Gaga" and there are too many "Lady Gaga" songs, she is at a loss as to which song she should listen to. In such a case, by using an analogy, she can ask something like "what Lady Gaga song corresponds to Not Afraid by Eminem?" If we can return search results for such a query, it would be helpful for her to try some songs of "Lady Gaga."

In an analogy search for MIR, given a query consisting of source artist a^s, source song s^s, and target artist a^t, our goal is to return a^t's song s^t where the relation between a^t and s^t corresponds to that between a^s and s^s. We compute the relation between an artist and a song based on the angle between their latent vectors and the ratio of their lengths because they respectively represent the qualitative and quantitative aspects of artists/songs as we described in Section 3.3. Intuitively, if the angle between a^s and s^s is similar to that between a^t and s^t, and the ratio of $\|\boldsymbol{\nu_{s^s}}\|$ to $\|\boldsymbol{\nu_{a^s}}\|$ is also similar to that of $\|\boldsymbol{\nu_{s^t}}\|$ to $\|\boldsymbol{\nu_{a^t}}\|$, we regard

Query			Rank	Song title
Source artist: Madonna			1	Yoü and I
Source song: Like a Prayer			2	Poker Face
Target artist: Lady Gaga			3	Telephone

Query			Rank	Song title
Source artist: Eminem			1	The Edge of Glory
Source song: Not Afraid			2	Bad Romance
Target artist: Lady Gaga			3	Yoü and I

Query			Rank	Song title
Source artist: The Beatles			1	Milk Cow Blues
Source song: That Means A Lot			2	Face
Target artist: Aerosmith			3	Temperature

Table 5. Ranking results of analogy search.

s^t as a good analogy search result of the query. However, because the degree of the scatter of songs in the feature space is different from one artist to another, we need to normalize the angles and ratios between an artist and its songs. Formally, given a^s, we first compute the angle between $\boldsymbol{\nu_{a^s}}$ and the vector of each of a^s's songs. The angles are then normalized to fit into the interval $[0, 1]$ by min-max normalization. Let θ_{s^s} denote the normalized angle of s^s. We also compute the ratio of the length of each of a^s's songs to $\boldsymbol{\nu_{a^s}}$'s length (i.e., $\|\boldsymbol{\nu_s}\|/\|\boldsymbol{\nu_{a^s}}\|$ where $s \in \mathcal{I}_{a^s}$). Again, min-max normalization is applied and let r_{s^s} be the normalized ratio of s^s. Similarly, we compute the normalized angles and ratios for each of a^t's songs. The score of $s^t \in \mathcal{I}_{a^t}$ for analogy search is then computed as follows:

$$f_{analogy}(a^s, s^s, a^t, s^t) = \gamma|\theta_{s^s} - \theta_{s^t}| + (1-\gamma)|r_{s^s} - r_{s^t}|,$$

where γ is a parameter to determine the weights on angle similarity and length-ratio similarity. Finally, a^t's songs are ranked in ascending order of $f_{analogy}(a^s, s^s, a^t, s^t)$.

In Table 5, we show example results where the top three songs are listed for each query. The value of γ is set to 0.5. In the top table, the source song is "Like a Prayer," which is a signature piece for "Madonna," and the target artist is "Lady Gaga." In this case, the signature pieces for "Lady Gaga" are ranked higher. Even when the source artist is "Eminem," whose music is largely different from that of "Lady Gaga," her signature pieces are retrieved for his signature piece "Not Afraid." When the source song is "That Means A Lot," which has a low PA with "The Beatles," songs with a low PA with "Aerosmith" are retrieved. From these results, we can say that this application can search for the target artist's songs that have a similar relationship between the source artist and the source song.

6. CONCLUSION

This paper analyzed song/artist latent features by embedding them into a same feature space. Based on the analysis results, we suggested three applications for song search. We acknowledge a limitation of this paper in that we did not quantitatively evaluate the search results of those applications. Nonetheless, we believe this study is worthwhile contribution as a first step toward leveraging latent vectors of songs and artists. In future work, we plan to quantitatively evaluate the usefulness of our proposed applications by conducting user studies. We also want other researchers to leverage the concepts of OS/PA and realize useful music information retrieval systems.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] M. Levy and M. Sandler, "A semantic space for music derived from social tags," in *Proceedings of the 8th International Conference on Music Information Retrieval*, ser. ISMIR '07, 2007, pp. 411–416.

[2] J. A. Russell, "A circumplex model of affect," *Journal of personality and social psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[3] S. Chapaneri and D. Jayaswal, "Structured prediction of music mood with twin gaussian processes," in *Proceedings of the 7th International Conference on Pattern Recognition and Machine Intelligence*, ser. PReMI '17, 2017, pp. 647–654.

[4] K. Tsukuda, K. Ishida, and M. Goto, "Lyric Jumper: A lyrics-based music exploratory web service by modeling lyrics generative process," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, ser. ISMIR '17, 2017, pp. 544–551.

[5] J. L. Moore, S. Chen, T. Joachims, and D. Turnbull, "Learning to embed songs and tags for playlist prediction," in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, ser. ISMIR '12, 2012, pp. 349–354.

[6] K. Pugatschewski, T. Köllmer, and A. M. Kruspe, "Playlists from the matrix - combining audio and metadata in semantic embeddings," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, ser. ISMIR '16, 2016.

[7] G. Karamanolakis, E. Iosif, A. Zlatintsi, A. Pikrakis, and A. Potamianos, "Audio-based distributional representations of meaning using a fusion of feature encodings," in *Proceedings of the 17th Annual Conference of the International Speech Communication Association*, ser. INTERSPEECH '16, 2016, pp. 3658–3662.

[8] J. Weston, S. Bengio, and P. Hamel, "Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval," *Journal of New Music Research*, vol. 40, no. 4, pp. 337–348, 2011.

[9] S. Rendle, "Factorization machines," in *Proceedings of the 2010 IEEE International Conference on Data Mining*, ser. ICDM '10, 2010, pp. 995–1000.

[10] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[11] C.-M. Chen, M.-F. Tsai, J.-Y. Liu, and Y.-H. Yang, "Using emotional context from article for contextual music recommendation," in *Proceedings of the 21st ACM International Conference on Multimedia*, ser. MM '13, 2013, pp. 649–652.

[12] A. Vall, M. Skowron, P. Knees, and M. Schedl, "Improving music recommendations with a weighted factorization of the tagging activity," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ser. ISMIR '15, 2015, pp. 65–71.

[13] O. Gouvert, T. Oberlin, and C. Févotte, "Matrix cofactorization for cold-start recommendation," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, ser. ISMIR '18, 2018, pp. 792–798.

[14] D. Yang, T. Chen, W. Zhang, Q. Lu, and Y. Yu, "Local implicit feedback mining for music recommendation," in *Proceedings of the 6th ACM Conference on Recommender Systems*, ser. RecSys '12, 2012, pp. 91–98.

[15] F. Yuan, G. Guo, J. M. Jose, L. Chen, H. Yu, and W. Zhang, "Boostfm: Boosted factorization machines for top-N feature-based recommendation," in *Proceedings of the 22Nd International Conference on Intelligent User Interfaces*, ser. IUI '17, 2017, pp. 45–54.

[16] ——, "Optimizing factorization machines for top-N context-aware recommendations," in *Proceedings of the 17th International Conference on Web Information Systems Engineering*, ser. WISE '16, 2016, pp. 278–293.

[17] M. Pichl, E. Zangerle, and G. Specht, "Improving context-aware music recommender systems: Beyond the pre-filtering approach," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, ser. ICMR '17, 2017, pp. 201–208.

[18] G. Vigliensoni and I. Fujinaga, "Automatic music recommendation systems: Do demographic, profiling, and contextual features improve their performance?" in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, ser. ISMIR '16, 2016, pp. 94–100.

[19] J. L. Moore, S. Chen, D. Turnbull, and T. Joachims, "Taste over time: The temporal dynamics of user preferences," in *Proceedings of the 14th International Society for Music Information Retrieval Conference*, ser. ISMIR '13, 2013, pp. 401–406.

[20] C. Chung, Y. Chen, and H. H. Chen, "Exploiting playlists for representation of songs and words for text-based music retrieval," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, ser. ISMIR '17, 2017, pp. 478–485.

[21] M. Schedl, "The LFM-1b dataset for music retrieval and recommendation," in *Proceedings of the 2016*

ACM on International Conference on Multimedia Retrieval, ser. ICMR '16, 2016, pp. 103–110.

[22] G. Vigliensoni and I. Fujinaga, "The music listening histories dataset," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, ser. ISMIR '17, 2017, pp. 96–102.

[23] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, ser. ISMIR '11, 2011, pp. 591–596.

[24] Y. Labs, "R2 - Yahoo! Music user ratings of songs with artist, album, and genre metainformation, v.1.0," https://webscope.sandbox.yahoo.com/catalog.php?datatype=r.

[25] Ò. Celma, *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space.* Springer, 2010.

[26] E. Zangerle, M. Pichl, W. Gassler, and G. Specht, "#nowplaying music dataset: Extracting listening behavior from twitter," in *Proceedings of the 1st International Workshop on Internet-Scale Multimedia Management*, ser. WISMM '14, 2014, pp. 21–26.

[27] D. Hauger, M. Schedl, A. Kosir, and M. Tkalcic, "The million musical tweet dataset - what we can learn from microblogs," in *Proceedings of the 14th International Society for Music Information Retrieval Conference*, ser. ISMIR '13, 2013, pp. 189–194.

[28] E. Pollastri, "An audio front end for query-by-humming systems," in *Proceedings of the 2nd International Symposium on Music Information Retrieval*, ser. ISMIR '01, 2001, pp. 65–72.

[29] T. Sorsa and K. Halonen, "Mobile melody recognition system with voice-only user interface," in *Proceedings of the 3rd International Conference on Music Information Retrieval*, ser. ISMIR '02, 2002, pp. 279–280.

[30] S. Pauws, "CubyHum: A fully operational "query by humming" system," in *Proceedings of the 3rd International Conference on Music Information Retrieval*, ser. ISMIR '02, 2002, pp. 187–196.

[31] A. Ito, S. Heo, M. Suzuki, and S. Makino, "Comparison of features for dp-matching based query-by-humming system," in *Proceedings of the 5th International Conference on Music Information Retrieval*, ser. ISMIR '04, 2004, pp. 297–303.

[32] D. Little, D. Raffensperger, and B. Pardo, "A query by humming system that learns from experience," in *Proceedings of the 8th International Conference on Music Information Retrieval*, ser. ISMIR '07, 2007, pp. 335–338.

[33] P. Ferraro, P. Hanna, L. Imbert, and T. Izard, "Accelerating query-by-humming on GPU," in *Proceedings of the 10th International Society for Music Information Retrieval Conference*, ser. ISMIR '09, 2009, pp. 279–284.

[34] C. de la Bandera, A. M. Barbancho, L. J. Tardón, S. Sammartino, and I. Barbancho, "Humming method for content-based music information retrieval," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, ser. ISMIR '11, 2011, pp. 49–54.

[35] E. Molina, L. J. Tardón, I. Barbancho, and A. M. Barbancho, "The importance of F0 tracking in query-by-singing-humming," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, ser. ISMIR '14, 2014, pp. 277–282.

[36] C. Wang, J. R. Jang, and W. Wang, "An improved query by singing/humming system using melody and lyrics information," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ser. ISMIR '10, 2010, pp. 45–50.

[37] A. Duda, A. Nürnberger, and S. Stober, "Towards query by singing/humming on audio databases," in *Proceedings of the 8th International Conference on Music Information Retrieval*, ser. ISMIR '07, 2007, pp. 331–334.

[38] J. R. Jang, C. Hsu, and H. Lee, "Continuous HMM and its enhancement for singing/humming query retrieval," in *Proceedings of the 6th International Conference on Music Information Retrieval*, ser. ISMIR '05, 2005, pp. 546–551.

[39] E. Allamanche, J. Herre, O. Hellmuth, T. Kastner, and C. Ertel, "A multiple feature model for musical similarity retrieval," in *Proceedings of the 4th International Conference on Music Information Retrieval*, ser. ISMIR '03, 2003, pp. 217–218.

[40] J. Aucouturier and F. Pachet, "Music similarity measures: What's the use?" in *Proceedings of the 3rd International Conference on Music Information Retrieval*, ser. ISMIR '02, 2002, pp. 157–163.

[41] H. Fujihara and M. Goto, "A music information retrieval system based on singing voice timbre," in *Proceedings of the 8th International Conference on Music Information Retrieval*, ser. ISMIR '07, 2007, pp. 467–470.

[42] F. Vignoli and S. Pauws, "A music retrieval system based on user driven similarity and its evaluation," in *Proceedings of the 6th International Conference on Music Information Retrieval*, ser. ISMIR '05, 2005, pp. 272–279.

[43] D. R. Turnbull, L. Barrington, G. Lanckriet, and M. Yazdani, "Combining audio content and social context for semantic music discovery," in *Proceedings of*

the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '09, 2009, pp. 387–394.

[44] P. Knees, T. Pohle, M. Schedl, and G. Widmer, "A music search engine built upon audio-based and web-based similarity measures," in *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '07, 2007, pp. 447–454.

[45] E. Brochu and N. d. Freitas, ""Name that song!": A probabilistic approach to querying on music and text," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, ser. NIPS'02, 2002, pp. 1529–1536.

[46] D. Lim, J. McAuley, and G. Lanckriet, "Top-N recommendation with missing implicit feedback," in *Proceedings of the 9th ACM Conference on Recommender Systems*, ser. RecSys '15, 2015, pp. 309–312.

[47] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei, "Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence," in *Proceedings of the 10th ACM Conference on Recommender Systems*, ser. RecSys '16, 2016, pp. 59–66.

[48] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, ser. UAI '09, 2009, pp. 452–461.

[49] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16, 2016, pp. 265–283.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, ser. ICLR '15, 2015.

[51] K. Tsukuda, S. Fukayama, and M. Goto, "ABCPRec: Adaptively bridging consumer and producer roles for user-generated content recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR'19, 2019, p. 1197–1200.

[52] W. Pan and L. Chen, "GBPR: Group preference based bayesian personalized ranking for one-class collaborative filtering," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, ser. IJCAI '13, 2013, pp. 2691–2697.

[53] R. He and J. McAuley, "VBPR: Visual bayesian personalized ranking from implicit feedback," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, ser. AAAI'16, 2016, pp. 144–150.

[54] Y. Zhang, A. Jatowt, and K. Tanaka, "Is tofu the cheese of Asia?: Searching for corresponding objects across geographical areas," in *Proceedings of the 26th International Conference on World Wide Web Companion*, ser. WWW '17 Companion, 2017, pp. 1033–1042.

[55] M. P. Kato, H. Ohshima, and K. Tanaka, "Content-based retrieval for heterogeneous domains: Domain adaptation by relative aggregation points," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '12, 2012, pp. 811–820.

DETECTING COLLABORATION PROFILES IN SUCCESS-BASED MUSIC GENRE NETWORKS

Gabriel P. Oliveira   **Mariana O. Silva**   **Danilo B. Seufitelli**
Anisio Lacerda   **Mirella M. Moro**
Universidade Federal de Minas Gerais, Brazil

{gabrielpoliveira,mariana.santos,daniloboechat,anisio,mirella}@dcc.ufmg.br
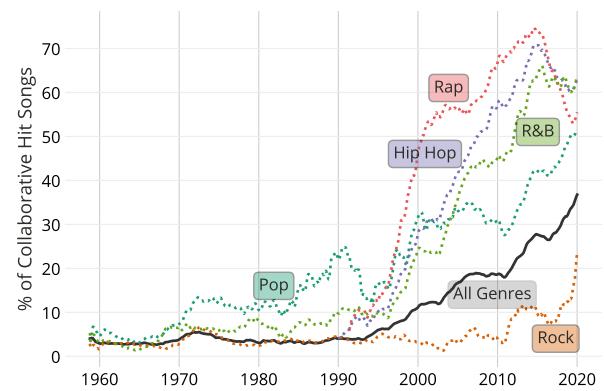
ABSTRACT

We analyze and identify collaboration profiles in success-based music genre networks. Such networks are built upon data recently collected from both global and regional Spotify weekly charts. Overall, our findings reveal an increase in the number of distinct successful genres from high-potential markets, pointing out that local repertoire is more important than ever on building the global music ecosystem. We also detect collaboration patterns mapped into four different profiles: *Solid*, *Regular*, *Bridge* and *Emerging*, wherein the two first depict higher average success. These findings indicate great opportunities for the music industry by revealing the driving power of inter-genre collaborations within regional and global markets.

1. INTRODUCTION

Artist collaborations are more popular than ever, as the landscape of the music industry becomes more complex. This widely adopted strategy is a strong force driving music nowadays, maintaining artists' presence and relevance in the market. Such connections usually help artists bridge the gap between styles and genres, overlapping new fan bases and consequently increasing their numbers. Figure 1 illustrates this phenomenon and highlights the growing trend in the number of collaborations within Billboard Hot 100 Charts. Although the general curve increases over time, genres such as *rap* and *hip-hop* present a collaboration rate higher than others (e.g., *pop* and *rock*). This contrast can be explained by the intrinsic nature of each music genre. For instance, *rap* and *hip-hop* artists frequently collaborate with the *pop* community, mainly as featured artists. Moreover, partnerships involving *pop* music may take place not only through intra-genre collaborations but also through inter-genres, bringing an additional dimension to their songs.

Musicians teaming up is nothing new but has risen far beyond the norm. Remaining an industry of creative growth, it is only natural for music (i.e., all musical scene members)

Figure 1: Historical frequency of collaborative hit songs for selected genres on Billboard Hot 100 Chart (1958 - 2020).

adapting to new conditions and redefining its layout. Not surprisingly, the Grammy categories were tightened (from 109 to 78, in 2012) as a result of its dynamic nature.[1] That is, the notion of categories and genres are blurred as never before. Through cross-genre collaboration, artists are naturally venturing into new domains and working outside of the category which they had originally been ascribed to. Such a collaboration phenomenon may be drastically reshaping music global environment, by challenging segments of certain genres to come up with something entirely new [1]. Moreover, this gradual revolution is becoming a driving force in creating a more collaborative scenario, making music one of the most innovative art forms.

As this creative market changes, it becomes more unpredictable; and doing both predictive and diagnostic analyses in such a context remains challenging. Still, we believe factors leading to an ideal musical partnership can be understood by exploring collaboration patterns that directly impact its success [1–3]. Hence, we aim to unveil the dynamics of cross-genre connections and collaboration profiles in success-based networks (i.e., connections formed by genres of artists who cooperate and create hit songs). We do so through the following research questions (RQ).

RQ1: *Does the regional aspect impact on popular genres and their hit songs?*

RQ2: *How has genre collaboration evolved over the past few years?*

---

[1] Grammy Award: https://en.wikipedia.org/wiki/Grammy_Award

RQ3: *Which are the potentially intrinsic factors and indicators that influence the collaboration success?*

In order to answer such questions, we first model genre collaboration in the music scenario as success-based networks (Section 3.1). Then, we build a novel dataset with data from global and regional markets (Section 3.2) and present the network science concepts and metrics required for understanding the paper (Section 3.3). Overall, our analyses and experiments over the networks reveal that:

(1) Individually analyzing regional markets is fundamental, as local genres play a key role on determining hit songs and popular artists (Section 4.1).

(2) In general, genre collaborations are increasing, with emerging local genres hitting global success – despite the differences in the evolution of regional markets (Section 4.2).

(3) Genre collaborations analyses describe three significant factors (*Attractiveness, Affinity* and *Influence*, Section 5.1) to uncover four profiles (*Solid, Regular, Bridge* and *Emerging*, Section 5.3).
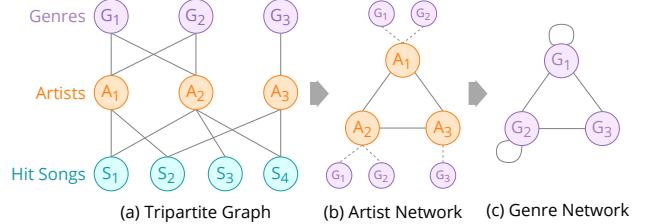
2. RELATED WORK

Genres are fundamental within the musical scenario by aggregating songs that share common characteristics. Hence, they are frequently used in the field of Music Information Retrieval (MIR), which aims to extract relevant information from music content. In fact, several tasks are genre-dependent or directly related to them, such as automatic genre classification, which has been largely studied by the MIR community over the years [4–8]. Nonetheless, there are also genre-aware studies assessing music source separation [9], genre modeling [10], preferences [11], disambiguation/translation [12, 13], new datasets [14] and ontologies [15]. Network science, the core of our methodology, has also been used to model genres into influence networks [2] and song communities [16].

Hit Song Science (HSS), which tackles the problem of predicting the popularity of a given song, is also an emerging field within MIR. Thus, different studies analyze the impact of acoustic and social features in musical success [17–20], some of them including genre information [21,22]. Moreover, Silva et al. [1] address collaboration as a key factor in success, using topological properties to detect relevant profiles in artist networks. Such an approach is novel and promising in HSS, but it is restricted to the artist and song levels. Therefore, studying collaboration from a genre perspective may reveal important information on how artists from different communities team up to make a new hit song. To the best of our knowledge, we are the first to build a success-based genre network and detect collaboration profiles within it, going deeper into the intrinsic factors that make up a successful collaboration.

3. METHODOLOGY

This work aims to detect collaboration profiles over music genre networks. Building a genre network (Section 3.1) re-

Figure 2: Reduction from the tripartite (a) to the one-mode Genre Collaboration Network (c). The intermediate step is an Artist Network with genre information (b). Artists and genres are linked when hit songs involve both nodes.

quires a proper dataset (Section 3.2), and finding its profiles needs different metrics (Section 3.3), as described next.

3.1 Genre Collaboration Network

A Collaboration Network is usually modeled as a graph formed by nodes (vertices) that may be connected through edges. For example, nodes represent artists and are connected by an edge if the respective pair of artists has collaborated in a song. Now, to analyze the interactions between genres, we model music collaboration as a tripartite graph, in which nodes are divided into three sets: genres, artists, and hit songs; i.e., the minimum elements to evaluate success. The building process of the genre network from the tripartite model is illustrated in Figure 2. Collaborative hit songs are sung by two or more artists, regardless of their participation (e.g., a typical *feat.* or a duet). We also equally consider all genres linked to an artist because they shape how such an artist is seen by fans and music industry.

For analyzing the interaction between musical genres, we reduce the tripartite model into a one-mode network in which nodes are exclusively genres. However, such a reduction is only possible by executing an intermediate step: building the artist collaboration network, Figure 2(b). In such a network, two artists are connected when both collaborate in one or more hit songs. The genres are not lost, as they are linked directly to the artists.

We may now build the final network by connecting the genres of artists who collaborate in the artist network. The edges are undirected and weighted by the number of hit songs involving artists from both genres, Figure 2(c). Also, self-loop edges are allowed, as there are hit songs from artists of the same genre. For example, the song *Old Town Road* [2] by Lil Nas X and Billy Ray Cyrus generates an edge between these artists in the intermediate network; and each of Lil Nas X's genres (*pop rap, country pop* and *hip hop*) is linked to Cyrus' only genre (*country*) with weight 1.

3.2 Dataset Building Process

Over recent years, the world has seen a dramatic change in the way people consume music, moving from physical records to streaming services. Since 2017, such services have become the main source of revenue within the global

---

[2] #1 Song of 2019 according to Billboard Year-End Hot 100 Chart: https://billboard.com/charts/year-end/2019/hot-100-songs

recorded music market. In fact, streaming revenues increased by 75.4% from then, reaching a total amount of US$ 11.4 billion by the end of 2019. [3] Thus, we build our dataset by using data from Spotify, the most popular global audio streaming service, with more than 286 million users across 79 markets. [4] It provides a weekly chart of the 200 most streamed songs in all its markets, and an aggregated global chart. We collect global and regional charts from January 2017 to December 2019, considering eight of the top 10 [5] music markets according to IFPI: United States, Japan [6], United Kingdom, Germany, France, Canada, Australia, and Brazil. We also use Spotify API [7] to gather information about the hit songs and artists present in the charts, such as all collaborating artists within a song (since the charts only provide the main ones) and their respective genres, which is the core of this work. Our final dataset contains 1,370 charts from 156 weeks, comprising 13,880 hit songs and 3,612 artists from 896 different music genres.

Then, a processing phase focuses on the artists' genres, because Spotify assigns a list of very specific genres to each artist. For example, Jay-Z (one of the most popular rappers) is assigned to both *east coast hip hop* and *hip hop* genres, which may be described only by *hip hop*. To simplify our modeling and further analyses, we choose to map all specific genres to more embracing and well-established *super-genres*. Note that the regional aspects are not lost in such a mapping, because our analyses are made separately for each considered market. Hence, the 896 existing genres are now mapped into 162 *super-genres*. The dataset and genre mapping are publicly available on our project page. [8]

3.3 Network Science Metrics

In this work, we use well-established network science metrics to analyze musical collaboration. Such metrics consider the network topological features, i.e., they relate to the network structure (nodes and edges) as follows. [9]

Degree and Weighted Degree. These metrics refer to the connectivity of each node in the network. The degree of a node is its amount of incident edges, and the weighted degree is the sum of the edges' weight. In our genre collaboration network, degree stands for the number of genre connections, and weighted degree represents the number of hit songs shared by both genres.

Clustering Coefficient (CC). Measures the tendency of neighbors of a node to be connected themselves. The higher its value, the more interconnected the node neighborhood.

Common Neighbors (CN). The number of neighbors that a given pair of nodes has in common in a network.

Neighborhood Overlap (NO). The ratio between the common neighbors of a given pair of nodes and the union set of their neighbors. Edges with low NO reveal local bridges in

Table 1: Most popular music genres in each considered market from 2017 to 2019.

	Genre	Songs	Arts.		Genre	Songs	Arts.		Genre	Songs	Arts.
Global	pop	1,715	424	**Canada**	pop	1,790	402	**UK**	pop	1,772	371
	hip hop	1,192	281		rap	1,762	209		hip hop	1,138	232
	rap	1,184	195		hip hop	1,511	232		rap	974	167
	pop rap	845	130		pop rap	1,355	149		dance pop	922	178
	dance pop	832	165		trap	1,139	154		pop rap	660	120
Australia	pop	1,646	411	**Germany**	hip hop	2,604	352	**Japan**	j-pop	797	163
	rap	873	165		pop	1,665	479		pop	705	210
	dance pop	822	171		rap	1,223	205		dance pop	438	103
	hip hop	792	191		dance pop	650	159		j-rock	312	72
	pop rap	657	133		pop rap	462	109		r&b	276	82
Brazil	pop	1,072	256	**France**	pop	3,138	470	**USA**	rap	2,057	231
	sertanejo	565	82		hip hop	2,660	285		hip hop	1,719	241
	brazilian funk	559	156		rap	2,526	245		pop	1,704	340
	dance pop	415	101		francoton	1,097	82		pop rap	1,518	139
	electro	307	93		dance pop	391	119		trap	1,370	172

the network, i.e., nodes traveling in "social circles", having almost no common connection.

Preferential Attatchment (PA). The probability of a given pair of nodes connecting in the future. The intuition is the more neighbors they have, the more likely they are to connect in the future.

Edge Betweenness (EB). The fraction of shortest paths that go through an edge in the network. Edges with a high score represent a bridge-like connector between two parts of the network, and their removal may affect the communication between others due to the lost common shortest paths.

Resource Allocation (RA). For a pair of nodes, it represents the fraction of a resource (e.g., information) that a node can send to another through its common neighbors.

4. EXPLORATORY ANALYSIS

We perform an exploratory analysis of the data collected in two main steps. First, we analyze Spotify charts for each market to detect popular genres (Section 4.1). Then, we characterize the genre collaboration network to understand the evolution of each market (Section 4.2).

4.1 Music Genres Overview

To answer *RQ1*, we analyze charts of eight markets over three years (see Methodology) in Table 1. Each country has its own musical inclinations, although the predominant genres are mostly *pop/pop rap*, *hip-hop*, and *rap*. Such preference may be due to the increasing number of collaboration songs among artists from different musical genres, as revealed in Figure 1: growing collaborations of *pop*, *rap*, *hip-hop*, and *R&B* in recent years. Also, except for *R&B*, they are the main genres at the top-5 genre lists on most markets; i.e., such genres are among both the most collaborative ones and the most listened on the globe. Moreover, there are three markets with local genres on their top-5 list: Brazil with *sertanejo* and *brazilian funk*; France with *francoton*; and Japan with *j-pop* and *j-rock*. Although local, such genres are potentially good choices for record companies to encourage musical collaborations. Note local engagement shapes the global environment, ensuring that music culture within such countries can develop and progress.

---

[3] IFPI Global Music Report 2019: https://gmr.ifpi.org/

[4] Spotify Company Info: https://newsroom.spotify.com/company-info/

[5] Data from South Korea and China was not available in Spotify.

[6] The first Japanese weekly chart is from August 31, 2017.

[7] Spotify API: https://developer.spotify.com/

[8] Project Bàde: https://bit.ly/proj-Bade

[9] For more information on such metrics, see references [23–25]

Table 2: Network characterization for global and three regional markets, representing the groups of countries with similar network evolution. Underlined values are the highest metric value for a specific market throughout the considered period.

| Metric	Global			USA (Group 1)			Brazil (Group 2)			UK (Group 3)		
	2017	2018	2019	2017	2018	2019	2017	2018	2019	2017	2018	2019
Number of genres (nodes)	72	79	89	76	73	83	58	63	61	74	76	79
Number of collabs (edges)	564	583	709	542	522	670	453	524	392	610	605	627
Average degree	15.7	14.8	15.9	14.3	14.3	16.1	15.6	16.6	12.9	16.5	15.9	15.9
Average degree (weighted)	256.9	247.4	236.7	324.6	287.9	241.4	136.1	133.0	95.3	216.5	203.6	159.5
Density	0.221	0.189	0.181	190	0.199	197	0.274	268	214	0.226	212	204
Average Clustering Coefficient	0.743	0.757	0.754	0.762	760	726	0.770	758	677	724	0.754	738
Number of self-loops	24	21	28	25	22	27	24	29	27	28	25	30
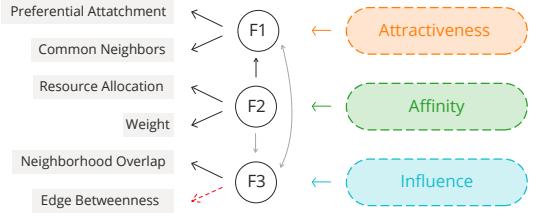
4.2 Network Characterization

After analyzing the charts, we build the genre collaboration network for each market and year to find out how genres connect to answer $RQ2$. With nine markets (global and eight countries) during three years, we analyze 27 networks [10]. For each network, we calculate basic statistics on its nodes and edges, as well as structural metrics (Section 3.3). Table 2 shows the results for selected markets.

First, the global genre networks reveal the world is more open to new successful genres (number of nodes/genres growth). Also, the number of genre connections (edges) increased considerably, meaning more collaborative hit songs are coming from artists whose genres are not linked in prior networks, opening up new opportunities for those genres to acquire new listeners. The networks average degree remains stable, while its weighted value decreases over the years. This could reveal a growth in the number of collaborations of well-established genres with emerging ones, represented by edges with low weight values (few hit songs). Still, such low-degree emerging genres may become popular shortly, expanding their collaborations to other unexplored genres. For instance, *k-pop* connections double as it spreads worldwide, approaching genres such as *reggaeton* (e.g., the collaboration between J-Hope from BTS and Becky G in the song *Chicken Noodle Soup*, September 2019).

For regional markets, we classify the countries into three groups, according to the similarities in networks' evolution: *(i)* USA and Canada; *(ii)* Brazil, France, Germany and Japan; *(iii)* UK and Australia. As the global network, countries in the first group have an increasing average degree and a decreasing average clustering coefficient, thus indicating a stronger tendency to diversify the inter-genre collaborations. Then, the second group includes non-English speaking countries with decreasing connectivity metrics in 2019, after a peak in 2018. The last group has countries in which more genres are becoming successful, while the connections are not increasing in the same proportion.

Overall, considering regional markets individually becomes more important for producers and record labels, as they are delivering more global hits over time. Their distinct behavior emphasizes the strength of cultural aspects on determining how music is consumed and the success of a given genre or artist. In each market, genre connections may reveal distinct profiles, which are an important tool for analyzing successful genre collaborations.

---

[10] All networks can be visualized in https://bit.ly/proj-Bade

Figure 3: EFA diagram. Solid and red dashed lines represent positive and negative correlations, respectively. Dark and lighter lines represent strong $[0.6 - 1.0]$ and weak $[< 0.6]$ correlations, respectively.

5. GENRE COLLABORATION PROFILES
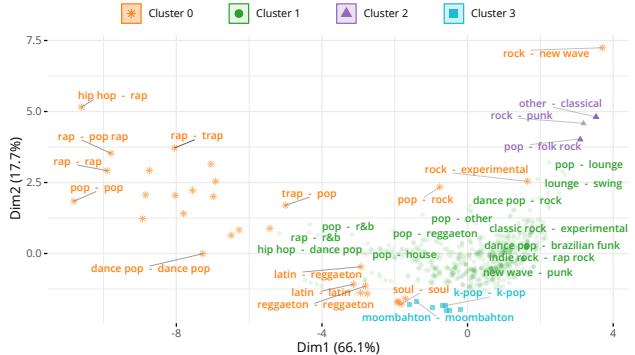
This section presents our approach to uncover significant factors that compose a successful music genre collaboration. Inspired by Silva et al. [1], we first extract information from the success-based networks by evaluating six edge-dependent metrics (Section 3.3). We perform an Exploratory Factor Analysis on such metrics to define factors in Section 5.1, and then perform cluster analysis in Section 5.2. Finally, we organize the found clusters into collaboration profiles in Section 5.3, to investigate the key driving factors on successful collaborations and then answer $RQ3$.

5.1 Exploratory Factor Analysis

Exploratory Factor Analysis (EFA) [26] is a statistical method designed to underline patterns of correlations among observed variables and extract latent factors. Generally, EFA identifies the number of common factors and the pattern of factor loadings (correlations). It assumes and asserts that manifest (observed) variables are expressed as a linear combination of factors and measurement errors. Each factor explains a particular variance in the variables and may find hidden data patterns. There are two main issues when executing an EFA: *(i)* determining the number of factors to retain for analysis, and *(ii)* selecting the final structure for how the measured variables relate to the factors. For the former, we use the Parallel Analysis criteria [27], which is based on random data simulation. The suggested number of factors to extract is then provided and based on examining the *scree plot* [28] of factors of the observed data with that of a random data matrix of the same size as the original. Finally, the EFA is performed using the well known Ordinary Least Squares (OLS) factoring method and an oblique rotation, allowing factors to correlate with each other.

We use EFA to identify the common factors and the

Figure 4: Clustering result for the USA network, in 2019, with examples of some genre collaborations in each cluster.

Figure 5: Collaboration profiles for all markets (2019). For additional radar plots, see Supplementary Material.

relationships among the edge-dependent metrics of all 27 success-based networks. Overall, the analysis results suggest a three-factor structure within those six metrics. A graphical representation of the emerging structure is in Figure 3. As the three factors are conceptually coherent, we labeled them as follows.

Attractiveness (F1). Factor 1 has high loads for both PA and CN metrics, with a positive correlation between them. Specifically, values close to 0 indicate that two nodes are not close and attracted, while higher values indicate closer nodes. Therefore, this factor corresponds to the predisposition of two nodes to connect in the future.
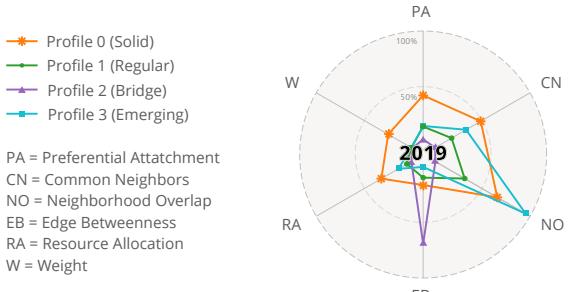
Affinity (F2). Factor 2 has high loads for both RA and W metrics, with a positive correlation between them. High values indicate strong social ties, and lower ones indicate weak ties. Hence, this factor measures both the frequency of collaboration between two nodes and the social strength.

Influence (F3). Factor 3 has high loads for both NO and EB metrics, with a negative correlation between them. Edges with low NO and high EB certainly consist of local bridges in the network. That is, they represent a bridge-like connector between two "social circles". Therefore, this factor corresponds to the importance level of an edge with access to different regions in the network.

5.2 Cluster Analysis

The second step of our approach is cluster analysis to group similar music genre connections based on the aforementioned factors. We use DBSCAN [29] as a clustering algorithm, which assigns data points to the same cluster if they are *density-reachable* from each other. Two important parameters are required for DBSCAN: ϵ defines the radius of neighborhood around a point x; and $MinPts$ (minimum points) is the minimum number of neighbors within the ϵ radius. To choose the optimal ϵ value, we use a method based on *k*-nearest neighbor distances, which calculates the average of the distances of every data point to its k nearest neighbors. In general, the value of k is specified by the user and corresponds to the $MinPts$ parameter. As a general rule, the $MinPts$ can be derived from the number of dimensions D in the dataset as $MinPts \geq D + 1$. Since we have six topological metrics, we set $MinPts = 7$.

Overall, four distinct clusters were detected in at least
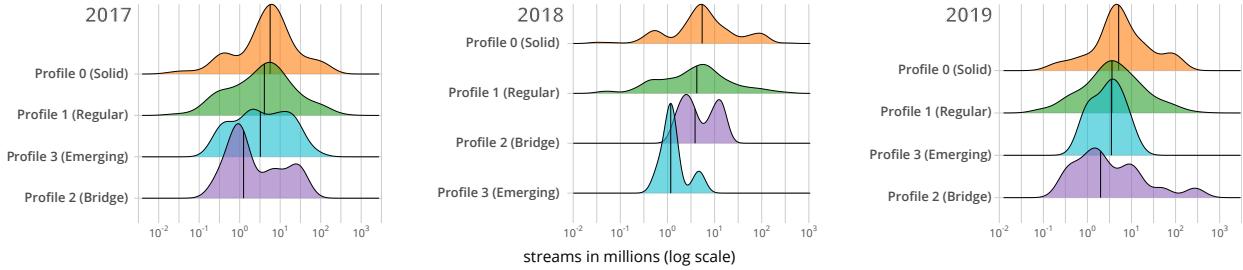
one of the 27 collaboration networks. As an example including all clusters, Figure 4 shows the result of the US network in 2019, where Cluster 0 groups the outliers identified by DBSCAN (data points in low-density regions, i.e., not associated with any proper cluster). Clusters 1 and 2 are slightly overlapping, but each covers groups of high-density data points, which is successful information in this analysis. We can also certainly conclude Cluster 3 is separate from the others. Next, we describe each cluster.

5.3 Collaboration Profiling

Now that we have detected a set of predominant clusters on all modeled networks, the next step is to look at their characteristics for profiling them and defining proper identities. First, for each network, we calculate the mean of the normalized metrics values grouped by each cluster id. Then, for each year, we plot radar charts for each profile with the mean values of each market present in that profile. Figure 5 shows such radar charts, where each cluster is represented by a polygon that exhibits its identity. To compare the metric values' magnitude of each cluster, we adopt the following scale: *low* is the bottom 30th percentile; *medium* is between 30th and 80th percentile; and *high* is the top 20th percentile. Such scale is based on the annual general values, i.e., considering the grouped normalized features of all markets by year.

The differences among the three plots represent minimal changes over the years. However, the distinct shapes show each cluster is *high* or *low* in certain features. Particularly, Cluster 0 presents collaborations with *high* values for *Attractiveness* and *Affinity* factors, but *medium* values for *Influence*. With a similar shape, Cluster 1 presents *medium* values for all four factors. On the other hand, Cluster 2 presents *high* values only for *Influence*, with *low Attractiveness* and *Affinity*. Finally, Cluster 3 is the group with major differences over the years: in general, its collaborations have *medium Attractiveness* and *Affinity*, and *low Influence*. Overall, each curve depicts a distinct collaboration profile, acting as a class descriptor of a cluster.

With the collaboration profiles settled, we can now answer $RQ3$. First, we analyze the distributions of success rate, and then the number of *intra-* and *inter*-genre collaborations for each profile. Here, we define success rate as the average of total streams of songs belonging to the music

Figure 6: Density ridgeline plots of streams in millions for each cluster. Darker vertical lines represent median values.

genres that compose the collaboration (edge) in that year. Figure 6 shows the success density ridgeline plots for each profile, indicating that Profiles 0 and 1 are composed of the most successful music genre collaborations, on average. With results from Table 3, in general, the most successful profiles are those composed of more inter-genre collaborations. Such a result may indicate a strong correlation between musical success and inter-genre collaborations. Indeed, by teaming up with one (or more) person of a different musical style in a song, both artists may draw from one another's fan bases; i.e., they may promote themselves to new public who could increase their fan base and audiences.

To summarize the characteristics of the collaboration profiles, we name each as follows.

– Profile 0 is *Solid Collaboration (Solid)*, composed of well-established collaborations between most popular genres (super-genres), which have been going on for decades. Examples include: *rap* and *hip-hop*, whose collaborative albums are hugely popular; and *hip-hop* and *pop*, whose separating line (between both genres) has become completely blurred in the last decade, mainly in the USA;

– Profile 1 is *Regular Collaboration (Regular)*, composed of the most common collaborations in all markets, which are very similar to solid collaborations but not as engaged. For instance, collaborations between *hip-hop/rap/pop* and *jazz/blues/soul*, which can be typical in many markets, but not as consolidated when compared to *Solid* ones;

– Profile 2 is *Bridge Collaboration (Bridge)*, composed of collaborations with high influence, representing bridge-like connectors between two regions of a network (mostly between divergent music styles). Such collaborations may be possible sources of investment to increase connectivity and strengthen ties among different audiences. One example is collaborations between *gospel* and others, such as *rap* and *MPB* (*Brazilian Popular Music*); and

– Profile 3 is *Emerging Collaboration (Emerging)*, formed mainly of collaborations between regional genres. Such partnerships generally occur within the same genre; possibly between one (or more) unknown artist and one (or more) established artist; or maybe in order to easily reach that genre audience. We propose the term *emerging* because such a profile can be seen as a transition phase for beginners, until they establish their fan bases. Examples of regional genres here include *k-pop* (popular music from South Korea), *moombahton* (fusion genre of *house* music and *reggaeton* (from Washington, D.C.), and *forró* (a popular musical genre from Brazilian Northeastern Region).

Table 3: Total number of *intra-* and *inter-*genre collaborations in each profile, from 2017 to 2019.

Collab	Solid			Regular		
	2017	2018	2019	2017	2018	2019
Inter-genre	140 (49%)	125 (42%)	103 (51%)	1,828 (99%)	1,916 (98%)	2,165 (94%)
Intra-genre	145 (51%)	174 (58%)	99 (49%)	23 (1%)	34 (2%)	128 (6%)

Collab	Bridge			Emerging		
	2017	2018	2019	2017	2018	2019
Inter-genre	10 (100%)	7 (100%)	16 (100%)	3 (7%)	1 (17%)	0 (0%)
Intra-genre	0 (0%)	0 (0%)	0 (0%)	40 (93%)	5 (83%)	7 (100%)

6. CONCLUSIONS

In this paper, we analyze and identify collaboration profiles in success-based music genre networks. Our results suggest that analyzing regional markets individually is fundamental, as local genres play a key role in determining hit songs and popular artists. Besides the differences in the evolution of regional markets, genre collaborations are also increasing, with emerging local genres achieving global success. Moreover, the networks' structures reveal three main factors that describe a genre collaboration: *Attractiveness*, *Affinity* and *Influence*. Analyzing such factors uncovers four different collaboration profiles: *Solid*, *Regular*, *Bridge* and *Emerging*, which act as class descriptors of successful partnerships. Overall, our results contribute to the understanding of the relation between cross-genre collaboration and hit songs.

Indeed, detecting genre collaboration profiles is a powerful way to assess musical success by describing similar behaviors within collaborative songs from multiple angles. Our findings may act as base material for further research tasks, e.g., prediction and recommendation. The former enables predicting the success of a given song/artist/album, while the latter can be used to point out potentially successful genre/artist collaborations. This not only benefits the MIR community, but also the music industry as a whole. In fact, music industry CEOs may maximize expected success by properly investing in potential artist/genre collaborations. Finally, artists may also profit by identifying the most suitable partnerships to lead the album to early stardom. In conclusion, this work sheds light on the science behind the collaboration phenomenon, providing potential impact to the music industry.

Future Work. We plan to consider other data sources and to expand the time period in order to better understand the markets' behavior, enhancing further analyses.

7. REFERENCES

[1] M. O. Silva, L. M. Rocha, and M. M. Moro, "Collaboration Profiles and Their Impact on Musical Success," in *Procs. of ACM/SIGAPP SAC*, Limassol, Cyprus, 2019, pp. 2070–2077.

[2] N. J. Bryan and G. Wang, "Musical influence network analysis and rank of sample-based music," in *ISMIR*, Miami, USA, 2011, pp. 329–334.

[3] C. Baccigalupo *et al.*, "Uncovering affinity of artists to multiple genres from social behaviour data," in *ISMIR*, Philadelphia, USA, 2008, pp. 275–280.

[4] T. Arjannikov and J. Z. Zhang, "An association-based approach to genre classification in music," in *ISMIR*, Taipei, Taiwan, 2014, pp. 95–100.

[5] I. Vatolkin, G. Rudolph, and C. Weihs, "Evaluation of album effect for feature selection in music genre recognition," in *ISMIR*, Malaga, Spain, 2015, pp. 169–175.

[6] S. Oramas *et al.*, "Multi-label music genre classification from audio, text and images using deep features," in *ISMIR*, Suzhou, China, 2017, pp. 23–30.

[7] A. Tsaptsinos, "Lyrics-based music genre classification using a hierarchical attention network," in *ISMIR*, Suzhou, China, 2017, pp. 694–701.

[8] S. S. Ghosal and I. Sarkar, "Novel approach to music genre classification using clustering augmented learning method (CALM)," in *AAAI MAKE*, ser. CEUR Workshop Proceedings, vol. 2600, 2020.

[9] C. Laroche *et al.*, "Genre specific dictionaries for harmonic/percussive source separation," in *ISMIR*, NYC, USA, 2016, pp. 407–413.

[10] M. Prockup *et al.*, "Modeling genre with the music genome project: Comparing human-labeled attributes and audio features," in *ISMIR*, Malaga, Spain, 2015, pp. 31–37.

[11] J. Bansal and M. Woolhouse, "Predictive power of personality on music-genre exclusivity," in *ISMIR*, Malaga, Spain, 2015, pp. 652–658.

[12] R. Hennequin, J. Royo-Letelier, and M. Moussallam, "Audio based disambiguation of music genre tags," in *ISMIR*, Paris, France, 2018, pp. 645–652.

[13] E. V. Epure, A. Khlif, and R. Hennequin, "Leveraging knowledge bases and parallel annotations for music genre translation," in *ISMIR*, Delft, the Netherlands, 2019, pp. 839–846.

[14] D. Bogdanov *et al.*, "The acousticbrainz genre dataset: Multi-source, multi-level, multi-label, and large-scale," in *ISMIR*, NYC, USA, 2019, pp. 360–367.

[15] H. Schreiber, "Genre ontology learning: Comparing curated with crowd-sourced ontologies," in *ISMIR*, NYC, USA, 2016, pp. 400–406.

[16] D. C. Corrêa, A. L. M. Levada, and L. da F. Costa, "Finding community structure in music genres networks," in *ISMIR*, Miami, UAS, 2011, pp. 447–452.

[17] L. Yang *et al.*, "Revisiting the problem of audio-based hit song prediction using convolutional neural networks," in *ICASSP*. IEEE, 2017, pp. 621–625.

[18] C. V. Araujo *et al.*, "Predicting music success based on users' comments on online social networks," in *WebMedia*, Brazil, 2017, pp. 149–156.

[19] F. Calefato *et al.*, "Collaboration success factors in an online music community," in *ACM GROUP*, Sanibel Island, USA, 2018.

[20] A. Cosimato *et al.*, "The conundrum of success in music: Playing it or talking about it?" *IEEE Access*, vol. 7, pp. 123 289–123 298, 2019.

[21] F. Abel *et al.*, "Analyzing the blogosphere for predicting the success of music and movie products," in *ASONAM*, Odense, Denmark, 2010, pp. 276–280.

[22] E. Zangerle *et al.*, "Hit song prediction: Leveraging low- and high-level audio features," in *ISMIR*, Delft, the Netherlands, 2019, pp. 319–326.

[23] M. E. J. Newman, *Networks: An Introduction*. Oxford University Press, 2010.

[24] D. Liben-Nowell and J. M. Kleinberg, "The link-prediction problem for social networks," *J. Assoc. Inf. Sci. Technol.*, vol. 58, no. 7, pp. 1019–1031, 2007.

[25] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[26] A. B. Costello and J. Osborne, "Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis," *Practical assessment, research, and evaluation*, vol. 10, no. 1, p. 7, 2005.

[27] L. G. Humphreys and R. G. Montanelli Jr, "An investigation of the parallel analysis criterion for determining the number of common factors," *Multivariate Behavioral Research*, vol. 10, no. 2, pp. 193–205, 1975.

[28] R. B. Cattell, "The scree test for the number of factors," *Multivariate behavioral research*, vol. 1, no. 2, pp. 245–276, 1966.

[29] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Procs. of KDD*, Portland, USA, 1996, pp. 226–231.

DEEP LEARNING BASED SOURCE SEPARATION APPLIED TO CHOIR ENSEMBLES

Darius Petermann[1]   **Pritish Chandna**[1]   **Helena Cuesta**[1]

Jordi Bonada[1]   **Emilia Gómez**[2,1]

[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona

[2] European Commission, Joint Research Centre, Seville

dariusarthur.petermann01@estudiant.upf.edu,

{pritish.chandna, helena.cuesta, jordi.bonada, emilia.gomez}@upf.edu

ABSTRACT

Choral singing is a widely practiced form of ensemble singing wherein a group of people sing simultaneously in polyphonic harmony. The most commonly practiced setting for choir ensembles consists of four parts; Soprano, Alto, Tenor and Bass (SATB), each with its own range of fundamental frequencies (F0s). The task of source separation for this choral setting entails separating the SATB mixture into the constituent parts. Source separation for musical mixtures is well studied and many deep learning based methodologies have been proposed for the same. However, most of the research has been focused on a typical case which consists in separating vocal, percussion and bass sources from a mixture, each of which has a distinct spectral structure. In contrast, the simultaneous and harmonic nature of ensemble singing leads to high structural similarity and overlap between the spectral components of the sources in a choral mixture, making source separation for choirs a harder task than the typical case. This, along with the lack of an appropriate consolidated dataset has led to a dearth of research in the field so far. In this paper we first assess how well some of the recently developed methodologies for musical source separation perform for the case of SATB choirs. We then propose a novel domain-specific adaptation for conditioning the recently proposed U-Net architecture for musical source separation using the fundamental frequency contour of each of the singing groups and demonstrate that our proposed approach surpasses results from domain-agnostic architectures.

1. INTRODUCTION

Choir music is a well-established and long-standing practice involving a body of singers performing together. Such ensembles are usually referred to as choir and may perform with or without instrumental accompaniment. A choir ensemble is usually structured by grouping the voices into four different sections, each depicting different frequency ranges for the singers; "Soprano" (260 Hz-880 Hz), "Alto" (190 Hz-660 Hz), "Tenor" (145 Hz-440 Hz), and "Bass" (90 Hz-290 Hz) [1]. This type of structural setting is usually referred to as a SATB setting. Although different variants of this structure exist, the SATB is the most well documented, with several conservatories across Europe dedicated to the study and practice of the art form, highlighting its cultural significance. This will be the main focal point of our study.

The segregation of a mixture signal into its components is a well researched branch of signal processing, known as source separation. For polyphonic music recordings, this implies the isolation of the various instruments mixed together to form the whole. With applications such as music remixing, rearrangement, audio restoration, and full source extraction, its potential use in music is of great appeal. While the task remains similar independently of the type of setting involved, the nature of the sources (e.g.: speech, musical instrument, singing voice) and their relations may entail various challenges and, consequently, require different separation methodologies to be employed.

The most studied case of musical source separation focuses on pop/rock songs, which typically have three common sources; vocals, drums, bass along with other instrumental sources which are usually grouped together as others. A large body of research [2–4] has been published in this field over the last few years, beginning with the consolidation of a common dataset for researchers to train and evaluate their models on. In 2016, *DSD100* [5] was first introduced and made available to the public and was later extended to *MUSDB18* [6], which comprises 150 full-length music tracks for a total of approximately 10 hours of music. To this day, *MUSDB18* represents the largest freely available dataset of its kind.

While source separation for the pop/rock case has come leaps and bounds in the last few years, it remains largely unexplored for the SATB choir case, despite its cultural importance. This is partly due to the lack of a consolidated dataset, similar to the *MUSDB18*, and partly due to the nature of the task itself. The sources to be separated in pop/rock have distinct spectral structure; the voice is a harmonic instrument and has a distinct spectral shape, defined by a fundamental frequency and its harmonic partials

and formants. The bass element to be separated also has a harmonic structure, but lacks the formants found in the human voice and has a much lower fundamental frequency than the human voice. In contrast, the spectrum of a percussive instrument is generally inharmnoic and energy is usually spread across the spectrum. In contrast, the sources to be separated in a SATB choir all have a similar spectral structure with a fundamental frequency, partials and formants. This makes the task more challenging than its more studied counter part. However, the distinct ranges of fundamental frequencies in the sources to be separated can be used to distinguish between them, a key aspect that we aim to explore in our study.

We build on top of some recently proposed Deep Neural Network (DNN) models to separate SATB monoaural recordings into each of their respective singing groups and then propose a specific adaptation to one of the models. The rest of the paper is organized as follow: Section 2 presents and investigates some of the recently proposed high performance deep learning based algorithms used for common musical source separation tasks, such as the U-Net [7] architecture and its waveform-based adaptation, Wave-U-Net [8]. Section 3 goes over the dataset curation carried out for this experiment. Section 4 presents our adaptation of the conditioned U-Net model described in [9], with a control mechanism conditioned on the input sources' fundamental frequency (F0). Section 5 defines the evaluation metrics and methodology used in this experiment. In Section 5.2 we evaluate and compare how existing models and our proposed adaptation perform on the task of source separation for SATB recordings. We then present and discuss the results. Section 6 finally concludes with a discussion around our experiment and provide comments on future research that we intend to carry out.

2. RELATED WORK

While source separation has remained relatively unexplored for the case of SATB choirs, a number of architectures have been proposed over the last few years for musical source separation in the pop/rock case. A comprehensive overview of all proposed models is beyond the scope of this study, but we provide a summary of some of the most pertinent models that we believe can easily be adapted to the case in study.

2.1 U-Net

The U-Net architecture [7], which was specifically developed to process and segment biomedical images, inspired many subsequent audio-related adaptations due to its unprecedented performance.

The original model includes an encoding path, which reduces the initial input into a latent representation (bottleneck) followed by a decoding path, which expends the channels' receptive field back into its original shape while concatenating the feature maps from the contracting path by the mean of skip connection layers.

One of the first paper to present a U-Net adaptation towards audio source separation was proposed by Jansson et al. [10], where they propose an architecture which specifically targets vocal separation performed on western commercial music (or pop music). The authors present an architecture directly derived from the original U-Net one, which takes spectrogram representations of the sources as input and aims at predicting a soft-mask for the targeted source (either vocal or instrumental). The predicted mask is then multiplied element-wise with the original mixture spectrogram in order to obtain the predicted isolated source. It is worth mentioning that for each of the given sources, a U-Net instance is trained in order to predict its respective mask. In the case of SATB mixtures, four U-Net instances are necessary in order to predict each of the four singing groups.

2.2 Conditioned-U-Net
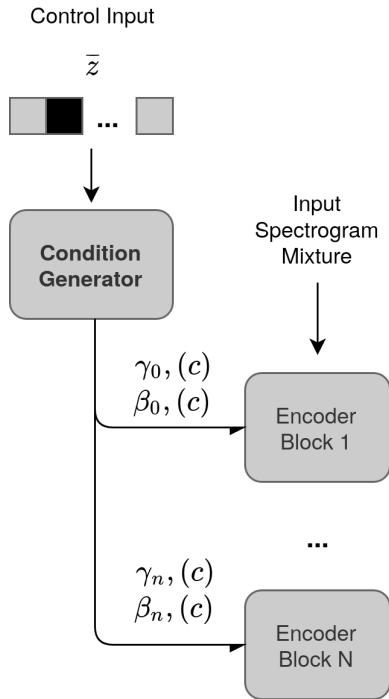
Depending on the nature of the separation task, its underlying process can easily lead to scaling issues. The conditioned U-Net (C-U-Net) architecture, described in [9], aims at addressing this limitations by introducing a mechanism controlled by external data which govern a single U-Net instance. C-U-Net does not diverge much from the initial U-Net one; as an alternative to the multiple instances of the model, each of which is specialized in isolating a specific source, C-U-Net proposes the insertion of feature-wise linear modulation (*FiLM*) layers [11], which represents an affine transform defined by two scalars - γ and β, across the architecture. This allows for the application of linear transformations to intermediate feature maps. These specialized layers conserve the shape of the original intermediate feature input while modifying the underlying mapping of the filters themselves.

$$FiLM(x) = \gamma(z)x + \dot\beta(z) \qquad (1)$$

In eq. (1), x is the input of the *FiLM* layer, γ and β the parameters that scale and shift x based on an external information, $\overline{z}$ [9]. γ and β modulates the feature maps according to an input vector $\overline{z}$, which describes the source to separate. The condition generator block described in Figure 1 represents a neural network embedding the one-hot encoding input $\overline{z}$ into the most optimal values to be used by the *FiLM* layer.

2.3 Wave-U-Net

In [8], the authors present a time-domain adaptation of the U-Net architecture, which performs the separation operation on the waveform. As the input is a one-dimensional signal, the feature maps are computed directly from the waveform samples through 1D convolution operations. Because Wave-U-Net takes raw waveforms as input, the initial U-Net model has to be adapted accordingly in order to accommodate for the input's nature. Consequently, the feature maps along both contracting and expanding paths are computed by the means of single-dimensional convolution layers. Both paths contain twelve convolutional layers, each, for a total of 24 layers. In the down-sampling

Control Input

Figure 1: C-U-Net control mechanism with the vector $\bar{z}$, a one-hot representation of the source to separate, which dictates the N sets of γ and β values to be used by the *FiLM* layer at each of the block of the encoding path, 1 to N.

path, the receptive field is reduced in half after each layer while the input feature maps are increased by a factor of 24 every time. On the other hand, in the up-sampling path the time-context is doubled after every convolutional layer while the feature maps are reduced, again by 24, after every layers. By that mean, the receptive field and the number of channels of the original input signal will remain preserved at the output stage. Although Wave-U-Net has proven to deliver satisfying results on common musical source separation tasks, the fact remains that waveform-based architectures in general require more data than their spectrogram-based counter parts.

3. DATASET

The training data we have curated for this experiment are composed of the following two datasets:

- Choral Singing Dataset [12] (CSD). Three songs performed by 16 singers from the Anton Bruckner Choir (SATB) [1].

- A proprietary dataset with 26 Spanish SATB songs by 4 singers, one for each part.

There are very few publicly available choir music datasets, thus our choice remains limited. To this day and to the best of our knowledge, there isn't any existing dataset which is specifically suited to our task, thus one

---

[1] https://zenodo.org/record/1286570#.XyGcHy-z3yU

---

of the subsidiary work of this experiment revolves around curating a proper and complete dataset to train our various models. For our experiment, we take advantage of the CSD [12]. This dataset was recorded in a professional studio and contains individual tracks for each of the 16 singers of a SATB choir, i.e. 4 singers per choir section. It comprises three different choral pieces: *Locus Iste*, written by Anton Bruckner, *Nĩno Dios d'Amor Herido*, written by Francisco Guerrero, and *El Rossinyol*, a Catalan popular song; all of them were written for 4-part mixed choir. The dataset is very well suited for our experiment as the isolated track for each individual singer will allow us to proceed the same way as in [13], that is to create artificial mixes by combining various stems from different groups together. Using different combinations of all 16 singers, we created 256 SATB quartets for each piece, which represent all possible combinations of singers taking into account the voice type restriction (i.e. exactly one singer per voice is needed).

The second dataset we use is a proprietary one including 26 songs for exactly one singer per part (i.e. 4 stems per song), which is a well-suited format for our task as well. All songs offered as part of this dataset are performed in Spanish and their length revolves around two to three minutes, for a total of 58 minutes of audio data.

Our curation work consists in consolidating these two datasets and make sure all the data that we are using remain consistent and well-formatted across all the audio stems, which includes length and amplitude normalization, as well as properties standardization. Most of the files included as part of the initial datasets were presented as 10-seconds long snippets as opposed to full-length songs, which isn't an ideal format to work with. Hence, some additional efforts have been devoted to turn these files in a more convenient and consolidated format.

4. APPROACH

Injecting domain knowledge in DNNs has been proved to be an effective way to learn complex input-output relations with high accuracy when the available data happen to be scarce and limited [14], such as found in our case.
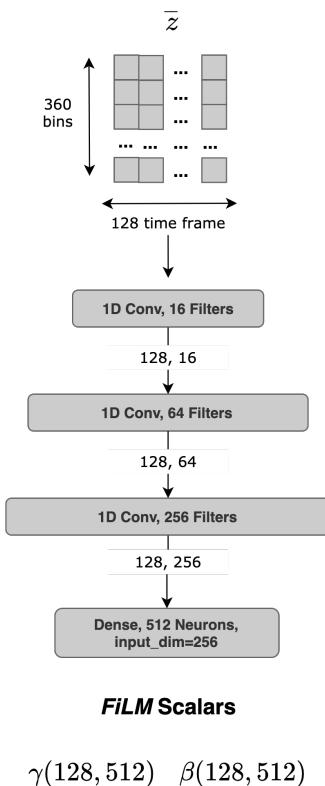
Each of the singing groups in a SATB recordings performs within its own respective frequency range, that is, the voices' F0 contour will rarely overlap across the various groups. This factor makes the sources' F0 a suitable discriminative feature, which could be injected in the DNN during the training stage and potentially improve the separation of the various singing groups in SATB recordings.

In this view, we propose to adapt the original C-U-Net architecture, which initially embeds the instruments to be separated, $\bar{z}$, in order to produce the various *FiLM* parameters (γ, β), and substitute the external control input data for the F0 track of the target source. The new control input vector $\bar{z}$ will thus hold time as well as frequency dimensions.

4.1 Control Input Representation
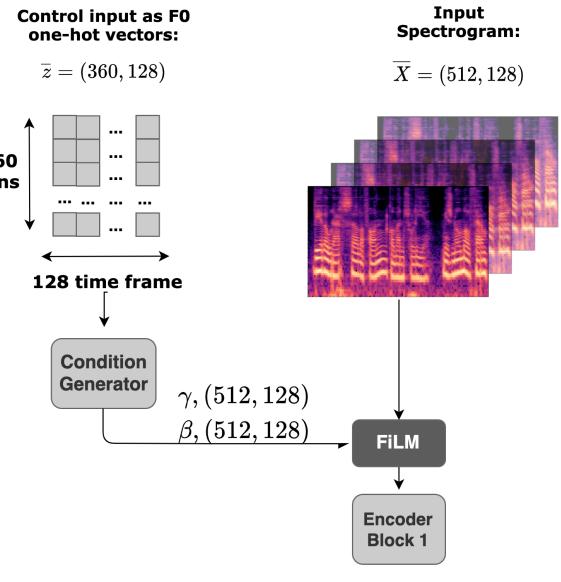
As previously mentioned, we use the frame-wise F0 as the external control data of our condition generator network. This entails that a few preliminary steps are required prior to proceeding to the training stage. We first automatically extract the sources' F0 track using the DIO algorithm [15]. Once the raw pitch tracks are obtained, we convert each time-step F0 into a one-hot encoded representation as postulated in [16], that is 60 frequency bins over 6 musical octaves, with a base frequency of 32.7 Hz Hertz, for a total of 360 frequency bins per time-step. As a result, for 128 time-steps, our control input will be in the shape of [128, 360].

Figure 2: Control model architecture. The convolution is performed across the frequency bins for each time-step. The dense layer provides a specific conditioning for each frequency bin.

4.2 Control Model

The control model used in our proposed architecture embeds the one-hot encoded CQT F0 representation for a given time-step into a set of transforms of identical shape as the spectrogram input. This is achieved by modeling the condition vectors as 1-D data with multiple feature channels. The condition vectors are then fed into a convolutional neural network (CNN) with a kernel of size 10 seizing contextual information from the adjacent time-steps. As a result, all input channels of the initial convolution contribute to all resulting feature maps in the output of the first convolutional layer. Finally a dense layer provides a

Figure 3: C-U-Net Control Mechanism adapted to our task, with the one-hot vector $\overline{z}$ depicting the various SATB singing groups' F0 contour.

specific conditioning for each frequency bin at each time-step, taking into account the contextual information previously captured by the CNN. Figure 2 shows the condition generator architecture in greater details.

As the temporal relation between the external control input data and the input spectrogram is crucial, it is important to apply these affine transformations while the receptive field of the input is still intact. Hence the *FiLM* layer is applied prior to the encoding path. Figure 3 shows the overall structure behind the proposed conditioning architecture.

We propose two variants of the architecture described above, each of which differs slightly in the way it embeds the control input data; the first variant applies a unique affine transform for each individual frequency bin at every input time-step. The resulting scalars in the output of the external CNN model will thus be in the shape [512, 128] for a given input spectrogram of the same time context. On the other hand, the second variant applies a single transform for all frequency bins at a given time-step, resulting in a set of scalars of shape [1, 128] for 128 spectrogram frames given in the input. We refer to the two approaches as "Domain-Specific Global" and "Domain-Specific Local", respectively and define them with the acronyms *C-U-Net D-S G* and *C-U-Net D-S L* in the rest of this paper. while the three models covered in Section 2 will be referred to as "Domain-Agnostic" models.

5. EVALUATION

To assess our proposed approach and show that injecting domain knowledge as control input data to the network improves its performance on SATB recordings, we evaluate the performances of three domain-agnostic state-of-the-art DNN models; U-Net, its waveform adaptation

Wave-U-Net, and the original C-U-Net. We then compare the results with the two domain-specific models proposed in Subsection 4.2. We evaluate the performance of a model by computing three metrics, SDR, SIR, and SAR [5], between the predicted and true audio sources. The three measurement metrics describe the overall quality of the separation, the level of interference with other sources as well as the amount of artifact added by the separation algorithm, respectively. The metrics are computed using the *mir_eval toolbox* [17] for each of the SATB singing groups.

5.1 Train - Test Split

Given the limited size of our data, we opted to set apart one song from the proprietary dataset as well as one singer per voice for each song from the CSD in order to build our first use-case test set. The rest of the data was used for training. This allowed us evaluate the model on unseen songs and singers. Our second test case contains unison singing, which was not seen at all during training. As such, we used the three songs from the CSD with all singers for evaluation.

5.2 Experiment Results

Subsections 5.2.1 and 5.2.2 present the performance results for the two test cases described earlier; that is, for the test set involving exactly one singer per part and the other involving exactly four singers per part, respectively. *C-U-Net D-A* refers to the domain-agnostic architecture while *C-U-Net D-S L* and *C-U-Net D-S G* refer to our two domain-specific adaptations. For testing, we use the oracle fundamental frequency of each of the sources, precomputed prior to model inference. In a complete source separation pipeline, we would complement our system with the multi-pitch algorithm proposed by [18].

5.2.1 Use-Case 1:

Table 1 portrays the mean SIR and SAR results on all the SATB parts and average for all five models mentioned in previous sections. Figure 4a details the SDR score distributions on our first test set. We observe that our two adaptations, *C-U-Net D-S L* and *C-U-Net D-S G*, call attention to a significant score gap between domain-agnostic and domain-specific models, with an average increase of about 1dB SDR and 1.5dB SIR between the two different approaches. These improvements underline an overall better quality of the predicted sources (SDR) as well as a decline in interference between the various predictions. Our domain-specific architecture hence demonstrates a better ability to cope with the correlated nature of the various SATB sources and seem to predict an appropriate spectral mask for each of them. We also observe that our proposed adaptations return the lowest SDR, SIR and SAR performances for the Bass part, specifically. This could be due to the fact that the Bass group, among all SATB groups, shares the highest number of harmonics with its other source counter parts.

We also note that the mean SDR substantially drops for the Tenor singing group across nearly all domain-agnostic

models, reaching a negative result with the *C-U-Net D-A* architecture (−1.25dB SDR). We speculate that the reason behind such decline can be directly related to the close nature of the F0 contours of both Alto and Tenor singing groups, making it harder for domain-agnostic architecture to distinguish between the two sources. This limitation brings yet another justification for the conditioning approach we have taken in this paper.

5.2.2 Use-Case 2:

Figure 4b as well as the bottom portion of Table 1 presents the SDR, SIR and SAR scores on the second use-case test set. We observe that the introduction of more complex mixtures involving a higher number of singers (i.e.: 16 singers in this case) decreases the performance of our proposed models, with an average SDR barely surpassing U-Net's for our *C-U-Net D-S G* model and levelling it out for the *C-U-Net D-S L* model. This can be attributed to the use of the mean of the various pitches present in a singing group, to represent the pitch of the unison. Since domain-agnostic models, such as the plain U-Net, don't hold this assumption, these architectures are less prone to errors when exposed to these type of mixture settings [2].
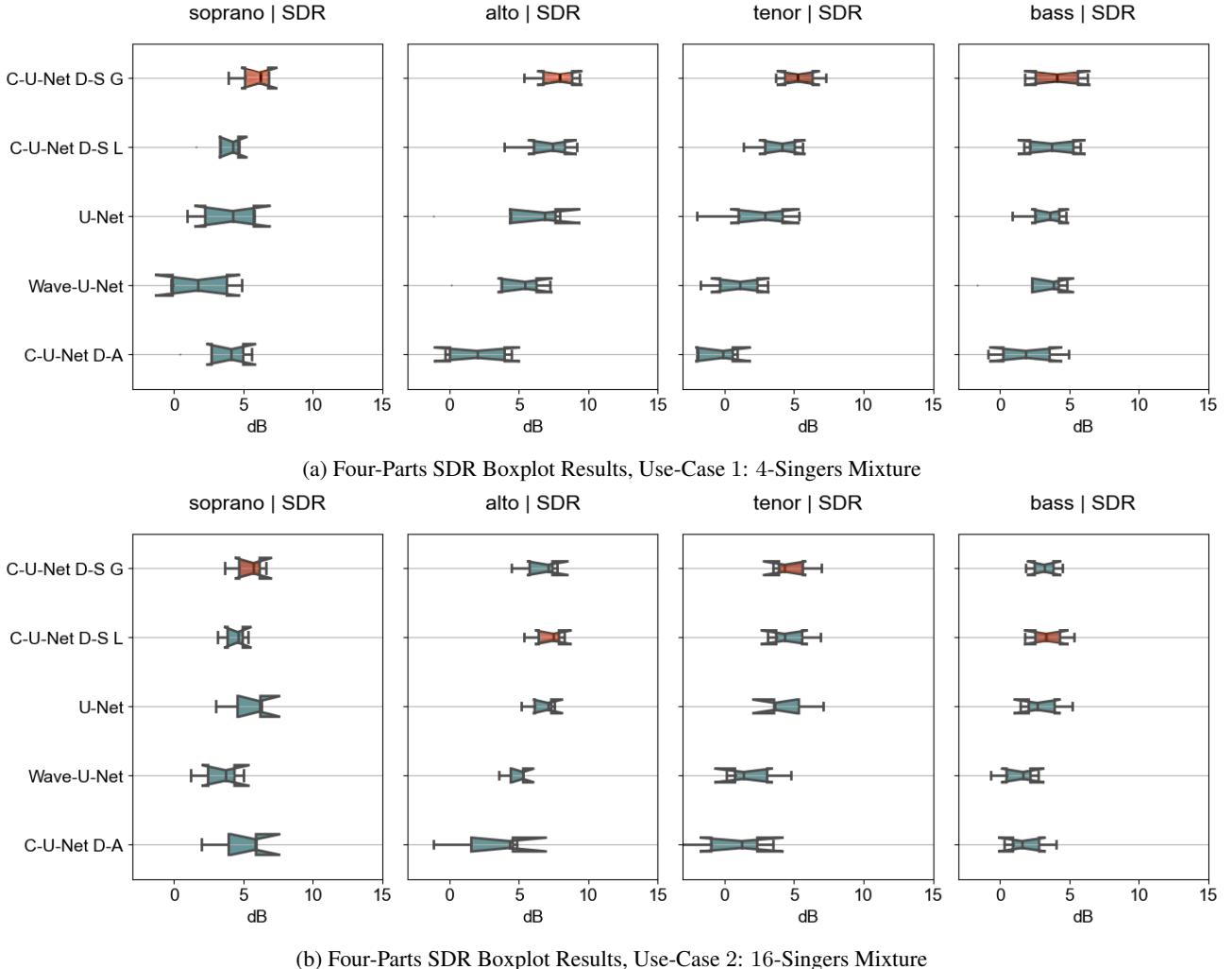
6. CONCLUSIONS AND FUTURE WORK

In this work we have presented the task of musical source separation applied to SATB choir recordings. We first described the consolidated dataset that we've specifically curated for this experiment and its potential use and application for future related research. We then assessed how well recent domain-agnostic deep learning based architectures for musical source separation performed on this task, given two different use-cases; 4-singers mixture and 16-singers mixture separation. An adaptation of the U-Net architecture was then proposed, consisting in conditioning some of the network parameters on the fundamental frequency contour of each of the SATB mixture sources. The preliminary results showed that taking advantage of domain-knowledge during the training process improved the performance on both of our proposed use-cases. For the evaluation presented in this paper, we use the oracle F0 is currently used as external control input data to the network. In a complete source separation pipeline, we plan on combining the task of multi-pitch tracking [18] with the system presented in this paper. We also plan on validating our evaluation with perceptual listening tests and exploring applications of the SATB separation. including remixing, transcription and transposition combined with the work presented in [19, 20].

7. ACKNOWLEDGEMENTS

---

[2] Audio examples showcasing how the inference of our proposed models compare against other state-of-the-art architectures are available online. On the same page is also included a link to our pre-trained models: `https://darius522.github.io/satb-source-separation-results/`

(a) Four-Parts SDR Boxplot Results, Use-Case 1: 4-Singers Mixture

(b) Four-Parts SDR Boxplot Results, Use-Case 2: 16-Singers Mixture

Figure 4: Boxplot SDR results on the five U-Net based models described in previous sections. Subfigure 4a shows the result distribution over the first use-case test set while 4b depicts the results for the second use-case. For each one of the SATB parts, the model performing with the highest median is indicated in a dark orange color.

Model	Test Use-Case 1 - SIR (dB)					Test Use-Case 1 - SAR (dB)				
	Soprano	Alto	Tenor	Bass	Avg.	Soprano	Alto	Tenor	Bass	Avg.
Wave-U-Net	5.99±2.4	9.19±2.9	4.62±2.1	8.49±3.5	7.07	5.36±1.7	7.11±2.4	4.79±1.4	4.89±1.4	5.54
U-Net	10.28±2.4	10.77±4.1	6.70±3.2	9.45±2.0	9.30	5.35±1.8	7.13±3.0	5.32±1.8	4.94±1.1	5.69
C-U-Net D-A	10.09±2.6	7.81±1.6	3.32±3.3	7.61±2.2	7.21	5.19±1.6	4.41±2.8	2.65±1.2	4.12±2.0	4.09
C-U-Net D-S L	9.71±1.7	12.37±1.5	9.89±2.2	9.71±1.7	10.42	5.44±1.0	8.75±2.0	5.58±1.3	5.51±1.7	6.32
C-U-Net D-S G	**12.72**±1.8	**14.04**±1.5	**11.79**±1.5	**9.78**±2.1	**12.08**	**7.02**±1.1	**9.02**±1.6	**6.86**±1.5	**5.93**±1.6	**7.21**

	Test Use-Case 2 - SIR (dB)					Test Use-Case 2 - SAR (dB)				
	Soprano	Alto	Tenor	Bass	Avg.	Soprano	Alto	Tenor	Bass	Avg.
Wave-U-Net	8.13±2.1	10.02±0.9	6.80±2.2	7.45±2.0	8.10	5.75±1.0	6.73±1.1	4.79±1.5	3.23±0.9	5.13
U-Net	**12.41**±1.8	13.11±1.2	10.26±1.1	8.50±2.3	11.07	6.31±1.4	7.97±1.0	**6.59**±1.8	5.27±1.1	6.54
C-U-Net D-A	11.99±1.8	9.08±2.8	5.65±3.1	7.60±2.0	8.58	5.77±1.7	4.60±2.9	3.39±1.8	4.15±1.2	4.48
C-U-Net D-S L	10.32±1.1	13.06±1.7	10.77±1.5	8.89±2.2	10.76	6.02±0.9	**8.59**±1.2	6.45±1.7	**5.59**±1.1	**6.66**
C-U-Net D-S G	12.08±1.5	**13.50**±2.5	**12.05**±1.3	**8.91**±2.1	**11.63**	**6.68**±1.2	7.70±1.3	6.17±1.6	5.17±0.8	6.43

Table 1: SIR and SAR mean and standard deviation results on the four SATB parts as well as their average for the five U-Net based models described in previous sections. The top table depicts the results obtained from the first use-case test set while the bottom one the second use-case test set.

8. REFERENCES

[1] M. Scirea and J. A. Brown, "Evolving four part harmony using a multiple worlds model," in *Proceedings of the 7th International Joint Conference on Computational Intelligence (IJCCI)*, vol. 1. IEEE, 2015, pp. 220–227.

[2] P. Chandna, M. Miron, J. Janer, and E. Gómez", "Monoaural Audio Source Separation Using Deep Convolutional Neural Networks," in *International Conference on Latent Variable Analysis and Signal Separation*, 2017, pp. 258–266.

[3] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix - a reference implementation for music source separation," *Journal of Open Source Software*, 2019. [Online]. Available: https://doi.org/10.21105/joss.01667

[4] F. Lluís, J. Pons, and X. Serra, "End-to-End Music Source Separation: Is it Possible in the Waveform Domain?" in *Proceedings of Interspeech 2019*, September 2019. [Online]. Available: http://dx.doi.org/10.21437/interspeech.2019-1177

[5] A. Liutkus, F.-R. Stöter, Z. Rafii, D. Kitamura, B. Rivet, N. Ito, N. Ono, and J. Fontecave, "The 2016 signal separation evaluation campaign," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2017, pp. 323–332.

[6] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," December 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical image computing and Computer-assisted Intervention*. Springer, 2015, pp. 234–241.

[8] D. Stoller, S. Ewert, and S. Dixon, "Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[9] G. Meseguer-Brocal and G. Peeters, "Conditioned u-net: Introducing a control mechanism in the u-net for multiple source separations." November 2019.

[10] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing Voice Separation with Deep U-Net Convolutional Networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[11] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "FiLM: Visual Reasoning with a General Conditioning Layer," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, April 2018.

[12] H. Cuesta, E. Gómez, A. Martorell, and F. Loáiciga, "Analysis of Intonation in Unison Choir Singing," in *Proceedings of the International Conference of Music Perception and Cognition (ICMPC)*, Graz (Austria), July 2018, pp. 125–130.

[13] H. Cuesta, E. Gómez, and P. Chandna, "A Framework for Multi-f0 Modeling in SATB Choir Recordings," in *Proceedings of the Sound and Music Computing (SMC) Conference*, Málaga (Spain), April 2019.

[14] M. Silvestri, M. Lombardi, and M. Milano, "Injecting domain knowledge in neural networks: a controlled experiment on a constrained problem," *ArXiv*, vol. abs/2002.10742, 2020.

[15] M. Morise, H. Kawahara, and H. Katayose, "Fast and reliable f0 estimation method based on the period extraction of vocal fold vibration of singing voice and speech," in *AES 35th International Conference on Audio for Games*, February 2009. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=15165

[16] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep Salience Representations for F0 Estimation in Polyphonic Music," in *Proceedings of the International Society for Music Information Retrieval Conference, (ISMIR)*, 2017.

[17] C. Raffel, B. Mcfee, E. J. Humphrey, O. N. Justin Salamon, D. Liang, and D. P. W. Ellis, "mir_eval: a transparent implementation of common mir metrics," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[18] H. Cuesta, B. McFee, and E. Gómez, "Multiple F0 Estimation in Vocal Ensembles using Convolutional Neural Networks," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montreal, Canada (Virtual), 2020.

[19] P. Chandna, H. Cuesta, and E. Gómez, "A Deep Learning Based Analysis-Synthesis Framework For Unison Singing," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montreal, Canada (Virtual), 2020.

[20] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "Content based singing voice extraction from a musical mixture," in *Proceedings of the 45th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2020, pp. 781–785.

A COMBINATION OF LOCAL APPROACHES FOR HIERARCHICAL MUSIC GENRE CLASSIFICATION

Antonio R. S. Parmezan[1]   **Diego Furtado Silva**[2]   **Gustavo E. A. P. A. Batista**[3]

[1] Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo, São Carlos, Brazil

[2] Departamento de Computação – Universidade Federal de São Carlos, São Carlos, Brazil

[3] School of Computer Science and Engineering – University of New South Wales, Sydney, Australia

`parmezan@usp.br, diegofs@ufscar.br, g.batista@unsw.edu.au`

ABSTRACT

Labeling a music recording according to its genre is an intuitive and familiar way to describe its content. Music genres are valuable information especially for music organization, personalized listening experience, and playlist generation. Automatically classifying music genres is a challenging endeavor due to the inherent ambiguity and subjectivity. Most efforts on music genre classification consider the complete independence between labels. However, music genres typically respect a hierarchical structure based on the influences or origins of each style. Conversely, many of the methods available for hierarchical classification are based on assumptions about the class hierarchy, such as the need for multiple children in each hierarchy's node, which may limit their use in music applications. Also, the local classifier per node approach that would be the most suitable for this scenario is costly regarding time and memory. In this paper, we present two local hierarchical classification approaches and show how to combine them to obtain a single one that is more robust and faithful to the music genre classification scenario. We evaluate our proposal in a music dataset hierarchically labeled with 120 music genres. As shown, compared to state-of-the-art approaches, our approach has a lower computational cost and can achieve competitive performances.

1. INTRODUCTION

The music genre is a convention used by humans to categorize and organize pieces of music. Besides being essential metadata for large databases of music distribution, the music genre resides in one of the most common descriptors employed in studies involving storage, retrieval, and usage of music knowledge [1–3].

The major problem with music genre information is that it is usually fuzzy and inaccurate. At the core of this question is human subjectivity, closely related to the several criteria for labeling music in genres [4]. Because of subjectivity, the music genre classification task becomes even more challenging, since it needs to deal with differences in interpretation and an unavoidable intersection between genres.

Besides, as there is no standard for labeling music, some music platforms may present a high number of genres. For example, the streaming music service Spotify [1] catalogs its music in over 1500 genres, which will eventually be updated and increased in quantity in the future. Also, a song classified as, for instance, "gothic metal" on Spotify may be labeled as "alternative rock" and "indie rock" on other platforms, such as Google Play Music [2] and Apple Music [3]. Finally, some music genres overlap in these services, while others have subgenres. Although there is no unique way of determining an item's music genre, the literature covers distinct approaches and methods to support this complex and many-sided task.

So far, most of the work in music information retrieval is only concerned with music genres as a flat classification problem [5–8]. A flat classifier seeks to associate each example with a class that belongs to a finite, devoid of structural dependencies and usually small, set of classes. However, the music genre classification calls for a genre taxonomy, *i.e.*, a hierarchical set of categories to be mapped onto a music collection.

In hierarchical music genre classification, supervised machine learning algorithms are designed to induce a hierarchical decision model. Such a model links the features of the examples to a class hierarchy, generally represented as a tree or a direct acyclic graph with varying specificity and generality levels. An advantage in assigning examples to hierarchically organized classes is that the closer to the root of the hierarchy a linkage occurs, the smaller the classifier error rate tends to be. Conversely, the obtained classification will be less specific and, therefore, less informative [9, 10].

Having the genres structured into a class hierarchy helps users to not only browse and retrieve music pieces but also navigate the collection according to the similarity of its content.

Most of the methods available for hierarchical classification are based on the local classifier per node approach.

---

[1] `https://www.spotify.com.`
[2] `https://play.google.com/music.`
[3] `https://www.apple.com/music.`

This approach resides in training a binary classifier for each node of the class hierarchy, except the root node, aiming to predict whether or not an example has the corresponding class [11]. The local binary dataset for a given node in the class hierarchy contains positive examples, those that have some relation to the class of the node in question, and negative examples, those related to the remaining classes. We can determine the sets of positive and negative examples in different ways; for instance, adopting heuristics based on set operations [12] or strategies based on nearest neighbors [13]. Depending on the number and disposition of the classes in the hierarchy, we may choose to use multiclass classifiers instead of binary ones. Thus, while the local classifier per parent node approach builds a multiclass classifier for each non-leaf class node of the hierarchy, the local classifier per level approach generates a multiclass classifier for each hierarchy level [11].

The multiclass approaches are much more efficient concerning time and memory than the binary one because they build fewer classifiers. On the other hand, they make some assumptions about the class hierarchy, such as the need for multiple children in each hierarchy's node. Although the assumptions help to avoid both the generation of one-class local datasets and class-membership inconsistency, they may limit the direct use of multiclass classifier-based hierarchical approaches in music applications.

In this paper, we combine the per node and per parent node approaches to obtain a single local hierarchical method that is more robust and faithful to the music genre classification scenario. We advocate using decision tree-based classification algorithms to build the local classifiers because they internally perform feature selection. As discussed throughout this work, this is a relevant consideration in hierarchical music genre classification since the features that most distinguish among classes tend to be different at each level of the class hierarchy.

We evaluate the proposed approach using a music dataset with 120 hierarchically organized music genres. Also, we compare our proposal with three well-known approaches in terms of performance and runtime. The results show that our proposal is computationally inexpensive and competitive against the traditional approaches.

The remainder of this paper is organized as follows: Section 2 gives a general view of the problem and summarizes research efforts in automatic music genre classification; Section 3 introduces the concepts of hierarchical classification; Section 4 describes our proposal, from the design of the hierarchical decision model to the prediction strategy adopted; Section 5 presents the experimental protocol, as well as the results and our discussion about them; Section 6 concludes this study and shows directions for future work.

2. RELATED WORK

Recognition of music genres is one of the most prominent research problems in music information retrieval. Studies point out that genre is the most chosen concept to guide the user browsing in music repositories [1, 14].

A music genre recognition system aims to categorize an audio signal with an unknown label into a previously known music genre from relevant features extracted from this audio. A classifier makes use of such characteristics to identify the music genre of the analyzed signal. The benefits of categorizing pieces of music in genres extend to many other tasks, such as organizing digital audio databases [15], building new search engines [16], and recommending songs [17].

Several approaches for audio-based music genre classification extract features associated with the timbral information, such as Mel-Frequency Cepstral Coefficients (MFCC), and sets of spectral and rhythmic characteristics [18].

There are three main categories of algorithms that can be applied in this context [19]. The first one represents the entire recording with one single set of features [5, 20]. The other two rely on classifying feature vectors extracted from short frames of the recordings and achieve the final result by an ensemble of these classifications. Some methods perform this approach directly on the frame-level features [21], while some other aggregate few consecutive frames, creating a new set of features usually given by the mean and standard deviations of the aggregated features [6, 22, 23].

More recently, researchers have used deep learning models to learn a feature representation that promises to advance the task of genre recognition significantly [24–27]. Although deep learning methods are computationally expensive, they allow the extraction of relevant audio features without having to depend on ad-hoc domain-dependent signal processing strategies [24].

The literature has mostly treated the automatic music genre classification as a problem of flat classification. In other words, most papers in this domain consider all the genres in the same hierarchy's level [5–8]. However, the music genre classification problem is better modeled with a taxonomy of genres. We describe below some notable works that have used class hierarchies to support the task mentioned above.

A binary approach, which uses a feature selection method on each generated local dataset and a Gaussian Mixture Model as a base-level classifier, was proposed in [28]. In this same direction, the authors in [29] applied an ensemble of Feed Forward Neural Networks and k-Nearest Neighbors (kNN) over the local binary datasets. For the kNN classifier, they employed a genetic algorithm feature selection mechanism.

Two datasets with content-based features and a standard local approach using Support Vector Machines classifiers were explored in [30]. Differently, the authors in [31] developed a local approach that adopts feature selection, multiple representations from the same object, and enables hierarchically multi-label classifications by using a two-layer labeling process.

The study reported in [32] involved two selective multiclass hierarchical methods. The first one selects the best feature set instead of the best classifier, while the second

one selects both the best classifier and the best representation simultaneously.

The authors in [33] proposed a novel approach to building a classification tree through subspace cluster analysis. On the other hand, hierarchical analysis of spectrograms was investigated in [34] to help classify music in genres.
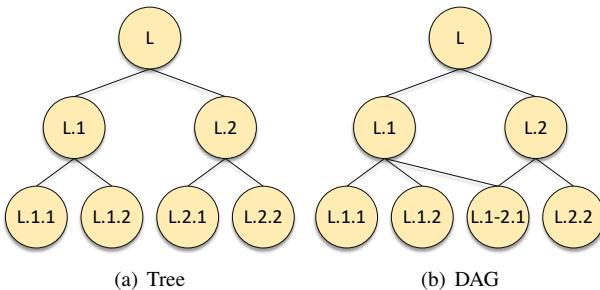
This paper presents a proposal that differs from the literature for having another point of view. Here we combine two local approaches to quickly and efficiently obtain a single hierarchical method that faithfully represents the music genre classification scenario.

3. HIERARCHICAL CLASSIFICATION

Flat classification differs from hierarchical one because in the latter the domain classes follow a logical organization. In flat classification, while the absence of interrelationships between classes characterizes some problems (single-label classification), the non-structural relationships between labels evidence others (multi-label classification). Structural dependencies, which express super or subclass relations, define hierarchical classification.

A dataset for hierarchical classification in the attribute-value table format comprises N pairs of examples $(\vec{x}_i, Y_i)$, where $\vec{x}_i = (x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_{i_M})$ and $Y_i \subset L = \{L, L.1, L.1.1, L.1.2, \ldots\}$. Specifically, each example $\vec{x}_i$ is represented by M predictive features (attributes) and has a set of labels Y_i for which there are relationships that obey a hierarchical class structure stipulated *a priori*. The class attribute, in turn, reflects the concept to be learned and described by the induced hierarchical models using supervised machine learning algorithms.

We can discern the hierarchical classification methods according to four central aspects [11]. The first one covers the type of hierarchical structure – tree or Direct Acyclic Graph (DAG) –, taken to depict the relationships among classes. In the tree structure (Figure 1(a)), each node, except the root node, is linked with at most one parent node. In the DAG structure (Figure 1(b)), each node, except the root node, can have one or more parent nodes.

(a) Tree   (b) DAG
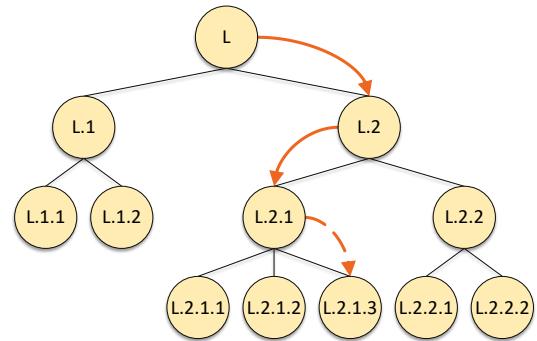
Figure 1. Hierarchical class structures.

The second aspect determines whether the algorithm can predict classes in one or more paths in the hierarchical structure. For instance, in the class hierarchy tree of Figure 1(a), if the model is able to predict both classes L.1.1 and L.1.2 for a provided example, which refers to the

paths L→L.1→L.1.1 and L→L.1→L.1.2, then it can predict multiple paths – Multiple Path Prediction (MPP). Conversely, the method performs Single Path Prediction (SPP) when this type of association is invalid.

The third aspect involves the hierarchical level at which the classification takes place. An algorithm can predict using only classes represented by leaf nodes – Mandatory Leaf-Node Prediction (MLNP) – or by using classes denoted by any internal or leaf node within the hierarchical structure – Non-Mandatory Leaf-Node Prediction (NMLNP). Figure 2 illustrates these two prediction strategies; the path L→L.2→L.2.1 portrays the NMLNP strategy, and the path L→L.2→L.2.1→L.2.1.3 indicates the MLNP tactic. We need to highlight that the NMLNP strategy is convenient mainly in applications that opt for the freedom to conduct a more generic classification, but with greater reliability.

Figure 2. Hypothetical class hierarchy.
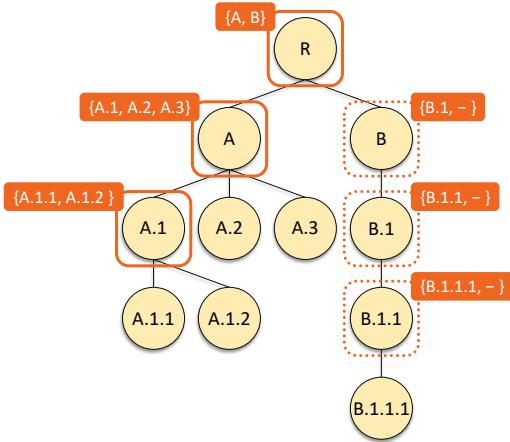
The fourth and final aspect concerns the way that machine learning methods deal with the hierarchical structure. We can group the approaches described in the literature into three broad categories: (i) flat approach, (ii) local approach, and (iii) global approach. Further details are available in [11].

4. PROPOSED APPROACH

One interesting research aspect that has been neglected by the music information retrieval and machine learning communities is the development and evaluation of the fusion of two or more hierarchical classification approaches. The motivation behind this idea arises because, unlike the global approach that generates a single classifier whose structure includes the entire class hierarchy [32, 35, 36], traditional approaches work with several local classifiers – binary or multiclass – to model the taxonomy of the problem's labels [9, 11].

Local approaches are generally preferred over the global ones due to the possibility of employing conventional supervised machine learning algorithms, which have been extensively tested and validated in flat classification tasks [37]. However, while some of the local approaches are computationally expensive in terms of time and memory, others make assumptions about the class hierarchy and, as a result, cannot be directly applied in some scenarios like the one treated here.

As aforementioned, the purpose of this paper is to combine the per node and per parent node local approaches to obtain a more efficient one. The justification for proposing a hybrid approach comes from the music genre classification problem, and we will explain it below using the class hierarchy tree of Figure 3.

Figure 3. Proposed approach: a hybrid local approach based on the per node and per parent node approaches. Squares with curved corners symbolize multiclass classifiers. Squares with dotted curved corners depict binary classifiers.

The class hierarchy of Figure 3 shows 11 classes that can be interpreted as music genres. A music genre can have one or more subgenres. If we adopted the local classifier per node approach, we would create a binary classifier for each class in the hierarchy, except for the root node, employing a set of positive examples – examples representing the current class – and a set of negative examples – examples that they are not associated with the current class. In this sense, to find the local training sets related to each class from the training dataset, several heuristics have been proposed [12, 13, 37]. The per node approach is suited to the task of music genre classification, but due to the construction of $|L| - 1$ classifiers, it is expensive regarding memory and processing.

On the other hand, if we were to use the local classifier per parent node approach, we would build, for each non-leaf node in the class hierarchy, a multiclass classifier to label new examples according to their subclasses. Therefore, in this approach, classifiers are also generated from sets of local training examples. Each local training set should be prepared so that the examples included therein are labeled only with the classes that will be differentiated by the multiclass classifier. The label of each example inserted in the training set selected for the classifier's induction must be generalized so that only the labels referring to the child classes of the analyzed node are present. Here a problem arises: if the approach were to encounter nodes B, B.1, and B.1.1 in Figure 3, it would generate one-class local datasets. Single paths like B→B.1→B.1.1→B.1.1.1 are essential in music genre classification since it demonstrates the evolution of the genre over time. Despite this

issue, the per parent node approach has better memory and processing than the per node one, and the resulting classifiers are less complex than the per level approach.

The local classifier per level approach, which creates a multiclass classifier for each level of the hierarchy, is not exempt from limitations. For the fourth level of the tree structure in Figure 3, such an approach would generate a local dataset containing only one class (B.1.1.).

In order to address the above issues, we propose a hybrid approach that represents the class hierarchy as a tree. We set up it as follows: for each internal node in the hierarchy with two or more children, we apply the local classifier per parent node approach and build a multiclass classifier with the children nodes, as indicated by the squares with curved corners in Figure 3. Otherwise, if the node has only one child, we apply the local classifier per node approach and generate a binary classifier, as symbolized by the squares with dotted curved corners in Figure 3. Note that we need to employ a strategy to choose the positive and negative examples from the local binary datasets. The literature shows that the more examples we consider in the learning phase, the better the induced classifier performs [12].

In this work, we suggest applying the "inclusive" heuristic to create local datasets [12]. This heuristic defines that the set of positive examples (S^+) is the most specific class and its descendants. In contrast, the set of negative examples (S^-) is all other classes except those in the set of positive examples and the ancestors of the most specific class. The output of the "inclusive" heuristic for node L.2.1 is: $S^+ = \{L.2.1, L.2.1.1, L.2.1.2, L.2.1.3\}$, and $S^- = \{L.1, L.1.1, L.1.2, L.2.2, L.2.2.1, L.2.2.2\}$.
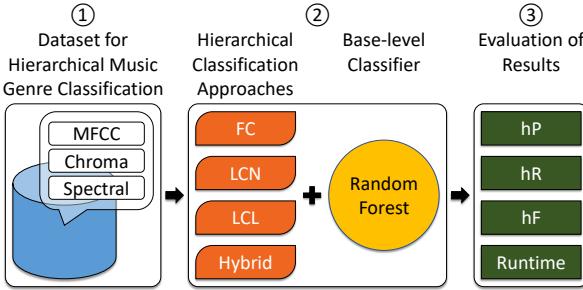
At each level of the hierarchy, groups of music genres are distinguished by their differences. Intuitively, the acoustic features that discriminate gothic metal from pop music diverge from those that differentiate country from gospel music. Hence, we can affirm that the classification of distinct music genres, like many other objects, benefits from different representations at distinct levels of the hierarchy. For this reason, we advocate applying learning algorithms based on decision trees to induce the local classifiers since they internally perform feature selection.

Finally, the hierarchy of music genres has some peculiarities that allow the classification to stop at internal nodes or go down up to the leaf nodes. To avoid inconsistencies in the classification step, we recommend using the top-down prediction strategy. In this strategy, an example is initially classified – based on the classifier's reliability (classification score) – among the first-level classes, and the subtrees of interest are only used to classify the examples at the other levels. The classification procedure is interrupted when the current classifier's reliability is less than a predefined $threshold$.

5. EXPERIMENTAL EVALUATION

This section presents an empirical assessment of our proposal and its comparison with three well-known approaches (Figure 4). First, we describe the considered

dataset. Next, we report the experimental setup regarding the adversary approaches, parameter setting, and evaluation measures. Then, we show and discuss the obtained results in terms of predictive performance and learning time.

Figure 4. High-level overview of the empirical evaluation.

5.1 Dataset

In our experiments, we used data from the Free Music Archive (FMA) [38]. This dataset contains 106,574 recordings, organized in 161 imbalanced genres.

We note that this paper's focus is not proposing novel features or comparing them in the classification scenario. For this reason, we considered the features available with the FMA dataset instead of extracting or learning features from the audios. Such a decision allowed us to compare the hierarchical classification approaches without relying on the gain provided by a better (or worse) feature set.

The features provided with the dataset, which we applied in our computational tests, are statistics from windowed MFCC, chroma, and spectral features extracted from 30 seconds in the middle of each recording using the LibROSA framework [39]. Specifically, these statistics are the mean, standard deviation, skew, kurtosis, median, minimum, and maximum.
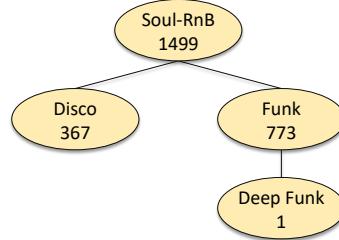
As noted in Section 2, music genre classification is usually performed by timbre-related features, such as MFCC and spectral characteristics. However, we also included the chroma-based features in our experiments to evaluate both the contribution of this kind of feature in the genre classification and our proposal's behavior when dealing with distinct music characteristics.

The FMA dataset has a default training-validation-test split. As we did not use the validation set to tune parameters, we merged it with the test examples. This operation provided us a training set that comprises 80% of the examples. Consequently, 20% of the remaining recordings belong to the test partition. Besides, this split is stratified and is guaranteed that there is no artist in the test set that also appears in the training set.

The genre hierarchy is an interesting characteristic to observe in this dataset. As the FMA allows the artists to label their songs themselves, the dataset presents a complex hierarchy of genres.

Another challenging factor present in this dataset is the class imbalance – "unbalanced with 1 to 38,154 tracks per genre" [38].

Figure 5 illustrates one branch of the class hierarchy, which presents the stated issues. It has nodes with single and multiple children, as well as genres with a significant difference regarding the number of tracks.

Figure 5. Example of genre hierarchy for the top-level Soul-RnB genre.

To make the hierarchical classification of single paths feasible, we address some issues found in the FMA dataset. In so many cases, the associated genres belong to more than one path in the genre hierarchy tree. In these cases, we select the genre for which the leaf node is at the lowest level. In case of ties, we kept that one in which the genre in the lowest level of each path comprises the higher number of tracks. We also removed unlabeled examples and examples from the test set whose classes were not present in the training set. Therefore, the dataset evaluated in this paper has 82,374 training examples and 15,681 test examples. These examples are described by 461 features and are associated with 120 classes organized hierarchically. The class hierarchy has four levels, each with 21, 107, 17, and 2 classes.

5.2 Compared Approaches and Parameter Settings

We compared the hybrid approach with three other well-known ones: (i) Flat Classifier (FC), (ii) Local Classifier per Node (LCN), and (iii) Local Classifier per Level (LCL). For FC, we assume each possible path in the label tree to be a class. For LCN, we use the "inclusive" heuristic to generate local datasets.

We adopted Random Forest [40] as a symbolic base-level classifier with the number of variables available for splitting at each tree node ($mtry$) equal to $\sqrt{(M-1)}$.

We considered the NMLNP strategy with $threshold = 0.5$. This setting means that the classification at the deepest levels is interrupted when the classification score is less than 0.5 or, in the case of per node classifiers, the predicted class is negative.

The experimental protocol execution comprised the use of the programming language R[4] with the following packages: caret, data.table, data.tree, and doParallel.

5.3 Evaluation Measures

In hierarchical classification problems, classes belonging to the levels furthest from the root node are generally more difficult to predict than classes associated with levels closest to the root node. In view of this, we assessed the quality

---

[4] https://www.r-project.org.

of the hierarchical classification approaches according to three hierarchical performance measures: (i) hierarchical Precision (hP), (ii) hierarchical Recall (hR), and (iii) hierarchical F-measure (hF). They are defined as follows:

$$hP = \frac{\sum_i |Y_i \cap \hat{Y}_i|}{\sum_i |\hat{Y}_i|} \quad (1) \qquad hR = \frac{\sum_i |Y_i \cap \hat{Y}_i|}{\sum_i |Y_i|} \quad (2)$$

$$hF = \frac{(\beta^2 + 1) \times hP \times hR}{\beta^2 \times hP + hR} \quad (3)$$

In Eqns. 1 and 2, $\hat{Y}_i$ denotes the set of labels predicted for a test example i and Y_i corresponds to the set of classes correct for this example. During the *hP* and *hR* computations, we need to discard the root node of the label hierarchy since, by definition, it is common to all the examples. The summations, in turn, are calculated over all the test examples.

As for Eqn. 3, β belongs to $[0, \infty)$ and refers to the importance assigned to the hP and hR values. Here, we established $\beta = 1$.

We must emphasize that the described measures are extended versions of the well-known metrics of precision, recall, and F-measure but tailored to the hierarchical classification scenario.

5.4 Results and Discussion

In order to check the performance of the hybrid method, we made comparisons with three other hierarchical classification approaches: FC, LCN, and LCL.

Our first evaluation criterion to be analyzed is predictive performance. Table 1 exhibits the three hierarchical performance measures obtained in the FMA dataset for the proposed method, as well as for the approaches used as baselines.

Performance measure	Hierarchical approaches			
	FC	LCN	LCL	Hybrid
hP	67.62	72.36	69.86	75.79
hR	66.10	71.96	69.46	77.38
hF	66.88	72.16	69.66	76.57

Table 1. Hierarchical predictive performance in % of the traditional approaches from the literature compared to our approach.

As shown in Table 1, the hybrid approach provided the best results, surpassing by a margin of approximately 4% the LCN scheme that is widely applied in the related literature. The poorest results came from the flat classifier. We expected this since such a model completely ignores the problem class hierarchy and, consequently, does not use domain knowledge to decompose the feature space of the problem in question into subproblems with a smaller number of classes.

After the predictive performance, our second evaluation criterion is runtime. In this work, runtime refers to learning time, *i.e.*, the time spent in inducing hierarchical classifiers over a dataset. Table 2 presents the runtime results achieved in the FMA dataset using both our proposal and the baseline approaches. We performed the tests on a server running 2.10 GHz Intel Xeon E5-2620 v4 processor (32-core) with 92GB RAM and operational system Debian 4.9.130-2 (64 bits) under the same processing conditions for all measurements. The times are indicated in minutes (min) or hours (h).

	Hierarchical approaches			
	FC	LCN	LCL	Hybrid
Learning time	48 min	92.44 h	4.41 h	3.53 h

Table 2. Time costs of the traditional approaches from the literature compared to our approach.

In Table 2, we can see that the hybrid method had a relatively shorter execution time than the LCN and LCL approaches. While LCN built 119 binary models and LCL four multiclass classifiers, our approach induced 27 models, 16 of which are multiclass and 11 binaries. We highlight that although our proposal has generated more classifiers than LCL, the induced models involved fewer examples and classes.

Even though the flat classifier's learning time was shorter than that of the hybrid method, the latter provided the best results in terms of predictive performance.

6. CONCLUDING REMARKS

In this paper, we introduced a novel approach for hierarchical music genre classification. The proposed method is based on combining local approaches to adapt the genre classification to more realistic hierarchies of music genre. Our results showed that the designed approach is better than the traditional ones in terms of predictive performance and execution time.

As future work, we intend to extend our method to deal with multiple labels in distinct paths in the hierarchy. Also, we plan to evaluate the use of different features and learning algorithms for the local classifications.

Finally, this research covered one real music genre hierarchy. As mentioned, distinct music platforms organize their labels in different ways. Thus, it is also our interest to perform a broad study on other real music genre structures to find specific contrivances that may aggregate information to improve hierarchical classifications in the context of music data.

7. ACKNOWLEDGEMENT

8. REFERENCES

[1] J.-J. Aucouturier and F. Pachet, "Representing musical genre: a state of the art," *Journal of new music research*, vol. 32, no. 1, pp. 83–93, 2003.

[2] E. Pampalk and M. Goto, "MusicSun: a new approach to artist recommendation," in *International Society for Music Information Retrieval Conference*, 2007, pp. 101–104.

[3] D. C. Corrêa and F. A. Rodrigues, "A survey on symbolic data-based music genre classification," *Expert Systems with Applications*, vol. 60, pp. 190–210, 2016.

[4] S. Lippens, J.-P. Martens, and T. De Mulder, "A comparison of human and automatic musical genre classification," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4. IEEE, 2004, pp. iv–iv.

[5] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[6] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl, "Aggregate features and ADABOOST for music classification," *Machine Learning*, vol. 65, no. 2-3, pp. 473–484, 2006.

[7] A. Holzapfel and Y. Stylianou, "Musical genre classification using nonnegative matrix factorization-based features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 424–434, 2008.

[8] I. Panagakis, E. Benetos, and C. Kotropoulos, "Music genre classification: a multilinear approach," in *International Society for Music Information Retrieval Conference*, 2008, pp. 583–588.

[9] J. G. Colonna, J. Gama, and E. F. Nakamura, "A comparison of hierarchical multi-output recognition approaches for anuran classification," *Machine Learning*, vol. 107, no. 11, pp. 1651–1671, 2018.

[10] A. R. S. Parmezan, V. M. A. Souza, and G. E. A. P. A. Batista, "Towards hierarchical classification of data streams," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, ser. Lecture Notes in Computer Science. Springer, 2019, vol. 11401, pp. 314–322, cIARP 2018.

[11] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1, pp. 31–72, 2011.

[12] R. Eisner, B. Poulin, D. Szafron, P. Lu, and R. Greiner, "Improving protein function prediction using the hierarchical structure of the gene ontology," in *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*. IEEE, 2005, pp. 1–10.

[13] T. Fagni and F. Sebastiani, "On the selection of negative examples for hierarchical text categorization," in *Language & Technology Conference*, 2007, pp. 24–28.

[14] J. H. Lee, H. Cho, and Y.-S. Kim, "Users' music information needs and behaviors: Design implications for music information retrieval systems," *Journal of the Association for Information Science and Technology*, vol. 67, no. 6, pp. 1301–1330, 2016.

[15] M. Torrens, P. Hertzog, and J. L. Arcos, "Visualizing and exploring personal music libraries," in *International Society for Music Information Retrieval Conference*, 2004.

[16] P. Knees, T. Pohle, M. Schedl, and G. Widmer, "A music search engine built upon audio-based and web-based similarity measures," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2007, pp. 447–454.

[17] P. Knees and M. Schedl, "A survey of music similarity and recommendation from music context data," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 10, no. 1, pp. 2:1–2:21, 2013.

[18] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "A survey of audio-based music classification and annotation," *IEEE transactions on multimedia*, vol. 13, no. 2, pp. 303–319, 2011.

[19] J. Bergstra, "Algorithms for classifying recorded music by genre," 2006, master's thesis, Université de Montréal.

[20] M. I. Mandel and D. Ellis, "Song-level features and support vector machines for music classification," in *International Society for Music Information Retrieval Conference*, 2005, pp. 594–599.

[21] C. Xu, N. C. Maddage, X. Shao, F. Cao, and Q. Tian, "Musical genre classification using support vector machines," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5. IEEE, 2003, pp. V–429.

[22] P. Ruvolo, I. Fasel, and J. R. Movellan, "A learning approach to hierarchical feature selection and aggregation for audio classification," *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1535–1542, 2010.

[23] S. Sigtia and S. Dixon, "Improved music feature learning with deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2014, pp. 6959–6963.

[24] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks," in *International Society for Music Information Retrieval Conference*, vol. 10, 2010, pp. 339–344.

[25] L. Deng, D. Yu *et al.*, "Deep learning: Methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[26] I.-Y. Jeong and K. Lee, "Learning temporal features using a deep neural network and its application to music genre classification," in *International Society for Music Information Retrieval Conference*, 2016, pp. 434–440.

[27] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2017, pp. 2392–2396.

[28] J. J. Burred and A. Lerch, "A hierarchical approach to automatic musical genre classification," in *International Conference on Digital Audio Effects*, 2003, pp. 8–11.

[29] C. McKay and I. Fujinaga, "Automatic genre classification using large high-level musical feature sets," in *International Society for Music Information Retrieval Conference*, 2004, pp. 525–530.

[30] T. Li and M. Ogihara, "Music genre classification with taxonomy," in *International Conference on Acoustics, Speech, and Signal Processing*, vol. 5. IEEE, 2005, pp. v–197.

[31] S. Brecheisen, H.-P. Kriegel, P. Kunath, and A. Pryakhin, "Hierarchical genre classification for large music collections," in *International Conference on Multimedia and Expo*. IEEE, 2006, pp. 1385–1388.

[32] C. N. Silla and A. A. Freitas, "Novel top-down approaches for hierarchical classification and their application to automatic music genre classification," in *IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2009, pp. 3499–3504.

[33] H. B. Ariyaratne and D. Zhang, "A novel automatic hierachical approach to music genre classification," in *International Conference on Multimedia and Expo*. IEEE, 2012, pp. 564–569.

[34] W. Du, H. Lin, J. Sun, B. Yu, and H. Yang, "A new hierarchical method for music genre classification," in *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics*. IEEE, 2016, pp. 1033–1037.

[35] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, vol. 73, no. 2, p. 185, 2008.

[36] J. Wehrmann, R. Cerri, and R. Barros, "Hierarchical multi-label classification networks," in *International Conference on Machine Learning*, 2018, pp. 5075–5084.

[37] B. Z. Santos, G. T. Pereira, F. K. Nakano, and R. Cerri, "Strategies for selection of positive and negative instances in the hierarchical classification of transposable elements," in *Brazilian Conference on Intelligent Systems*. IEEE, 2018, pp. 420–425.

[38] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: a dataset for music analysis," in *International Society for Music Information Retrieval Conference*, 2017, pp. 316–323.

[39] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "LibROSA: audio and music signal analysis in python," in *Python in Science Conference*, 2015, pp. 18–25.

[40] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*, 3rd ed. California: Morgan Kaufmann, 2011.

MULTITASK LEARNING FOR INSTRUMENT ACTIVATION AWARE MUSIC SOURCE SEPARATION

Yun-Ning Hung
Center for Music Technology
Georgia Institute of Technology
amyhung@gatech.edu

Alexander Lerch
Center for Music Technology
Georgia Institute of Technology
alexander.lerch@gatech.edu

ABSTRACT

Music source separation is a core task in music information retrieval which has seen a dramatic improvement in the past years. Nevertheless, most of the existing systems focus exclusively on the problem of source separation itself and ignore the utilization of other —possibly related— MIR tasks which could lead to additional quality gains. In this work, we propose a novel multitask structure to investigate using instrument activation information to improve source separation performance. Furthermore, we investigate our system on six independent instruments, a more realistic scenario than the three instruments included in the widely-used MUSDB dataset, by leveraging a combination of the MedleyDB and Mixing Secrets datasets. The results show that our proposed multitask model outperforms the baseline Open-Unmix model on the mixture of Mixing Secrets and MedleyDB dataset while maintaining comparable performance on the MUSDB dataset.

1. INTRODUCTION

Music source separation has long been an important task for Music Information Retrieval (MIR) with numerous practical applications. By isolating the sound of individual instruments from a mixture of music, source separation systems can be used, for example, as a pre-processing tool for music transcription [22] or for audio remixing [37]. They also enable special applications such as the automatic generation of karaoke tracks by separating vocals from the accompaniment, stereo-to-surround upmixing, and instrument-wise equalization [1, 24].

Most of the current source separation systems use deep learning approaches to estimate a spectral mask for each independent instrument, then apply the mask to the mixture audio for separation. Although the utilization of deep learning has improved source separation performance dramatically, one problem of this approach is the limited amount of training data for the prevalent supervised learning approaches. More specifically, the datasets need to comprise

of the separated tracks of each instrument, which renders most easily accessible music data useless as it is already mixed. Multiple open-source datasets attempt to address this issue [5, 11, 17]. MUSDB [23] is nowadays the most frequently used dataset for the training and evaluation of source separation systems. In addition to the limited size, the main shortcoming of MUSDB is the limited number of instrument tracks: 'Bass,' 'Drums,' 'Vocals,' and 'Other.' Other datasets show other drawbacks, for instance, the iKala and MIR-1K datasets only contain short clips of music instead of complete songs [5, 11].

In addition to the data challenge, one potential issue with most existing music source separation systems is that they exclusively focus on the source separation task itself. Harnessing the information of other MIR tasks by incorporating them into source separation, however, has not been explored in-depth. For example, Instrument Activation Detection (IAD) can help determine which time frame contains the target instrument, while pitch detection can help determine which frequency bins are more likely to contain a harmonic series [13, 19]. This kind of multitask learning approach has been reported to be efficient for multiple other MIR tasks. Böck et al. achieve state-of-the-art performance for both tempo estimation and beat tracking by learning these two tasks at the same time [4]. Bittner et al. show that by estimating multi-f0, melody, bass line, and vocals at the same time, the system outperforms its single-task counterparts on all four tasks [2]. Similar results have been reported for simultaneously estimating score, instrument activation, and multi-f0 [12]. However, only recently was multitask learning successfully applied to source separation by combining it with pitch estimation [28].

In this paper, we propose a novel multitask learning structure to explore the combination of IAD and music source separation. By training for both tasks in an end-to-end manner, the estimated instrument labels can be used during inference as a weight for each time frame. The goal is both to suppress the frames not containing the target instrument and to correct a potentially incorrectly estimated mask. To increase the size of the available training data, we leverage two open-source large-scale multi-track datasets (MedleyDB [3] and Mixing Secrets [8]) in addition to MUSDB to evaluate on a larger variety of separable instruments. We refer to the combination of these two datasets as the *MM dataset*.

In summary, the main contributions of this work are
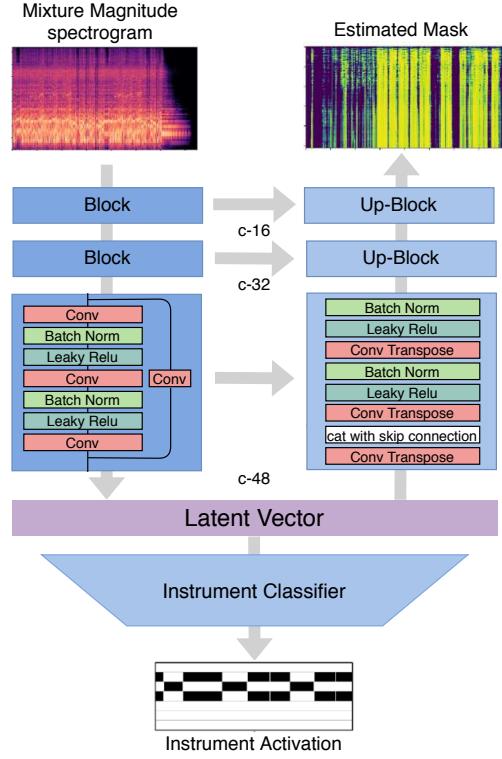
- the systematic investigation of the first multi-task source separation deep learning model that incorporates source separation with IAD in the spectral domain,

- the application of the IAD predictions during inference, and

- the presentation of the first open-source model that separates up to six instruments instead of the four tracks (3 independent instruments) featured by MUSDB.

2. RELATED WORK

State-of-the-art systems for music source separation are all based on deep learning due to proven superior performance. Uhlich et al. presented one of the pioneering works using a Deep Neural Network (DNN) architecture for music source separation [35], and Nugraha et al. used a DNN architecture and fully-connected layers for multichannel music source separation [21]. In the following years, more deep learning related systems were introduced. For example, Takahashi and Mitsufuji used recurrent neural networks to deal with temporal information [36], while others proposed the U-net structure for multiple separation tasks [14]. The U-net structure had been previously found useful for image segmentation [26] and treats the decomposition of a musical audio signal into the target and accompanying instrument tracks analogous to image-to-image translation. Takahashi et al. presented a dense LSTM that achieved the highest score in the SiSEC2018 [31] competition [34]. To preserve high resolution information, Liu and Yang introduced dilated 1-D convolution and a GRU network to replace pooling [16]. Different from other approaches using spectrograms as the input representation, Défossez et al. experimented on time-domain waveform source separation and showed that results comparable to spectrogram-based source separation systems are achievable [6]. "Spleeter," based on a U-net model structure, is currently regarded as one of the most powerful source separation systems [10]. It should be noted that —although the pre-trained model is freely available— Spleeter is trained on a proprietary, publicly unavailable dataset.

Stöter et al.'s "Open-Unmix" is frequently used as modern benchmark system on the MUSDB dataset [23]. It is a well documented open-source music source separation system with a recurrent architecture that achieves good separation results [32].

Most of the methods mentioned above are trained and evaluated on the open-source dataset used in SiSEC2018 competition [31]: MUSDB [23]. As mentioned above, one of the main problems of the MUSDB dataset is that it has only a limited amount of songs and instrument categories: it only includes three separable independent instruments. To include more separable instruments, Miron et al. proposed a score-informed system able to separate four classical instruments by training it on synthetic renditions [19]. However, their system is limited to classical music and requires the musical score for separation. While Spleeter is able to separate four independent instruments and Uhlich et al.
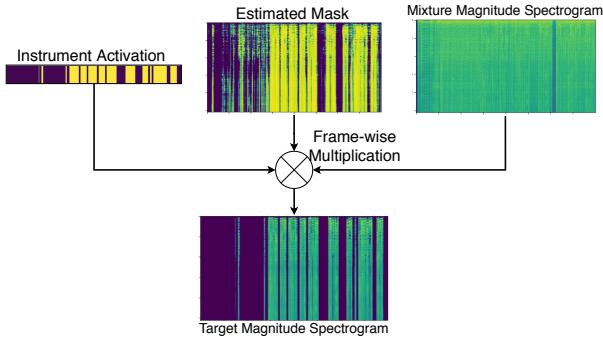
Figure 1. Multitask model structure for our proposed source separation system. Block is the residual block composed of convolutional layers while Up-Block is the residual block composed of transposed convolutional layers. "c" is the number of features.

constructed a dataset which contains nine separable instruments [35], both their datasets are not publicly available.

To explore the possibility of separating unseen instruments, Seetharaman et al. proposed to use instrument class labels as a condition to cluster time-frequency bins for different instruments into the embedding space [28]. Their work showed that the system can also separate unseen instruments during testing by sampling from the learned embedding space. Lee et al. proposed to use audio queries for music separation [15]. The learned feature vector from the audio query acts as the condition to inform the separation of the target source. Manilow et al. introduced a multitask learning structure for source separation, instrument classification, and music transcription [18]. They showed that by jointly learning these three tasks, the source separation quality increased. Furthermore, the network seems to generalize better to unseen instruments. Other works such as [29, 33] combine instrument activity with source separation, however, they utilized either the predicted activity or ground truth as an input condition instead of learning in an end-to-end manner.

3. METHOD

We propose a U-net-based [27] multitask structure to incorporate instrument recognition with music source separation, which we refer to as Instrument Aware Source Separation (IASS) system. An overview of the model is shown in

Figure 2. Using instrument activation as a weight to filter the estimated mask, which will be used to multiply with the mixture of the magnitude spectrogram.

Figure 1. Although the multitask approach shows similarities with previous approaches (compare [18]), we design our model with a different goal: instead of just learning a joint representation using the multitask structure, our model uses estimated labels from multitask learning during inference to improve source separation estimation.
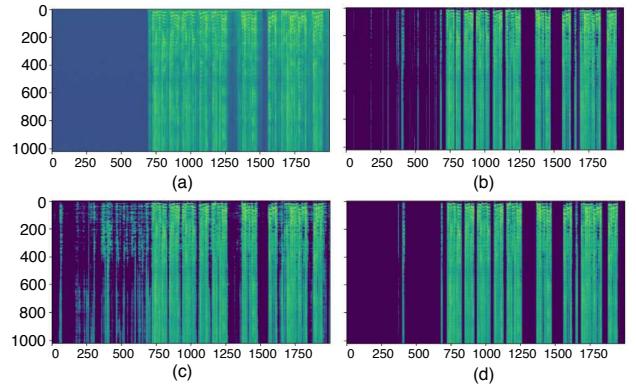
3.1 Model structure

The U-net structure has been found useful for image decomposition [26], a task with general similarities to source separation. The skip connections of U-net enable the model to learn from both high-level and low-level features leading to its success for music source separation [10, 14, 30].

Our model differs from previous U-net-based source separation systems by using a residual block instead of a CNN in each layer. The residual block allows the information from the current layer to be fed into a layer 2 hops away and deepens the structure. Each encoder and decoder contains three blocks with each block containing three convolutional or transposed convolutional layers, respectively, two batch normalization layers, and two leaky ReLU layers. The multitask objective is achieved by attaching a CNN classifier to the latent vector. This classifier predicts the instrument activity and has four transposed convolutional layers and three batch normalization layers in between. The last convolutional and transposed convolutional layers in each block feature a (3,1) filter size for up-sampling or down-sampling, while the others have a (3,3) filter size. During training, we use a Mean Square Error (MSE) loss for source separation and a Binary Cross-Entropy (BCE) loss for the prediction of the instrument activity. A hyperparameter α is manually tuned to balance these two loss functions:

$$L = L_{\text{MSE}} + \alpha L_{\text{BCE}}. \tag{1}$$

After successful training, the predicted instrument activity is used as a binary weight to multiply with the magnitude spectrogram along the time dimension. By doing so, the instrument labels are able to suppress the frames not containing any target instrument as shown in Figure 2. However, this binary instrument mask has two potential problems. First, false negatives of the predicted labels

Figure 3. First 2000 frames of the separated vocal track from the song *Angels In Amplifiers — I'm Alright* from the MUSDB-HQ dataset, visualizing different post-processing methods for applying instrument activity as a weight on the predicted magnitude spectrogram: (a) ground truth spectrogram, (b) predicted spectrogram without post-processing, (c) predicted spectrogram with raw predicted instrument labels as a weight, (d) predicted spectrogram with smoothed predicted instrument labels.

might mistakenly suppress wanted components in the spectrogram. Figure 3 (b) exemplifies that around frames 800 and 1000 with gaps caused by false negative prediction within a continuous sound. Even if the gaps have only a the length of a few milliseconds, it will have negative impact on the perceived quality. Second, the binary mask might cause repeated abrupt switching between silence and sound, which might lead to artifacts such as musical noise further decreasing the perceived quality. To address these problems, a median filter is applied to smooth the predicted instrument activities. The influence is discussed in Sect. 4.

An implementation of our system is publicly available online.[1]

3.2 Data representation

We extract magnitude spectrograms with a window length and hop size of 4096 and 1024 samples, respectively, at a sample rate of 44100 Hz for the input of our source separation model. The same magnitude spectrogram is extracted for the target audio reference. The instrument activity ground truth is at the frame level, meaning there is a binary label for each instrument to show whether the instrument is active or not in each time frame. Instrument labels have the same time resolution as the input spectrogram. We use the original activation probability computed by both datasets [3, 8], and binarize the activation with a threshold of 0.5 as suggested.

4. EXPERIMENT

To show the efficiency of our proposed model, we first compare our model with the baseline Open-Unmix model [32] on the MUSDB-HQ dataset. Note that we choose

---

[1] https://biboamy.github.io/Source_Separation_Inst

Method	Vocals	Bass	Drums	Other
IASS 3 blocks	6.46	4.18	5.56	4.19
IASS 4 blocks	6.51	4.25	5.15	4.38

Table 1. SDR score for IASS source separation performance with 3 and 4 residual blocks.

MUSDB-HQ instead of MUSDB because we want to obtain a high-quality separation system without potential coding artifacts. The audio of MUSDB is encoded in a lossy format while MUSDB-HQ provides the raw audio data. Other than that, there is no difference between the two datasets. However, since MUSDB-HQ is a newly released dataset it complicates comparing our results to other approaches directly as most of the previous systems have not been evaluated on the MUSDB-HQ dataset. Both the baseline and the proposed method are then evaluated on the MM dataset with six different instruments. For each source, a separate model is trained for both the baseline and our proposed method. Finally, an ablation study is conducted to investigate the impact of instrument labels and median filter on the source separation results. We train our models with the Adam optimizer with a learning rate of 0.001 and apply early stopping if the validation loss does not change for 100 epochs.

4.1 Dataset

Two open-source datasets, MUSDB-HQ [25] and the combination of Mixing Secrets [8] and MedleyDB [3] dataset (MM dataset) are used for the experiments. MUSDB is the most widely used dataset for music source separation and contains four separated tracks: 'Bass,' 'Drums,' 'Vocals,' and 'Others.' The dataset has 150 full-length stereo music tracks. We use the data split proposed by Stöter [32]: 86/14/50 songs for training, validation, and testing, respectively. Since data augmentation has been proven to be helpful [34], the following data augmentation is applied during the training. First, one track is randomly selected from each source and multiplied with a random gain factor ranging from 0.25 to 1.25. The starting point of each track is randomly chosen for chunking into a clip with length of 6 s. Finally, the chunked audio clips from each source are remixed for training. Since the original MUSDB-HQ dataset does not include instrument activation labels, we apply the energy tracking method proposed for MedleyDB [3] with a threshold of 0.5 to obtain the frame-level binary instrument activity labels.

The MM dataset contains 585 pieces of songs (330 from MedleyDB and 258 from Mixing Secrets) with more than 100 instruments and their individual tracks. We use the training and testing split proposed by Gururani et al. [9] for training (488 songs) and evaluating (100 songs) our system. The most frequently occurring 6 instruments are picked as target instruments: 'Bass,' 'Drums,' 'Vocals,' 'Electrical Guitar,' 'Acoustic Guitar,' and 'Piano.' One of the problems with this dataset is that not all of the songs provide parameters on how to remix the individual tracks into the mixture. Therefore, the volume of each track is adjusted

	Method	SDR	SIR	SAR	ISR
Vocals	Open-Unmix	6.11	13.21	6.75	12.43
	IASS	**6.46**	**14.70**	**6.98**	**14.30**
Bass	Open-Unmix	**4.48**	**8.23**	**5.40**	**10.29**
	IASS	4.18	7.30	4.52	6.85
Drums	Open-Unmix	5.02	10.17	6.05	10.55
	IASS	**5.56**	**10.74**	**6.86**	**10.92**
Other	Open-Unmix	**4.23**	**9.90**	3.88	7.34
	IASS	4.19	8.78	**4.70**	**9.32**

Table 2. BSS metrics for Open-Unmix and IASS on the MUSDB-HQ dataset.

to the same loudness (RMS) during training before applying the random gain as detailed above. In addition, each track is downmixed to a single channel. Furthermore, the data augmentation technique introduced above is applied to generate a large number of training samples from the MM dataset. We construct two separate groups of songs. One contains all the tracks including the target instrument, while another contains the tracks without the target instrument ("accompaniments"). There are total of 128 target tracks for 'Acoustic Guitar,' 189 for 'Piano,' 325 for 'Electrical Guitar,' 374 for 'Vocals,' 468 for 'Drums,' and 458 for 'Bass.' During training, we randomly select 1 to 5 tracks in the accompaniment pool to mix with the target instrument. By doing so, we can generate various combinations of training "songs." The random chunking approach applied to MUSDB-HQ is also applied on MM dataset. The testing set also balances the loudness of each track. We filter out the songs from the testing set which do not contain any of the 6 target instruments, resulting in 20 songs with 'Piano,' 23 songs with 'Acoustic Guitar,' 54 songs with 'Electrical Guitar,' 71 songs with 'Vocals,' 76 songs with 'Bass,' and 81 songs with 'Drums.'

4.2 Evaluation

To reconstruct the waveforms from the resulting magnitude spectrograms, we multiply the magnitude spectrogram with the phase of the original complex spectrograms and apply the inverse short-time Fourier transform on the complex spectrogram. We do not use any post-processing such as Wiener filtering here to focus on the raw result without potentially confounding quality gains in post-processing. Therefore, the Open-Unmix post-processing is disabled for the evaluation.

The quality of the source separation is evaluated with the four most frequently used objective metrics: source to distortion ratio (SDR), source to interference ratio (SIR), source to artifact ratio (SAR), and Image to Spatial distortion Ratio (ISR) [7]. We use the *museval* package for calculating the evaluation metrics [31].

4.2.1 Source separation on MUSDB-HQ

To allow for comparison with other systems trained on the MUSDB, the first experiment reports the result on MUSDB-HQ.

	Open-Unmix	IASS	IBM	input-SDR
Vocals	3.68	4.78	6.49	-6.24
Elecgtr	1.55	1.77	4.56	-5.90
Acgtr	0.95	1.29	3.38	-6.65
Piano	1.08	1.91	3.63	-6.31
Bass	4.04	5.26	5.34	-5.77
Drums	4.45	4.89	6.23	-6.05

Table 3. SDR score for Open-Unmix, IASS, an ideal binary mask and input-SDR.

In a first preliminary experiment we investigate whether adding more residual blocks influences the performance of the proposed method. We report the SDR score for the four sources of the MUSDB-HQ dataset in Table 1. The performance with three residual blocks and with four residual blocks is comparable on all instruments. For training efficiency, we use three residual blocks in the following experiment.
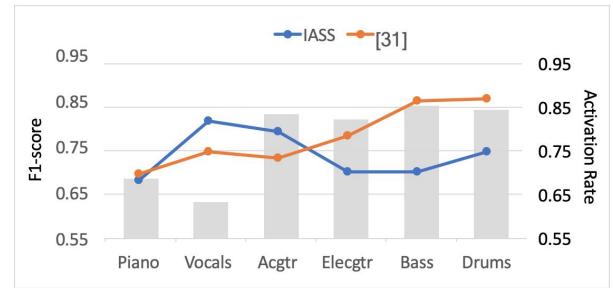
Table 2 shows the results of our proposed model compared to Open-Unmix. Our model outperforms the Open-Unmix model on 'Vocals' and 'Drums', performs equally on 'Other', and slightly worse on 'Bass'. This might be because 'Bass' most likely to appear throughout the songs. As a result, the improvement of using the instrument activation weight is limited. The imbalanced activity might also impact the instrument classifier.

4.2.2 Source separation on MM

The results for the MM dataset are summarized in Table 3. We re-trained the Open-Unmix model on the MM dataset by using the default training setting provided with their code. The results for the Ideal Binary Mask (IBM) (source code [31]) represent the best case scenario. The worst case scenario is represented by the results for input-SDR, which is the SDR score when using mixture as the input. Compared to MUSDB-HQ, the MM dataset has a larger amount of training data. It can be observed from Table 3 that our proposed model generally achieves better source separation performance on six instruments. We can also observe a trend that both models have higher scores on 'Drums,' 'Bass,' and 'Vocals' than on 'Electrical Guitar,' 'Piano,' and 'Acoustic Guitar.' This might be attributed to the fact that 'Guitar,' 'Piano,' and 'Acoustic Guitar' have fewer training samples (cf. Sect. 4.1). Another possible reason is that the more complicated spectral structure of polyphonic instruments such as 'Guitar' and 'Piano' make the separation task more challenging.

4.2.3 Instrument activity detection

While our system's source separation performance was the primary concern, the accuracy of the instrument predictions is also of interest. Our classifier output is compared to the model proposed by Gururani et al. [9], which was trained and evaluated on the same MM dataset. Note that this comparison is still not completely valid as their system uses multi-label prediction while our model is single-label.

Figure 4. IAD result for both Gururani et al.'s method (orange) and our IASS (blue) with label aggregation. Indicated in gray is the activation rate (percentage of the training frames containing positive activity labels).

Still, it can provide some insights into how well our system predicts the instrument activity. As Gururani et al.'s system predicts instrument labels with a time resolution of 1 s, the output resolution of our prediction has to be reduced. For each second, all the estimated activations are aggregated by calculating their median. Furthermore, instrument subcategories from their work are combined. For example, female and male singers are combined into 'Vocals,' electrical bass and double bass are combined into 'Bass,' electrical and acoustic piano are combined into 'Piano' and clean and distorted electrical guitar are combined into 'Electrical Guitar.' We report the AUC score in Figure 4.

We can make the following observations. First, 'Piano,' 'Electrical Guitar,' and 'Bass' tend to have lower detection rates. This might be because all these instrument categories include both acoustic and electric instruments which the model might easily confuse with the background music. This might also influence the source separation performance. The result can explain that in Table 4, 'Vocals' have the highest increase in the average score when applying instrument labels since 'Vocals' has better instrument detection accuracy. In contrast, 'Bass' has a lower increase since it has poorer instrument detection results. Second, from Figure 4 we can observe that 'Vocals' and 'Piano' have a lower activation rate, which means the model has fewer sound samples containing 'Vocals' and 'Piano' during training. This aligns with the highest SIR score increase on 'Vocals' and 'Piano' in Table 4 when instrument activation is added, since instrument activation can help suppress the interference at the non-active frames. This also shows the potential of our model to be used on instruments which only appear in the song infrequently.

4.3 Ablation study

In this experiment, we investigate the impact of the instrument labels on our model. First, the IASS model is trained and evaluated without using instrument labels, i.e., as a standard U-net without instrument classifier on the latent vector. The model will only be updated by the MSE between the ground-truth magnitude spectrogram and predicted spectrogram ($\alpha = 0$). For testing, all instrument "predictions" are set to 1. Second, the IASS model is trained
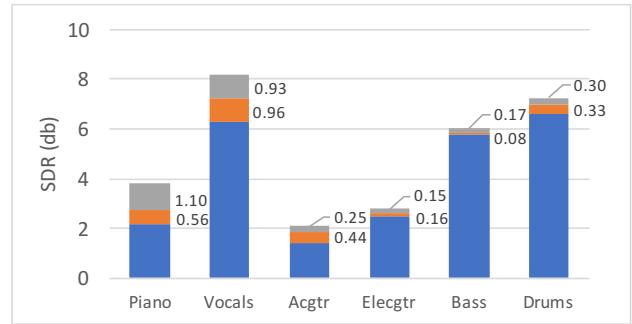
	Train	Test	SDR	SIR	SAR	Avg
Vocals	✗	✗	4.26	8.58	4.48	5.77
	✓	✗	3.94	8.48	4.69	5.70
	✓	✓	**4.78**	**11.62**	**5.31**	**7.24**
Elecgtr	✗	✗	1.75	0.61	**4.94**	2.46
	✓	✗	**1.82**	1.27	4.29	2.43
	✓	✓	1.77	**1.64**	4.48	**2.63**
Acgtr	✗	✗	1.11	0.75	2.42	1.43
	✓	✗	1.15	0.48	**2.52**	1.38
	✓	✓	**1.29**	**1.80**	2.45	**1.85**
Piano	✗	✗	1.55	2.97	2.13	2.31
	✓	✗	1.70	3.16	2.06	2.22
	✓	✓	**1.91**	**4.17**	**2.15**	**2.74**
Bass	✗	✗	4.10	**8.34**	4.74	5.72
	✓	✗	4.12	7.82	**5.10**	5.68
	✓	✓	**4.34**	8.13	**5.10**	**5.85**
Drums	✗	✗	4.50	9.54	5.15	6.40
	✓	✗	4.38	9.87	4.95	6.40
	✓	✓	**4.89**	**10.72**	**5.26**	**6.96**

Table 4. Ablation study for IASS source separation performance training and evaluating with (✓) or without (✗) instrument labels.

with instrument labels but evaluated without instrument labels. This is a traditional multitask scenario: the model will be trained with both the MSE and the BCE losses in Eq. (1). However, during evaluation, the output magnitude spectrogram is not weighted by the instrument activity (predictions equal 1). Third, we include the IASS results from Table 3 —computed with both losses and using the instrument predictions as mask weights— for convenience.

The results are shown in Table 4. It can be observed that using instrument labels as a weight generally leads to a better performance than without using instrument labels. The result also somewhat unexpectedly shows that training with instrument detection loss influences source separation performance, as the average quality score is often lower when training with the multitask loss. One possible reason for this is that adding the IAD sub-task forces more information to be passed to the bottom layers, where the resolution is compressed. We argue, however, that the multitask learning structure does bring an extra benefit to the system: using the instrument activity predictions as a weight leads to better separation quality. Figure 3 visualizes the effect on one of the songs from the MUSDB-HQ dataset. This song does not have any vocals before 16 s, which is around time frame 700. Subfigure (a) shows the ground truth magnitude spectrogram before applying instrument labels while (c) and (d) show the predicted spectrograms after applying the instrument activations or the smoothed instrument activations, respectively. Both the false positive predictions in the beginning before time frame 700 as well as the false negative predictions around frames 800 and 1000 have been repaired by using smoothed activations. This result is consistent with the results in Table 4 where SIR has the highest increase: interferences are more successfully suppressed.

Furthermore, we investigate the influence of median

Figure 5. Ablation study for IASS source separation performance with or without median filter on instrument labels. The orange bar shows the increase of the average score (average of SDR, SIR, and SAR) after applying median filtering. Gray shows the average score increase when using instrument ground truth labels instead of estimated labels.

filtering the predicted instrument activity on our results. Figure 5 shows the performance of our proposed source separation model with and without applying the median filter on the predicted instrument activities. Using the median filter generally increases performance across all instruments as it eliminates spurious prediction errors.

Finally we are using the oracle ground truth labels instead of the estimated labels as the weight. As we can observe from Figure 5, using the ground truth labels brings an average score increase in all instruments, especially for vocals and piano. This can be seen as the upper-bound best case scenario of our instrument-activity-weighted model and emphasizes the potential for improvement when combining instrument prediction with source separation.

5. CONCLUSION

In this paper, we proposed a novel multitask structure combining instrument activation detection with multi-instrument source separation. We utilize a large dataset to evaluate on various instruments and show that our model achieves equal or better separation quality than the baseline Open-Unmix model. The ablation study also shows that using instrument activation as a weight is able to correct the false estimation from the source separation task and improve source separation performance. In summary, the main contributions of this work are the proposal of a multitask learning structure combining IAD with source separation, and insights into using new open-source datasets to increase the number of separable instrument categories.

We have identified several directions for future extensions of this model. First, we plan to increase the number of target instruments by combining synthesized data with the MM dataset especially for underrepresented instrument classes. Second, we plan to incorporate other tasks, such as multi-pitch estimation, into our current multi-task structure [20]. Third, we will explore using multi-label instrument detection to separate multiple instruments at the same time. Lastly, we will explore post-processing methods such as Wiener filter to improve our system's quality.

6. ACKNOWLEDGMENT

7. REFERENCES

[1] Tülay Adali, Christian Jutten, Arie Yeredor, Andrzej Cichocki, and Eric Moreau. From raw audio to a seamless mix: creating an automated DJ system for Drum and Bass. *IEEE Signal Processing Magazine*, pages 16–17, 2014.

[2] Rachel M. Bittner, Brian McFee, and Juan Pablo Bello. Multitask learning for fundamental frequency estimation in music. *arXiv preprint arXiv:1809.00381*, 2018.

[3] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 155–160, 2014.

[4] Sebastian Böck, Matthew E. P. Davies, and Peter Knees. Multi-task learning of tempo and beat: Learning one to improve the other. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 569–576, 2019.

[5] Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Jyh-Shing Roger Jang. Vocal activity informed singing voice separation with the ikala dataset. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 718–722, 2015.

[6] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.

[7] Cédric Févotte, Rémi Gribonval, and Emmanuel Vincent. Bss_eval toolbox user guide – revision 2.0. In *IRISA Technical Report 1706, Rennes, France*, 2005.

[8] Siddharth Gururani and Alexander Lerch. Mixing secrets: a multi-track dataset for instrument recognition in polyphonic music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017. Late-breaking paper.

[9] Siddharth Gururani, Cameron Summers, and Alexander Lerch. Instrument activity detection in polyphonic music using deep neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 569–576, 2018.

[10] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. Spleeter: A fast and state-of-the-art music source separation tool with pre-trained models. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018. Late-breaking paper.

[11] Chao-Ling Hsu and Jyh-Shing Roger Jang. On the improvement of singing voice separation for monaural recordings using the mir-1k dataset. *IEEE Transactions on Audio, Speech, and Language Processing*, 18:310–319, 2010.

[12] Yun-Ning Hung, Yi-An Chen, and Yi-Hsuan Yang. Multitask learning for frame-level instrument recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 381–385, 2019.

[13] Yun-Ning Hung and Yi-Hsuan Yang. Frame-level instrument recognition by timbre and pitch. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 569–576, 2018.

[14] Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep U-Net convolutional networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, page 323–332, 2017.

[15] Jie Hwan Lee, Hyeong-Seok Choi, and Kyogu Lee. Audio query-based music source separation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 878–885, 2019.

[16] Jen-Yu Liu and Yi-Hsuan Yang. Dilated convolution with dilated gru for music source separation. In *International Joint Conferences on Artificial Intelligence*, pages 4718–4724, 2019.

[17] Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. The 2016 signal separation evaluation campaign. In *International conference on latent variable analysis and signal separation*, pages 323–332, 2017.

[18] Ethan Manilow, Prem Seetharaman, and Bryan Pardo. Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 771–775, 2020.

[19] Marius Miron, Jordi Janer, and Emilia Gómez. Monaural score-informed source separation for classical music using convolutional neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 569–576, 2017.

[20] Tomoyasu Nakano, Kazuyoshi Yoshii, Yiming Wu, Ryo Nishikimi, Kin Wah Edward Lin, and Masataka Goto. Joint singing pitch estimation and voice separation based on a neural harmonic structure renderer. *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 160–164, 2019.

[21] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. Multichannel music separation with deep neural networks. In *European Signal Processing Conference (EUSIPCO)*, pages 1748–1752, 2016.

[22] Jouni Paulus and Tuomas Virtanen. Drum transcription with non-negative spectrogram factorisation. In *European Signal Processing Conference*, pages 1–4, 2005.

[23] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, December 2017. [Online] https://doi.org/10.5281/zenodo.1117372.

[24] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stoter, Stylianos Ioannis Mimilakis, Derry FitzGerald, and Bryan Pardo. An overview of lead and accompaniment separation in music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(8):1307–1335, 2018.

[25] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. MUSDB18-HQ - an uncompressed version of MUSDB18, August 2019.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2015.

[28] Prem Seetharaman, Gordon Wichern, Shrikant Venkataramani, and Jonathan Le Roux. Class-conditional embeddings for music source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 301–305, 2019.

[29] Olga Slizovskaia, Leo Kim, Gloria Haro, and Emilia Gomez. End-to-end sound source separation conditioned on instrument labels. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 306–310, 2019.

[30] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. *arXiv preprint arXiv:1806.03185*, 2018.

[31] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. The 2018 signal separation evaluation campaign. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 293–305, 2018.

[32] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Open-unmix-a reference implementation for music source separation. *Journal of Open Source Software*, 2019.

[33] Rupak Swaminathan and Alexander Lerch. Improving singing voice separation using attribute-aware deep network. *International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 60–65, 2019.

[34] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation. In *International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 106–110, 2018.

[35] Stefan Uhlich, Franck Giron, and Yuki Mitsufuji. Deep neural network based instrument extraction from music. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2135–2139, 2015.

[36] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 261–265, 2017.

[37] Len Vande Veire and Tijl De Bie. From raw audio to a seamless mix: creating an automated dj system for drum and bass. *EURASIP Journal on Audio, Speech, and Music Processing*, 2018(1):13, 2018.

AUTOMATIC COMPOSITION OF GUITAR TABS BY TRANSFORMERS AND GROOVE MODELING

Yu-Hua Chen[1,2,3], **Yu-Hsiang Huang**[1], **Wen-Yi Hsiao**[1], **and Yi-Hsuan Yang**[1,2]

[1] Taiwan AI Labs, Taiwan, [2] Academia Sinica, Taiwan, [3] National Taiwan University, Taiwan

`r08946011@ntu.edu.tw`, `{yshuang,wayne391,yhyang}@ailabs.tw`
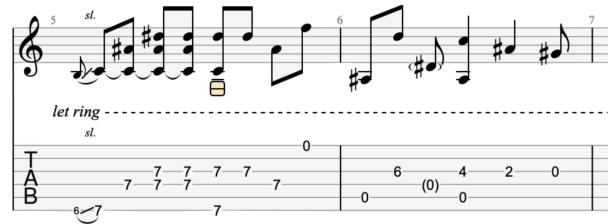
ABSTRACT

Deep learning algorithms are increasingly developed for learning to compose music in the form of MIDI files. However, whether such algorithms work well for composing guitar tabs, which are quite different from MIDIs, remain relatively unexplored. To address this, we build a model for composing fingerstyle guitar tabs with Transformer-XL, a neural sequence model architecture. With this model, we investigate the following research questions. First, whether the neural net generates note sequences with meaningful note-string combinations, which is important for the guitar but not other instruments such as the piano. Second, whether it generates compositions with coherent rhythmic groove, crucial for fingerstyle guitar music. And, finally, how pleasant the composed music is in comparison to real, human-made compositions. Our work provides preliminary empirical evidence of the promise of deep learning for tab composition, and suggests areas for future study.

1. INTRODUCTION

Thanks to the cumulative efforts in the community, in recent years we have seen great progress in using deep learning models for automatic music composition [8]. An important body of research has been invested on creating piano compositions, or more generally keyboard style music. For instance, the "Music Transformer" presented by Huang *et al.* [19] employs 172 hours of piano performances to learn to compose classical piano music. Another group of researchers extends that model to generate pop piano compositions from 48 hours of human-performed piano covers [20]. They both use a MIDI-derived representation of music and describe music as a sequence of event tokens such as `NOTE-ON` and `NOTE-VELOCITY`. While the MIDI format works the best for representing keyboard instruments and less for other instruments (for reasons described below), Donahue *et al.* [14] and Payne [31] show respectively that it is possible for machines to learn from a set of MIDI files to compose multi-instrument music.

There are, however, many other forms of musical notation that are quite different from the staff notation assumed by keyboard music. For example, the tabulature, or "tab" for short, is a notation format that indicates instrument fingering rather than musical pitches. It is common for fretted stringed instruments such as the guitar and ukulele, and free reed aerophones such as the harmonica. It makes more sense for people playing such instruments to read the tabs, as they suggest how to move the fingers.

Figure 1. An example of fingerstyle guitar tab composed by human, along with the corresponding staff notation.

As shown in Figure 1, a tab contains information such as the fingering configuration on the fretboard (six strings for the case of the guitar) as well as usage of the left-hand or right-hand playing techniques. Such information is usually missing in the corresponding staff notation and MIDI files. Learning to automatically compose guitar music directly from MIDI files, though possible, has the limitation of ignoring the way people play these instruments. However, to our best knowledge, little has been done to use tabs to train a deep generative model.

To investigate the applicability of modern deep learning architectures for composing tabs, we compile a new dataset of 333 TAB files of "fingerstyle guitar" (including originally fingerstyle guitar music and fingerstyle adaptation) [3], and modify the data representation of the Music Transformer [19] to make the extended model learn to compose guitar tabs. With this model, we aim to answer three research questions (RQs):

- Whether the neural network learns to generate not only the note sequences but also the fingering of the notes to be played on a fretboard, from reading only the tabs (instead of, for example, watching videos demonstrating how people play the guitar)?

- Whether the neural network generates compositions with coherent "groove," or the use of rhythmic patterns over time [13, 32, 39]? It is generally assumed that the layers of a neural network learn abstractions of data on their own to perform the intended

task, e.g., to predict the next events given the history. However, in music, groove is usually indirectly implied according to the arrangement of notes along the time axis, instead of explicitly specified in either a MIDI or TAB file. Therefore, it remains to be studied whether the model can do better if it has to explicitly handle bar-level GROOVING events, inserted into the training data as a high-level information in some way, or if such a modification is not needed. This is in particular relevant in the context of fingerstyle composition, as in fingerstyle a guitarist has to take care of the melody, chord comping, bass line and rhythm simultaneously [3].

- Finally, how the compositions generated by the neural network compare with human-composed guitar tabs, when both rendered into audio waveforms and presented to human listeners? This gives us a direct evaluation of the effectiveness of the neural network in modeling guitar music.

We provide audio rendition of examples of the generated tabs (using a guitar synthesizer of a DAW called Ample Sound [1]) at `https://ss12f32v.github.io/Guitar-Transformer-Demo/`, along with a video recording of a guitarist playing a generated tab.

In what follows, we review some related work in Section 2, and then present the tab dataset in Section 3. After that, we describe in Section 4 the methodology for modeling and learning to compose guitar tabs. We present the result of objective and subjective evaluations addressing the aforementioned research questions in Section 5.

2. RELATED WORK

2.1 Guitar-related Research in MIR

In the music information retrieval (MIR) community, research concerning guitar is often related to automatic guitar transcription [5, 7, 9, 16, 18, 21, 22, 27, 34, 46] and playing technique detection [4, 10, 37, 38]. For example, Su *et al.* [38] built a convolution neural network (CNN) model for detecting the playing techniques associated with the string-pressing hand, and incorporated that for transcribing audio recordings of unaccompanied electric guitar performances. Rodríguez *et al.* [34] presented a model for transcribing Flamenco guitar falsetas, and Abeßer and Schuller [5] dealt with the transcription of solo bass guitar recordings. We note that, while automatic transcription concerns with recovering the tab underlying an audio guitar performance, our work deals with automatic composition of original guitar tabs in the symbolic domain, and therefore does not consider audio signals.

As there are multiple fret positions to play the same note on a guitar, it may not be easy for a novice guitar learner to play a guitar song without the corresponding tab. Automatic suggestion of the fingering given a human-made "lead sheet," a symbolic format that specifies the melody and chord sequence but not their fingering, has therefore been a subject of research. Existing work has explored the use of hidden Markov models, genetic algorithm, and

neural networks to predict the fingering by examining its playing difficulty for a guitarist, viewing the task as an optimal path finding problem [6, 28, 35, 40]. While such prior arts can be considered as performing a MIDI-to-TAB conversion, our work aims to model TABs directly.

Xi *et al.* developed the GuitarSet [45], a set of 360 audio recordings of a guitar equipped with the hexaphonic pickup. The special pickup is able to capture the sound from each string individually, making it possible for a model to learn to perform multipitch estimation and tabulature fingering arrangement at the same time. Using the dataset, Wiggins and Kim [43] built such a model with CNN, achieving 0.83 F-score (i.e., the harmonic average of precision and recall) for multipitch estimation, and 0.90 for identifying the string-fret combinations of the notes. While the dataset is relevant for guitar transcription, its recordings are all around 12–16 bars in length only, which seems to be too short for deep generative modeling.

McVicar *et al.* [24–26] used to build sophisticated probabilistic systems to algorithmically compose rhythm and lead guitar tabs from an input chord and key sequence. Our work differs from theirs in that we aim to build a general-purpose tab composition model using modern deep generative networks. An extra complexity of our work is that we experiment with fingerstyle guitar, a type of performance that can be accomplished by a single guitarist.

2.2 Transformer Models for Automatic Composition

The Transformer [41] is a deep learning model that is designed to handle ordered sequences of data, such as natural language. It models a word sequence $(w_1, w_2, \ldots w_T)$ seen in the training data by factorizing the joint probability into a product of conditionals, namely, $P(w_1) \cdot P(w_2|w_1) \cdot \cdots \cdot P(w_T|w_1, \ldots, w_{T-1})$. During the training process, the model optimizes its parameters so as to correctly predict the next word w_t given its preceding history $(w_1, w_2, \ldots w_{t-1})$, for each position t in a sequence.

Following some recent work on recurrent neural network (RNN)-based automatic music composition [29, 42], Huang *et al.* [19] viewed music as a language and for the first time employed the Transformer architecture for modeling music. Given a collection of MIDI performances, they converted each MIDI file to a time-ordered sequence of musical "events," so as to model the joint probability of *events* as if they are *words* in natural language (see Section 4.1 for details of such events). The Transformer with relative attention was shown to greatly outperform an RNN-based model, called PerformanceRNN [29], in a subjective listening test [19], inspiring the use of Transformer-like architectures, such as Transformer or Transformer-XL [12], in follow-up research [11, 14, 20, 31, 44]. [1]

There are lots of approaches to automatic music composition, deep learning- and non-deep learning based included [8, 15, 30]. We choose to consider only the Transformer architecture here, to study whether we can translate its strong result in modeling MIDIs to modeling TABs.

---

[1] We note that it is debatable whether music and language are related. We therefore envision that some other new architectures people will come up with in the future might do a much better job than Transformers in modeling music. This is, however, beyond the scope of the current work.

	# tabs	# bars	# bars per tab	# events per tab
training	303	24,381	80±41	5,394±3,116
validation	30	2,593	74±35	5,244±3,183

Table 1. Statistics of the dataset; the last two columns show the mean and standard deviation values across each set. Please see Table 2 for definitions of the events.

3. FINGERSTYLE GUITAR TAB DATASET

There have been some large-scale MIDI datasets out there, such as the Lakh MIDI dataset [33] and BitMidi [2]. The former, for example, contains 176,581 unique MIDI files of *full songs*. In contrast, existing datasets of tabs are usually smaller and shorter, as they are mainly designed for learning the mapping between tabs and audio (i.e., for transcription research), rather than for generative modeling of the structure of tabs. The tabs in the GuitarSet [45], for example, are performances of *short excerpts of songs*, typically 12–16 bars in length, which are not long.

For the purpose of this research, we compile a guitar tab dataset on our own, focusing on the specific genre of fingerstyle guitar. Specifically, we collect digital TABs of *full songs*, to facilitate language modeling of guitar tabs. We go through all the collected TABs one-by-one and filter out those that are of low quality (e.g., with wrong fingering, obvious annotation errors), or are not fingerstyle (e.g., have more than one tracks). We also discard TABs that are not in standard tuning, to avoid inconsistent mapping between notes and fingering. As shown in Table 1, this leads to a collection of 333 TABs, each with around 80 bars. This includes TABs of famous professional fingerstyle players such as Tommy Emmanuel and Sungha Jung. All the TABs are in 4/4 time signature, and they can be in various keys. We reserve 30 TABs for validation and performance evaluation, and use the rest for training.

Please note that, similar to the MIDI files available in Lakh MIDI [33], the TAB files we collect do not contain *performance* information such as expressive variations in dynamics (i.e., note velocity) and micro-timing [23, 29]. To increase velocity variation, we use Ample Sound [1] to add velocity to each note by its humanization feature. We do not deal with micro-timing in this work.
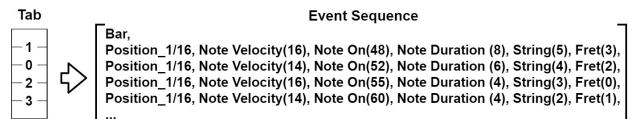
3.1 Fingerstyle

It is interesting to focus on only fingerstyle guitar in the context of this work, as we opt for validating the effectiveness of Transformers for single-track TABs first, before moving to modeling multi-track performances that involve at least a guitar (e.g., a rock song). We give a brief introduction of fingerstyle guitar below.

Fingerstyle [3] is at first a term that describes using fingertips or fingernails to pluck the strings to play the guitar. Nowadays, the term is often used to describe an arrangement method to blend multiple parts of musical elements or tracks, which are initially played by several instruments, into the composition of one guitar track. Therefore, a guitarist playing fingerstyle has to simultaneously take care of

category/type	description
NOTE-ON	45 different pitches (E2–C6)
NOTE-DURATION	multiples of the 32th note (1–64)
NOTE-VELOCITY	note velocity as 32 levels (1–32)
POSITION	temporal position within a bar; multiples of the 16th note (1–16)
BAR	marker of the transition of bars
STRING	6 strings on a tab
FRET	20 fret positions per string
TECHNIQUE	5 playing techniques: slap, press upstroke, downstroke, and hit-top
GROOVING	32 grooving patterns

Table 2. The list of events adopted for representing a tab as an event sequence. The first five are adapted from [19,20], whereas the last four are tab-specific and are new. We have in total 45+64+32+16+1+6+20+5+32=231 unique events.

Figure 2. An example of the result of "TAB-to-event" conversion needed for modeling a tab as a sequence. Here, we show the resultant event representation of a C chord.

the melody line, bass line, chord comping and the rhythmic groove. Groove, in particular, is important in fingerstyle, as it is now only possible to work on the rhythmic flow of music with a single guitar and the use of the two hands. We hence pay special attention to groove modeling in this work (see Section 4.3).

4. MODELING GUITAR TABS

In this section, we elaborate how we design an event representation for modeling guitar tabs, or more generally tabs of instruments played by string strumming.

4.1 Event Representation for MIDIs: A Quick Recap

In representing MIDIs as a sequence of "events," Huang *et al.* [20] considered, amongst others, the following event tokens. Each note is represented by a triplet of NOTE-ON, NOTE-DURATION, and NOTE-VELOCITY events, representing the MIDI note number, quantized duration as an integer multiple of a *minimum duration*, and discrete level of note dynamics, respectively. The minimum duration is set to the 32th note. The onset time of the notes, on the other hand, is marked (again after quantization) on a time grid with a specific *resolution*, which is set to the 16th note as in [19]. Specifically, to place the notes over the 16-th note time grid, they use a combination of POSITION and BAR events, indicating respectively the position of a note onset within a bar, among the 16 possible locations, and the beginning of a new bar as the music unfolds over time. This

event representation has been shown effective in modeling pop piano [20]. We note that the time grid outlined with this combination of POSITION and BAR events can also contribute to modeling the rhythm of fingerstyle guitar.

4.2 Event Representation for Tabs

To represent TABs, we propose to add, on top of the aforementioned five types of events for MIDIs,[2] the following three new types of fingering-related events: STRING, FRET, TECHNIQUE, and a type of rhythm-related events: GROOVING. We introduce the first three below, and the last in the next subsection. Table 2 lists all the events considered, whereas Figure 2 gives an example of how we represent a C chord with such an event representation.

We use the first 20 frets of the 6 strings in the collected TABs, i.e., each string can play 20 notes. The pitch range of the strings overlaps, so a guitarist can play the same pitch on different strings, with moderate but non-negligible difference in timbre. The fingering of the notes also affects playability [46]. In standard tuning, the strings can play 45 different pitches, from E2 to C6.

In our implementation, we adopt the straightforward approach to account for the various possible playing positions of the notes—to add STRING and FRET tokens right after the NOTE-ON tokens in the event sequence representing a tab. We note that the FRET tokens are actually *redundant*, in that the combination of NOTE-ON and STRING alone is sufficient to determine the fret position to use. However, in pilot studies we found the inclusion of FRET makes the model converges faster at the training time.

Specifically, instead of a 3-tuple representation of a note as the case in MIDIs, we use a 5-tuple note representation that consists of successive tokens of NOTE-VELOCITY, NOTE-ON, NOTE-DURATION, STRING and FRET for TABs. As such five tokens always occur one after another in the training sequences, it is easy for a Transformer not to miss any of them when generating a new NOTE-ON event at the inference time, according to our empirical observation of the behavior of the Transformers.
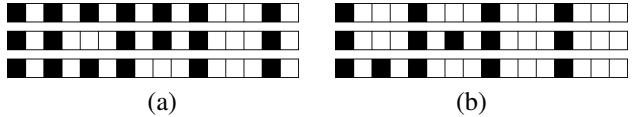
However, as we do not impose constraints on the association between NOTE-ON and STRING, it remains to be studied whether a Transformer can learn to compose tabs with reasonable note-string combinations. This is the subject of the **1st RQ** outlined in Section 1.

As for the TECHNIQUEs, we consider the following five right-hand techniques: slap, press, upstroke, downstroke, and hit-top, which account for ∼1% of the events in our training set. The inclusion of other techniques, such as sliding and bending, is left as a future work.

Similar to [19, 20], we consider the 16th note as the resolution of onset times, which is okay for 4/4 time signature. Increasing the resolution further to avoid quantization errors and to enhance expressivity is also left to the future.

4.3 Groove Modeling

Groove can be in general considered as a rhythmic feeling of a changing or repeated pattern, or "humans' pleasurable

---

[2] Huang *et al.* [20] actually considered the Chord and Tempo events additionally; we found these two types of event less useful in modeling tabs, according to preliminary experiments.

(a)   (b)

Figure 3. Samples of 16-dim hard grooving patterns assigned to 2 different clusters (a), (b) by $k$means clustering.

urge to move their bodies rhythmically in response to music" [36]. Unlike the note-related or time-related events, groove is usually *implicitly* implied as a result of the arrangement of note onsets over time, instead of be explicitly specified in either a MIDI or TAB file. Hence, it might be possible for a Transformer to learn to compose music with reasonable groove, without we *explicitly* inform it what groove is. We refer to this baseline variant of our Transformer as the **no grooving** version, which considers all the events listed in Table 2 but GROOVING.

However, as a tab is now represented as a sequence of events, it is possible to add groove-related events to help the model make sense of this phenomenon. Since our event representation has the BAR events to mark the bar lines, we can ask the model to learn to generate a "bar-level" GROOVING event right after a BAR event, before proceeding to generate the actual content of the bar. Whether such a groove-aware approach benefits the quality of the generated tabs is the subject of our **2nd RQ**.
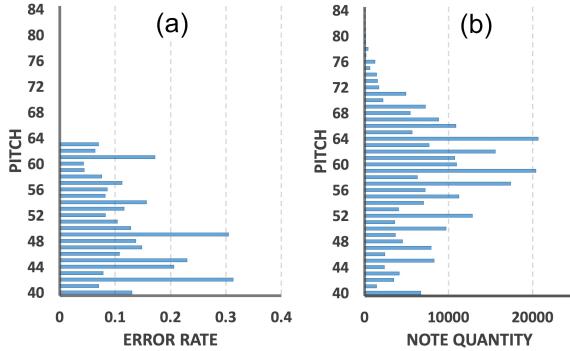
To implement such an approach, we need to come up with 1) a bar-level grooving representation of symbolic music, and 2) a method to convert the grooving representation, which might be a vector, to a discrete event token.

In this work, we represent groove by the *occurrence of note onset* over the 16-th time grid, leading to the following four grooving representations of music.

- **Hard grooving**: A 16-dim binary vector marking the presence of (at least one) onset per each 16 positions of a bar. A popular pattern in our dataset, for example, is $[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]$, meaning onsets on beats only.

- **Soft grooving**: A soft version that considers the number of onsets (but disregarding the velocity values) for each position, normalized by the maximum in the bar, leading to a 16-dim real-valued vector.

- **Multi-resolution hard (or soft) grooving**: Variants of the last two that additionally consider corresponding down-sampled 8-dim and 4-dim vectors to emphasize the beats (e.g., counting only the onsets on beats), and then concatenate the vectors together, yielding a 28-dim vector (i.e., 16+8+4).

To convert the aforementioned grooving patterns to events, a discretization is needed. Among various possible approaches, we experiment with the simplest idea of grouping the grooving patterns seen in the training set into a number of clusters. We can then use the ID of the cluster a grooving pattern is associated with for the GROOVING event of that grooving pattern. For simplicity, we employ the classic $k$means algorithm [17] here, setting k to 32. Please see Figure 3 for an example of the clustering result.

Figure 4. Distributions of (a) the error rate for each note in the string arrangement prediction of our model, and (b) the counts of each note in the training set.

	string (high-pitched ↔ low-pitched)					
	1st	2nd	3rd	4th	5th	6th
(a) accuracy	100%	99%	97%	94%	91%	90%
(b) pitch 42	∼0%	∼0%	10%	∼0%	27%	63%
(c) pitch 57	∼0%	6%	65%	26%	∼0%	∼0%
(d) pitch 69	85%	14%	∼0%	∼0%	∼0%	∼0%

Table 3. (a) The average accuracy of our model in associating each `STRING` with a `NOTE-ON`, broken down by string; (b–d) The string-relevant output probability estimated by our model for three different pitches.

4.4 Transformer-XL-based Architecture

Following [14, 20], we use the Transformer-XL [12] for the architecture of our model. Unlike the Transformer used in [19], the Transformer-XL gains a longer receptive filed with a segment-level recurrence mechanism, thereby seeing further into the history and benefiting from the extended memory. We base our implementation on the open source code of [20], adopting many of their settings. For example, we also set the sequence length and recurrence length to 512 events, and use 12 self-attention layers and 8 attention heads. The model has in total ∼41M trainable parameters. The training process converges within 12 hours on a single NVIDIA V100 GPU, with batch size 32.

5. EVALUATION

5.1 Experiment 1: On Fingering

The 1st RQ explores how a Transformer learns the association between notes and fingering, without human-assigned prior knowledge/constraints on the association. For simplicity, we use the **no grooving** variant of our model here.

A straightforward approach to address this RQ is to let the model generates randomly a large number of event sequences (i.e., compositions) and examine how often it generates a plausible `STRING` event after a `NOTE-ON` event. Table 3(a) shows the average note-string association accuracy calculated from 50 generated 16-bar tabs, broken down into six values according to `STRING`. To our mild disappointment, the accuracy, though generally high, is not

	Hard accuracy ↑		Soft distance ↓	
	mean	max	mean	min
hard grooving	76.2%	82.4%	56.3	44.6
soft grooving	76.9%	83.0%	**56.2**	**43.7**
multi-hard	**79.0%**	**85.7%**	57.8	44.3
multi-soft	74.6%	81.1%	64.7	52.9
no grooving	70.0%	80.1%	58.6	47.7
training data	**82.1%**	**89.5%**	**43.8**	**28.6**
random	64.9%	71.3%	70.6	59.6

Table 4. Objective evaluation on groove coherence.

perfect. This indicates that some post-processing is still needed to ensure the note-string association is correct.

As Table 3(a) shows larger errors toward the 6th string, we also examine how the errors distribute over the pitches. Interestingly, Figure 4(a) shows that the model makes mistakes only in the low end; the fingering prediction is good for pitches (i.e., MIDI numbers) from 64 to 84.

It is hard to find out why exactly this is the case, but we present two more observations here. First, we plot in Figure 4(b) the popularity of these pitches in the training set. The Pearson correlation coefficient between the note quantity and the error rate is weak, at 0.299, suggesting that this may not be due to the sparseness of the low-pitched notes. Second, we show in Table 3(b)–(d) the note-string association output probability estimated by our model for three different pitches. Interestingly, it seems the model has the tendency to use neighboring strings for each pitch. For example, pitch 42 is actually a bass note playable on the 6th string, and it erroneously "leaks" mostly to the 5th string.

5.2 Experiment 2: On Groove
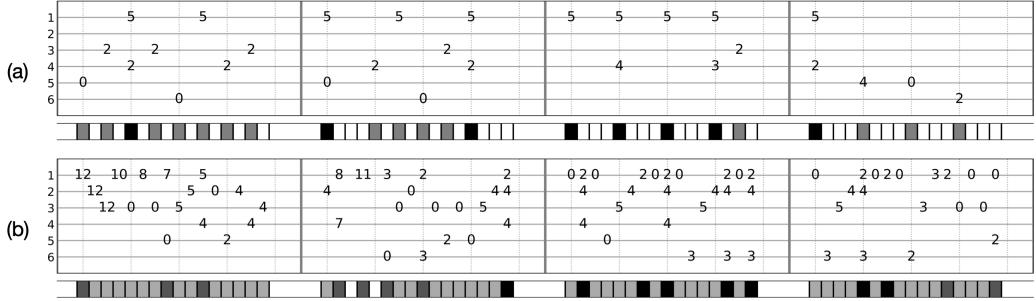
Figure 5 gives two examples of tabs generated by the hard grooving model. It seems the grooving is consistent across time in each tab. But, how good it is?

The 2nd RQ tests whether the added `GROOVING` events help a Transformer compose tabs with better rhythmic coherence. We therefore intend to compare the performance of models trained with or without `GROOVING` for generating "continuations" of a given "prompt."
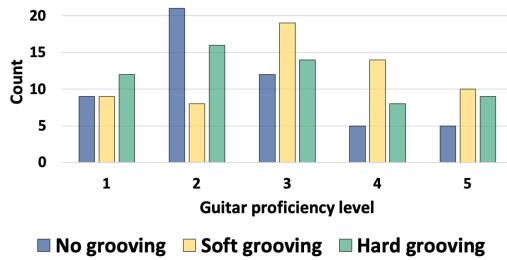
We consider both objective and subjective evaluations here. For the former, we compare the models trained with `GROOVING` events obtained with each of the four vector-quantized grooving representations described in Section 4.3. We ask the models to generate 16-bar continuations following the first 4 bars of the 30 tabs in the validation set. The performance of the models is compared against that of the 'no-grooving' baseline, the 'real' continuations (of these 30 tabs), and a 'random' baseline that picks the next 16 bars from another tab at random from the validation set. The last two are meant to set the high-end and low-end performances, respectively. For fair comparison, we also project the note onsets of the validation data onto the 16th-note grid underlying our training data.

We consider the following two simple objective metrics:

- **Hard accuracy**: Given the hard grooving patterns $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ of the prompt, and those of the

Figure 5. Segments of 2 tabs randomly generated by the hard grooving model; below each tab—the soft grooving patterns.

Figure 6. Result of the first user study asking subjects to choose the best among the three continuations generated by different models, with or without GROOVING, given a man-made prompt. The result is broken down according to the self-report guitar proficiency level of the subjects.

continuation $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_M)$, where both $\mathbf{x}_i$ and $\mathbf{y}_j$ are in $\{0,1\}^K$, $N = 4$, $M = 16$, $K = 16$, we compare the similarity between $\mathbf{X}$ and $\mathbf{Y}$ by

$$\text{mean}_{i=(1,\ldots,N)} \frac{1}{MK} \sum_{j=1}^{M} \sum_{k=1}^{K} \text{XNOR}(x_i^{(k)}, y_j^{(k)}), \tag{1}$$

where $\text{XNOR}(\cdot, \cdot)$ returns, element-wisely, whether the k-th element of $\mathbf{x}_i$ and $\mathbf{y}_j$ are the same. Alternatively, we replace the mean aggregator by $\max$, to say it is good enough for $\mathbf{y}_j$ to be close to any $\mathbf{x}_i$.

- **Soft distance**: We consider instead the soft grooving patterns $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_j$, and compute the distance between them as $\text{mean}_{i=(1,\ldots,N)} \frac{1}{M} \sum_{j=1}^{M} \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_j\|_2^2$. We can similarly replace mean by the $\min$ function.

Table 4 shows that, consistently across different metrics, groove-aware models outperform the no-grooving model. Moreover, the scores of the groove-aware models are closer to the high end than to the low end. It is also important to note that, there is still a moderate gap between the best model's composition and the real data, which has to be further addressed in the future work.

Figure 6 shows the result of the subjective evaluation, where we present the audio rendition (using a guitar synthesizer) of the aforementioned 16-bar continuations to human listeners, and ask them to choose the one they like the most, among those generated by the 'no-grooving,'

	Real	No grooving	Hard grooving
MOS	3.48±1.16	2.80±1.03	3.43±1.12

Table 5. Result of the second user study (in mean opinion score, from 1 to 5) comparing audio renditions of real tabs and machine-composed tabs by two variants of our model.

'soft-grooving,' and 'hard-grooving' models. We divide the response from 57 participants by their self-report proficiency level in guitar. Figure 6 shows that professionals are aware of the difference between groove-aware and no-grooving models. According to their optional verbal response, groove-aware models continue the prompts better, and generate more pleasant melody lines.

5.3 Experiment 3: On Comparison with Real Tabs

Finally, our last RQ involves another user study where we ask participants to rate, on a Likert five-point scale how they like the audio rendition of the continuations, this time including the result of real continuations. For groove-aware models, we consider hard-grooving only, for its simplicity and also for reducing the load on the subjects. Much to our surprise, the average result from 23 participants (see Table 5) suggests that hard-grooving compositions are actually on par with real compositions. We believe this result has to be taken with a grain of salt, as it concerns with only fairly short pieces (i.e., 16 bars) that do not contain performance-level variations. Yet, it provides evidence showing the promise of deep learning for tab composition.

6. CONCLUSION

In this paper, we have presented a series of evaluations supporting the effectiveness of a modern neural sequence model, called Transformer-XL, for automatic composition of fingerstyle guitar tabs. The model still has troubles in ensuring the note-string association and the rhythmic coherence of the generated tabs. How well the model generates tabs of plausible long-term structure is not yet studied. And, much of the expression in guitar music is left unaddressed. Much work are yet to be done to possibly redesign the network architecture and the tab representation. Yet, we hope this work shows promises that inspire more research on this intriguing area of research.

7. REFERENCES

[1] Ample sound. https://www.amplesound.net/en/index.asp.

[2] The BitMidi dataset. https://github.com/feross/bitmidi.com.

[3] Fingerstyle guitar. https://en.wikipedia.org/wiki/Fingerstyle_guitar/.

[4] J. Abeßer, H. Lukashevich, and G. Schuller. Feature-based extraction of plucking and expression styles of the electric bass guitar. In *Proc. IEEE International Conference on Acoustics, Speech, & Signal Processing*, pages 2290–2293, 2010.

[5] J. Abeßer and G. Schuller. Instrument-centered music transcription of solo bass guitar recordings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(9):1741–1750, 2017.

[6] S. Ariga, S. Fukayama, and M. Goto. Song2Guitar: A difficulty-aware arrangement system for generating guitar solo covers from polyphonic audio of popular music. In *Proc. International Society for Music Information Retrieval Conference*, pages 568–574, 2017.

[7] A. M. Barbancho, A. Klapuri, L. J. Tardon, and I. Barbancho. Automatic transcription of guitar chords and fingering from audio. *IEEE Trans. Audio, Speech and Language Processing*, 20(3):915–921, 2012.

[8] J.-P. Briot, G. Hadjeres, and F. Pachet. *Deep Learning Techniques for Music Generation, Computational Synthesis and Creative Systems*. Springer, 2019.

[9] G. Burlet and I. Fujinaga. Robotaba guitar tablature transcription framework. In *Proc. International Society for Music Information Retrieval Conference*, 2013.

[10] Y.-P. Chen, L. Su, and Y.-H. Yang. Electric guitar playing technique detection in real-world recordings based on F0 sequence pattern recognition. In *Proc. International Society for Music Information Retrieval*, 2015.

[11] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel. Encoding musical style with transformer autoencoders. *arXiv preprint arXiv:1912.05537*, 2019.

[12] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.

[13] S. Dixon, F. Gouyon, and G. Widmer. Towards characterisation of music via rhythmic patterns. In *Proc. International Society for Music Information Retrieval*, pages 509–516, 2004.

[14] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley. LakhNES: Improving multi-instrumental music generation with cross-domain pre-training. In *Proc. International Society for Music Information Retrieval*, pages 685–692, 2019.

[15] J. D. Fernández and F. Vico. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48(1):513–582, 2013.

[16] S. Gorlow, M. Ramona, and F. Pachet. Decision-based transcription of Jazz guitar solos using a harmonic bident analysis filter bank and spectral distribution weighting. *arXiv preprint arXiv:1611.06505*, 2016.

[17] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108, 1979.

[18] A. Hrybyk and Y. E. Kim. Combined audio and video analysis for guitar chord identification. In *Proc. International Society for Music Information Retrieval Conference*, 2010.

[19] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music Transformer: Generating music with long-term structure. In *Proc. International Conference on Learning Representations*, 2019.

[20] Y.-S. Huang and Y.-H. Yang. Pop Music Transformer: Beat-based modeling and generation of expressive Pop piano compositions. In *Proc. ACM International Conference on Multimedia*, 2020.

[21] E. J. Humphrey and J. P. Bello. From music audio to chord tablature: Teaching deep convolutional networks toplay guitar. In *Proc. IEEE International Conference on Acoustics, Speech & Signal Processing*, pages 6974–6978, 2014.

[22] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller. Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters. In *Proc. International Conference on Digital Audio Effects*, 2014.

[23] A. Lerch, C. Arthur, A. Pati, and S. Gururani. Music performance analysis: A survey. In *Proc. International Society for Music Information Retrieval Conference*, pages 33–43, 2019.

[24] M. McVicar, S. Fukayama, and M. Goto. AutoLeadGuitar: Automatic generation of guitar solo phrases in the tablature space. In *Proc. International Conference on Signal Processing*, pages 599–604, 2014.

[25] M. McVicar, S. Fukayama, and M. Goto. AutoRhythmGuitar: Computer-aided composition for rhythm guitar in the tab space. In *Proc. International Computer Music Conference*, 2014.

[26] M. McVicar, S. Fukayama, and M. Goto. AutoGuitarTab: Computer-aided composition of rhythm and lead guitar parts in the tablature space. *IEEE/ACM*

Transactions on Audio, Speech, and Language Processing, 23(7):1105–1117, 2015.

[27] J. Michelson, R. M. Stern, and T. M. Sullivan. Automatic guitar tablature transcription from audio using inharmonicity regression and bayesian classification. *Journal of The Audio Engineering Society*, 2018.

[28] E. Mistler. Generating guitar tablatures with neural networks. *Master of Science Dissertation, The University of Edinburgh*, 2017.

[29] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 2018.

[30] G. Papadopoulos and G. Wiggins. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *Proc. AISB Symposium on Musical Creativity*, pages 110–117, 1999.

[31] C. Payne. MuseNet. `https://openai.com/blog/musenet/`, 2019.

[32] G. Peeters. Rhythm classification using spectral rhythm patterns. In *Proc. International Society for Music Information Retrieval*, pages 644–647, 09 2005.

[33] C. Raffel and D. P. W. Ellis. Extracting ground truth information from MIDI files: A MIDIfesto. In *Proc. International Society for Music Information Retrieval*, pages 796–802, 2016. [Online] `https://colinraffel.com/projects/lmd/`.

[34] S. Rodríguez, E. Gómez, and H. Cuesta. Automatic transcription of Flamenco guitar falsetas. In *Proc. International Workshop on Folk Music Analysis*, 2018.

[35] S. I. Sayegh. Fingering for string instruments with the optimum path paradigm. *Computer Music Journal*, 13(3):76–84, 1989.

[36] O. Senn, L. Kilchenmann, T. Bechtold, and F. Hoesl. Groove in drum patterns as a function of both rhythmic properties and listeners' attitudes. *PLOS ONE*, 13:1–33, 06 2018.

[37] L. Su, L.-F. Yu, and Y.-H. Yang. Sparse cepstral and phase codes for guitar playing technique classification. In *Proc. International Society for Music Information Retrieval*, 2014.

[38] T.-W. Su, Y.-P. Chen, L. Su, and Y.-H. Yang. TENT: Technique-embedded note tracking for real-world guitar solo recordings. *International Society for Music Information Retrieval*, 2(1):15–28, 2019.

[39] L. Thompson, S. Dixon, and M. Mauch. Drum transcription via classification of bar-level rhythmic patterns. In *Proc. International Society for Music Information Retrieval*, pages 187–192, 2014.

[40] D. R. Tuohy and W. D. Potter. Guitar tablature creation with neural networks and distributed genetic search. In *Proc. International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 2006.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[42] E. Waite, D. Eck, A. Roberts, and D. Abolafia. Project Magenta: Generating long-term structure in songs and stories, 2016. `https://magenta.tensorflow.org/blog/2016/07/15/lookback-rnn-attention-rnn/`.

[43] A. Wiggins and Y. E. Kim. Guitar tablature estimation with a convolutional neural network. In *Proc. International Conference on Music Information Retrieval*, pages 284–291, 2019.

[44] S.-L. Wu and Y.-H. Yang. The Jazz transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures. In *Proc. International Society for Music Information Retrieval Conference*, 2020.

[45] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello. GuitarSet: A dataset for guitar transcription. In *Proc. International Conference on Music Information Retrieval*, pages 453–460, 2018. [Online] `https://github.com/marl/guitarset/`.

[46] K. Yazawa, D. Sakaue, K. Nagira, K. Itoyama, and H. Okuno. Audio-based guitar tablature transcription using multipitch analysis and playability constraints. In *Proc. IEEE International Conference on Acoustics, Speech & Signal Processing*, pages 196–200, 2013.

A COMPUTATIONAL ANALYSIS OF REAL-WORLD DJ MIXES USING MIX-TO-TRACK SUBSEQUENCE ALIGNMENT

Taejun Kim[1]   **Minsuk Choi**[1]   **Evan Sacks**[2]
Yi-Hsuan Yang[3]   **Juhan Nam**[1]

[1] Graduate School of Culture Technology, KAIST, South Korea
[2] 1001Tracklists, United States
[3] Research Center for IT Innovation, Academia Sincia, Taiwan

`{taejun,minsukchoi,juhan.nam}@kaist.ac.kr, evan@1001tracklists.com, yang@citi.sinica.edu.tw`

ABSTRACT

A DJ mix is a sequence of music tracks concatenated seamlessly, typically rendered for audiences in a live setting by a DJ on stage. As a DJ mix is produced in a studio or the live version is recorded for music streaming services, computational methods to analyze DJ mixes, for example, extracting track information or understanding DJ techniques, have drawn research interests. Many of previous works are, however, limited to identifying individual tracks in a mix or segmenting it, and the sizes of the datasets are usually small. In this paper, we provide an in-depth analysis of DJ music by aligning a mix to its original music tracks. We set up the subsequence alignment such that the audio features are less sensitive to the tempo or key change of the original track in a mix. This approach provides temporally tight mix-to-track matching from which we can obtain cue-points, transition length, mix segmentation, and musical changes in DJ performance. Using 1,557 mixes from *1001Tracklists* including 13,728 tracks and 20,765 transitions, we conduct the proposed analysis and show a wide range of statistics, which may elucidate the creative process of DJ music making.

1. INTRODUCTION

A Disc Jockey (DJ) is a musician who plays a sequence of existing music tracks or sound sources seamlessly by manipulating the audio content based on musical elements. The outcomes can be medleys (mix), mash-ups, remixes, or even new tracks, depending on how much DJs edit the substance of the original music tracks. Among them, creating a mix is the most basic role of DJs. This involves curating music tracks and their sections to play, deciding the order, and modifying them to splice one section to another as a continuous stream. In each step, DJs consider various elements of the tracks such as tempo, key, beat,

chord, rhythm, structure, energy, mood and genre. These days, DJs create the mix not only for a live audience but also for listeners in music streaming services.

Recently, imitating the tasks of DJ using computational methods has drawn research interests [1–7]. On the other hand, efforts have been made to understand the creative process of DJ music making. In the perspective of reverse engineering, tasks extracting useful information from real-world DJ mixes can be useful in such a pursuit. In the literature, at least the following tasks have been studied. (1) *Track identification* [8–10]: identifying which tracks are played in DJ music which can be either a mix or a manipulated track. (2) *Mix segmentation* [11, 12]: finding boundaries between tracks in a DJ mix. (3) *Mix-to-track alignment* [13, 14]: aligning the original track to an audio segment in a DJ mix. (4) *Cue point extraction* [14]: finding when a track starts and ends in a DJ mix. (5) *Transition unmixing* [13, 14]: explaining how DJs apply audio effects to make a seamless transition from one track to another. However, the previous studies only focused on solving the tasks usually with a small dataset and did not provide further analysis using extracted information from the tasks. For example, Sonnleitner et al. [8] used 18 mixes for track identification. Glazyrin [11] and Scarfe et al. [12] respectively collected 103 and 339 mixes with boundary timestamps for mix segmentation. The majority of previous studies concentrated on identification and segmentation and few studies on the other three tasks used artificially generated datasets [13, 14].

To address the need of a large-scale study, we collected in a total of 1,557 real-world mixes and original tracks played in the mixes from *1001Tracklists*, a community-based DJ music service. [1] The mixes include 13,728 unique tracks and 20,765 transitions. However, tracks used in DJ mixes usually include various versions so-called "extended mix", "remix", or "edit". Also, a few tracks in track-lists of the collected dataset are annotated incorrectly by users. Therefore, an alignment algorithm is required to ensure that the collected tracks are exactly the same versions as the ones used in the mixes. More importantly, the alignment will be a foundation for further computational analysis of DJ mixes. With these two motivations, we

---

[1] `https://www.1001tracklists.com`

Summary statistic	All	Matched
The number of mixes	1,564	1,557
The number of unique tracks	15,068	13,728
The number of played tracks	26,776	24,202
The number of transitions	24,344	20,765
Total length of mixes (in hours)	1,577	1,570
Total length of unique tracks (in hours)	1,038	913
Average length of mixes (in minutes)	60.5	60.5
Average length of unique tracks (in minutes)	4.1	4.0
Average number of played tracks in a mix	17.1	15.5
Average number of transitions in a mix	14.5	12.9

Table 1. Statistics of the *1001Tracklists* dataset. The original dataset size is denoted as 'All' and the size after filtering as 'Matched'.

set up the mix-to-track subsequence dynamic time warping (DTW) [15] such that the mix can be aligned with the original tracks in presence of possible tempo or key changes. The warping paths from the DTW provide temporally tight mix-to-track matching from which we can obtain cue points, transition lengths, and key/tempo changes in DJ performances in a quantitative way. To evaluate the performances of the alignment and the cue point extraction methods simultaneously, we evaluate mix segmentation performances regarding the extracted cue points as boundaries dividing two adjacent tracks in mixes, comparing them to human-annotated boundaries. Furthermore, by observing the performance changes depending on the three different types of cue points, we analyze the human annotating policy of track boundaries.

Although DJ techniques are complicated and different depending on the characteristics of tracks, there has been common knowledge for making seamless DJ mixes. However, to the best of our knowledge, the domain knowledge has never been addressed in the literature with statistical evidence obtained by computational analysis. In this study, we analyze the DJ mixes using the results from the subsequence DTW mentioned above for the following hypotheses: 1) DJs tend not to change tempo and/or key of tracks much to avoid distorting the original essence of the tracks. 2) DJs make seamless transitions from one track to another considering the musical structures of tracks. 3) DJs tend to select cue points at similar positions in a single track.

The analysis is performed based on the results obtained from the subsequence alignment and provides insights statistically for tempo adjustment, key transposition, track-to-track transition lengths, and agreements of the cue points among DJs. We hope that the proposed analysis and various statistics may elucidate the creative process of DJ music making. The source code for the mix-to-track subsequence DTW, the cue point analysis and the mix segmentation is available at the link. [2]

2. THE DATASET

Our study is based on DJ music from *1001Tracklists*. We obtained a collection of DJ mix metadata via direct personal communication with *1001Tracklists*. Each entry of

mixes contains a list of track, boundary timestamps and genre. It also contains web links to the audio files of the mixes and tracks. We downloaded them separately from the linked media service websites on our own. We found a small number of web links to tracks are not correct and so filtered them out by a mix-to-track alignment method automatically (see Section 3.3). The boundary timestamps of tracks in a mix are annotated by the users of *1001Tracklists*.

Table 1 summarizes statistics of the dataset. The original size of the dataset is denoted as 'All' and the size after filtering as 'Matched' in Table 1. Note that the number of played tracks is greater than the number of unique tracks as a track can be played in multiple mixes. The dataset includes a variety of genres but mostly focuses on House and Trance music. More detailed statistics of the dataset are available on the companion website. [3]

3. MIX-TO-TRACK SUBSEQUENCE ALIGNMENT

The objective of mix-to-track subsequence alignment is to find an optimal alignment path between a subsequence of a mix and a track used in the mix. This alignment result will be the basis of diverse DJ mix analysis concerning the cue point, track boundary, key/tempo changes and transition length. We also use it for removing non-matching tracks. This section describes the detail of computational process.

3.1 Feature Extraction

When DJs create a mix, they often adjust tempo and/or key of the tracks in the mix or add audio effects to them. Live mixes contain more changes in timbre and even other sound sources such as the voices from the DJ. In order to address the acoustic and musical variations between the original track and the matched subsequence in the mix, we use beat synchronous chroma and mel-frequency cepstral coefficients (MFCC). The beat synchronous feature representations enable tempo invariance and dramatically reduces the computational cost in the alignment. The aggregation of the features from the frame level to the beat level also smooths out local timbre variations. The chroma feature, on the other hand, facilitates key-invariance as circular shift of the 12-dimensional vector corresponds to key transposition. The MFCC feature captures general timbre characteristics. We used Librosa [4] to extract the chroma and MFCC features with the default options except that the dimentionality of MFCC was set to 12 and the type of chroma was to chroma energy normalized statistics (CENS) [16].
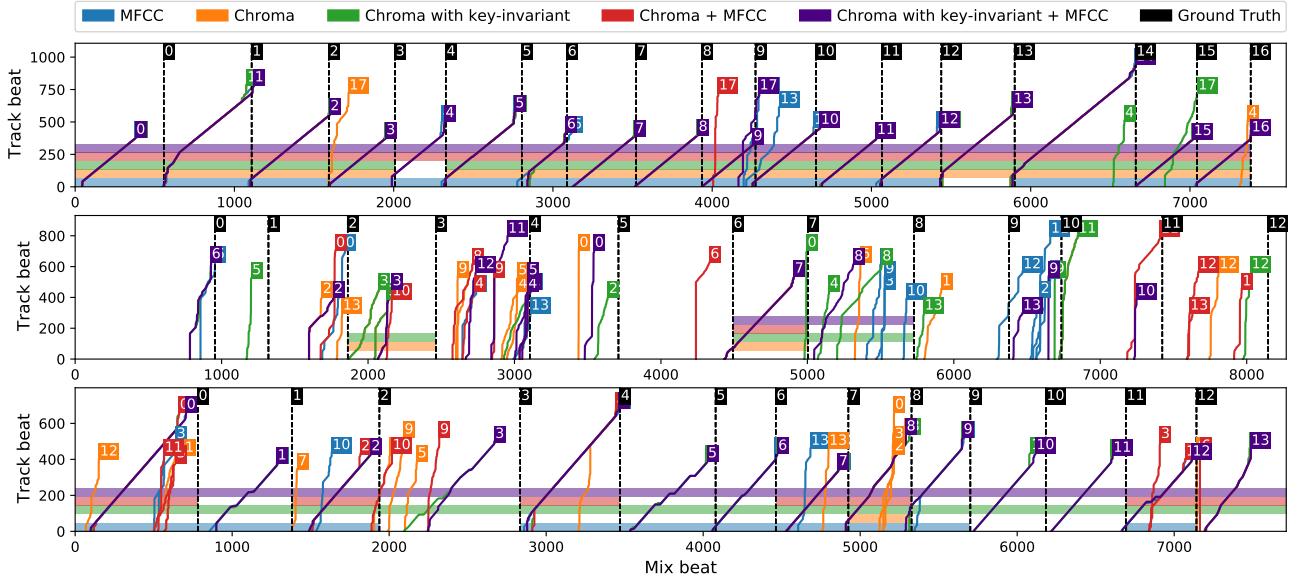
3.2 Key-Invariant Subsequence DTW

We compute the alignment by applying subsequence DTW to the beat synchronous features [15]. We used an implementation from Librosa, adopting the transposition-invariant approach from [17]. Specifically, we calculated 12 versions of chroma features by performing all possible

---

Figure 1. Visualizations of the result of a DTW-based mix-to-track subsequence alignment between a mix and the original tracks played in that mix. The colored solid lines show the warping paths of the alignment depending on the input feature, and whether or not applying the transposition-invariant method on the subsequence DTW. The tagged numbers on warping paths and ground truth boundaries indicate played and timestamped indices in the mix, respectively. A colored bar at the bottom of the figures is added if the alignment of the method is considered successful according to the match rate. (Top) A correctly matched example. (Middle) An unsuccessful example, due to the low sound quality of the mix. (Bottom) The alignment can be improved using the key-invariant chroma. Best viewed in color.

circular shifts on the original track side and select the one with the lowest matching cost in the subsequence DTW. This result returns not only the optimal alignment path but also the key transposition value of the original track.

Figure 1 shows three examples of the alignment results when different combinations of features (MFCC, chroma, and key-invariant chroma) are used. When the alignment path of the subsequence satisfies a match rate (described in Section 3.3), we put a color strip corresponding to each feature in the bottom of the figure. Since we use beat synchronous representations for them, the warping paths become diagonal with a slope of one if a mix and a track are successfully aligned. The top panel in the figure shows an successfully aligned example for the most of tracks and features where all warping paths have straight diagonal paths.[5] The middle panel shows a failing example because sounds from crowds are also recorded in the mix.[6] The bottom panel shows a example where chroma with circular shift distinctively works better others as the DJ frequently uses key transposition on the mix.[7]

3.3 Filtering Using Match Rates

As stated above, we can measure the quality of the alignment from the warping path. Ideally, when every single move on the path is diagonal, that is, one beat at a time for both track and mix axis, we will obtain a perfect straight diagonal line. However, the acoustic and musical changes deform the path. We define the ratio of the diagonal moves

in a mix (one move per beat) as the *match rate* and use it for filtering out incorrectly annotated tracks. We experimentally chose 0.4 as a threshold. The size of the dataset after the filtering is denoted as "Matched" in Table 1. We only use the matched tracks for the analysis in this paper.

4. CUE POINT EXTRACTION

Cue points are timestamps in a track that indicate where to start and end the track in a mix. Determining the cue points of played tracks is an essential task of DJ mixing. This section describes extracting cue points using the warping paths obtained from the aforementioned mix-to-track subsequence alignment.

4.1 Term Definitions

We first define terms related to cue points. In the context of the track-to-track transition, a *cue-out* point is a timestamp that the previous track starts fading out and the next track starts fading in, and a *cue-in* point is when the previous track is fully faded out and only the next track is being played. The *transition* region is defined as the time interval from the cue-out point of the previous track to the cue-in point of the next track. Additionally, we define a *cue-mid* point as the middle of a transition, which can technically be considered as a boundary of the transition.
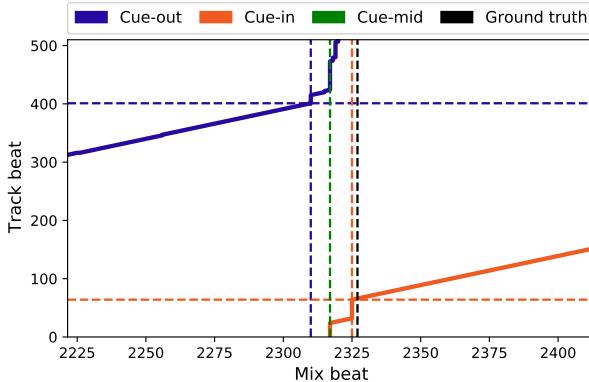
4.2 Methods

The mix-to-track alignment results naturally yield cue points of matched tracks. Figure 2 shows an example of extracted cue points (a zoomed-in view of the top figure in

---

[5] `https://1001.tl/14jltnct`
[6] `https://1001.tl/15fulzc1`
[7] `https://1001.tl/bcx2z0t`

Figure 2. A zoomed-in view of a visualization of mix-to-track subsequence alignment explaining the three types of extracted cue points. The two solid lines indicate warping paths representing alignment between the mix and tracks. The vertical colored dotted lines represent the extracted cue points on the mix and the horizontal dotted lines represent the points on each track. The vertical black dotted line is a human-annotated ground truth boundary between the two tracks. The solid lines are the fourth and fifth warping paths from the top of Figure 1. Best viewed in color.

Figure 1). The two alignment paths drift from the diagonal lines in the transition region (between 2310 and 2324 in mix beat) because the two tracks cross-fades. Based on this observation, we detect the cue-out point of the previous track by finding the last beat where preceding 32 beats have diagonal moves in the alignment path. Likewise, we detect the cue-in point of the next track by finding the first beat where succeeding 32 beats have diagonal moves in the alignment path.

5. MIX SEGMENTATION

The goal of mix segmentation is to divide a continuous DJ mix into individual tracks, which can enhance the listening experience and can be a foundation of further analysis or learning of DJ mixes. Since DJs make seamless transitions, it is difficult to notice that a track is fading in or out. To quantitatively measure how difficult it is, a study analyzed how accurate humans are at creating the boundary timestamps and found that the standard deviation of the human disagreement for track boundaries in mixes is about 9 seconds, which implies it is difficult to find the optimal boundaries even for humans [12]. Furthermore, the ambiguous definition of the boundary and long lengths of transitions makes it difficult to annotate the boundary timestamps [8].

5.1 Cue Point based Estimation

Given the extracted cue point so far, we can estimate the track boundaries with three possible choices. The first is the position that the next track fully appears (cue-in point), the second is the position that previous track starts to disappear (cue-out point), and the last is the middle of the transition (cue-mid point). By comparing each of them
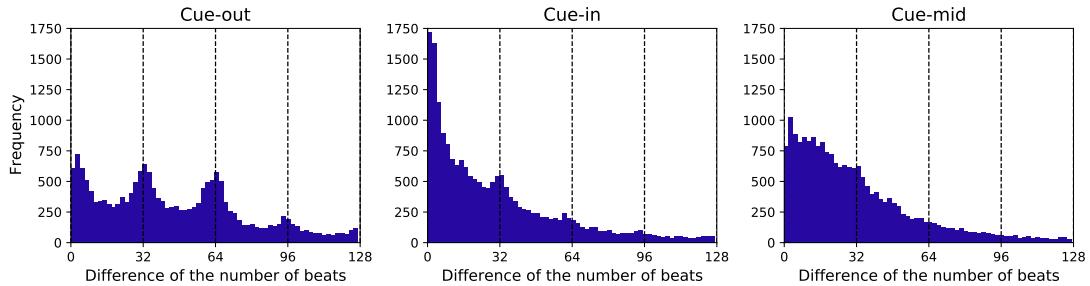
with human-annotated boundary timestamps, we can measure which type of cue point humans tend to consider as a boundary.

Figure 3 shows three histograms where each of them is computed from the differences between human-annotated boundary timestamps and one of the cue point types in beat unit. The overall trend shows that the distribution of cue-in point is mostly skewed towards zero. Interestingly, the distribution of cue-out point has more distinctive peaks around every 32 beat than the distribution of cue-in point. Considering the histogram of the transition length has peaks at every 32 beat as shown in Figure 6, this reflects that human annotators tend to label cue-in points as a boundary compared to cue-out (note that the transition length is computed by subtracting the cue-in point from the cue-out point). On the other hand, the distribution of cue-mid point has a gradually decreasing curve without peaks. While this distribution looks like having better estimates than the cue-out point, Table 3 shows an opposite result. That is, in terms of the number of cue points closest to the human annotations, the cue-out point is the second and the cue-mid point is the worst among the three types. These results indicate that the cue-mid point is a safe choice. That is, although the cue-mid point is least likely to be a boundary as shown in Table 3, the difference between the estimate and human annotation is relatively small because it is the middle of the transition region.

Table 2 shows the difference between human-annotated boundary timestamps and one of the cue point types in terms of median time (in seconds) on the left side. The overall trend confirms that the cue-in is the best estimate of track boundary and the cue-mid is a safer choice than the cue-out. The table also shows the result of "cue-best". This is computed with the minimum difference among the three cue point types for each of the transition region. The result shows that the median time differences are dramatically decreased to 4-5 seconds. Table 2 also shows the difference between human-annotated boundary timestamps and the cue-in point in terms of hit rates on the right side. The hit rates are computed the ratio of correct estimates given a tolerance window. If the estimate is within the tolerance window on the human-annotated boundary timestamp, it is regarded as a correct estimate. We set three tolerance windows (15, 30, and 60 seconds) considering that the average tempo of tracks in the dataset is 127 beat per minute (BPM) and then the tolerance windows approximately correspond to 32, 64, 128 beats (multiples of a phrase unit). The result shows that the best hit rate with the 30 second window (about 64 beats) is above 80%. Given the long transition time as shown in Figure 6, the cue-in point may be considered as a reasonable choice.

5.2 Effect of Audio Features

Table 2 also compares the median time difference between human-annotated boundary timestamps and one of the cue point types for different audio features used in the subsequence DTW. In general, the chroma features are a better choices than MFCC (p-value of t-test < 0.001 for chroma

Figure 3. Histograms of distances to ground truth boundaries in the number of beats depending on the type of the cue point. The dotted lines are plotted at every 32 beats which is usually considered as a phrase in the context of dance music.

Feature	Median time difference (in seconds)				Cue-in hit rate		
	Cue-out	Cue-in	Cue-mid	Cue-best†	15 sec	30 sec	60 sec
MFCC	27.92	14.27	13.55	5.340	0.5187	0.7591	0.9023
Chroma	23.85	11.80	12.33	**4.230**	0.5837	0.7973	0.9286
Chroma with key-invariant	23.87	11.77	12.37	4.240	0.5843	0.7968	0.9282
Chroma + MFCC	23.41	11.48	**12.16**	4.380	0.5866	0.8035	0.9284
Chroma with key-invariant + MFCC	**23.38**	**11.40**	**12.16**	4.380	**0.5881**	**0.8040**	**0.9288**

Table 2. Mix segmentation performances depending on the type of cue point and the input feature used to obtain the warping paths. Median time differences between cue points and ground truths are shown on the left side and hit rates of cue-in points with thresholds in seconds are shown on the right side. "Key-invariant" indicates applying the key transposition-invariant method for the DTW. The best score of each criteria is shown in **bold**. $\dagger$ indicates the scores are computed using the best score among the three cue types.

Cue-out	Cue-in	Cue-mid
6,151 (30%)	10,844 (52%)	3,770 (18%)

Table 3. The number of ground truth boundary timestamps closest to the type of cue point.

with or without key-invariant). When both of chroma and MFCC are combined, the median time difference slightly reduces but it is statistically insignificant (p-value of t-test > 0.1). However, we observed that the subsequence DTW does not work well for some genres such as Techno which only contain drum and ambient sounds. This might can be improved by using MFCCs with a large number of bins or using mel-spectrograms. The use of key-invariant chroma generally does not make much difference because key transposition does not performed frequently as discussed in Section 6.2.

6. MUSICOLOGICAL ANALYSIS OF DJ MIXES

We hypothesize that DJs share common practices in the creative process in terms of tempo change, track-to-track transition, and cue point selection. In this section, we validate them using the results from the mix-to-track subsequence alignment and the cue point extraction.
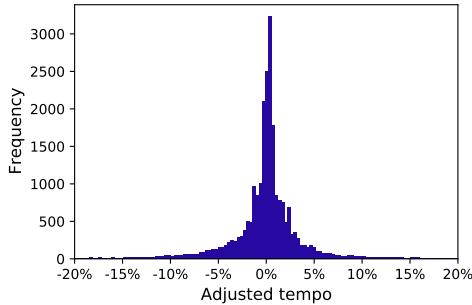
6.1 Tempo Adjustment

We compare the estimated tempo of the original track to the tempo of each audio segment where the track is played in a mix. Figure 4 shows a histogram of percentage dif-

ferences of tempo between the original track and the audio segment in the mix. For example, a difference of 5% indicates the tempo of the original track is increased by 5% while played in the mix. As shown in the histogram, the adjusted tempo has an double exponential distribution, which means the adjusted tempo values are skewed towards zero. In detail, 86.1% of the tempo are adjusted less than 5%, 94.5% are less than 10%, and 98.6% are less than 20%. If one implements an track identification system for DJ mix that is robust to tempo adjustment, this distribution could be a reference.
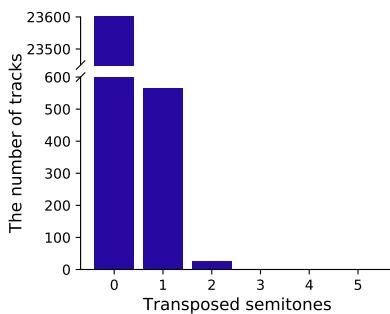
6.2 Key Transposition

A function so-called "master tempo" or "key lock" that preserves pitch despite tempo adjustments is activated by default in modern DJ systems such as stand-alone DJ systems, DJ softwares, and even turntables for vinyl records. Therefore, key transposition is usually performed when a DJ intentionally wants to change the key of a track. As mentioned in Section 3.2, the transposition-invariant DTW can provide the number of transposed semitones as a by-product. We computed the statistics of key transposition using them (using DTW taking both MFCCs and key-invariant chroma). Figure 5 shows a histogram of key transposition between the original track and the audio segment in the mix. Only 2.5% among the total 24,202 tracks are transposed and, among those transposed tracks, 94.3% of them are only one semitone transposed. This result indicates that DJs generally do not perform key transposition much and leave the "master tempo" function turned on in most cases.

Figure 4. A histogram of adjusted tempo of tracks in mixes.
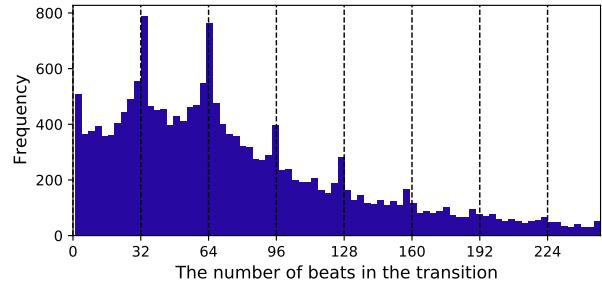
Figure 5. The number of tracks depending on the number of semitones in mixes.

Figure 6. A histogram of the transition lengths in number of beats. The dotted lines are plotted at every 32 beats.

Figure 7. A histogram of distances between cue points of a single track in the number of beats.

6.3 Transition Length
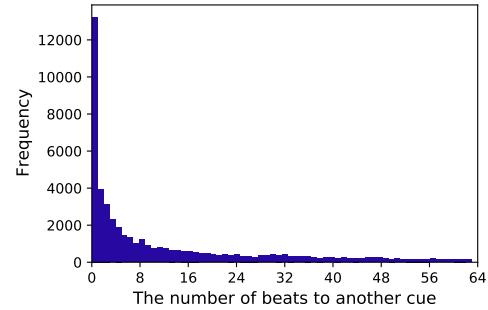
Once we extract cue-in and cue-out points in the transition region, we can calculate the transition length. This can provide some basic hints on how DJ makes the track-to-track transition in a mix. Figure 6 shows a histogram of transition lengths in the number of beats. We annotated the dotted lines every 32 beat which is often considered as a phrase in the context of dance music. The histogram has peaks at every phrase. This indicates that DJs consider the repetitive structures in the dominant genres of music when they make transitions or set cue points.

6.4 Cue Point Agreement among DJs

Deciding cue points of played tracks is a creative choice in DJ mixing. Observing the agreement of cue points on a single track among DJs may elucidate the possibility of finding some common rules. To the end, we collected all extracted cue points for each track and computed the statistics of deviations in cue-in points and cue-out points among DJs. Specifically, we computed all possible pairs and their distances separately for cue-in points and cue-out points. Since the two distributions were almost equal, we combined them into a single distribution in Figure 7. From the results, 23.6% of the total cue point pairs have zero deviation. 40.4% of them were within one measure (4 beats), 73.6% were within 8 measures and 86.2% were within 16 measures. This indicates that there are some rules that DJs share in deciding the cue points. It would be interesting to perform detailed pattern analysis to estimate the cue points using this data in future work.

7. CONCLUSIONS

We presented various statistics and analysis of 1,557 real-world DJ mixes from *1001Tracklists*. Based on the mix-to-track subsequence DTW, we conducted cue point analysis of individual tracks in the mixes and showed the possibility of common rules in the music making that DJs share. We also investigated mix segmentation by comparing the three types of cue point to human-annotated boundary timestamps and showed that humans tend to recognize cue-in points of the next tracks as boundaries. Finally, we showed the statistics of tempo and key changes of the original tracks in DJ performances. We believe this large-scale statistical analysis of DJ mixes can be beneficial for computer-based research on DJ music. The cue point analysis can be the ground for the precise definition of cue points and the tempo and key analysis can provide a guideline of the musical changes during the DJ mixing.

As a future work, we plan to estimate cue points within a track as a step towards automatically generating a mix [3, 4]. The cue point estimation has many application such as DJ software and playlist generation on music streaming services. This will require structure analysis or segmentation of a single music track, which is an important topic in MIR. Furthermore, we plan to analyze the transition region in a mix to investigate DJ mixing techniques. For example, it is possible to estimate the gain changes in the cross-faded region by comparing the two adjacent original tracks and the mix [13, 14]. The methods can be extended to the spectrum domain. Such detailed analysis of mixing techniques will allow us to understand how DJs seamlessly concatenate music tracks and provide a guide to develop automatic DJ systems.

8. ACKNOWLEDGEMENT

9. REFERENCES

[1] Y.-T. Lin, C.-L. Lee, J.-S. Jang, and J.-L. Wu, "Bridging music via sound effects," in *2014 IEEE International Symposium on Multimedia.* IEEE, 2014, pp. 116–122.

[2] R. M. Bittner, M. Gu, G. Hernandez, E. J. Humphrey, T. Jehan, H. McCurry, and N. Montecchio, "Automatic playlist sequencing and transitions," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 442–448.

[3] D. Schwarz, D. Schindler, and S. Spadavecchia, "A heuristic algorithm for DJ cue point estimation," in *Proc. Sound and Music Computing (SMC) Conference*, 2018.

[4] L. V. Veire and T. De Bie, "From raw audio to a seamless mix: creating an automated DJ system for drum and bass," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2018, no. 1, p. 13, 2018.

[5] A. Kim, S. Park, J. Park, J.-W. Ha, T. Kwon, and J. Nam, "Automatic DJ mix generation using highlight detection," in *International Society for Music Information Retrieval Conference (ISMIR), Late-Breaking Paper*, 2017.

[6] Y.-S. Huang, S.-Y. Chou, and Y.-H. Yang, "Generating music medleys via playing music puzzle games," in *Proc. AAAI Conference on Artificial Intelligence*, 2018.

[7] ——, "DJnet: A dream for making an automatic DJ," in *International Society for Music Information Retrieval Conference (ISMIR), Late-Breaking Paper*, 2017.

[8] R. Sonnleitner, A. Arzt, and G. Widmer, "Landmark-based audio fingerprinting for DJ mix monitoring." in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 185–191.

[9] P. S. Manzano, "Audio fingerprinting techniques for sample identification in electronic music," Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany, 2016.

[10] P. Lopez Serrano Erickson, "Analyzing sample-based electronic music using audio processing techniques," Ph.D. dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany, 2019.

[11] N. Glazyrin, "Towards automatic content-based separation of DJ mixes into single tracks," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 149–154, soure code available at [Online] https://github.com/nglazyrin/MixSplitter.

[12] T. Scarfe, W. Koolen, and Y. Kalnishkan, "Segmentation of electronic dance music," *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, vol. 22, no. 3, p. 4, 2014, soure code available at [Online] https://github.com/ecsplendid/DanceMusicSegmentation.

[13] L. Werthen-Brabants, "Ground truth extraction & transition analysis of DJ mixes," Master's thesis, Ghent University, Ghent, Belgium, 2018.

[14] D. Schwarz and D. Fourer, "Methods and datasets for DJ-mix reverse engineering," in *Proc. International Symp. on Computer Music Multidisciplinary Research (CMMR)*, 2019, pp. 426–437.

[15] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications.* Springer, 2015, ch. 7.2.3. Subsequence DTW.

[16] M. Müller and S. Ewert, "Chroma toolbox: MATLAB implementations for extracting variants of chroma-based audio features," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[17] M. Müller and M. Clausen, "Transposition-invariant self-similarity matrices." in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2007, pp. 47–50.

Paper Session 6

MODELING AND ESTIMATING LOCAL TEMPO: A CASE STUDY ON CHOPIN'S MAZURKAS

Hendrik Schreiber, Frank Zalkow, Meinard Müller

International Audio Laboratories Erlangen, Germany

{hendrik.schreiber,frank.zalkow,meinard.mueller}@audiolabs-erlangen.de

ABSTRACT

Even though local tempo estimation promises musicological insights into expressive musical performances, it has never received as much attention in the music information retrieval (MIR) research community as either beat tracking or global tempo estimation. One reason for this may be the lack of a generally accepted definition. In this paper, we discuss how to model and measure local tempo in a musically meaningful way using a cross-version dataset of Frédéric Chopin's Mazurkas as a use case. In particular, we explore how tempo stability can be measured and taken into account during evaluation. Comparing existing and newly trained systems, we find that CNN-based approaches can accurately measure local tempo even for expressive classical music, if trained on the target genre. Furthermore, we show that different training–test splits have a considerable impact on accuracy for difficult segments.

1. INTRODUCTION

While *global* tempo is well defined for music with little or no tempo variability [1], this is less so the case for *local* tempo, especially for expressive classical music. Composer markings like *rubato* (expressive, local tempo change) or *ritardando* (slow down) indicate continuous or even abrupt tempo changes, leading to one or more segments with stable tempi and segments of tempo instability in between. Figure 1, for example, shows tempo markings for Frédéric Chopin's Mazurka Op. 68, 3 (details are discussed in Section 2). Naïvely, one may model local tempo for such a piece as one of two extremes: at the *micro level*, as an instantaneous value, e.g., as the Inter Beat Interval (IBI) between two consecutive beats, or at the *macro level*, by averaging the number of beats over a longer period of time. For expressive music, both approaches have disadvantages. IBIs exhibit a large variance, and averaging beat counts may underestimate the tempo, because expression leads more often to longer than shorter IBIs [2]. Repp therefore attempts to find a definition for the *basic tempo* [3], i.e., the implied tempo the instantaneous tempo
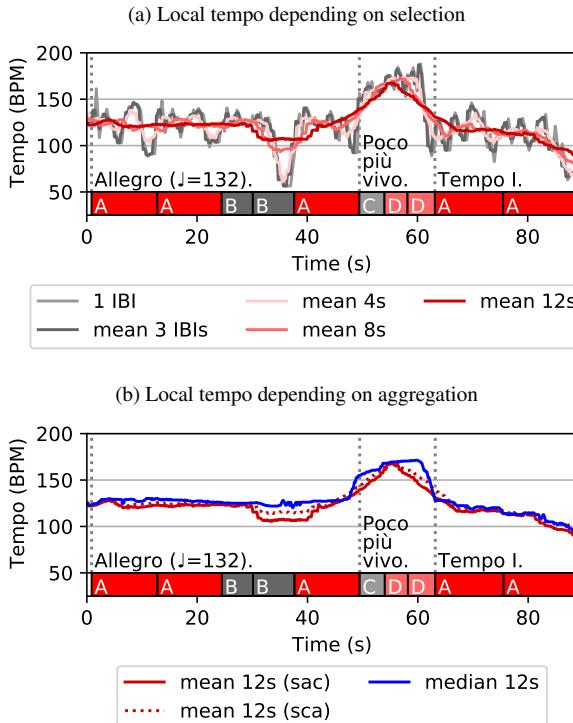
Work	Measures	Beats	Recordings
Op. 17, 4	132	396	62
Op. 24, 2	120	360	64
Op. 30, 2	65	193	34
Op. 63, 3	77	229	88
Op. 68, 3	61	181	50

Table 1: Dataset overview [13]: Number of measures, beats, recordings for five Chopin Mazurkas.

varies around. In [2], he suggests to derive the basic tempo from the first quartile of eighth-note Inter Onset Intervals (IOIs). Similarly, Dixon [4] proposes IOI clustering, using centroids as tempo hypotheses. Grosche and Müller [5] propose yet another approach by defining local tempo as the mean of three consecutive IBIs, which is identical to using Inter Measure Intervals (IMIs) for pieces in ¾ time. The same method is also used by Chew and Callender [6]. In summary, local tempo is usually modeled by aggregating local pulse information, but there appears to be no clear consensus on how. Even though local tempo estimates are popular intermediate features for beat trackers (e.g., [7,8]), few works explicitly estimate and evaluate local tempo estimates. Peeters [9] simply measures whether 75% of the estimated local tempi match the annotated global tempo. In subsequent work [10], he compares the median of local tempi with a global ground truth. A similar approach is taken in [11]—after beat tracking, the median IBI is used as global tempo and then evaluated. Similar to global tempo evaluation, Grosche and Müller [5] compute the accuracy of their IMIs allowing a 4% tolerance and certain integer factors. Schreiber and Müller [12] only provide visualizations for local tempo estimates. To our knowledge, there is no commonly accepted evaluation procedure. Even less researched than local tempo is tempo *stability*, usually only referred to as a precondition for global tempo estimation [1]. Grosche et al. [13] mention that beat trackers tend to have problems with the first and last few beats of Mazurkas due to boundary problems, and observe increased error-levels caused by sudden tempo changes, but as far as we know no measure for local tempo stability has been proposed.

Modeling local tempo, determining its stability, and estimating it automatically from audio are problems at the intersection of music information retrieval (MIR) and computational musicology. We believe that all three problems have to be solved together in order to provide use-

(a) Local tempo depending on selection

(b) Local tempo depending on aggregation

Figure 1: Local reference tempo depending on **(a)** selection and **(b)** aggregation functions for Op. 68, 3 (Cohen, 1997) with section boundaries and score tempo markings.

ful tools for computational music performance analysis (MPA) [14]. Such tools can, for example, be used to determine how well a given performance matches the score—similar to how it has been done for dynamics [15]. Studies like [16], comparing relative local tempo variations within performance collections, could be enhanced by using absolute tempo information.

Working towards this goal, we investigate how to model local tempo (Section 2) and tempo stability (Section 3) for expressive music using Mazurkas by Chopin. As our main contribution, we estimate local tempi using neural network-based approaches, adapt these approaches to our use case, and explore their behavior and potential (Section 4). In our evaluation, we focus on identifying error classes and sources, and in particular the effect of stability. In Section 5, we discuss our findings and draw conclusions.

2. LOCAL TEMPO

Cancino-Chacón et al. [17] see the *global tempo* of a performance as the approximate rate at which musical events happen throughout that performance. In contrast, *local tempo* refers to the rate of events within a smaller time window and can therefore be regarded as local deviation from the global tempo. In accordance with this definition, we are interested in a musically meaningful, single-value description of a segment of limited length. We can define this length musically, e.g., as three consecutive IBIs [5, 6], or physically, e.g., as 6 s or 8 s segments [9, 10]. In either case, we first *select* beat events, because they fall into a time

span, and then *aggregate* them. For example, we may use the mean or the median of all IBIs falling into a 4 s interval. One purpose of this aggregation is to be able to largely ignore *expressive timing*, which can be defined as deviations of individual beat events from the local tempo [17], e.g., rolled or arpeggiated chords [18]. Note that, in this work, we are not attempting to find the *most* suitable selection and aggregation functions (see [3]), but merely discuss options and aim to establish a framework that can be used for such an endeavor. To illustrate different choices, we use Chopin's Op. 68, 3 (piano: Cohen, 1997) as an example. It is one of over 2,700 recordings of 49 Mazurkas by Chopin collected by the Mazurka Project.[1] Of all collected recordings, 298 recordings of five Mazurkas have been manually beat-annotated [19]. We refer to this subset as the *Mazurka-5* dataset. It contains between 34 and 88 different versions of each of the five Mazurkas (Table 1).
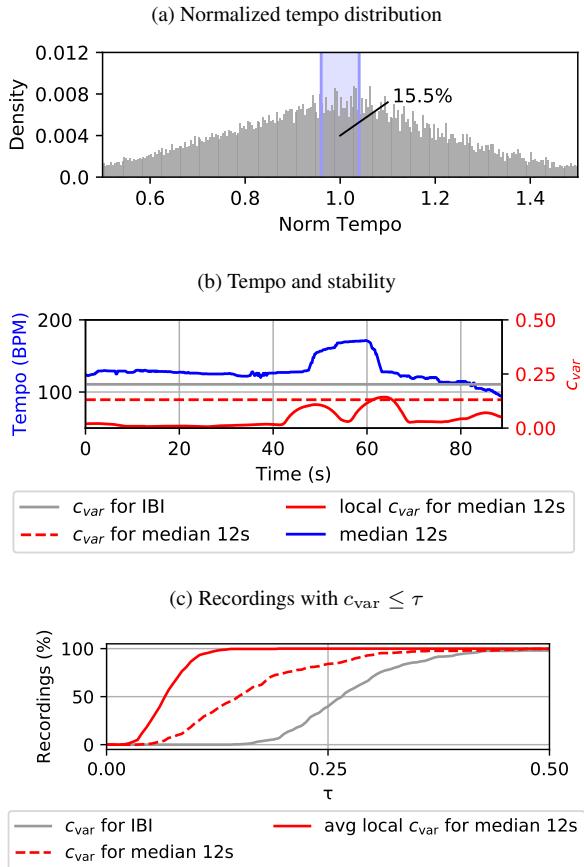
Our example, Op. 68, 3 (Cohen, 1997), consists of four different musical sections A to D (Figure 1). While the score does not contain section markers,[2] it explicitly specifies two tempo changes: at the start of section C from *Allegro, ma non troppo (♩=132)* (fast, but not too fast), to *Poco più vivo* (a little more lively), and back to *Tempo I* after the second D-section. Figure 1a depicts the effects of different selection functions using the mean for aggregation. We see that defining local tempo as individual IBIs leads to very high variance. Using three consecutive IBIs smoothes the tempo curve slightly. The shown tempo curves based on 4 s, 8 s, and 12 s segments progressively lead to less variance. While the 4 s tempo curve still follows the phrasing closely (distinct minima at the end of each musical section), this is less the case for the curves based on longer segments. This is especially obvious at the end of the 2nd B section at 38 s.

Figure 1b shows the differences between using mean and median as aggregation function. The tempo curves for mean show local over-smoothing in transitional sections, leading to a triangular shape in the more lively CDD-section from $50 - 60$ s. Because of the edge-preserving property of median-filtering, the median curve captures sudden tempo changes better. The CDD-section resembles a rectangle, i.e., a high tempo plateau. At the same time, the local minimum at the end of the 2nd B disappears. Thus, the median curve corresponds to the composer's markings.

So far we first selected IBIs, aggregated them, and then converted the result to BPM (selection → aggregation → conversion: sac). As an alternative, we could have first converted IBIs to BPM and then aggregated them (selection → conversion → aggregation: sca). When using mean, the result is not the same. For sections with changing tempo (Figure 1b, $30 - 70$ s), local tempo values are lower when we first average and then convert (sac, solid red line) as opposed to first convert and then average (sca, dotted red line). Note that the median is unaffected by this issue.

---

[1] http://www.mazurka.org.uk/

[2] Section markers were added by us to allow an easier discussion.

(a) Normalized tempo distribution

(b) Tempo and stability

(c) Recordings with $c_{\mathrm{var}} \leq \tau$

Figure 2: **(a)** Per recording normalized tempo distribution with percentage of values between 0.96 and 1.04 (light-blue area). **(b)** Local tempo (blue line) and stability (c_{var}) for Op. $68, 3$ (Cohen, 1997). c_{var} based either on IBIs (gray line), the (sampled) median tempo over $12\,\mathrm{s}$ intervals (dashed red line), or the averaged local c_{var} over $12\,\mathrm{s}$ segments of median tempi (solid red line). **(c)** Percentage of recordings with $c_{\mathrm{var}} \leq \tau$.

3. TEMPO STABILITY

For the evaluation of global tempo estimation one typically requires recordings with approximately constant tempi [1], i.e., a certain degree of tempo stability. Since local tempo estimation is in fact global tempo estimation for very short segments, we seek to quantify local tempo stability in order to conduct an informed evaluation of our experiments in Section 4. As a first approach to describe tempo stability quantitatively on the intra-track level, we convert all *Mazurka-5* IBIs to tempo values and normalize them by dividing by their respective track's average. Figure 2a depicts the resulting normalized histogram.[3] Only 15.5% of the *Mazurka-5*'s normalized tempi are in the interval between 0.96 and 1.04—the often used $\pm 4\%$ tolerance interval for stable tempi [1]. For comparison, 90.9% of the *Ballroom* [1,20] dataset's normalized tempi are in the same interval. Obviously, the two datasets are very different w.r.t. intra-track tempo stability.

While the $\pm 4\%$ interval is illustrative when categoriz-

ing stable vs. unstable, it is a rather arbitrary threshold. Arguably, the standard deviation of a track's normalized tempi is better suited to describe intra-track tempo variability. It is identical to the coefficient of variation,[4] which is defined as the ratio between the standard deviation σ and the mean μ:

$$c_{\mathrm{var}} = \frac{\sigma}{\mu}. \tag{1}$$

We show this IBI-based c_{var}-value for our example Op. $68, 3$ (Cohen, 1997) as a horizontal gray line in Figure 2b. As discussed in Section 2, instantaneous tempo values like IBIs tend to overestimate the variance of a musically meaningful local tempo for expressive music. From a musical point of view, it is therefore more appropriate to analyze tempo stability of Mazurkas not based on individual IBIs, but on the basic tempo, which—*for the purpose of this discussion*—we approximate with the median tempo over $12\,\mathrm{s}$ segments (Figure 2b, blue line). Sampling the local median tempo allows us to calculate an arguably more appropriate c_{var} (Figure 2b, dashed red line), which lies well below the gray line, indicating higher stability. This however, still ignores the fact that Mazurkas may contain multiple sections with stable but different tempi. We can take this into account by calculating local coefficients of variation for short segments of the median-based tempo curve. The solid red curve in Figure 2b shows the results for overlapping $12\,\mathrm{s}$-segments. For most of the recording it is very low. Only in the transitional regions, at the beginning and end of the CDD-section, we see higher values. Note that by averaging the local c_{var} we can obtain a measure for intra-*segment* stability, while the two track-level c_{var} measures represent intra-*track* stability. Figure 2c depicts how many recordings of our dataset have a c_{var} below a threshold τ for all three ways of calculating it. The comparison shows that for *Mazurka-5* intra-segment variability is far smaller than intra-track variability.

4. EXPERIMENTS
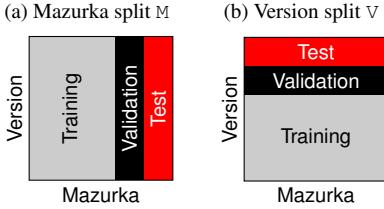
We now investigate how different local tempo estimation systems perform when tested with *Mazurka-5*. We consider the following systems: The RNN-based beat tracking system Böck[5] [21] (estimated beats are aggregated identically to the ground truth), the CNN-based tempo estimation system DeepTemp[6] [22], and the system DT-Maz, which is set up identically to DeepTemp, but has been trained on *Mazurka-5* recordings instead of Pop/Rock, EDM, and Ballroom music. Based on our observations in Section 2 and informal experiments with several segment lengths, we model the local tempo with median-aggregated IBIs from $11.9\,\mathrm{s}$ segments.[7] As mentioned in Section 2, we do not claim that this is the best possible selection or aggregation, but a reasonable configuration.

---

[3] The comb pattern is a consequence of the $10\,\mathrm{ms}$ resolution of the original annotations.

[4] Also known as CV or relative standard deviation (RSD).

[5] https://github.com/CPJKU/madmom with default parameters.

[6] Scaled with model sizing parameter $k = 16$, see [22] for details.

[7] We chose $11.9\,\mathrm{s}$ instead of the previously used $12\,\mathrm{s}$ for practical reasons. The system DeepTemp is already trained on $11.9\,\mathrm{s}$.

Figure 3: Dataset splitting into training, validation, and test sets.

4.1 Setup
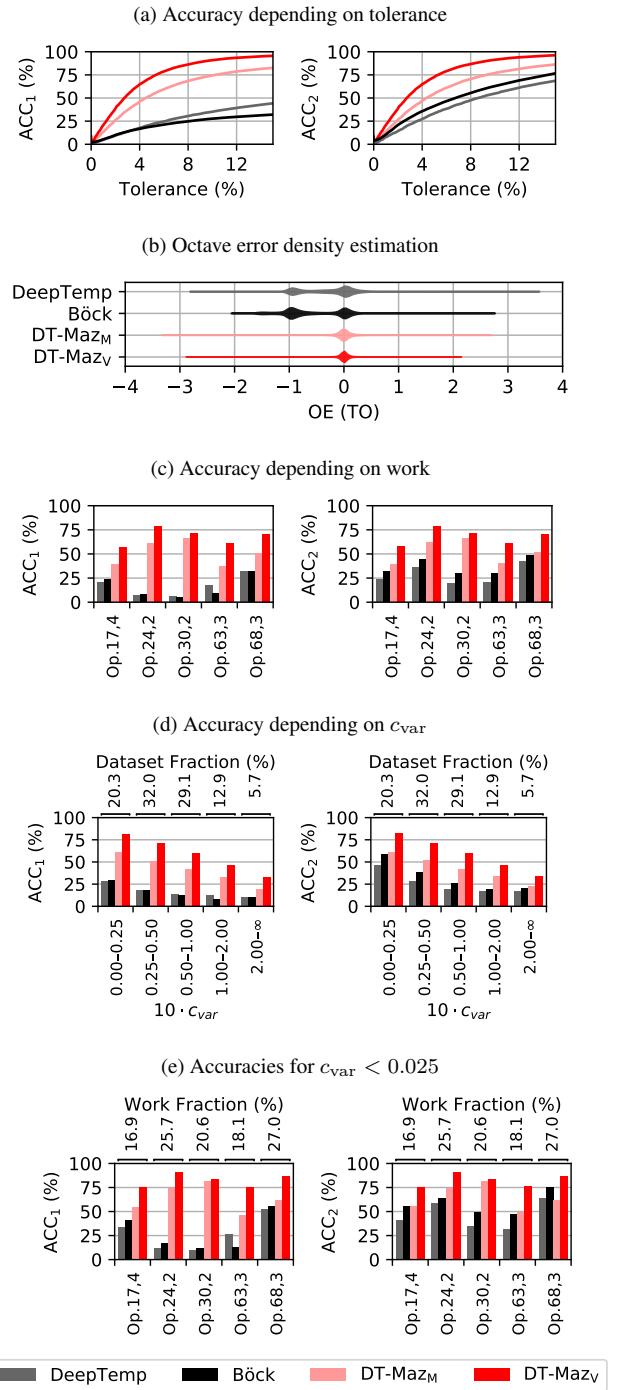
We trained DT-Maz from scratch [8] on *Mazurka-5* recordings using 5-fold cross validation with two different kinds of splits, M for Mazurka and V for version (or performance). For M, each split contains all versions of one Mazurka (Figure 3a). For V, each split consists of a disjoint 5th of all versions of each of the five Mazurkas (Figure 3b). During training, three splits were used as training data and one for validation. The remaining 5th split was used for testing. Each split was used exactly once for validation or testing. We refer to the models trained on M-splits as DT-Maz$_M$ and to the V-split models as DT-Maz$_V$. The employed training procedure was very similar to [12]. Audio is first converted to mel-magnitude-spectrograms. Then samples with the dimensions $F \times T$ are used as network input. $F = 40$ being the number of frequency bins covering the frequency range $20 - 5,000$ Hz, and $T = 256$ being the number of time frames with a length of 46 ms per frame, corresponding to 11.9 s. We further use scale & crop data augmentation [12] with time scale factors drawn from $\mathcal{N}(1, 0.1)$, but limited to $[0.7, 1.3]$ to avoid extreme distortions. After augmentation, samples are standardized to zero mean and unit variance. Like [12], we use categorical crossentropy as loss, because we cast tempo estimation as a classification problem, predicting tempo as one of 256 linearly spaced classes ranging from 30 to 255 BPM. [9] Adam [23] is used as optimizer with a batch size of 32 and an initial learning rate of 0.001. The rate is halved once the validation loss stops improving and training is continued with the best performing model up to that point (stepwise annealing). We repeat this at most 10 times. If reduction does not lead to a lower validation loss three times in a row, training is stopped. To avoid overfitting to longer recordings, we ensure that samples from all training recordings are presented with the same frequency.

4.2 Evaluation

To evaluate, we estimate the tempo for a sliding segment with length 11.9 s (256 frames) and a hop size of 186 ms (4 frames) over all recordings. As metric we use ACC$_1$ (tempo accuracy) and ACC$_2$ (accuracy allowing so-called *octave errors*, i.e., estimates that are wrong by the factor 2, ½, 3 or ⅓) from the global tempo estimation
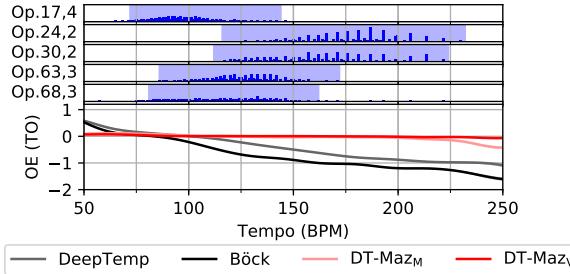
Figure 4: **(a)** Local ACC$_1$ and ACC$_2$ depending on accuracy tolerance. **(b)** Density estimation for OE. **(c)** Local ACC$_1$ and ACC$_2$ for the five Mazurkas. **(d)** Local ACC$_1$ and ACC$_2$ considering c_{var} ranges. **(e)** Accuracies for segments with $c_{var} < 0.025$.

task [1], which are meant for music with low intra-track tempo variability. This is reasonable, because we apply the metric locally for each segment, so that the tolerance does not have to correspond to intra-track, but to intra-segment variability, and as we have shown in Section 3, intra-segment variability is relatively low. Nevertheless, we consider the typical 4% tolerance an arbitrary threshold and therefore plot accuracy values for the tolerance interval

Figure 5: **(top)** Sweet octaves in light-blue. Local tempo histograms in dark-blue. **(bottom)** Estimates of generalized additive models fit to OE/tempo-pairs

$0-15\%$ in Figure 4a. For both variants of `DT-Maz`, ACC_1 values are higher than for the other systems, regardless of tolerance. Not surprisingly, ACC_1 values are also generally higher for higher tolerances.[10] The best performing system for the tolerances 4%, 8%, and 12% is `DT-Maz`$_V$ with remarkable 64.6%, 86.4%, and 93.5%. The worst performing system is `Böck`, with 16.8%, 24.8%, and 29.7%. For ACC_2 the best performing system is also `DT-Maz`$_V$ with 64.8%, 86.8%, and 94.0%, and the worst performing system is `DeepTemp` with 27.3%, 47.5%, and 61.2%. In the following paragraphs we discuss the most prominent errors, namely octave errors, tempo stability related errors, and problems with specific musical properties.

Tempo Octave. Using violin plots, Figure 4b depicts kernel density estimates (KDE) of the octave error OE defined as

$$\text{OE}(y, \hat{y}) = \log_2 \frac{\hat{y}}{y}, \qquad (2)$$

with $y, \hat{y} \in \mathbb{R}_{>0}$ as the ground truth and estimate. Identifiable by the very dense section around -1 Tempo Octaves (TO), `DeepTemp` and `Böck` suffer most from underestimating the actual tempo. As Figure 4c shows, octave errors are not evenly distributed among the five Mazurkas. Op. 24, 2 and Op. 30, 2 are much more affected than the other three. This can be partially explained by the fact that on average versions for Op. 24, 2 and Op. 30, 2 are much faster. Their *sweet octaves* [24], i.e., the tempo octave most tempo values are in, are $[116, 232)$ and $[112, 224)$ BPM, while the sweet octaves for Op. 17, 4, Op. 63, 3, and Op. 68, 3 are $[72, 144)$, $[86, 172)$, and $[81, 162)$ BPM (Figure 5, top). A closer investigation shows that for the tested Mazurkas, both `DeepTemp` and `Böck` lean towards negative octave errors for higher tempi, revealing an *octave bias* [24]. This is visualized in Figure 5, bottom. It shows the estimates of generalized additive models (GAM) that are fit to measured OE per reference tempo. It illustrates what kind of estimation error we can expect depending on a given true tempo. For tempi greater than 100 BPM, `Böck` and `DeepTemp` tend to suffer from negative octave errors.

Stability. Figure 4d shows that accuracy is higher when considering only segments with low c_{var}-values—our proxy for tempo variability. When only considering
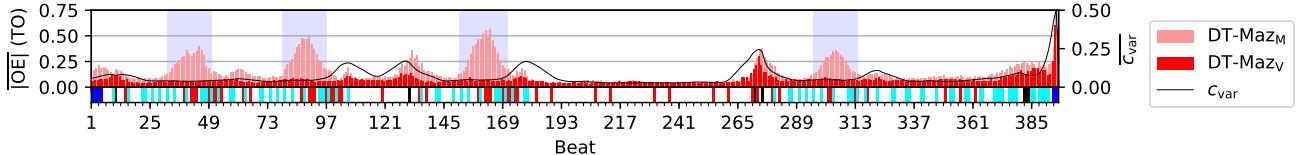
relatively stable segments with $c_{\text{var}} < 0.025$ (Figure 4e), the accuracy scores for all five Mazurkas increase substantially. The comparison of `DT-Maz`$_M$ and `DT-Maz`$_V$ shows that `DT-Maz`$_M$ performs much worse for some Mazurkas (Op. 17, 4, Op. 63, 3, and Op. 68, 3) than `DT-Maz`$_V$. Apparently, differences in stability cannot fully explain differences in accuracy for the five works.

Musical Properties. We have seen in Figure 4e that even for stable segments, `DT-Maz`$_V$ performs better than `DT-Maz`$_M$. To find out why, we exploit beat annotations for each recording of the five Mazurkas. They allow us to compute stability and the absolute octave error $|\text{OE}|$ for $11.9\,\text{s}$ segments with a beat at their center, i.e., stability and error on a musical time axis. Using musical time, we can summarize errors and stability measures *cross-version* by averaging per beat over all recordings of a given Mazurka. Figure 6 shows the results for Op. 17, 4 and as expected, the $\overline{c_{\text{var}}}$-curve roughly correlates with errors by both `DT-Maz`$_M$ and `DT-Maz`$_V$. For `DT-Maz`$_M$ we see four additional peaks around beats 42, 89, 162, and 305 (highlighted in light-blue). These peaks loosely correlate with the occurrence of dense mixtures of ornamented beats (red) and weak bass beats (cyan), i.e., piece-dependent musical properties (classification from [13]), which are apparently the main reason for the difference in accuracy. Trained on the V-split, `DT-Maz`$_V$ was able to learn piece-specific musical properties and generalize them across versions. This implies that expecting `DT-Maz`$_M$'s accuracy levels is more realistic when using either model on unseen Mazurkas.

5. DISCUSSION AND CONCLUSIONS

With five of Chopin's Mazurkas as use case, we have shown that local tempo for expressive music can be modeled using `median` aggregated IBIs, and tempo stability can be measured using the coefficient of variation (c_{var}) of local tempo values. Using these tools, we have found that the five Chopin Mazurkas exhibit high intra-track tempo variability, but low intra-segment variability, i.e., the local tempo is relatively stable and thus musically meaningful. This has allowed us to conduct a local tempo estimation experiment. As was to be expected, a standard beat-tracker like `Böck` and a tempo estimation CNN like `DeepTemp`—trained on Pop, EDM, and Ballroom music—perform relatively poorly for Mazurkas. Even when ignoring tempo octave errors, the results are by far inferior to those achieved by the same kind of CNN as `DeepTemp`, but trained on recordings from the target genre. It is reasonable to assume that training the `Böck` system on Mazurkas would also improve performance substantially—at the price of a strong genre bias. More interestingly, we have been able to confirm a relationship between estimation accuracy and tempo stability measured in c_{var}. Arguably, segments with a very high c_{var} may not have a meaningful local tempo and should therefore be excluded from local tempo evaluation. Another valuable insight results from the comparison of local accuracy results for `DT-Maz`-models trained on either the piece-wise M- or the performance-wise V-split. It

---

[10] To keep the evaluation concise, the reported local accuracy in all following accuracy figures use 4% tolerance.

Figure 6: Averaged c_{var} and $|OE|$ for Op. 17, 4 around beats with classifications from [13]: non-event beats (black ■), boundary beats (blue ■), ornamented beats (red ■), and weak bass beats (cyan ■). High-error sections, unexplained by tempo instability, are highlighted in light-blue.

allows identification of piece-specific, musically difficult passages. When training and testing on the `V`-split, the network apparently has a chance to learn these piece-specific features not covered by data augmentation. One might also argue, `DT-Maz`$_V$ overfits to the pieces ("cover song effect" [25]). While usually seen as a negative effect, we exploit this to learn about our dataset by contrasting results with `DT-Maz`$_M$.

As with all deep learning systems, performance depends largely on the training data. For a production system, one is therefore well advised to use a larger and more diverse training set than we did in this case study.

6. FUTURE WORK

We consciously refrained from attempting to find ideal segment lengths and aggregation functions. We would therefore welcome studies on larger corpora of expressive music that search for optimal selection and aggregation functions as well as c_{var} ranges useful for meaningful evaluations.

7. REFERENCES

[1] F. Gouyon, A. P. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.

[2] B. H. Repp, "Diversity and commonality in music performance: An analysis of timing microstructure in Schumann's "Träumerei"," *The Journal of the Acoustical Society of America*, vol. 92, no. 5, pp. 2546–2568, 1992.

[3] ——, "On determining the basic tempo of an expressive music performance," *Psychology of Music*, vol. 22, no. 2, pp. 157–167, 1994.

[4] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, pp. 39–58, 2001.

[5] P. Grosche and M. Müller, "Extracting predominant local pulse information from music recordings," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1688–1701, 2011.

[6] E. Chew and C. Callender, "Conceptual and experiential representations of tempo: effects on expressive performance comparisons," in *Proceedings of the International Conference on Mathematics and Computation in Music (MCM)*. Springer, 2013, pp. 76–87.

[7] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[8] A. Gkiokas, V. Katsouros, G. Carayannis, and T. Stafylakis, "Music tempo estimation and beat tracking by applying source separation and metrical relations," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, 2012, pp. 421–424.

[9] G. Peeters, "Time variable tempo detection and beat marking," in *Proceedings of the International Computer Music Conference (ICMC)*, Barcelona, Spain, 2005.

[10] ——, "Template-based estimation of time-varying tempo," *EURASIP Journal on Advances in Signal Processing*, 2007.

[11] J. L. Oliveira, F. Gouyon, L. G. Martins, and L. P. Reis, "IBT: A real-time tempo and beat tracking system," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 291–296.

[12] H. Schreiber and M. Müller, "A single-step approach to musical tempo estimation using a convolutional neural network," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 98–105.

[13] P. Grosche, M. Müller, and C. S. Sapp, "What makes beat tracking difficult? A case study on Chopin

Mazurkas," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 649–654.

[14] A. Lerch, C. Arthur, A. Pati, and S. Gururani, "Music performance analysis: A survey," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 33–43.

[15] K. Kosta, O. F. Bandtlow, and E. Chew, "Dynamics and relativity: practical implications of dynamic markings in the score," *Journal of New Music Research*, vol. 47, no. 5, pp. 438–461, 2018.

[16] J. Peperkamp, K. Hildebrandt, and C. C. S. Liem, "A formalization of relative local tempo variations in collections of performances," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 158–164.

[17] C. E. Cancino-Chacón, M. Grachten, W. Goebl, and G. Widmer, "Computational models of expressive music performance: A comprehensive and critical review," *Frontiers in Digital Humanities*, vol. 5, 2018.

[18] M. Fu, G. Xia, R. B. Dannenberg, and L. A. Wasserman, "A statistical view on the expressive timing of piano rolled chords," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, 2015, pp. 578–583.

[19] C. S. Sapp, "Hybrid numeric/rank similarity metrics," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Philadelphia, USA, 2008, pp. 501–506.

[20] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, 2013, pp. 227–232.

[21] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016, pp. 255–261.

[22] H. Schreiber and M. Müller, "Musical tempo and key estimation using convolutional neural networks with directional filters," in *Proceedings of the Sound and Music Computing Conference (SMC)*, Málaga, Spain, 2019, pp. 47–54.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference for Learning Representations (ICLR)*, San Diego, California, USA, 2015.

[24] H. Schreiber and M. Müller, "A post-processing procedure for improving music tempo estimates using supervised learning," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 235–242.

[25] H. Schreiber, C. Weiß, and M. Müller, "Local key estimation in classical music recordings: A cross-version study on Schubert's Winterreise," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020, pp. 501–505.

LEARNING TO READ AND FOLLOW MUSIC IN COMPLETE SCORE SHEET IMAGES

Florian Henkel[1]   **Rainer Kelz**[2]   **Gerhard Widmer**[1]

[1] Institute of Computational Perception, Johannes Kepler University Linz, Austria
[2] Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria

florian.henkel@jku.at

ABSTRACT

This paper addresses the task of score following in sheet music given as unprocessed images. While existing work either relies on OMR software to obtain a computer-readable score representation, or crucially relies on prepared sheet image excerpts, we propose the first system that directly performs score following in full-page, completely unprocessed sheet images. Based on incoming audio and a given image of the score, our system directly predicts the most likely position within the page that matches the audio, outperforming current state-of-the-art image-based score followers in terms of alignment precision. We also compare our method to an OMR-based approach and empirically show that it can be a viable alternative to such a system.

1. INTRODUCTION

Score following is a fundamental task in MIR and the basis for applications such as automatic accompaniment [1, 2], automatic page turning [3] or the synchronization of live performances to visualizations [4, 5]. These applications require a real-time capable system that aligns a musical performance to a symbolic score representation in an online fashion. To solve this, existing systems either require a computer-readable score representation (e. g. extracted using Optical Music Recognition (OMR) [6]) or rely on fixed-size (small) snippets of sheet images.

Models from the latter category are by design only capable of handling fixed-sized excerpts of the sheet image due to a limited action space to predict the next position in the score. This is a severe constraint, as the sheet image snippet has to (at least partly) correspond to the incoming audio excerpt. If it does not match the audio anymore (due to some tracking error), no proper prediction can be formed. To overcome this limitation, we attempt score following directly in the full sheet image, enabling the system to observe the whole page at any given time. This makes the problem significantly more challenging, e. g., due to repetitive musical score structures, compared

to locally constrained systems that only look at snippets.

To the best of our knowledge, we present the first system that requires neither OMR nor any other form of score pre-processing and directly follows musical performances in full sheet images in an end-to-end fashion. [1]
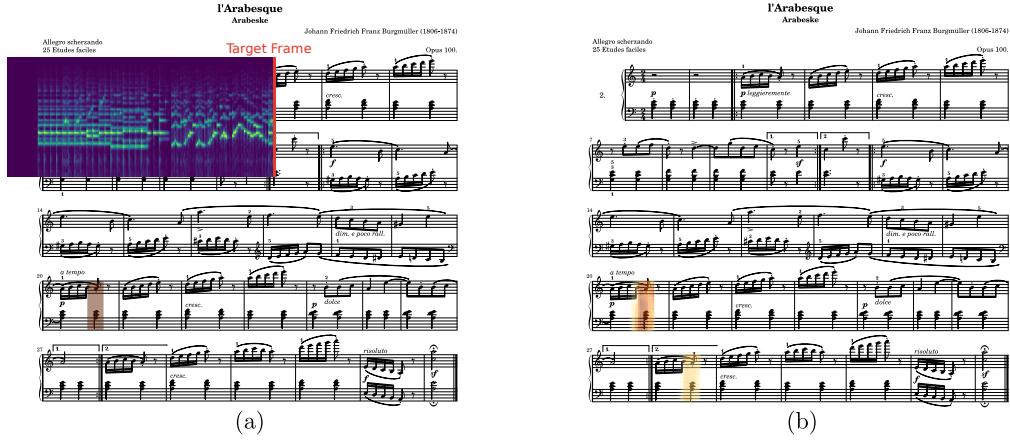
Specifically, we formulate score following as a *referring image segmentation task* and introduce an appropriate model architecture in Section 3, including a conditioning mechanism in the form of a Feature-wise Linear Modulation (FiLM) layer [7] as a central building block. In Section 4, we demonstrate the system on polyphonic piano music, taken from the MSMD dataset [8]. To analyze its generalization capabilities, we also test it on real musical performances taken from the MSMD test split, in Section 5. The results will show that our model outperforms current state-of-the-art image based trackers in terms of alignment precision, but also that it currently lacks robustness across different audio conditions.

2. RELATED WORK

Score following approaches can be broadly categorized into those that rely on the presence of a computer-readable score representation, such as MusicXML or MIDI, and those that try to do without such a symbolic representation. In the former category, techniques like Dynamic Time Warping (DTW) [4, 9, 10] and Hidden Markov Models (HMMs) [11–13] are applied to achieve robust and reliable tracking results. The main issue with these approaches is the need for computer-readable score representations, which must either be created manually in a tedious and time consuming process, or automatically extracted using OMR. In the OMR case, the faithfulness of the symbolic score to what is depicted on the sheet image strongly depends on the quality of the OMR system, which may introduce errors that impede the actual score following task. Empirical evidence for this claim was published in [14], where a DTW-based score following system that relied on MIDI scores extracted via an OMR system had difficulties tracking synthetically created test data.

Several recent publications deal with the latter category, and investigate score following in the context of non-computer-readable score representations, represented as raw sheet images. In [15], the authors propose a multi-

---

[1] Code and data will be made available on-line: https://github.com/CPJKU/audio_conditioned_unet

Figure 1. Score Following Task as modelled in this work: Given a score (sheet image) and an incoming musical performance (audio), the goal is to predict the current location in the score in real time. The audio is fed to the tracker frame by frame and the system processes the last 40 frames to predict the position for the latest frame (the *Target Frame* marked red in (a)). The ground truth (bounding box around current score position; see (a)) is given as a binary segmentation mask. The system predicts a probability for each pixel to correspond to the current audio; thus it highlights those regions that are most likely to match the correct location. Ideally, this should be only a single region. However, in (b) we see that such a prediction is not perfect: while the highest probability is assigned to the correct position in staff four, there is also a small likelihood in the last staff, as the notes are the same for both locations. To predict the location correctly, the system needs to consider the whole audio up to the current point in time, which motivates our design choices introduced in Section 3.

modal deep neural network to predict the position within a sheet snippet based on a short audio excerpt. In [16] and [14], score following is formulated as a reinforcement learning (RL) problem, where the RL agent's task is to adapt its reading speed in an unrolled sheet image, conditioned on an audio excerpt. One of the limitations of all these methods is that they require the score to be represented in an unrolled form, i. e., staves need to be detected in the sheet image, cut out and presented to the score following system in sequence.

To overcome this, [17] introduced a system that directly infers positions within full sheet images for monophonic piano music. However, the temporal aspect of score following was neglected altogether — based on an audio excerpt all possible matching positions in the full sheet image are highlighted, including those that were already played — making it interesting preliminary work, but not a reasonable score following system. In the following we build upon their foundation and incorporate long term audio context, proposing the first fully capable score following system that works on entire sheet images, without needing any pre-processing steps.
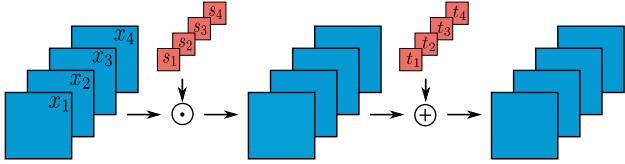
3. SCORE FOLLOWING AS A REFERRING IMAGE SEGMENTATION TASK

Similarly to [17], we model score following as an image segmentation task — more specifically, as a *referring* image segmentation task. In computer vision, the goal of referring image segmentation is to identify a certain region or object within an image based on some language expression [18, 19]. It shows similar characteristics as the multimodal approach to score following — we want to locate the position in the sheet image that the incoming audio refers to, meaning we treat the audio as the language expression, and the score image as the entity to reason about. More

precisely, our modeling setup is as follows: based on the incoming musical performance up to the current point in time, the task of the model is to predict a segmentation mask for the given score that corresponds to this currently played music, as shown in Figure 1. The ground truth for this task is chosen to be a region around the current position in the score with a fixed width and a height depending on the height of the staff. While the size of this mask can be arbitrarily chosen, we define it such that it provides a meaningful learning target first and foremost.

A challenging question arising with such a setup is how to combine the different input modalities, audio and score. While [14–16] learn a latent representation for both input modalities which are subsequently concatenated and further processed, we follow the direction of [17] instead, and employ a *conditioning mechanism* that directly modulates the activity of feature detectors that process the score image. The former setup would be problematic due to the increasing number of parameters. Furthermore, this design is able to retain the fully-convolutional property of our model, i. e., if desired one could process sheet images of arbitrary resolution. [2] In contrast to [17], we apply the conditioning mechanism on top of a recurrent layer to provide a longer temporal context. This permits the audio input up until the current point in time to guide the segmentation towards the corresponding position in the score image. We argue that it is necessary for this task to have such a long temporal audio context in order to form more reliable predictions. For example, it is common to have repeating note patterns in the score spanning over a longer period of time in the audio. Existing trackers that use only a fixed-size audio input are not able to distinguish between such patterns, if they exceed the given audio context.

---

[2] Note that while we do not investigate this further and work with fixed-sized sheets, this could be useful in a real world application.

Figure 2. Sketch of the FiLM layer [7]. The layer scales and translates the feature maps $\mathbf{x}$ using learned functions $\mathbf{s}(\mathbf{z})$ and $\mathbf{t}(\mathbf{z})$, respectively. $\mathbf{z}$ is an additional, external input carrying the conditioning information.

Audio (Spectrogram) 78×40
2 x (Conv(3, 1, 1)-24 - LN - ELU) - MP(2)
2 x (Conv(3, 1, 1)-48 - LN - ELU) - MP(2)
2 x (Conv(3, 1, 1)-96 - LN - ELU) - MP(2)
2 x (Conv(3, 1, 1)-96 - LN - ELU) - MP(2)
Conv(1, 0, 1)-96 - LN - ELU
Dense(32) - LN - ELU

Table 1. The context-based encoder used for the experiments. Conv(f, p, s)-k denotes a convolutional layer with k $f \times f$ kernels, padding of p and stride s. We use layer normalization (LN) [20], ELU activation [21] and max pooling (MP) with a pool size of 2×2. The output of the last layer is fed into a LSTM as shown in Figure 3. The network resembles the one used in [8].

3.1 Feature-wise Linear Modulation

The Feature-wise Linear Modulation (FiLM) layer is a simple linear affine transformation of feature maps, conditioned on an external input [7]. The purpose of using this layer is to directly interfere with the learned representation of the sheet image by modulating its feature maps, assisting the convolutional neural network to focus only on those parts that are required for a correct segmentation. In our case, the external input $\mathbf{z}$ is the hidden state of a recurrent layer that takes as input an encoded representation of an audio excerpt. This encoded representation is created by a neural network, e. g., as depicted in Table 1. The FiLM layer itself is defined as

$$f_{\text{FiLM}}(\mathbf{x}) = \mathbf{s}(\mathbf{z}) \cdot \mathbf{x} + \mathbf{t}(\mathbf{z}), \qquad (1)$$

where $\mathbf{s}(\cdot)$ (for scaling) and $\mathbf{t}(\cdot)$ (for translation) are arbitrary vector-valued functions implemented as neural networks. Their values depend on the conditioning vector $\mathbf{z}$, and together they define an affine transform of the tensor $\mathbf{x}$ which refers to the collection of feature maps of a particular convolutional layer, after normalization. The affine transformation is performed per feature map, meaning that each feature map k is scaled and translated by $s_k(\cdot)$ and $t_k(\cdot)$, respectively, with k identifying the k-th output of the two functions. The number of output values for $\mathbf{s}(\cdot)$ and $\mathbf{t}(\cdot)$ is the same as the number of feature maps contained in $\mathbf{x}$, denoted by K (cf. Figure 2).

3.2 Model Architecture

Our model is based on a U-Net architecture similar to the one used in [22] for detecting musical symbols in sheet images. U-Nets were originally introduced for medical image segmentation, to segment an image into different parts, by classifying each pixel into either foreground or

background [23]. This fits naturally to our interpretation of the score following task, as a process of segmenting the sheet image into a region that corresponds to the current position in the audio input and labelling everything else as background. The overall architecture, shown in Figure 3, resembles the one proposed in [17], with several important differences. Based on the empirical findings reported in [17], we decide to incorporate conditioning information from the audio input in blocks B-H, leaving only blocks A and I without conditioning. However, we substitute the transposed convolutions in the decoder part of the network with bilinear upsampling operations with a factor of two, followed by a 1×1 convolution, both aimed at reducing checkerboard artifacts in the segmentation [24]. Due to the small batch size used during training (cf. Section 4.3) as well as the presence of the recurrent layer, we replace batch normalization with layer normalization [20].

For deriving the conditioning information from the audio input, we test two different spectrogram encoders. One takes a spectrogram snippet with a length of 40 frames, corresponding to two seconds of audio; the spectrogram is processed by the network shown in Table 1, which is roughly similar to the one used in [8]. The other version takes as input a *single spectrogram frame*, using a dense layer with 32 units, layer normalization and ELU activation function. The output of the encoders is fed to an LSTM [25] layer with 128 units and its hidden state is then used as the external input $\mathbf{z}$ in the FiLM layers.

4. EXPERIMENTS
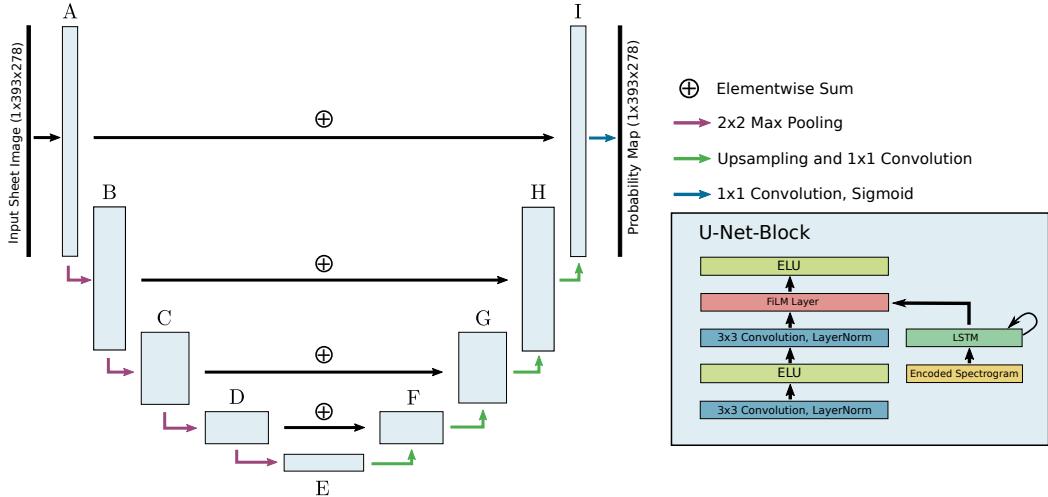
To study the properties of our proposed approach, we conduct experiments on polyphonic piano music provided by the MSMD [8] dataset. While this section is mainly concerned with comparing data augmentation and different architectures, later on in Section 5 we will investigate the generalization capabilities of the system in terms of 16 real piano recordings from the MSMD test split. We will also contextualize the proposed system with related work described in [14], which we use as baselines for comparison.

4.1 Data

We use the *Multi-modal Sheet Music Dataset (MSMD)* [8], a standard dataset for such evaluations, comprising polyphonic piano music from various composers including Bach, Mozart, and Beethoven. The sheet music is typeset with Lilypond [3] and the audio tracks are synthesized from MIDI using Fluidsynth [4] together with a piano sound font. The original MSMD splits used by [14] encompass 354 train, 19 validation and 94 test pieces. The precise alignments between audio and sheet music in this dataset are created automatically. Despite that, it turned out that some of the pieces still contain alignment errors. We manually identified and fixed most of these errors, including wrongly aligned notes and missing or wrongly detected staves. One piece from the train set was removed, because we were not

---

[3] http://lilypond.org/
[4] http://www.fluidsynth.org/

Figure 3. Audio-Conditioned U-Net architecture. Each block (A-I) consists of two convolutional layers with ELU activation and layer normalization. The FiLM layer is placed before the last activation function. The spectrogram encoding given by the output of the network shown in Table 1 is passed through a recurrent layer. The hidden state of this recurrent layer is then used for conditioning in the FiLM layer. Each symmetric block has the same number of filters starting with 8 in block A and increasing with depth to 128 in block E.

able to fix it. Thus, the cleaned dataset consists of 353 train, 19 validation, and 94 test pieces, which will be made publicly available. If a piece consists of several pages, each page is treated as a single piece and the original MIDI information is partitioned accordingly.[5] Altogether, we have 945 train, 28 validation and 125 test pages. The rendered score images have a resolution of 1181×835 pixels, are downscaled by a factor of three to 393×278 pixels, and are used as the input to the U-Net. Preliminary tests showed that the downscaling does not significantly impact the performance, and benefits the speed of the training process. For the ground truth annotations, we rely on the automatic notehead alignment described in [8]. The notehead alignments yield (x, y) coordinate pairs in the sheet image, which are further adjusted for our purposes such that the y coordinates correspond to the middle of the staff the respective note belongs to. Given these coordinates, we create a binary mask with a width of 10 pixels and an adaptive height depending on the height of the staff (see Figure 1). The task of the U-Net is now to infer a segmentation mask given the image of the score together with the conditioning information derived from the audio input. Note that in theory it should be possible to directly predict x and y coordinates instead of a segmentation mask, however as shown in [26] this is a much harder task, and we were not able to achieve acceptable performance so far, even using their proposed *CoordConv* layer.

The audio is sampled with 22.05 kHz and processed at a frame rate of 20 frames per second. The DFT is computed for each frame with a window size of 2048 samples and then transformed with a semi-logarithmic filterbank that processes frequencies between 60 Hz and 6 kHz, yielding 78 log-frequency bins. Lastly, the spectrogram bins are standardized to zero mean and unit variance. The audio conditioning network is presented either with 40 consec-

utive frames (two seconds of audio) or a single frame at a time. We use the `madmom` python library for all signal processing related computations [27].

4.2 Baselines and Evaluation Measures

In the following, we will present a series of experiments, comparing the new proposed full-page tracking system to several baselines (in order to better understand the importance of some of our design choices) as well as to related state-of-the-art approaches from the literature.

First, we evaluate two different spectrogram encoders, as introduced in Section 3.2, vis-à-vis a baseline version of our system that does not have the capability to summarize all the audio up to the current point in time, i.e., that does not have memory in the form of an RNN. We do this in order to obtain empirical evidence for our argument that having access to long term temporal information is highly beneficial for obtaining good approximate solutions to the score following task. The two different encoders are denoted as context-based (CB) and frame-based (FB), using 40 spectrogram frames and a single frame, respectively. The baseline without temporal context uses the CB encoder and replaces the RNN layer with a fully connected layer of the same size. In the following this baseline will be denoted as NTC (no temporal context).

The *evaluation measures* used for this comparison are of a geometric kind (bounding box pixel error and distance on printed score page), in order to focus on the new challenge of full-page orientation: we measure the pixel-wise evaluation metrics *Precision*, *Recall* and F_1-score that were also used in [17], and the mean and median alignment error between ground truth and prediction in centimeters, both with the network output thresholded at 0.5. To calculate the alignment error between the ground truth and the predicted probability mask (recall Figure 1), we calculate the center of mass over all pixels for both masks and compute the euclidean distance between the two centers to

---

[5] This is mainly done to facilitate the training procedure. In an application, this could be solved by some simple 'hack' that turns pages when the score follower reaches the end of a page.

obtain the alignment error in pixels. Given a resolution of 72 dpi, the error is converted to centimeter using a factor of 0.0352 cm/pixel, under the assumption that the score image is printed on a sheet of DIN A4 paper.

In the second experiment we compare our system to alternative state-of-the-art approaches from the literature: the first approach is based on an OMR system that extracts symbolic MIDI information from the sheet image. The symbolic MIDI information is then synthesized to audio. The system subsequently computes chroma features with a feature rate of 20 Hz from both the synthesized and the performance audio, and applies audio-to-audio alignment using a form of online DTW [28]. This is the method described in [14] and will be abbreviated as OMR in the upcoming result table. The second and third approach, described in Section 2, are a multi-modal localization network (MM-Loc) [15] and a Reinforcement Learning (RL) agent [14, 16], both working with sheet image snippets.

The *evaluation measure* for this will be of a more music-related kind (temporal tracking error in the performance), reflecting the intended purpose of the systems (score following), and permitting a direct comparison with alternative methods. Similarly to [9, 10], we compute, for each note onset, the absolute time difference between prediction and ground truth. We set 5 threshold values, ranging from 0.05 to 5 seconds, and report the cumulative percentage of notes tracked with an error up to the given threshold. Given the ground truth alignment from note onsets to the corresponding notehead coordinates in the sheet image, we can interpolate from the predicted positions in the sheet image back to the time domain. This is straightforward for MM-Loc and the RL agent, because they both already use an unrolled score derived from the groundtruth, whereas the proposed method requires further processing. We first need to compute the center of mass of the segmented region to obtain x, y coordinates. We map the y coordinate to the closest staff, and apply a similar interpolation as before in an unrolled score to get the time difference between the predicted and actual position in the score.

For evaluating the OMR baseline we face a problem that has already been noted in [14] — we do not have the required groundtruth alignment between the OMR-extracted score and the performance. Given that only onset positions are evaluated, we are justified to assume a perfect alignment between score and audio, if for each unit of time in the audio a constant distance in the score sheet is travelled. If the OMR system makes no errors, the alignment between OMR score and performance is a diagonal in the DTW global cost matrix, correcting the overall tempo difference by a linear factor. As in [14], we evaluate the OMR-based system by measuring the offset of the actual tracking position relative to the perfect alignment.

4.3 Experimental Setup

All models are trained using the same procedure. We optimize the *Dice* coefficient loss [29], which is more suitable than e.g., *binary cross-entropy*, as we are facing an imbalanced segmentation problem with far more unimpor-

tant background pixels than regions of interest. To optimize this target we use Adam [30] with default parameters, an initial learning rate of $1e^{-4}$ and L^2 weight decay with a factor of $1e^{-5}$. If the conditioning architecture involves an LSTM, we use a batch-size of 4 and a sequence length of 16. For the audio conditioning model without a temporal context we use a batch size of 64. The weights of the network are initialized orthogonally [31] and the biases are set to zero. If the loss on the validation set does not decrease for 5 epochs, we halve the learn rate and stop training altogether when the validation loss does not decrease over a period of 10 epochs or the maximum number of 100 epochs is reached. The model parameters with the lowest validation loss are used for the final evaluation on the test set. Similar to [17], we perform data augmentation in the image domain by shifting the score images along the x and y axis. To investigate whether tempo augmentation improves model performance, we train all models without tempo augmentation as well as with 7 different tempo change factors ranging from 0.5 up to 1.5.

4.4 Results

In Table 2, we compare different conditioning architectures, no long term temporal context (NTC), a context of 40 frames (CB) and a single frame (FB) in combination with an LSTM, respectively. We observe that the NTC model has the lowest performance, both in terms of the pixel-wise measures, as well as in terms of its alignment error. A possible reason for this could be ambiguities in the sheet image, since audio excerpts could match several positions in the score. The results for CB and FB support our initial claim that a long term temporal context is required for this task. While both models achieve a good performance, CB outperforms FB in all measures. On average, the alignment error is around 1.25 cm and the median is at 0.51 cm, meaning that half of the time our model is less than 0.51 cm away from the true position. Furthermore, we observe that tempo augmentation improves the results for all models.

In Table 3, we compare our best model from Table 2 to several baselines from the literature in terms of the cumulative percentage of onsets that are tracked with an error below a given threshold. We observe that the context-based proposed model (CB) outperforms all baselines except for the highest threshold. This suggests that our method is very precise on one hand, but on the other hand is not able to track all onsets with a timing error below five seconds.

5. REAL PERFORMANCES

To test the generalization capabilities of the system under real recording conditions, we evaluate our best model on the 16 piano recordings (corresponding to 25 score pages) from the MSMD test split introduced in [14], for which we also manually corrected some of the alignments. We compare again to the baselines introduced in Section 4.2, which are likewise evaluated using the corrected alignments. In line with [14], we compare four different settings with in-

MSMD (125 test pages)						
	TA	**P**	**R**	**F$_1$**	$\overline{d}_{cm}$	$\widetilde{d}_{cm}$
NTC	✗	0.696	0.665	0.678	3.70	2.37
	✓	0.770	0.740	0.754	2.78	1.61
CB	✗	0.810	0.790	0.799	1.62	0.73
	✓	**0.854**	**0.835**	**0.843**	**1.25**	**0.51**
FB	✗	0.790	0.768	0.778	1.82	1.21
	✓	0.820	0.816	0.816	1.58	0.80

Table 2. Different conditioning architectures with/without tempo augmentation (TA): no temporal context (NTC), context-based (CB) and frame-based (FB). For each model the parameters with lowest validation loss are chosen for evaluation on the test set. Measures: pixel-wise precision (P), recall (R) and F_1, and mean ($\overline{d}_{cm}$) and median ($\widetilde{d}_{cm}$) of alignment error in centimeters.

MSMD (125 test pages)				
Err. [sec]	OMR [14]	MM-Loc [32]	RL [14]	CB
≤ 0.05	44.7%	44.6%	40.9%	**73.3%**
≤ 0.10	51.9%	49.2%	43.3%	**74.7%**
≤ 0.50	76.0%	82.2%	79.7%	**85.2%**
≤ 1.00	85.0%	86.0%	87.8%	**88.5%**
≤ 5.00	**97.4%**	92.0%	97.2%	93.7%

Table 3. Our best model (CB) vs. existing baselines, in terms of onsets tracked with an error below a given threshold. For the RL agent we report the average over 10 runs due to its stochastic policy. In contrast to [14], OMR, MM-Loc and RL do not stop tracking if they fall out of a given tracking window.

creasing difficulty. The first is the same synthetic setting as in Section 4. The second setting uses the performance MIDI synthesized with the same piano synthesizer used during training. The third uses the audio of the "direct out" audio output of the "Yamaha AvantGrand N2" hybrid piano used for recording, and the last one uses the audio recorded via a room microphone.

Table 4 summarizes the results. Overall, we observe that the proposed system (CB) achieves more precise results in terms of time difference (i.e., higher percentages for the tighter error thresholds) in three out of four settings. For the last setting we observe a worse performance, which indicates that our model has possibly overfit to the synthesized audio and is not yet robust enough. OMR yields very robust results in all scenarios, which is possibly due to the used chroma features. While the results are not as precise, it outperforms the other methods for higher threshold values.

A possible explanation for this is that our model has more freedom in being able to perform big jumps on the sheet image paper, thus increasing the error possibility. Models relying on sheet snippets are not designed to perform such jumps and thus can also not make very extreme errors. Furthermore, our model is more sensitive to the audio representation fed into the conditioning mechanism, as it influences the convolutional filters in multiple layers that process the sheet image. Overall, we assume that this is an issue of the synthetic dataset which can be tackled by training on more diverse performances and a more robust audio model for the conditioning mechanism.

Err. [sec]	OMR [14]	MM-Loc [32]	RL [14]	CB
	Original MIDI Synthesized (Score = Performance)			
≤ 0.05	37.1%	41.6%	36.5%	**69.8%**
≤ 0.10	46.1%	44.2%	38.2%	**70.6%**
≤ 0.50	74.9%	77.6%	72.9%	**80.6%**
≤ 1.00	**86.8%**	79.9%	79.8%	82.4%
≤ 5.00	**99.6%**	90.3%	96.5%	89.1%
	Performance MIDI Synthesized			
≤ 0.05	28.9%	47.2%	23.4%	**56.5%**
≤ 0.10	39.8%	49.0%	24.8%	**58.1%**
≤ 0.50	71.7%	**83.2%**	54.5%	80.9%
≤ 1.00	83.4%	**86.1%**	64.0%	84.4%
≤ 5.00	**98.8%**	96.0%	81.2%	90.1%
	Direct Out			
≤ 0.05	22.6%	33.8%	27.7%	**40.0%**
≤ 0.10	33.0%	35.4%	29.1%	**41.6%**
≤ 0.50	**70.3%**	59.7%	60.7%	64.2%
≤ 1.00	**83.9%**	63.4%	73.3%	69.3%
≤ 5.00	**99.3%**	75.3%	95.5%	81.1%
	Room Recording			
≤ 0.05	**22.6%**	20.7%	19.2%	9.4%
≤ 0.10	**32.2%**	24.3%	20.6%	10.5%
≤ 0.50	**70.2%**	54.1%	46.6%	21.5%
≤ 1.00	**82.7%**	57.3%	58.7%	26.2%
≤ 5.00	**97.4%**	70.2%	89.1%	44.3%

Table 4. Comparing best performing model to several baselines on a set of 16 real piano recordings (25 pages) from the MSMD test split. Model evaluation is as described in Table 3, with the difference that for the RL agent we report the average over 50 runs due to its stochastic policy and the smaller sample size.

6. DISCUSSION AND CONCLUSION

We have proposed the first end-to-end trained score following system that directly works on full sheet images. The system is real-time capable due to a constant runtime per step, it compares favorably with existing baselines on synthetic polyphonic piano music, and sets the new state of the art for sheet-image-based score following in terms of temporal alignment error. However, there are still generalization problems for real piano recordings. While the model shows a much more precise alignment in most scenarios, we see a performance deterioration over different recording conditions. This will need to be solved in the future, either with a more robust audio model, or a data augmentation strategy that incorporates reverberation effects. Future work will also require testing on scanned or photographed sheet images, to gauge generalization capabilities of the system in the visual domain as well. As there is currently no dataset consisting of scanned sheet images with precise notehead to audio alignments, it will be necessary to curate a test set. The next step towards a system with greater capabilities, is to either explicitly or implicitly incorporate a mechanism to handle repetitions in the score as well as in the performance. We assume that the proposed method will be able to acquire this capability quite naturally from properly prepared training data, although we suspect its performance will heavily depend on its implicit encoding of the audio history so far, i.e., how large an auditory context the recurrent network is able to store.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] C. Raphael, "Music Plus One and Machine Learning," in *Proc. of the International Conference on Machine Learning (ICML)*, Haifa, Israel, 2010, pp. 21–28.

[2] A. Cont, "A Coupled Duration-Focused Architecture for Real-Time Music-to-Score Alignment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 6, pp. 974–987, 2010.

[3] A. Arzt, G. Widmer, and S. Dixon, "Automatic Page Turning for Musicians via Real-Time Machine Listening," in *Proc. of the European Conference on Artificial Intelligence (ECAI)*, Patras, Greece, 2008, pp. 241–245.

[4] A. Arzt, H. Frostel, T. Gadermaier, M. Gasser, M. Grachten, and G. Widmer, "Artificial Intelligence in the Concertgebouw," in *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, Buenos Aires, Argentina, 2015, pp. 2424–2430.

[5] M. Prockup, D. Grunberg, A. Hrybyk, and Y. E. Kim, "Orchestral Performance Companion: Using Real-Time Audio to Score Alignment," *IEEE Multimedia*, vol. 20, no. 2, pp. 52–60, 2013.

[6] J. Calvo-Zaragoza, J. Hajič jr., and A. Pacha, "Understanding Optical Music Recognition," *Computer Research Repository, abs/1908.03608*, 2019.

[7] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *32nd AAAI Conference on Artificial Intelligence*, New Orleans, USA, 2018.

[8] M. Dorfer, J. Hajič jr., A. Arzt, H. Frostel, and G. Widmer, "Learning Audio–Sheet Music Correspondences for Cross-Modal Retrieval and Piece Identification," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, 2018.

[9] A. Arzt, "Flexible and Robust Music Tracking," Ph.D. dissertation, Johannes Kepler University Linz, 2016.

[10] S. Dixon, "An On-Line Time Warping Algorithm for Tracking Musical Performances," in *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh, Scotland, 2005, pp. 1727–1728.

[11] A. Cont, "Realtime Audio to Score Alignment for Polyphonic Music Instruments using Sparse Non-Negative Constraints and Hierarchical HMMS," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5, Toulouse, France, 2006, pp. 245–248.

[12] N. Orio, S. Lemouton, and D. Schwarz, "Score Following: State of the Art and New Developments," in *Proc. of the International Conference on New Interfaces for Musical Expression (NIME)*, Montreal, Canada, 2003, pp. 36–41.

[13] E. Nakamura, P. Cuvillier, A. Cont, N. Ono, and S. Sagayama, "Autoregressive Hidden Semi-Markov Model of Symbolic Music for Score Following," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, 2015, pp. 392–398.

[14] F. Henkel, S. Balke, M. Dorfer, and G. Widmer, "Score Following as a Multi-Modal Reinforcement Learning Problem," *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, 2019.

[15] M. Dorfer, A. Arzt, and G. Widmer, "Towards Score Following in Sheet Music Images," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, 2016, pp. 789–795.

[16] M. Dorfer, F. Henkel, and G. Widmer, "Learning to Listen, Read, and Follow: Score following as a Reinforcement Learning Game," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 784–791.

[17] F. Henkel, R. Kelz, and G. Widmer, "Audio-Conditioned U-Net for Position Estimation in Full Sheet Images," in *Proc. of the 2nd International Workshop on Reading Music Systems*, Delft, The Netherlands, 2019, pp. 8–11.

[18] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy, "Generation and Comprehension of Unambiguous Object Descriptions," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016, pp. 11–20.

[19] L. Ye, M. Rochan, Z. Liu, and Y. Wang, "Cross-Modal Self-Attention Network for Referring Image Segmentation," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, USA, 2019, pp. 10 502–10 511.

[20] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[21] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," in *Proc. of the International Conference on Learning Representations (ICLR) (arXiv:1511.07289)*, San Juan, Puerto Rico, 2016.

[22] J. Hajič jr., M. Dorfer, G. Widmer, and P. Pecina, "Towards Full-Pipeline Handwritten OMR with Musical Symbol Detection by U-Nets." in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 225–232.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional Networks for Biomedical Image Segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI).* Munich, Germany: Springer, 2015, pp. 234–241.

[24] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and Checkerboard Artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[25] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the CoordConv solution," in *Advances in Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2018, pp. 9605–9616.

[27] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "Madmom: A new python audio and music signal processing library," in *Proc. of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 2016, pp. 1174–1178.

[28] M. Müller, *Fundamentals of Music Processing*. Springer Verlag, 2015.

[29] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV).* IEEE, 2016, pp. 565–571.

[30] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. of the International Conference on Learning Representations (ICLR) (arXiv:1412.6980)*, San Diego, USA, 2015.

[31] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013.

[32] M. Dorfer, "Multimodal Deep Representation Learning and its Application to Audio and Sheet Music," Ph.D. dissertation, Johannes Kepler Universität Linz, 2018.

UNCOVERING AUDIO PATTERNS IN MUSIC WITH NONNEGATIVE TUCKER DECOMPOSITION FOR STRUCTURAL SEGMENTATION

Axel Marmoret[1]   **Jérémy E. Cohen**[1]   **Nancy Bertin**[1]   **Frédéric Bimbot**[1]

[1]Univ Rennes, Inria, CNRS, IRISA, France.

axel.marmoret@irisa.fr

ABSTRACT

Recent work has proposed the use of tensor decomposition to model repetitions and to separate tracks in loop-based electronic music. The present work investigates further on the ability of Nonnegative Tucker Decompositon (NTD) to uncover musical patterns and structure in pop songs in their audio form. Exploiting the fact that NTD tends to express the content of bars as linear combinations of a few patterns, we illustrate the ability of the decomposition to capture and single out repeated motifs in the corresponding compressed space, which can be interpreted from a musical viewpoint. The resulting features also turn out to be efficient for structural segmentation, leading to experimental results on the RWC Pop data set which are potentially challenging state-of-the-art approaches that rely on extensive example-based learning schemes.

1. INTRODUCTION

A common problem in Music Information Retrieval domain (MIR) is the design of musical content representations and features able to capture meaningful information in relation to a particular aspect of music. While short-term features are dominant in the literature, higher-scale features aiming to describe medium-term patterns and long-term structural properties tend to be much less addressed.

Recent work by Smith and Goto [1] has proposed the use of tensor decomposition to model repetitions in loop-based electronic music, with the purpose of separating tracks in audio content. In this paper, we explore the ability of the method to provide a sparse description of music by capturing and characterizing patterns at the bar-scale level in western pop songs in their audio form. As a testbed, we evaluate the effectiveness of the new features for structural segmentation, *i.e.* the task of retrieving the boundaries of the various musical sections (such as verses, choruses, intros, bridges...) which form a music piece.

We first recall (in section 2) the mathematical theory of the tensorial model called Nonnegative Tucker Decom-

position (NTD), and we provide detailed illustrations and interpretation of the NTD components on the audio recording of a well-known pop song. We then (in section 3) elaborate on a number of practical considerations related to NTD, which are needed to be taken into account when applying the model to real music data. In the last part of the article (sections 4 and 5), we report on experiments and results obtained with the NTD-derived features for structural segmentation of the RWC Pop Music data set [2].

2. NONNEGATIVE TUCKER DECOMPOSITION

2.1 Time-Frequency-Bar Tensor

Music in its audio form is often represented in the time-frequency domain as a spectrogram, *i.e.* a 2-dimensional matrix (further denoted as X). Along the x-axis, the temporal dimension unfolds, discretized as signal frames, while the y-axis is a frequency-related dimension (such as modules of the Fourier coefficients, pitches, constant-Q transforms, wavelet coefficients...).
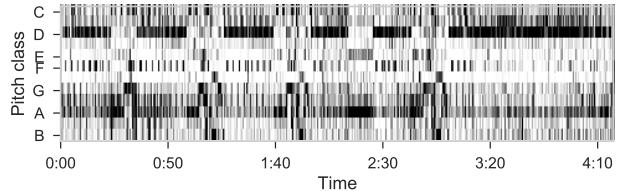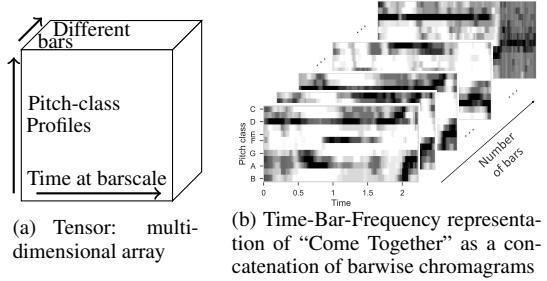
Figure 1: Chromagram of "Come Together", by The Beatles.

In this work, we describe music as chromagrams. Conventionally, the 12 rows represent the energy distribution across the song for each semi-tone of the classical western music scale, where a note and all its octave counterparts are represented in the same row. An example chromagram is shown in Figure 1.

In the tensorial approach, the temporal dimension is broken up into two distinct dimensions: a low-scale dimension representing time in terms of frame index normalized at the bar scale, and a high-scale time dimension representing the bar index within the entire piece. This new viewpoint makes it possible to represent a song as a third-order tensor $\mathcal{X}$ of size $F \times T \times B$, F being the size of the frequency dimension (12 in the case of chromas), T the number of frames used to describe bars (local time scale) and B the number of bars in the song (global time scale). We call $\mathcal{X}$ the Time-Frequency-Bar representation of the song which, from a data structure viewpoint, is the recasting of X as a 3D array. Tensor $\mathcal{X}$ can be seen as the concatenation

(a) Tensor: multi-dimensional array

(b) Time-Bar-Frequency representation of "Come Together" as a concatenation of barwise chromagrams

Figure 2: Principle and illustration of a Time-Bar-Frequency representation as a third-order tensor.

of local time-frequency representations, each of them characterizing the content of a bar, as illustrated on Figure 2. Note that, as bars can be of different lengths in absolute time, the frame hop depends on each bar, and is defined so that all bars contain the same number of frames.

2.2 Mathematical Model and Formalism

Let us denote as $\mathcal{X}$ the Time-Frequency-Bar representation of a song, with dimensions $F \times T \times B$. Assuming chroma coefficients are all nonnegative, $\mathcal{X}$ is also a nonnegative tensor. Computing a Nonnegative Tucker Decomposition (NTD) of $\mathcal{X}$ consists in finding 3 nonnegative factor matrices W, H and Q (corresponding to the three "modes" of the Time-Frequency-Bar tensor) and a nonnegative core tensor $\mathcal{G}$ which relates the three modes as of how to combine them to reconstruct (an approximation of) $\mathcal{X}$.

The dimensions $F' \times T' \times B'$ of the core tensor $\mathcal{G}$ are usually set to be lower than those of $\mathcal{X}$ (*i.e.* $F', T', B' \leq F, T, B$ respectively). As a consequence, matrices W, H and Q are respectively of dimensions $F \times F'$, $T \times T'$ and $B \times B'$ and they can be understood as transformed and compressed representations of the raw information conveyed across the three dimensions of the full tensor.

In conventional tensor-product notation [3], the approximation of $\mathcal{X}$ can be written in compact form as:

$$\mathcal{X} \approx \mathcal{G} \times_1 W \times_2 H \times_3 Q . \qquad (1)$$
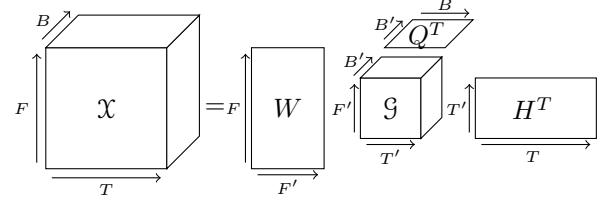
which rewrites, using element-wise notation, as:

$$\mathcal{X}(f,t,b) \approx \sum_{f',t',b'=1}^{F',T',B'} \mathcal{G}(f',t',b')W(f,f')H(t,t')Q(b,b') \qquad (2)$$

In particular, any given bar of index b is represented as:

$$\mathcal{X}(:,:,b) \approx W \left(\sum_{b'=1}^{B'} Q(b,b')\mathcal{G}(:,:,b') \right) H^T \qquad (3)$$

Figure 3 depicts a schematic 3-D representation of a NTD. NTD core dimensions F', T' and B' are assumed to be known (or set empirically) prior to the decomposition. As they are lower than the dimensions of their respective mode of the tensor, NTD achieves information compression via nonlinear dimensionality reduction. Indeed, for an original tensor of size $F \times T \times B$ (*i.e.* comprising $F.T.B$ numerical values), the NTD decomposition

Figure 3: Nonnegative Tucker Decomposition of tensor $\mathcal{X}$ in factor matrices W, H, Q, and core tensor $\mathcal{G}$, with their dimensions.

will total $F.F' + T.T' + B.B' + F'.T'.B'$ values. For example, in the decomposition presented further in Figure 4, the original tensor contains 102528 real positive values ($F, T, B = 12, 96, 89$), while only 3626 for the NTD ($F', T', B' = 12, 12, 10$).
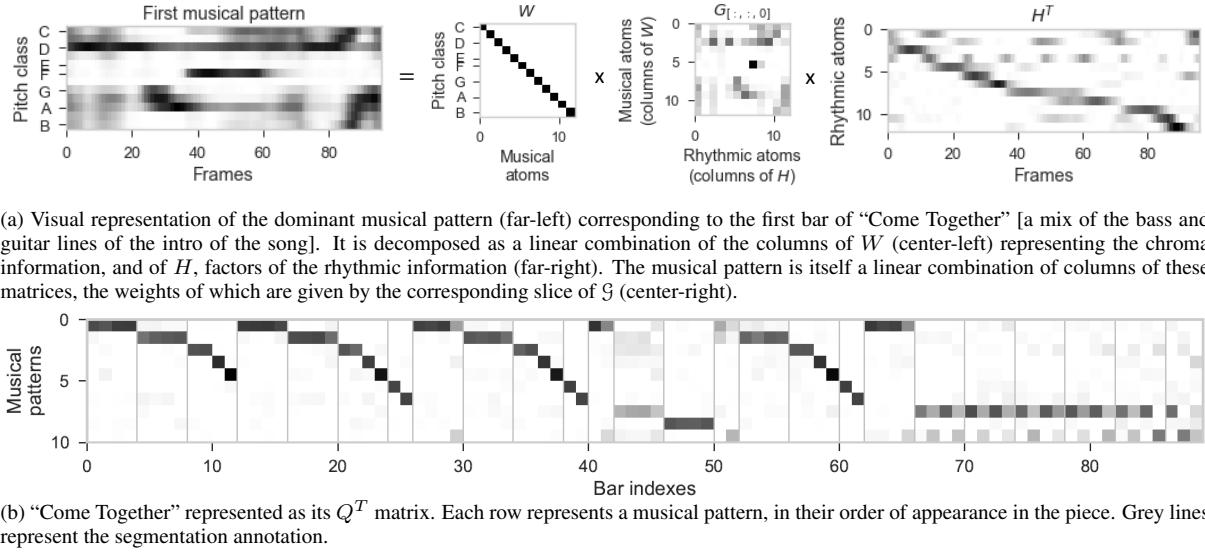
2.3 Interpretation of the NTD

Loosely speaking, music can be viewed as composed of musical events (notes, percussive sounds, ...) occurring non-randomly in bars. Under that assumption, bars can be modeled as the combination of a limited set of time-frequency templates along time within a bar, according to some rhythmic values, such as "half notes" or "beamed eight notes" for example. This is the purpose of a conventional musical score, where the major part of symbolic information represents pitch and rhythm. Following that idea, it is a very popular goal in MIR to design methods for turning back musical content (in audio form) into a sparse combination of musical events and temporal activations, as is the case, for instance, with Nonnegative Matrix Factorization (NMF) for music transcription [4]. Moreover, music often contains repetitions: different bars can entirely or partly share similar content. For instance, beside almost identical repetitions, some instrumental lines can reoccur in different contexts: an identical bass line in a verse and in a guitar solo, for example.

Combining these observations, we assume that each bar can be represented as the nonnegative combination of a few "musical patterns" (as NMF would do), where a "musical pattern" is itself a sparse combination of musical events and rhythmic activations at the bar scale (for example a melodic line, or a drum fill). Repetitions imply that some of these musical patterns should appear in several bars across a piece. NTD offers an ideal framework to model these properties for music decomposition, musical patterns being efficiently and sparsely shared across bars.

In the NTD, the W matrix represents the musical events, such as the most recurrent notes or chords. H represents rhythmic activations at the bar scale, for example 4 quarter notes on the beats. Then, each 2D "slice" of the core $\mathcal{G}$ linking these two matrices defines a musical pattern, as a linear combination of some of their columns (musical and rhythmic atoms): for example bass drum hits on the on-beats and snare hits on the off-beats. Finally, Q indicates, for each bar, the combination of musical patterns forming it (generally a few) and their respective intensity.

Figure 4 provides a detailed example of the various NTD components stemming from "Come Together" by the Beatles. While the upper part of the figure illustrates the

(a) Visual representation of the dominant musical pattern (far-left) corresponding to the first bar of "Come Together" [a mix of the bass and guitar lines of the intro of the song]. It is decomposed as a linear combination of the columns of W (center-left) representing the chroma information, and of H, factors of the rhythmic information (far-right). The musical pattern is itself a linear combination of columns of these matrices, the weights of which are given by the corresponding slice of $\mathcal{G}$ (center-right).

(b) "Come Together" represented as its Q^T matrix. Each row represents a musical pattern, in their order of appearance in the piece. Grey lines represent the segmentation annotation.

Figure 4: Visualizations of the NTD of Come Together by The Beatles, with ranks $T' = 12$ and $B' = 10$.

dominant musical pattern for the first bar together with its decomposition in NTD, the lower part depicts the description of the entire piece via the Q^T matrix of the NTD. This example has been obtained from the chromagram represented in Figure 1. Because the song is expressed on the 12-chroma scale, we expect little compressibility with respect to this dimension. We hence simplify the model by fixing W to the 12-size identity matrix. This means that each semi-tone is represented by one and only one column of W. For higher dimensions or different representations, columns of W could represent a wider range of harmonic or percussive sounds, chords, or any other frequency pattern. Conversely, ranks T' and B' (respectively the second dimension for H and Q) are adjustable parameters of the model. In the decomposition presented in Figure 4, they have been set to $T' = 12$ and $B' = 10$. All columns of H and all slices of the core linking the factor matrices W and H, which define the musical patterns, are l_2 normalized (*i.e.* divided by their standard deviation).

3. PRACTICAL INSIGHTS ON THE NTD

In this section, we discuss a number of considerations which are bound to have an impact on the actual result of the NTD-based representations and must therefore be taken into account in practical situations.

3.1 NTD Algorithm

NTD can typically be computed by minimizing the following non-convex objective function with respect to the non-negative matrices W, H, Q and the core tensor $\mathcal{G}$:

$$\|\mathcal{X} - \mathcal{G} \times_1 W \times_2 H \times_3 Q\|_F^2 \qquad (4)$$

While a direct global minimization of Eqn (4) is not tractable in general, a standard approach in the literature is to resort to alternating optimization. Following [5], we solve Eqn (4) for W, H, Q and $\mathcal{G}$ alternatively. It can be shown that each of these steps means solving a matrix non-

negative least-square problem of the form:

$$\min_{Z \geq 0} \|Y - AZ\|_F^2 \qquad (5)$$

for some matrices Y, A, Z. This problem is convex, and it is possible to solve it exactly, or up to an arbitrary precision. An efficient algorithm for solving matrix nonnegative least squares with high precision is the Hierarchical Alternating Least Squares, and we used an accelerated variant of it to speed-up computation [6]. The problem of updating $\mathcal{G}$ is also a nonnegative least squares problem, but not a matrix one. Therefore, to update the core tensor $\mathcal{G}$, we used a proximal gradient with optimal step [7, Ch. 10].

It can be shown that the proposed alternating algorithm is guaranteed to converge to a stationary point of the objective function (4), since it boils down to an alternating proximal gradient algorithm with optimal step [8]. In practice, we used a stopping criterion based either on a fixed maximal number of iterations or on a fixed minimal tolerance of improvement between two successive updates. The entire code, along with experimental notebooks, are published and open-source [1] . Under this implementation, computing the NTD for "Come Together" (4'16" song) with our algorithm takes approximately 15 seconds on a laptop with an Intel® Core(TM) i7 processor and 16GB of RAM.

3.2 Robustness of the NTD

At least two issues with the NTD make the output of any algorithm highly dependent on the initialization. First, there might be several solutions $W, H, Q, \mathcal{G}$ that provide the same (or a very similar) estimate $\hat{\mathcal{X}} \approx \mathcal{X}$. This problem, known as identifiability deficiency, has been little studied for NTD, and established identifiability conditions are very restrictive [9]. Moreover, these conditions are hard to check in practice. Therefore it is unreasonable to assess the identifiability of the NTD in our application. As a consequence, this means that there might be infinitely many

---

[1] https://gitlab.inria.fr/amarmore/musicntd/-/tree/0.1.0

solutions to minimizing Eqn (4) that are, from an optimization point of view, equally satisfying. Second, even in the case where the NTD is identifiable, the cost function of (4) is highly non-convex, and local algorithms can only hope to recover a local minimum at best.

These two issues combined give rise to a high dependency of the solution on the initial condition: from two different initializations, two different results – most probably non-identifiable local minima – are likely to be obtained. We have observed such situations in our investigations, with various initializations indeed resulting in different outputs. However, in most cases the decomposition would provide results that were reasonably interpretable from a musical perspective. In particular, when we initialized the algorithm with the absolute values of the Higher Order SVD [10] computed with the Tensorly toolbox [11], the procedure consistently provided satisfying results for segmentation, as detailed further in our experiments.

3.3 Rank Selection

The ranks F', T' and B' of the decomposition are crucial parameters of the NTD model. Indeed, low ranks tend to over-compress information in the data, failing to uncover relevant structural information in the song, while high ranks may give too much importance to details in the data, resulting in the inability of the model to group similar patterns in a same class of representations.

As developed further in section 5.4, our experiments indicate that the optimal ranks are probably specific for each song, which can be easily understood as a consequence of the diversity of intrinsic variability across music pieces. Providing an efficient method for selecting the ranks is a challenging topic, left to future work.

4. NTD-BASED SEGMENTATION

To study further the relevance of the NTD representation, we evaluated it in the context of structural segmentation. To our knowledge, this is the first attempt to exploit tensorial representations for this purpose.

4.1 Autosimilarity for Describing Structure

The autosimilarity matrix $X^T X$ of a music piece (X being its time-frequency representation) is commonly used in structural segmentation. Indeed, similar portions of the piece are likely to have high correlation values. A high density of high values around the diagonal is expected in passages with strong internal similarities, whereas low local correlations would indicate a change in homogeneity. In the ideal case, structural segments appear as consistent blocks with a high level of internal correlation while segment boundaries are points connecting such blocks, surrounded by zones of low cross-correlation.

Nonetheless, music signals usually generate dense autosimilarity matrices, as dissimilar segments in the musicological/perceptive sense (for example a guitar line on the chorus opposed to one in the verse) may still be close

in terms of signal properties. While similar parts generate high correlation blocks, it can be harder to characterize segments boundaries when the same instruments are played in all segments (even when playing different lines).

In the present work, we replace $X^T X$ by an autosimilarity matrix $\tilde{Q}\tilde{Q}^T$ computed from the row-wise normalized Q matrix (denoted $\tilde{Q}$), and study its capacity to provide an efficient representation for structural segmentation.
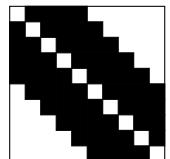
Our assumption is that bar descriptions provided by $\tilde{Q}$ provide a better contrast between similar and dissimilar musical constituents. For instance, we expect two different lines of the same instrument to generate different musical patterns, resulting in lower similarity, whereas compressive effects of NTD will increase correlation of similar events in the transformed space. In that sense, NTD can be seen as a way to uncover piece-dependent features for describing bars, which can then be used to group the bars according to their relative similarity.

Figure 5 depicts the "barwise" autosimilarity matrix of the chromagram X of "Come Together": the content of each bar of the signal has been vectorized, and similarity is computed between these barscale vectors. This matrix is compared to the autosimilarity of the $\tilde{Q}$ matrix, presented on Figure 4b. This figure visually supports the hypothesis that autosimilarity matrices are sparser when computed from the matrix Q rather than from the chromas X. Still, highly similar blocks seem to be preserved.

4.2 A Segmentation Algorithm Using Autosimilarity

To assess this hypothesis, we implemented a segmentation algorithm based on the principle of a sliding convolution kernel along the diagonal of the autosimilarity matrix.
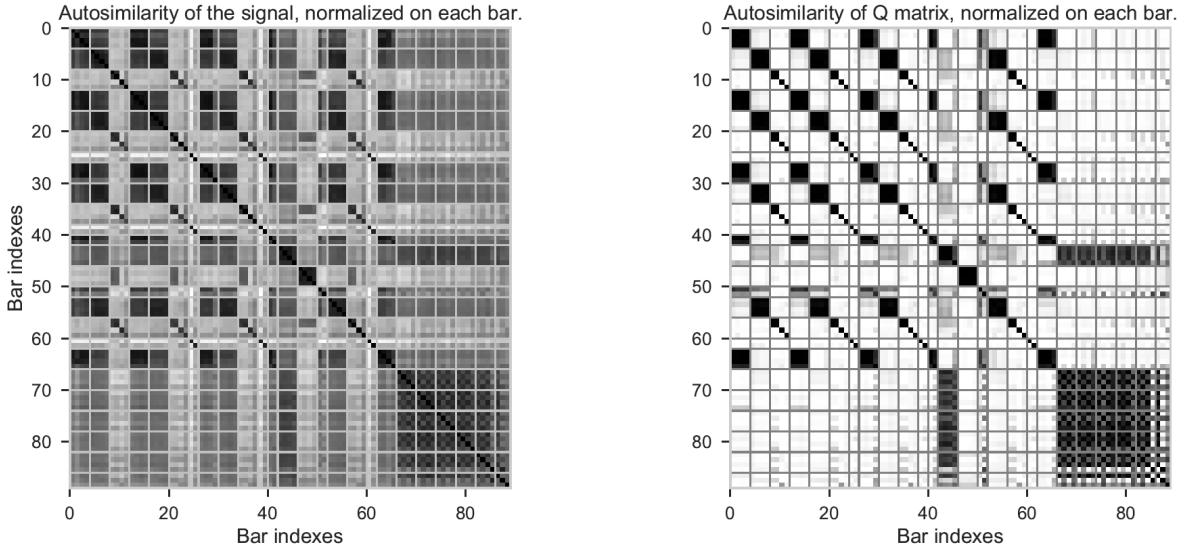
This kernel is a square binary matrix, whose entries are non-zero only on the lower and upper 4 subdiagonals around the main diagonal (Figure 6). In other terms, denoting k_{ij} the kernel elements, $k_{ij} = 1$ if $1 \leq |i - j| \leq 4$. Otherwise, $k_{ij} = 0$.

Figure 6: Kernel of size 10

For every possible segment (b_1, b_2), a kernel of size $n = b_2 - b_1 + 1$ is convolved with the corresponding autosimilarity sub-matrix (restricted to the bars between b_1 and b_2) which is then normalized by the size of the segment. This leads to a raw convolution score: $c_{b_1,b_2} = \frac{1}{n} \sum_{i,j=0}^{n-1} k_{ij} a_{i+b_1, j+b_1}$. The kernel aims at detecting local similarities within the 8 bars surrounding each bar. The more similar this surrounding is, the higher the score.

In addition, we combine the kernel score with a regularity penalty $p(n)$, depending on the size n of the segment. Indeed, in pop music in general (and in the MIREX10 RWC Pop annotations in particular [12]), the distribution of musical segment sizes (in bars) tend to be centered around 8, and they are more likely to be even than odd. In the experiments reported in the next section, we set empirically, $p(8) = 0$, $p(n) = \frac{1}{4}$, if n is a multiple of 4, $p(n) = \frac{1}{2}$ if n is a multiple of 2, and $p(n) = 1$ if n is odd. This penalty modifies the raw convolution score as

Figure 5: Barwise l_1-normalized autosimilarity matrices for "Come Together" (0: white - 1: black). Left: barwise chromagram autosimilarity - Right: autosimilarity of $\tilde{Q}$ matrix from Figure 4b. Grey horizontal and vertical lines represent the segmentation annotation.

follows:

$$c'_{b_1,b_2} = c_{b_1,b_2} - \lambda p(n) c_{k8}^{max} \qquad (6)$$

where c_{k8}^{max} is the maximum of the raw convolution score over all restrictions of size 8 bars within the piece, in order to cope with potential discrepancies in sparsity across autosimilarity matrices for different pieces. In practice, λ is fitted by cross-validation.

Finally, segment boundaries are found by a dynamic programming algorithm, inspired from [13]: it keeps the sequence of segments maximizing the global cost defined as the sum of all segment costs.

5. EXPERIMENTS

The proposed method was applied to the $\tilde{Q}^T$ representation and tested on the "structural segmentation" task, as defined in the MIREX campaigns [14], on the 100 songs from the RWC Pop database [2]. MIREX10 annotations [12] serve as the reference segmentation (1680 segments). We compare our results with state-of-the-art methods listed below.

5.1 Related Work

In the context of structural segmentation, numerous methods try to detect segment boundaries from the autosimilarity matrix, or from an "affinity matrix" derived from it.

The use of autosimilarity for segmenting music structure probably traces back to Foote [15]. In this work, structural boundaries are detected by applying a kernel along the diagonal, as described above. Foote's kernel though aims at detecting "novelty" in the signal's autosimilarity matrix, by comparing inter-similarity between the near past and near future at the current point. A high novelty should indicate a low inter-similarity between past and future, hinting towards a boundary between segments.

More recently, convex NMF was used for segmenting a pre-processed autosimilarity matrix [16]. A variant of NMF decomposition is used to enforce the feature space

(here, similarity between different bars) to be contracted in convex combinations of columns of the autosimilarity matrix. Factorization results are thus interpreted as the most similar bars, which can then be processed into sections.

Spectral clustering can also be used. In [17], an affinity matrix is computed from the signal, where the similarity is obtained with k-nearest neighbors and time-proximity rules. Then, interpreting this matrix as a graph, and its values as vertices connectivity, this method studies the eigenvectors of its Laplacian. These eigenvectors can be interpreted as principally connected vertices, forming cluster classes for segmentation.

We primarily compare the NTD method with these techniques for two reasons. First, they are implemented in the MSAF toolbox [20]. Second, they reach state-of-the-art performance among "blind" methods for structural segmentation, *i.e.* methods which, like NTD, do not resort to extensive training from examples. Note that the segmentation results we obtained with MSAF, though, are slightly worse ($\approx 3/4\%$) than those obtained at MIREX 2016 [21], possibly due to evolutions of the toolbox itself in the interval. We did not tune any of the default parameters.

As current state-of-the-art, we selected the algorithm from [18] since it ranked first in this task in the last MIREX campaigns. However, as opposed to the previous methods, it requires supervised training from many examples.

5.2 Downbeat-Synchronous Alignment

By construction, the boundaries estimated by the NTD-based approach are aligned on downbeats, which is not the case for the techniques we use as baseline comparisons. As segments generally start and end on downbeats of the song, this alignment could induce a bias favouring our technique. To compensate for this, in addition to the segmentation scores computed with the original boundaries, we compute the scores after having aligned boundaries on the closest downbeat. We call this condition "Aligned on downbeats".

Algorithm		$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
NTD-based autosimilarity		53.3%	62.1%	56.6%	66.8%	78.1%	71.1%
Barwise chromagram autosimilarity		43.1%	45.7%	43.9%	64.8%	68.0%	65.8%
Foote Novelty [15]	Original	29.7%	22.3%	25.1%	63.9%	48.6%	54.5%
	Aligned on downbeats	42.0%	30.0%	34.5%	67.1%	47.7%	55.0%
CNMF [16]	Original	22.8%	21.5%	21.5%	46.8%	45.1%	44.7%
	Aligned on downbeats	31.6%	28.1%	28.8%	50.7%	45.4%	46.5%
Spectral Clustering [17]	Original	31.2%	30.5%	29.4%	60.7%	60.8%	58.1%
	Aligned on downbeats	49.2%	45.0%	45.0%	65.5%	60.6%	60.3%

Table 1: Averaged segmentation scores, and their comparison with several "blind" reference methods.

Algorithm	$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
NTD, with "oracle ranks" for each song	67.1%	78.2%	71.5%	78.5%	90.2%	83.1%
Neural Networks [18], results from MIREX 2015 [19]	80.4%	62.7%	69.7%	91.9%	71.1%	79.3%

Table 2: Averaged segmentation scores in the "oracle ranks" condition, compared to the current state-of-the-art (non-blind) method.

In addition, we also processed the barwise autosimilarity obtained directly from the chromagram, in order to measure the impact of the NTD-derived representation vs the raw time-frequency representation.

5.3 Implementation Details

RWC Pop signals are sampled at 44100Hz. Bars were estimated by the madmom toolbox [22]. Chromas were extracted from the Constant-Q Transform of the signal with 32-sample hop using Librosa [23], then mapped to 96 equally spaced chroma vectors per bar. This results in a chromagram X with 12 rows and $96 \times B$ columns. Tensors were handled with the Tensorly toolbox [11]. We use our own implementation of the NTD algorithm (see Section 3.1). Segmentation performance was computed with the mir_eval toolbox [24].

5.4 Results

Segmentation performance is evaluated with metrics based on "hit-rate". The hit-rate considers a boundary as correct if it coincides with a boundary in the reference segmentation within some time window. From the count of correct and incorrect segment boundaries, we compute Precision, Recall and F-measure. Tolerance windows were chosen to be 0.5s and 3s, in line with MIREX standards.

As mentioned in Section 3.3, the ranks of the NTD strongly influence the decomposition and, consequently, the segmentation results. Ranks T' and B' are treated as adjustable parameters, and can vary between 12 to 48, with a step of 4. W is fixed to the 12-size identity matrix. The impact of T' and B' is investigated under two rank selection conditions.

In the first condition (Table 1), the RWC Pop data set is divided in two subsets (songs with odd vs even ID number), which are alternatively used as tuning (for global optimization of the ranks and the penalty parameter λ) and test data sets, in a 2-fold cross-validation fashion. Results shown in the table are averaged over the two folds. Hence, in this condition, all songs of a test data subset are decomposed with the same ranks, namely $T', B' = 40, 28$ for odd songs, and $48, 24$ for even ones.

In the second condition, presented in Table 2, the NTD ranks are fitted a posteriori on each song individually: for each tolerance value, separately, we select the ranks leading to the best F-measure for the given song. This is called the "oracle ranks" condition, corresponding to the situation where a "perfect" rank selection procedure would exist. Resulting scores provide an (optimistic) performance upper bound.

These two tables exhibit very competitive results. In the first (and most realistic) condition, NTD-based segmentation performance exceeds those of the reference "blind" methods segmentation. In the "oracle ranks" condition, the NTD provides higher F-measures than the state-of-the-art, showing strong potential for the technique, provided an efficient rank selection method is eventually developed.

6. CONCLUSION AND FUTURE WORK

Designing relevant audio features from music remains one of the key questions in many MIR tasks. In this paper, we have proposed a three-way tensor representation of music in frequency, short-term (frames) and mid-term (bars), and means to decompose it under the low-rank Nonnegative Tucker Decomposition (NTD) model. This decomposition turns out to be able to provide a compressed representation of interest, capturing salient patterns in music.

We have illustrated the benefits of the method in a structural segmentation task. The NTD-based representation allows to compute a new type of autosimilarity matrix which exhibits a better contrast than those directly computed on 2D time-frequency representations, and seems well-suited to identify musical patterns at the "right" time-scale for the task. Experimental results are promising and show a potential to compete with state-of-the-art approaches, may they be "blind", or greedier on training data.

Additional research is required to consolidate the technique. First, as our experiments show, a rank selection criterion would drastically improve segmentation performance. Second, the model does not yet incorporate the notion of proximity between patterns themselves. In parallel, a number of theoretical questions on model identifiability and algorithmic convergence also remain open.

7. REFERENCES

[1] J. B. Smith and M. Goto, "Nonnegative tensor factorization for source separation of loops in audio," in *2018 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018, pp. 171–175.

[2] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Popular, Classical and Jazz Music Databases," in *ISMIR*, vol. 2, 2002, pp. 287–288.

[3] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[4] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.* IEEE, 2003, pp. 177–180.

[5] A. H. Phan and A. Cichocki, "Extended HALS algorithm for nonnegative tucker decomposition and its applications for multiway analysis and classification," *Neurocomputing*, vol. 74, no. 11, pp. 1956–1969, 2011.

[6] N. Gillis and F. Glineur, "Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization," *Neural computation*, vol. 24, no. 4, pp. 1085–1105, 2012.

[7] A. Beck, *First-order methods in optimization.* SIAM, 2017.

[8] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014.

[9] G. Zhou, A. Cichocki, Q. Zhao, and S. Xie, "Efficient nonnegative tucker decompositions: Algorithms and uniqueness," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 4990–5003, 2015.

[10] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[11] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "Tensorly: Tensor learning in python," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 925–930, 2019.

[12] F. Bimbot, G. Sargent, E. Deruty, C. Guichaoua, and E. Vincent, "Semiotic description of music structure: An introduction to the quaero/metiss structural annotations," in *53rd Int. Conf. Audio Engineering Society*, 2014.

[13] G. Sargent, F. Bimbot, and E. Vincent, "Estimating the structural segmentation of popular music pieces under regularity constraints," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 2, pp. 344–358, 2016.

[14] J. S. Downie, "The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research," *Acoustical Science and Technology*, vol. 29, no. 4, pp. 247–255, 2008.

[15] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *2000 IEEE Int. Conf. on Multimedia and Expo. ICME2000. Proc. Latest Advances in the Fast Changing World of Multimedia*, vol. 1. IEEE, 2000, pp. 452–455.

[16] O. Nieto and T. Jehan, "Convex non-negative matrix factorization for automatic music structure identification," in *2013 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2013, pp. 236–240.

[17] B. McFee and D. Ellis, "Analyzing song structure with spectral clustering," in *ISMIR*, 2014, pp. 405–410.

[18] T. Grill and J. Schlüter, "Music boundary detection using neural networks on combined features and two-level annotations," in *ISMIR*, 2015, pp. 531–537.

[19] MIREX, "Results in the structural segmentation task at the 2015 MIREX contest," 2015. [Online]. Available: https://nema.lis.illinois.edu/nema_out/mirex2015/results/struct/mrx10_2/summary.html

[20] O. Nieto and J. P. Bello, "Systematic exploration of computational music structure research," in *ISMIR*, 2016, pp. 547–553.

[21] MIREX, "Results in the structural segmentation task at the 2016 MIREX contest," 2016. [Online]. Available: https://nema.lis.illinois.edu/nema_out/mirex2016/results/struct/mrx10_2/summary.html

[22] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proc. 24th ACM Int. Conf. on Multimedia*, Amsterdam, The Netherlands, 10 2016, pp. 1174–1178.

[23] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. 14th python in science Conf.*, vol. 8, 2015.

[24] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "mir_eval: A transparent implementation of common MIR metrics," in *ISMIR*, 2014.

MOVING IN TIME: COMPUTATIONAL ANALYSIS OF MICROTIMING IN MARACATU DE BAQUE SOLTO

Matthew E. P. Davies [1,4]   **Magdalena Fuentes**[2]   **João Fonseca**[3]
Luís Aly[3,4]   **Marco Jerónimo**[3]   **Filippo Bonini Baraldi**[5,6]

[1] University of Coimbra, CISUC, DEI, Portugal
[2] CUSP, MARL, New York University, USA
[3] University of Porto, Faculty of Engineering, Portugal
[4] INESC TEC, Portugal
[5] Ethnomusicology Institute (INET-md), FCSH, Universidade Nova de Lisboa, Portugal
[6] Centre de Recherche en Ethnomusicologie (CREM-LESC), Paris Nanterre University, France
`mepdavies@dei.uc.pt`

ABSTRACT

"Maracatu de baque solto" is a Carnival performance combining music, poetry, and dance, occurring in the Zona da Mata Norte region of Pernambuco (Northeast Brazil). Maracatu percussive music is strongly repetitive, and is played as loud and as fast as possible. Both from an MIR and ethnomusicological perspective this makes a complex musical scene to analyse and interpret. In this paper we focus on the extraction of microtiming profiles towards the longer term goal of understanding how rhythmic performance in Maracatu is used to promote health and well-being. To conduct this analysis we use a set of recordings acquired with contact microphones which minimise the interference between performers. Our analysis reveals that the microtiming profiles differ substantially from those observed in more widely studied South American music. In particular, we highlight the presence of dynamic microtiming profiles as well as the importance of the choice of time-keeper instrument, which dictates how the performances can be understood. Throughout this work, we emphasize the importance of a multidisciplinary approach in which MIR, audio engineering, and ethnomusicology must interact to provide meaningful insight about this music.

1. INTRODUCTION

"Maracatu de baque solto", also known as "Maracatu rural", is a Carnival performance combining music, poetry, and dance, occurring in the Zona da Mata Norte region of Pernambuco (Northeast Brazil). Most inhabitants of this region, dominated by the sugar cane monoculture, are rural workers with very modest income, who invest most of

Figure 1: Maracatu de baque solto "Leão de Ouro de Condado" during a Carnival parade in Tupaoca (Pernambuco), Feb. 28th, 2017. Photo credit: Filippo Bonini Baraldi.

their time and money to participate in the Carnival "desfile" (parade), taking place every year in Recife, the state capital. More than 100 groups of Maracatu de baque solto, of different sizes (ranging from 15-20 members up to 200 members) are currently active, each with their own headquarters ("sede") which are generally linked to individual families. A Maracatu performance is shown in Fig. 1.

Maracatu de baque solto differs from another Carnival performance with a similar name, Maracatu "de baque virado," not only in terms of its musical and choreographic features, but also because it has remained a very local cultural practice. Indeed, while Maracatu de baque virado, like other music from Pernambuco (e.g., forró, coco de roda, ciranda), has recently spread out nationally and internationally, Maracatu de baque solto (hereafter shortened to Maracatu) is only performed within a radius of about $100\,\mathrm{km}^2$ and is strongly linked to the local afro-indigenous spiritual and religious practices, specifically, the jurema-umbanda worship [1]. This local dimension explains why, barring a few exceptions, it remains a largely understudied cultural expression. To the best of our knowledge no in-depth analysis of its music has ever been realised.

Previous field research conducted in Condado, a small city located in the Zona da Mata Norte region, suggests that

the protective function of Maracatu is locally expressed by two key concepts: "consonância" (consonance) and "fechar" (closure) [2, 3]. Consonance points to a particular way of behaving, of dancing together, and of producing sounds. It is associated with an idea of high interpersonal coordination, as opposed to the idea of "desmantelo" (fracture and breaking up). The expression "fechar o Maracatu" (closing the Maracatu) refers to various aesthetic strategies used to protect the individual and the community from the threats that are at stake during Carnival. The percussive nucleus of Maracatu plays as fast and loud as possible, saturating the acoustical space and "filling" each metrical position in order to avoid any silence. To this end, the study of systematic microtiming, i.e., intentional deviations from strict metronomic timing, can be a promising first step towards understanding Maracatu performances.

In December 2019, 13 members of a Maracatu group "Leão de Ouro de Condado" were invited to Lisbon. Several public performances were organised which culminated in a two-part event: a parade in which musicians and dancers moved through the streets, followed by a fixed location, outdoor performance. We focus on the recordings obtained from the latter, and perform an exploratory analysis of the microtiming in the percussionists' performances.

Our research is part of the emerging topic of computational ethnomusicology [4, 5], and falls within a growing body of work examining the role of microtiming and rhythmic structure and its relationship to groove and musical embodiment [6–10]. It also intersects with the existing MIR literature on the analysis of rhythm in South American music [11–16]. In our specific context, we face two prominent, interconnected challenges. In the absence of formal theories about Maracatu, there is little basis on which to pose research questions that computational analysis (e.g., using MIR techniques) could address. From a more practical perspective, the very nature of Maracatu performance: musicians in tight proximity to one another, playing very loudly, and moving between multiple ad-hoc external locations, creates technical difficulties for the high-quality signal acquisition necessary for temporally-precise analysis of musical timing.

Our methodology to address these challenges is to conduct the research from a strongly multidisciplinary perspective. We directly leverage technical expertise in audio engineering for signal acquisition, music signal processing and machine learning for computational rhythm analysis, and ethnomusicology to guide the interpretation of the findings based on long-term field research. By necessity, this leads to an exploratory approach concerning the presence and use of microtiming in Maracatu, where the computational analysis can be used as a means to infer new understanding about the musical practice.

To enable the isolated analysis of the microtiming of each percussionist, we use contact microphones attached to each individual instrument. We adapt a state-of-the-art approach for microtiming analysis [12] and investigate both "within-instrument" microtiming, where onset and beat information are specific to a given instrument,

and "between-instrument" microtiming, where a "time-keeping" instrument provides the beat reference. Our main findings demonstrate that the choice of the time-keeper is critical and can change the interpretation of the performance. Furthermore, we observe microtiming profiles that change dynamically within given pieces of music, and between pieces of music. Thus, the notion of a single characteristic microtiming profile appears not to apply to the set of Maracatu recordings under investigation here.

The remainder of this paper is structured as follows. Section 2 provides an overview of the musical structure and instruments of Maracatu. Section 3 describes the signal acquisition process. Section 4 summarizes the microtiming modelling approach, with our main findings presented in Section 5. We conclude the paper in Section 6 with a reflection on the broader impact of the research.
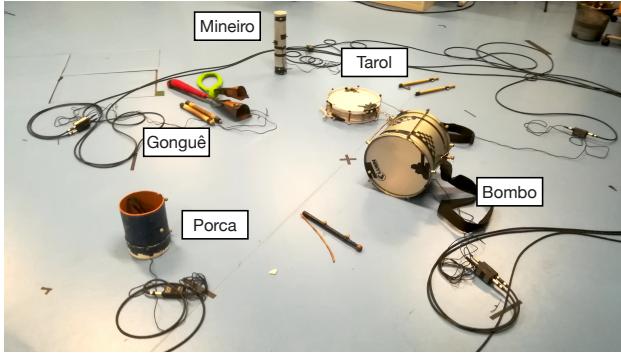
2. MARACATU DE BAQUE SOLTO

Maracatu is a combination of various elements: two to four wind instruments (trumpet and trombone), played by "musicos" (musicians), and a nucleus of five percussionists called the "terno." During performances, the musicos and terno act in close cooperation with a poet (the "mestre de apito"). When the poet improvises short verses about 30 s in duration, the musicians remain silent and the dancers and public remain still. When the poet finishes his verses and blows his whistle ("apito"), the percussionists and the wind musicians play for about the same duration and everybody dances. During this time, the poet prepares his next verses. This alternating pattern remains throughout the performance, which may last up to eight hours.

Maracatu percussive music is highly repetitive, and played as loud and as fast as possible. These features give rise to a very strong euphoria in the dancers and the public. Just a few rhythmical patterns exist in Maracatu and depend on the metrical form that the poet is following: the main genres are "marcha" and "samba," although the samba of Maracatu has nothing to do with the well-known samba music of Brazil. In Maracatu, marcha and samba indicate both the metrical subdivision of the poet's couplets, the rhythm played by the terno, and the melodies played by the musicos. Various melodies may be associated to the marcha pattern and/or samba pattern, depending on the melodic line that the poet chooses to sing his verses.

In Maracatu, the five percussion instruments of the terno, shown in Fig. 2, are: *Bombo* – a bass drum-like instrument played with two sticks, one for each side. We refer to Bombo High as the upper skin, and Bombo Low as the lower. *Onguê* – an iron instrument comprised of two bells of different pitches. We refer to Gonguê High as the higher, and Gonguê Low as the lower pitched bell. *Porca* – a friction drum, played with a damp cloth holding the stick. *Tarol* – similar to a small snare drum, but thinner. *Mineiro* – a metal tube filled with beads or other small objects, which is shaken to create a rattle-type sound.

Following transcriptions in [17] for both the marcha and samba, the Porca plays a regular quarter note pattern. Likewise for the marcha, the lower bell of the Gonguê has

Figure 2: The instruments of the terno (with identifying labels overlaid) including the connection of the contact microphones.

the same pattern. Thus, for marcha we assume that either could provide a time-keeping role.

3. DATA ACQUISITION

In this work, we wished to analyse each percussionist's performance in isolation. While multiple microphone setups have been successfully used to acquire separated audio data for rhythmic and microtiming analysis [18], the physical arrangement of the Maracatu percussionists in a tight circle, together with the very loud playing style, makes this impractical and highly prone to "spillage." In turn, this would make any subsequent annotation and microtiming analysis extremely challenging. A promising alternative is to consider the use of contact microphones.

For this study, we used the Schertler Basik Set universal contact microphone. Each microphone includes a phantom-power adaptor box which delivers a line-level signal, with a 60 Hz–15 kHz frequency response. The sensitivity on the instrument is −34 dB (time-averaged sound level). We found these microphones provided high-quality audio recordings with minimal spillage and distortion.

For the microphone placement, we sought to balance the optimum location for sound capture while minimizing any impact to each musician's playing style. Given the small size of the Schertler pickup (less than 1 cm in diameter), this aspect was relatively straightforward.

We placed two pickups on the Gonguê – one per bell, and two on the Bombo – one per skin. For the remainder, the Tarol, Porca, and Mineiro we used a single pickup. The contact microphones were individually connected to a Motu UltraLite-mk3 Hybrid (USB/Firewire) with nominal gain at the input and no further processing. The recording session consisted of discrete, synchronised tracks, one per microphone, recorded in a Pro Tools 2019 mixing session configured to record at 44.1 kHz, with 16-bit depth.

We used this setup in the fixed location performance, rather than the parade. Over the total performance duration of 48 minutes, we partitioned the performance into 34 individual pieces (i.e., editing out the poetry), with a mean duration of 39.4 s. Of these 34 pieces, 28 were marcha, 5 were samba, and in one piece the percussionists played

samba, while the musicos played marcha.

4. MICROTIMING MODEL
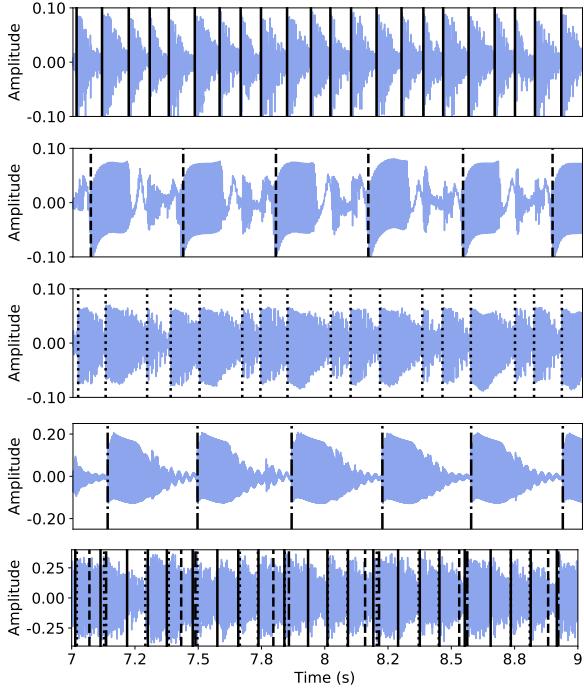
In order to undertake any assessment of the microtiming present in recordings, we must obtain precise temporal markers which indicate the note onset positions. Following existing work in microtiming analysis [12, 14] it is necessary to create a reference beat grid against which to compare the locations of performed onsets with quantised beat and/or sub-beat positions. In this way, each beat interval can be assigned a normalised duration of 1, and thus a rhythmic pattern containing four equal sub-divisions would occur at normalised positions 0, 0.25, 0.50, and 0.75. When summarising this information over multiple beats, it is possible to observe systematic microtiming patterns–called microtiming profiles [12]–, where, in the case of Brazilian samba, the third and fourth sub-divisions of the beats have been shown to occur ahead of their quantised position [12, 14]. Note that because the relative position of onsets is normalized with respect to the beat interval, the modelling of microtiming profiles is independent of tempo changes, which allows us to visualise their change over time in a consistent manner. In this section, we first describe the means by which onsets and beat annotations were obtained, followed by the technique used to estimate microtiming profiles through time.

4.1 Onset Annotation

The 34 pieces in the Maracatu performance totalled approximately 22 minutes. Across all 7 channels, this led to a total of 238 contact microphone signals to be analysed. As shown in Fig. 3, even though the separation between channels is mostly very good, some spillage still occurred. This is especially prominent between the bells of the Gonguê which are physically connected. We also observed spillage where one instrument had yet to begin playing and the vibrations carried through the air were picked up thanks to the extremely high sensitivity of the contact microphones.

Given our proposed signal acquisition process using contact microphones, we assumed that largely isolated percussion tracks would be relatively straightforward for a well-known onset detection method using deep neural networks [19]. On this basis, we hoped to be able obtain reliable onset information in a semi-automatic way, where minor corrections (shifts, insertions, and deletions) could be performed. In practice, we found that the rather unusual waveform shapes of the Gonguê, Porca, and Mineiro events created numerous problems for the onset detection system and thus provided little benefit over manual annotation from scratch. Indeed, the recordings of the Mineiro were so challenging to annotate in a precise and consistent way, that we chose not to include them at this stage of our analysis. Ultimately, we selected four instruments: two instruments with time-keeping roles (Porca and Gonguê Low) and two rhythmically expressive instruments in which to observe microtiming (Tarol and Bombo High).

To provide the final onset annotations we followed the

Figure 3: Illustration of signals with annotated onsets in approximately one bar. Top to bottom: Tarol with annotations (solid line); Porca with annotations (dashed line); Bombo High with annotations (dotted line); Onguê Low with annotations (dash-dotted line); Audio mixture with overlaid onset times of the four instruments.

methodology applied to Brazilian samba [12] and adapted an existing deep neural network [20] and retrained it specific to each instrument using a subset of manually annotations. Even using *instrument-adapted* networks, some manual correction was required, both to contend with issues of temporal localization as well as extra and missed detections. As a coarse indication of the annotation effort, we highlight 51 onsets in just 2 s in the lowest plot of Fig. 3, with $\sim 45,000$ over the four instruments.

4.2 Microtiming Estimation

Our microtiming modelling is based on the approach in [12], which models microtiming profiles per beat as a multi-dimensional variable **m**, where each component m_i with $i = 1, ..., N$, with N the total number of onsets per beat, describes the evolution over time of the relative position of onset i with respect to the beat. The model in [12] estimates the beat and onset likelihoods, and infers the beat positions and associated microtiming profiles jointly using conditional random fields (CRFs). In this work we follow these ideas, but instead of inferring beat and microtiming profiles jointly with a CRF as in [12], we perform the beat and onset inference offline, and then group the onsets and beats using Algorithm 1. The reason for this is two-fold: our proposed approach is simpler and computationally cheaper than the CRF approach, with the limitation that it would not be robust in presence of noisy signals or mixtures, which is not the case here. Also, since we are in-

Algorithm 1: Microtiming modelling

> **Input:** b, o, τ, r
> **Output:** **m**, t
> **for** $i \leftarrow 1$ *to len(b)-1* **do**
> $\quad \Delta b \leftarrow b^{(i+1)} - b^{(i)}$;
> $\quad t_{ini} \leftarrow b^{(i)} - \tau \times \Delta b$;
> $\quad t_{end} \leftarrow b^{(i+1)} - \tau \times \Delta b$;
> $\quad o_{beat} \leftarrow o[t_{ini} < o < t_{end}]$;
> $\quad$ **if** *len(o_{beat}) $< r$ **and** o_{beat} is not empty* **then**
> $\quad\quad o_{temp} \leftarrow range(0, 1, 1/r) + t_{ini}$;
> $\quad\quad$ **for** $j \leftarrow 1$ *to len(o_{beat})* **do**
> $\quad\quad\quad k_{min} \leftarrow \arg\min_k(|o_{beat}^{(j)} - o_{temp}^{(k)}|)$;
> $\quad\quad\quad o_{fix}[k_{min}] \leftarrow o_{beat}^{(j)}$;
> $\quad\quad$ **end**
> $\quad\quad o_{beat} \leftarrow \text{interp}(o_{fix}[nan], o_{fix}[\sim nan])$
> $\quad$ **else**
> $\quad\quad$ **continue;**
> $\quad$ **end**
> $\quad$ **for** $j \leftarrow 2$ *to len(o_{beat})* **do**
> $\quad\quad v_{IOI}^{(j-1)} \leftarrow o_{beat}^{(j)} - o_{beat}^{(j-1)}$
> $\quad$ **end**
> $\quad$ **m**$^{(i)} \leftarrow v_{IOI} / \Delta b$;
> $\quad t^{(i)} \leftarrow b^{(i)}$
> **end**
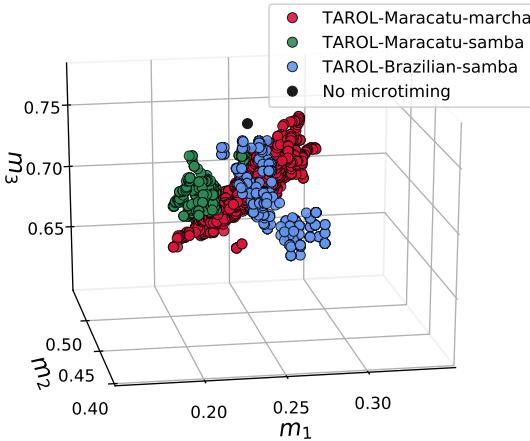
terested in both within-instrument and between-instrument microtiming, we need a flexible model that allows changing the beat reference, which we can do since we compute beats and onsets offline and integrate them afterwards.

Algorithm 1 is structured as follows: for each beat b we obtain the onsets o_{beat} that fall within the beat interval $[t_{ini}, t_{end}]$ by a tolerance given by τ, and check if we have the expected number of onsets (denoted by r in Algorithm 1). If not, we deduce which onsets are missing by comparing the given onsets with a template containing evenly-distributed positions (0.25, 0.5 and 0.75 in the case of three expected onsets). Next, we interpolate the missing onsets for better visualisation of the microtiming pattern. Finally, we obtain the microtiming profile **m** by dividing all inter-onset-intervals by the beat interval length. We exclude beats with more than the expected number of onsets.

Microtiming deviations and their variation across time have been studied in the context of time-keeper instruments (e.g., in Brazilian samba [12], Uruguayan candombe [18], and jazz [21]). While previous approaches focus on within-instrument rhythmic patterns, we also explore the microtiming generated between time-keepers and non-time-keeper instruments. We apply a similar strategy to that in [12] to analyse the profiles visually.

Both existing work and Algorithm 1 assume that there is *one* main rhythmic pattern played during most of the recording. This hypothesis holds in most of the recordings we obtained, however, unlike other examples such as the BRID dataset [22] where it holds to a great extent, in Maracatu, it is not true for samba.

For the analysis of between-instrument microtiming de-

Figure 4: Comparison of Maracatu sub-genres and Tarol (Caixa) in Brazilian samba.

viations, we follow the existing literature about Maracatu [17] and use the Porca and Gonguê Low onsets as beat references for the other instruments since they play the role of time-keepers and are meant to play the beat. For the within-instrument analysis for Tarol and Bombo, we estimated the beats using the *madmom* library [23] [1], which performed well in the solo tracks, and we adjusted the final beat positions to the closest onset.

5. MICROTIMING ANALYSIS

Marcha vs. samba: Both by listening to the recordings and inspection of the microtiming profiles, we discovered that samba has far greater variation in rhythmic patterns than marcha. In addition, the beat estimation revealed a noticeable difference in tempo, with marcha approximately $165\,\mathrm{bpm}$ where as samba was faster at around $180\,\mathrm{bpm}$. Transcriptions from [17] also state that there is greater variation in the notated rhythmic patterns for samba compared to marcha. From the perspective of drawing robust conclusions about microtiming profiles, which, using our approach, rely quite strongly on a consistent rhythmic pattern, we were only able to conduct reliable analysis for Tarol within-instrument microtiming. To enable a high-level comparison, which also includes the use of Tarol (referred to as Caixa in [12]) in Brazilian samba we present a scatter plot of m_1 vs. m_2 vs. m_3 in Fig. 4. Perhaps the most striking observation is that for marcha, the Tarol shows a much wider variation in the m_1 and m_3 dimensions, where as the limited data we obtain for Maracatu samba occupies a tighter cluster, and Brazilian samba varies more prominently over m_2. This indicates a different use of microtiming for Tarol in our recordings both within Maracatu sub-genres and compared to Brazilian samba.

Microtiming profiles in marcha: In Fig. 5 we observe both the within- and between-instrument microtiming analysis for Tarol (left column) and Bombo High (right column) for recording #28. For the between-instrument

analysis we use the Gonguê Low and Porca as time-keepers (middle and bottom rows respectively). When comparing between-instrument and within-instrument profiles, we observe one additional trace for both instruments: the deviation at the beat level (which is normalised out for within-instrument analysis). Referring back to Fig. 3 we can see microtiming profiles consistent with a sub-division of the beat into four $1/16$th notes for Tarol, with the second $1/16$th note (m_1) not played for Bombo High.

Looking at the microtiming profiles through time, we see more fluctuation in Tarol compared to Bombo High. In particular for Tarol, the smoothed microtiming profiles move above and below the quantised positions indicating a dynamic use of microtiming. Furthermore, a direct comparison of Tarol with Bombo High illustrates different profiles. Across the entire set of recordings, we found the following median profiles: Tarol [0.25, 0.47, 0.715], and Bombo High [0.46, 0.69].

When contrasting the time-keepers, we observe much greater variation when the Porca beats are used as reference compared to Gonguê Low, including an unexpected downward trend for both Tarol and Bombo High. While not present in all marcha, we observed several similar instances, which can be attributed to the beats of the Porca being played *ahead* of the beat and slowly aligning in phase by the end. Looking again at Fig 3, we see the Porca annotations are earlier than the other instruments consistently by upto $20\,\mathrm{ms}$. A possible explanation may be that the Gonguê is louder and thus takes a more prominent time-keeping role, allowing the Porca to take a more expressive role. Regardless, we assert the importance of analysing the behaviour of the time-keeper instruments.
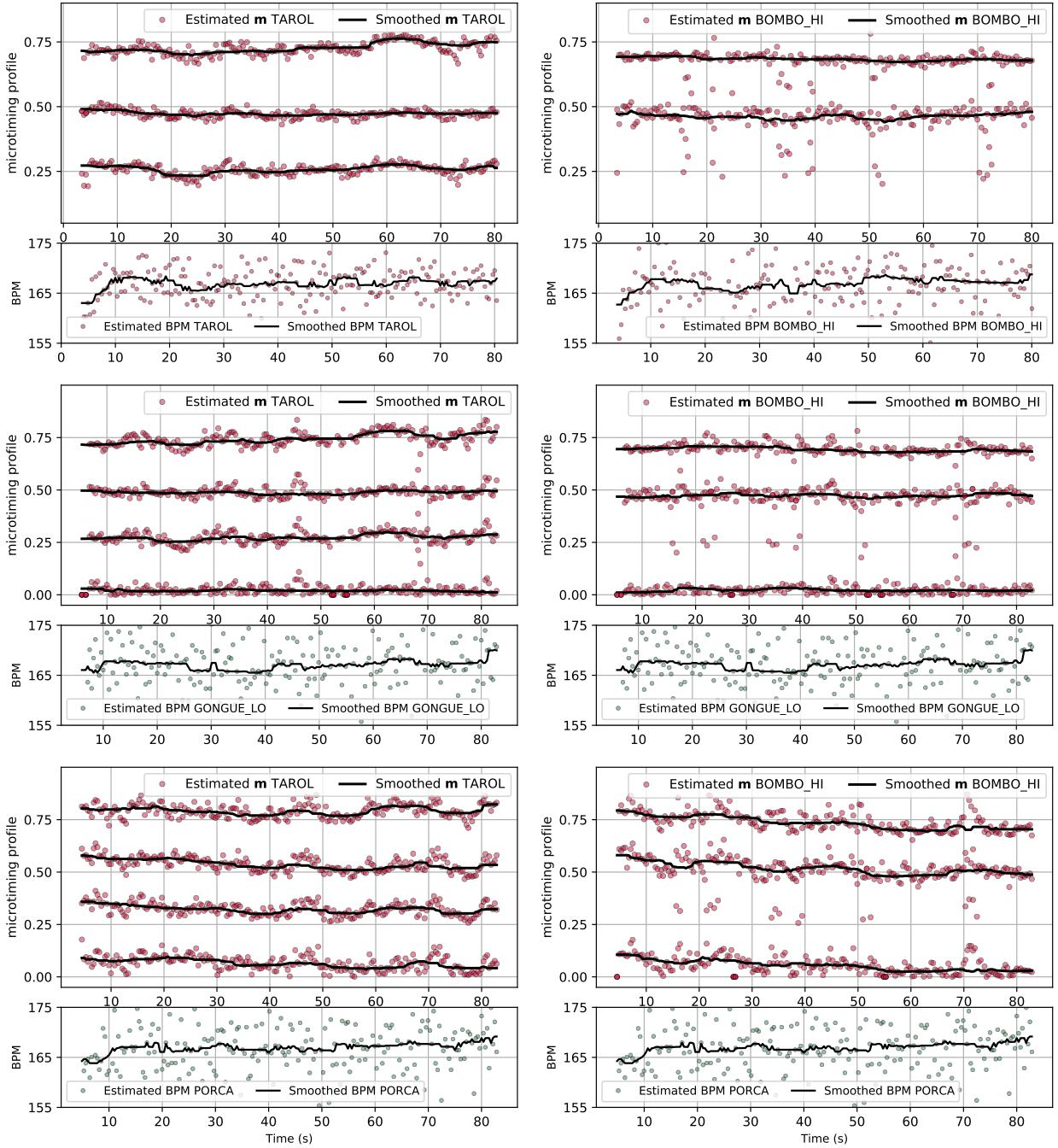
6. REFLECTIONS

One objective of current ethnomusicological research is to reveal how musicians of different cultures develop strategies for playing together that differ in subtle ways to those in Western culture. These strategies are often implicit, associated to verbal categories that express local views on how music "should sound" in a particular cultural context [25]. These verbal categories often rely on non-musical concepts and metaphors, often related to other sensorial domains (vision, taste, etc.) [26].

Ethnographic field research is a first, necessary step for unveiling subtle strategies of playing together that are at stake in a particular musical culture. Formal analysis of live performances is then needed to understand to what acoustic reality these local concepts refer. To this end, MIR techniques, such as microtiming analysis, can provide innovative solutions for exploring qualities of the music that would otherwise be hard to describe.

In the Zona da Mata region, Carnival is not a simple distraction but rather a ritual involving a complex set of mystical beliefs and social concerns. At this time of the year invisible negative "entidades" (entities) are believed to be more active and even dangerous, and interpersonal relations are marked by feelings of envy and jealousy. Carnival is therefore considered as a threat both for individuals and

---

[1] We used the model that implements Böck et al. [24] in version 0.16.1.

Figure 5: Microtiming profiles in recording #28 for Tarol and Bombo High, left and right respectively. The microtiming estimations use within-instrument, Gonguê Low and Porca beats as reference in top, middle and bottom plots respectively.

the community. Maracatu de baque solto is a performance-ritual that allows people to overcome these risks.

It is important to stress that the Maracatu musicians had never travelled outside of Brazil, and bringing them to Europe was logistically complicated. In addition, the signal acquisition and onset annotation were also non-trivial, meaning several challenges needed to be overcome before even beginning to analyse the microtiming in Maracatu in a computational way. Nevertheless, within the confines of a small dataset, limited to one set of musicians, our preliminary analysis revealed new findings on the use of microtiming, in particular its dynamic nature in Maracatu.

In the long term, our aim is to understand what it means to play in "consonance" and to "close the Maracatu." Both concepts point to subtle manners of producing sounds collectively, that differ form the ones observed in other musical contexts. Since no formal analyses of Maracatu music have been previously realised, this paper is a first attempt to understand how musicians rhythmically interact during a live performance. In future research, we intend to play back the recordings to the musicians to understand if "consonancia" and "closure" can be associated to the specific microtiming profiles highlighted in this paper.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] L. Garrabé, "Les rythmes d'une culture populaire: les politiques du sensible dans le maracatu-de-baque-solto, Pernambuco, Brésil," Ph.D. dissertation, Université Paris 8, 2010, (in French).

[2] M. Acselrad, *Viva Pareia! Corpo, dança e brincadeira no Cavalo-Marinho de Pernambuco.* Recife: Editora Editora Universitária UFPE, 2013, (in Portuguese).

[3] F. Bonini Baraldi, "Inveja e corpo fechado no Maracatu de baque solto pernambucano," *Submitted.* (in Portuguese).

[4] E. Gómez, P. Herrera, and F. Gómez-Martin, "Computational ethnomusicology: perspectives and challenges," *Journal of New Music Research*, vol. 42, no. 2, pp. 111–112, 2013.

[5] G. Tzanetakis, "Computational ethnomusicology: a music information retrieval perspective," in *Proc. of Joint ICMC\SMC Conference*, 2014, pp. 112–117.

[6] M. E. P. Davies, G. Madison, P. Silva, and F. Gouyon, "The effect of microtiming deviations on the perception of groove in short rhythms," *Music Perception: An Interdisciplinary Journal*, vol. 30, no. 5, pp. 497–510, 2013.

[7] A. Hofmann, B. C. Wesolowski, and W. Goebl, "The tight-interlocked rhythm section: Production and perception of synchronisation in jazz trio performance," *Journal of New Music Research*, vol. 46, no. 4, pp. 329–341, 2017.

[8] L. Kilchenmann and O. Senn, "Microtiming in swing and funk affects the body movement behavior of music expert listeners," *Frontiers in psychology*, vol. 6, p. 1232, 2015.

[9] B. Merker, "Groove or swing as distributed rhythmic consonance: introducing the groove matrix," *Frontiers in Human Neuroscience*, vol. 8, 2014.

[10] M. A. Witek, E. F. Clarke, M. Wallentin, M. L. Kringelbach, and P. Vuust, "Syncopation, body-movement and pleasure in groove music," *PloS one*, vol. 9, no. 4, p. e94446, 2014.

[11] T. M. Esparza, J. P. Bello, and E. J. Humphrey, "From genre classification to rhythm similarity: Computational and musicological insights," *Journal of New Music Research*, vol. 44, no. 1, pp. 39–57, 2015.

[12] M. Fuentes, L. S. Maia, M. Rocamora, L. W. P. Biscainho, H. C. Crayencour, S. Essid, and J. P. Bello, "Tracking beats and microtiming in afro-latin american music using conditional random fields and deep learning," in *Proc. of the 20th Intl. Society for Music Information Retrieval Conf.*, 2019, pp. 251–258.

[13] L. S. Maia, M. Fuentes, L. W. P. Biscainho, M. Rocamora, and S. Essid, "SAMBASET: A Dataset of Historical Samba de Enredo Recordings for Computational Music Analysis," in *Proc. of the 20th Intl. Society for Music Information Retrieval Conf.*, 2019, pp. 628–635.

[14] L. Naveda, F. Gouyon, C. Guedes, and M. Leman, "Microtiming patterns and interactions with musical properties in samba music," *Journal of New Music Research*, vol. 40, no. 3, pp. 225–238, 2011.

[15] L. Nunes, M. Rocamora, and L. W. P. Jure, L.and Biscainho, "Beat and downbeat tracking based on rhythmic patterns applied to the Uruguayan Candombe drumming," in *Proc. of the 16th Intl. Society for Music Information Retrieval Conf.*, 2015, pp. 264–270.

[16] M. Rocamora, L. Jure, M. Fuentes, L. S. Maia, and L. W. P. Biscainho, "CARAT: Computer-aided Rhythmic Analysis Toolbox," in *Late-Breaking Demo Session of the 20th Intl. Society for Music Information Retrieval Conf.*, 2019.

[17] C. de Oliveira Santos, T. S. Resende, and P. M. Keays, *Batuque Book: Maracatu Baque Virado e Baque Solto.* Recife: Edição do autor, 2009.

[18] M. Rocamora, "Computational methods for percussion music analysis: The Afro-Uruguayan Candombe drumming as a case study," Ph.D. dissertation, Universidad de la República (Uruguay). Facultad de Ingeniería, 2018.

[19] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long-short term memory neural networks," in *Proc. of the 11th Intl. Society for Music Information Retrieval Conf.*, 2010, pp. 589–594.

[20] M. E. P. Davies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," in *Proc. of the 27th European Signal Processing Conf.*, 2019.

[21] C. Dittmar, M. Pfleiderer, S. Balke, and M. Müller, "A swingogram representation for tracking micro-rhythmic variation in jazz performances," *Journal of New Music Research*, vol. 47, no. 2, pp. 97–113, 2018.

[22] L. S. Maia *et al.*, "A novel dataset of Brazilian rhythmic instruments and some experiments in computational rhythm analysis," in *AES Latin American Congress of Audio Engineering (AES LAC)*, 2018, pp. 53–60.

[23] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new python audio and music signal processing library," in *24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, Oct. 2016, pp. 1174–1178.

[24] S. Böck and M. Schedl, "Enhanced beat tracking with context-aware neural networks," in *Proc. Int. Conf. Digital Audio Effects*, 2011, pp. 135–139.

[25] N. Fernando and D. Rappoport, Eds., *Cahiers d'ethnomusicologie: Le Goût Musical*, 28, 2015.

[26] F. Bonini Baraldi, E. Bigand, and T. Pozzo, "Measuring Aksak Rhythm and Synchronization in Transylvanian Village Music by Using Motion Capture," *Empirical Musicology Review*, vol. 10, no. 4, pp. 265–291, 2015.

MULTILINGUAL MUSIC GENRE EMBEDDINGS FOR EFFECTIVE CROSS-LINGUAL MUSIC ITEM ANNOTATION

Elena V. Epure
Deezer Research

Guillaume Salha
Deezer Research

Romain Hennequin
Deezer Research

`research@deezer.com`

ABSTRACT

Annotating music items with music genres is crucial for music recommendation and information retrieval, yet challenging given that music genres are subjective concepts. Recently, in order to explicitly consider this subjectivity, the annotation of music items was modeled as a translation task: predict for a music item its music genres within a target vocabulary or taxonomy (tag system) from a set of music genre tags originating from other tag systems. However, without a parallel corpus, previous solutions could not handle tag systems in other languages, being limited to the English-language only. Here, by learning multilingual music genre embeddings, we enable cross-lingual music genre translation without relying on a parallel corpus. First, we apply compositionality functions on pre-trained word embeddings to represent multi-word tags. Second, we adapt the tag representations to the music domain by leveraging multilingual music genres graphs with a modified retrofitting algorithm. Experiments show that our method: 1) is effective in translating music genres across tag systems in multiple languages (English, French and Spanish); 2) outperforms the previous baseline in an English-language multi-source translation task.

1. INTRODUCTION

Music genres are a key characteristic of music items [1, 2]. In music streaming services, user profiles and interests can be expressed through music genres, tracks and artists can be grouped in genre-specific collections, and content-based recommender systems frequently exploit music genres as item tags. However, music genres are difficult to infer due to their subjective nature. Based on their music preferences, musicological knowledge and culture, people inconsistently associate genres to music items [3–5]. Thus, annotating music items with genres for providing personalized recommendation and retrieval is challenging.

Acknowledging this subjectivity and the absence of a unique genre definition, recent works [6,7] framed the music genre annotation as a *translation*. More precisely, given

music items annotated with music genres originating from multiple source tag systems such as folksonomies, editorial vocabularies or taxonomies, the goal was to predict the equivalent music genres within a target tag system. In a supervised setup, the translation relied on a parallel corpus of music items jointly annotated with music genres from the source and target tag systems [6,7]. In an unsupervised setup, when the parallel corpus was unavailable, a solution centered on taxonomy alignment was proposed [6].

However, the translation of music genres between *multilingual* sources remains unaddressed when a parallel corpus is unavailable. The only past unsupervised solution [6] relied on heuristics specific to the English language, making its adaptation to multilingual tags a challenge. Here, we propose to perform the unsupervised cross-lingual translation by leveraging multilingual music genre embeddings. Also, our method to learn these embeddings could be straightforwardly applied to new languages.

The proposed method is further summarised. First, by acknowledging the compositional nature of music genres (i.e. the meaning of multi-word music genres can be often derived from the meaning of each word), we learn music genre embeddings by applying compositionality functions to pre-trained word vectors [8–10]. Moreover, as these pre-trained vectors are often trained on language-specific text, we need to align them across languages [11, 12].

Second, we fit the obtained music genre embeddings into the music domain. The embeddings learnt on general-language corpora could sometimes be semantically ambiguous. For instance, *house* is closer to *building* than to *music* and *jazz* is more similar to *folk* than to *bebop* in fast-Text [8]. To tackle this problem, we create a music genre knowledge graph from multilingual DBpedia [13] that contains multilingual genres as nodes and exhibits different types of music genre relations through its edges. Then, we use *retrofitting* [14] to encode the relational knowledge from the semantic graph in the embeddings. In this work, we modify the original retrofitting algorithm [14] to distinguish between two types of relations: equivalence (e.g. *dnb* and *drum'n'bass*) and other types of relatedness such as sub-genres, derivative genres, fusion genres, stylistic origins. Besides, we use retrofitting to learn embeddings for music genres that do not exist in the pretrained embedding vocabulary by exploiting their graph relations (e.g. *ethnotronica* and *chillstep* are not in the pretrained fastText vocabulary).

We evaluate the proposed method in two experiments.

First, we collect a new parallel corpus of music items annotated with music genres in three languages (English, French and Spanish) and demonstrate the effectiveness of our method for unsupervised cross-lingual music genre translation. Second, we show that using the embeddings learnt with our method outperforms the previous baseline [6] in a music genre translation task between multiple English-language tag systems.

2. PROBLEM FORMULATION AND RELATED WORK

Annotating music items from song lyrics or audio content has concentrated significant research efforts in the music information retrieval community [7, 15, 16]. Most existing works fix a tag system and focus on general music genres like *jazz* or *pop* [3–5]. Nonetheless, the dissimilarity of music genre tag systems and their use in annotations has been recently put forward for consideration, together with the need to take into account tags with increased granularity [15, 17]. In this direction, two previous works [6, 7] have framed the music genre annotation as a tag translation task between various music genre tag systems.

Specifically, given a set $\mathcal{S}$ of *source tag systems*, $S = \cup_{E \in \mathcal{S}} E$ the union of all tags across all source tag systems, $\mathcal{P}$ the partitions of S and T a *target tag system*, the goal is to define a translation scoring $f : \mathcal{P}(S) \to \mathbb{R}^{|T|}$ which estimates a score for each target tag from a set of source tags drawn from S. While in this notation a tag system refers to a set of tags, more general representations such as music genre graphs or taxonomies can also include relations between tags [18–20].

Hennequin et al. [7] proposed two translation strategies, both relying on the existence of a parallel corpus of music items annotated with music genres. Epure et al. [6] addressed also the unsupervised case, when such a parallel corpus was absent, and designed a knowledge-based method to learn tag embeddings. This method relied on multiple building blocks corresponding to tag normalization, the construction of an integrated music genre graph bringing together all source and target tag systems, and a taxonomy alignment algorithm mapping each music genre on DBpedia [13] tags. The DBpedia-related building block yielded music genre vectors quantifying the relatedness of the tag under consideration to each DBpedia music genre. For translation, considering $\{s_1, \ldots, s_K\}$ source tags and any target tag t, f was computed using cosine similarity:

$$f_t(\{s_1, s_2, \ldots, s_K\}) = \sum_{k=1}^{K} \frac{\mathbf{s}_k^T \mathbf{t}}{||\mathbf{s}_k||_2 ||\mathbf{t}||_2}, \qquad (1)$$

where $\mathbf{s}_k$ and $\mathbf{t}$ are the vectors corresponding to each s_k, respectively t and $|| \cdot ||_2$ is the Euclidian L2-norm.

In the previous unsupervised work, Epure et al. [6] focused on English-language music genres, claiming that the extension of the knowledge-based method to include multilingual tag systems was feasible since it relied on multilingual DBpedia. While we agree that it is feasible, the extent to which the introduced method could be easily

changed to support other languages is questionable. Both normalizing tags and mapping music genres into the DBpedia space rely on language-specific heuristics. For instance, in normalization, heuristics referring to the length of tokens were used. However, the average word length, hence what is considered as a short or medium-length token depends on the language [21]. Then, mapping music genres on English DBpedia genres is limiting because some tags may exist in two languages but not in the English DBpedia. Computing directly the degree of relatedness of a source tag to a target tag could be a better alternative.

3. A MULTI-STEP METHOD FOR LEARNING MUSIC GENRE EMBEDDINGS

In this work, we propose a method to learn multilingual music genre embeddings that can be easily extended to new languages and support cross-lingual translation. The first step is to deduce initial embeddings for multi-word music genres by leveraging pre-trained multilingual word embeddings (Section 3.1). However, directly using these music genre embeddings in cross-lingual translation is prone to under-perform because:

- the embeddings often correspond to the most common word senses (e.g. *country* can refer to *nations* or *rock* could be closer in meaning to *stone*) and they are not disambiguated against the music domain.

- some music genres could contain rare words which are absent from the pre-trained model vocabulary, resulting in potentially unknown tag embeddings.

To address these issues, we complement distributional concept representations with semantics from knowledge bases that expose concept relations. Thus, in a second step, we assemble a multilingual music genre graph (Section 3.2). Then, we adjust the initial tag embeddings to encode the tag relations from the collected graph, ensuring the domain adaptation. For this, but also to learn embeddings for concepts with unknown vocabulary words, we use *retrofitting* [14], which we modify to reflect the different types of music genre relations (Section 3.3).

3.1 Initializing Music Genre Embeddings

Under the music genre translation framework introduced in Section 2, the main goal boils down to quantifying the degree of relatedness of two textual tags. This task is widely popular in the natural language processing (NLP) community and contemporary approaches resort to expressing the relatedness as distance between corresponding word embeddings [8–10]. The mapping of words on embeddings is guided by the distributional hypothesis [22], which states that words in similar contexts are likely to have similar meanings. Word embeddings have been proven effective in capturing word syntactic and semantic similarities and in improving downstream NLP tasks such as natural language understanding [23] and information retrieval [24].

In order to measure the relatedness of multilingual words using embeddings learnt from monolingual corpora,

an alignment between the language-specific embedding spaces is required. Through the alignment [25], we ensure that multilingual word embeddings are projected into a common space where they are comparable. Practically, a mapping function between two monolingual word embedding spaces is learnt, for instance by using a bilingual lexicon [12]. Effective alignments have been also found using orthogonal Procrustes [11, 25].

Starting from multilingual word vectors, we discuss strategies to initialize the music genre embeddings. Music genres can contain multiple words. We claim that the compositionality principle, stating that the meaning of a multi-word expression is dictated by the meaning of each word, often holds for our case. For instance, *Dance pop* is related to *dance* and *pop* or *Balada romántica* is a type of *ballad* which is *romantic* [1]. The contemporary approach for compositional embeddings is to learn a function which derives the embeddings of a multi-word expression from the embeddings of its words [26]. The function is learnt by minimizing the distance for each multi-word expression between its distributional embedding and its embedding computed from its word embeddings. Obtaining the distributional embedding for multi-word music genres would be however challenging because sufficiently large corpora with all tags in multiple languages are required.

For this reason, the first music genre initialization strategy we propose consists of a simple compositionality function such as averaging word embeddings (*avg*). Let $V = \{c_1, c_2, ..., c_n\}$ be the multilingual vocabulary, c_i being a concept composed of at least one word. We aim to compute $\hat{\mathbf{Q}} \in \mathbb{R}^{n \times d}$, the embedding matrix for the vocabulary V, where $\hat{\mathbf{q}}_i \in \mathbb{R}^d$ denotes the embedding of concept c_i. If c_i is composed of the following words, $\{w_1, w_2, \ldots, w_M\}$, $\hat{\mathbf{q}}_i$ can be computed as $\frac{1}{M} \sum_{m=1}^{M} \mathbf{w}_m$, where $\mathbf{w}_m$ is the embedding of the word w_m. Of note is that if c_i contains words absent from the pretrained word embedding vocabulary, the d-dimensional null vector, $\mathbf{0}_d$, is used as a default.

The second music genre initialization strategy we propose exploits the fact that some words in a compounded expression may be more illustrative than others. The more frequently a word is observed in a corpus, the more likely it is that the word is common for a language and semantically less informative (e.g. *music* in *post industrial music*). Thus, the compositional embedding computation of a multi-word expression can be modified such that the contribution of each word embedding is inversely proportional to its frequency. Pre-trained word embeddings are generally released sorted by decreasing word corpus frequency. Let z_{w_m} be the rank of w_m in this vocabulary. Then, based on the Mandelbrot's generalization [27] of the Zipf's law [28], its frequency f_{w_m} can be estimated to $f_{w_m} = 1/(z_{w_m} + 2.7)$.

Further, we rely on the smooth inverse frequency (*sif*) based averaging proposed by Arora et al. [29] to compute the multi-word expression embeddings. This method is aligned with our previous observations and proven highly effective compared to more complex neural network-based

models on a large diversity of NLP tasks [29]. Given f_{w_m} the estimated frequency of the word w_m and a a fixed hyper-parameter [2], $\hat{\mathbf{q}}_i$ is computed as:

$$\overline{\mathbf{q}}_i = \frac{1}{M} \sum_{m=1}^{M} \frac{a}{a + f_{w_m}} \mathbf{w}_m \qquad (2)$$

$$\hat{\mathbf{q}}_i = \overline{\mathbf{q}}_i - \mathbf{u}\mathbf{u}^T \overline{\mathbf{q}}_i \qquad (3)$$

where $\mathbf{u}$ is the first singular vector from the singular value decomposition of $\overline{\mathbf{Q}}$ obtained with the Equation 2 [30].

3.2 Assembling a Multilingual Music Genre Graph

Previous related work [6] created a music genre graph by integrating multiple English-language music genre tag systems and a crawled sub-graph of DBpedia through a node English-language based normalization step. The other music genre tag systems were Lastfm, Tagtraum and Discogs, used in the 2018 MediaEval AcousticBrainz Genre Task [17]. Here, we bypass the language-specific heuristics normalization and propose a more robust alternative. We crawl a multilingual DBpedia music genre sub-graph and use its words as basis for normalizing new tag systems.

We further detail how we assemble the DBpedia-based music genres graph. We set the seeds for crawling from: 1) DBpedia entities of type *MusicGenre*, 2) the music genres of the multilingual DBpedia-based music item corpus (described in Section 4.1), 3) synonyms of the music genres of the previous two sources, linked through the *wikiPageRedirects* relation. We discover new potential music genres by crawling DBpedia entities linked to the seeds through one of the relations: *wikiPageRedirects*, *stylisticOrigin*, *musicSubgenre*, *derivative* and *musicFusionGenre* [3]. During crawling, seeds are updated with discovered entities that were not visited before, and the crawling goes on until no seeds are left. This is applied for each language. Finally, all music genres discovered as yet are connected to their equivalent tags in other languages, when possible (the DBpedia relation *sameAs*). In a post-processing step, we remove music genre nodes written as free-style text, which do not have DBpedia pages, and the connected components which do not contain at least one high-confidence music genre (empirically, we noticed that the highest-confidence tags were those from the music item corpus).

To ensure that tag systems with different music genre spellings benefit from the multilingual graph, we define a normalization which we apply to both the graph nodes and new tags. First, we tokenize each tag by non-alphanumeric characters. Further, as in [6], we create prefix trees to split multi-word tags such as *sludgemetal* or *indierock*. Nevertheless, we do not necessarily aim at a grammatically correct split, but at one based on lemmatized DBpedia music genre words [4]. Namely, if *sludgemetal* is already among

---

[1] Exceptions from the principle also exist (e.g. *hard rock*).

[2] Experimentally, it has been shown that $a = 10^{-3}$ is a suitable choice when using different types of pre-trained word embeddings [29].

[3] These relation names correspond to the English-language DBpedia. They have often translated versions in DBpedia in other languages.

[4] Through lemmatization, we retrieve the base form of inflected words using spacy (https://spacy.io). Most genre words are generally in their base form (e.g. *jazz*). However, some other words benefit from this (e.g. *Northern / North* or *children / child*)

the DBpedia music genre words, then there is no further split and we expect its initial embedding to be corrected though the embeddings of its graph neighbors as explained in the next section (Section 3.3).

3.3 Retrofitting Music Genre Embeddings

Retrofitting [14] has been proposed as a post-processing step to improve concept embeddings by leveraging existing semantic lexicons or knowledge graphs (e.g. WordNet [31]). More precisely, concept embeddings are modified to also encode concept relations [14, 32–34].

Let $G = (V, E)$ be the graph capturing the semantic relations between the nodes in V through a set of edges $E \subseteq V \times V$. The objective of retrofitting is to learn $\mathbf{Q} \in \mathbb{R}^{n \times d}$, the new concept embeddings, such that each new embedding $\mathbf{q}_i \in \mathbb{R}^d$ does not stray too far from the initial distributional embedding $\hat{\mathbf{q}}_i$, but also becomes closer to the new embeddings of the neighbour vertices $\mathbf{q}_j \in \mathbb{R}^d, j : (i, j) \in E$. The objective function to minimize is then [14]:

$$\Phi(\mathbf{Q}) = \sum_{i \in V} \left(\alpha_i ||\mathbf{q}_i - \hat{\mathbf{q}}_i||_2^2 + \sum_{j:(i,j) \in E} \beta_{ij} ||\mathbf{q}_i - \mathbf{q}_j||_2^2 \right) \tag{4}$$

where α_i and β_{ij} are positive scalars specifying the importance given to each component, the initial embedding and each graph neighbor. As Φ is convex with respect to $\mathbf{Q}$, a solution minimizing the objective function Φ is found in [14] via an iterative strategy derived from Jacobi iteration algorithm [35] that converges for such graph-based propagation problem [35, 36]. More precisely, until convergence, $\mathbf{q}_i$ is iteratively updated as follows:

$$\mathbf{q}_i \leftarrow \frac{\sum_{j:(i,j) \in E} (\beta_{ij} + \beta_{ji})\mathbf{q}_j + \alpha_i \hat{\mathbf{q}}_i}{\sum_{j:(i,j) \in E} (\beta_{ij} + \beta_{ji}) + \alpha_i} \tag{5}$$

where $\mathbf{Q}$ is initialized to $\hat{\mathbf{Q}}$. Equation (5) is not the same as the original one [14]. We observed that, when applying the Jacobi method to optimize equation (4), the contributing terms in the partial derivative with respect to the node i are those where i appears as source as well as target node in the inner sum, leading to a different update. In [36, 37], the same conclusion referring to a corrected update rule, different from the initial proposal, is reached.

Faruqui et al. [14] set $\alpha_i = 1$ and $\beta_{ij} = \frac{1}{\text{degree}(i)}$ for $(i, j) \in E$, where degree(i) is the number of neighbors i has in the graph G, or 0 for $(i, j) \notin E$. This choice was largely adopted in other related works [32, 38]. Speer and Chin [39] proposed to use a modified version of retrofitting to learn embeddings for unknown vocabulary concepts which are present in the knowledge graph. For this case, α_i is set to 0 for all unknown vocabulary concepts. This results in $\mathbf{q}_i$ being updated by averaging the embeddings of its neighbours at each iteration. Despite the change in the update rule we made, compared to the original work, we retain this choice of hyper-parameters as being a reasonable default, and defer the investigation of a more principled way to pick α_i and β_{ij} to future work.

We further modify retrofitting to take advantage of the different types of music genre relations. On one hand, mu-

sic genres can be semantically equivalent to other music genres (the relation types *wikiPageRedirects* and *sameAs*). On the other hand, music genres can be related to other music genres, but not semantically equivalent (e.g. *stylisticOrigin*). The change we propose for computing these new embeddings ($\mathbf{Q}_{\overline{\beta}}$) is through the coefficients β_{ij}, making them dependent on music genre relation types:

$$\overline{\beta}_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E_\epsilon \subset E \\ \beta_{ij} & \text{if } (i, j) \in E - E_\epsilon \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where E_ϵ contains edges which represent equivalence relations (*wikiPageRedirects*, *sameAs*); $E - E_\epsilon$ contains edges with the remaining relation types (*stylisticOrigin*, *musicSubgenre*, *derivative*, *musicFusionGenre*).

4. EXPERIMENTS

We evaluate the effectiveness of the learnt music genre embeddings, first, in a new cross-lingual music genre translation scenario (Section 4.3) and, second, in an existing English-language multi-source music genre translation task [6, 17] (Section 4.4). The languages we focus on for the cross-lingual annotation are English (**En**), French (**Fr**) and Spanish (**Es**). We start by presenting the parallel corpora used in the experiments (Section 4.1). Then, we discuss the detailed evaluation setup (Section 4.2). The results show that our music genre vectors are highly effective for cross-lingual translation and lead to improved results on the past unsupervised Enligh-language translation task [6].

4.1 Datasets

For the cross-lingual translation experiment, we relied on DBpedia [13] to collect a parallel corpus. During an initial manual analysis, we noticed that DBpedia music artists or works could have associated quite different music genres across languages. We present a few examples in Table 1. Also, when the tags used in annotations were equivalent, they were sometimes partially translated (e.g. *Rock_alternatif* in **Fr**), while other times they maintained the same form as in English (e.g. *Soft_rock* in **Es**). We collected DBpedia entities of type *MusicalArtist*, *Band*, or *MusicalWork* with music genres associated in at least two languages. Then, in a post-processing step, we filtered out the music items with tags that appeared less than 16 times.

For the English-language multi-source translation, we use an existing dataset [6, 17], which contains tracks annotated with English-language music genres from three sources. Discogs (**Dc**) tags are provided by editors per album, and automatically propagated to each track [17]. Lastfm (**Lf**) and Tagtraum (**Tt**) tags are created by Internet users per track. We show in Table 2 the number of music items and unique music genres in the new multilingual and the past English-language multi-source parallel corpora.

4.2 Evaluation Setup

We evaluated our models by translating music genre tags associated with tracks from multiple source tag systems to

Title	Type	En	Fr	Es
Morning View	Album	Alternative_metal, Funk_rock Alternative_rock, Post-grunge	Rock_alternatif	Metal_alternativo Rock_experimental
Jimi Hendrix	Artist	Hard_rock, Psychedelic_rock Blues, Rhythm_and_blues	Rock_psychédélique Blues_rock, Hard_rock	Blues_rock, Rock_psicodélico Hard_rock
Julio Iglesias	Artist	Dance-pop, Latin_music Adult_contemporary_music	Pop_française	Pop_latino, Balada_romántica Soft_rock, Adult_contemporary

Table 1. Examples of DBpedia music items annotated with music genres. The tag choices are inconsistent across sources. Tags may be adapted to a language (e.g. *Pop_latino* in **es**) or may keep the same form in all languages (e.g. *Hard_rock*).

	En	Fr	Es	Dc	Lf	Tt
Music items (tracks, albums, artists)	48 146	30 611	34 918	1 098 336	686 978	589 583
Unique music genres	489	338	491	315	327	296

Table 2. Number of music items and unique music genres in the multilingual and the English-language parallel corpora.

a target tag system. The translation scoring function computes a score for each tag of the target tag system as the degree of relatedness of the target tag to the input set of source tags. Compared to Equation 1, the translation scoring function we use here averages the cosine similarities between each source and target tag embeddings:

$$\hat{f}_t(\{s_1, s_2, \ldots, s_K\}) = \frac{1}{K} f_t(\{s_1, s_2, \ldots, s_K\}) \quad (7)$$

Like in other multi-label prediction tasks [15, 40], we use a ranking metric in evaluation, namely the area under the receiver operating characteristic curve (AUC). We macro-average the scores and report their mean and standard deviations computed over 4 folds. We split the multi-label data in a stratified way, balancing the overall number of music items and tag distribution across the folds [41].

For each experiment, English-language multi-source and cross-lingual, we have three input tag systems represented as partially aligned music genre graphs. For the multi-source translation, we assemble a graph from the English-language DBpedia music genre sub-graph and the input taxonomies, Discogs, Lastfm and Tagtraum. For the cross-lingual translation, the new music genre graph, which was assembled as described in Section 3.2, has 10748 tags in **En**, 2905 in **Fr** and 3988 in **Es**. The translation is performed using annotations from combinations of two out of three tag systems to the kept-out tag system. We also retain in evaluation annotations which are only from one of the two selected source tag systems.

In each experiment, we compare the *avg* and *sif* strategies to initialize the music genre vectors. We report results when using directly the initial embeddings ($\hat{\mathbf{Q}}$) in translation or retrofitted with the original method ($\mathbf{Q}$) or with our modified version ($\mathbf{Q}_{\bar{\beta}}$). As for the choice of pre-trained word embeddings, we use multilingual fastText [10] which we align with the method proposed by Joulin et al. [42].

4.3 Results on Cross-Lingual Genre Translation

In Table 3, we present the results of the cross-lingual music genre translation. The baseline we propose estimates the relatedness of two tags to be the length of their shortest

path in the multilingual DBpedia-based music genre graph. As a reminder, this graph is partially aligned, meaning that some music genres have equivalent tags in other languages. As shown in Table 3 in parentheses, the baseline scores are quite high proving that the graph is fairly effective for cross-lingual translation on this dataset. Even so, we are able to exceed these scores by a large margin with our music genre embeddings, initialized with *sif* and retrofitted to take into account the music genre relations.

When comparing the initialization strategies, we can observe that directly using *sif* embeddings in translation outperforms the baseline, while *avg* yields lower AUC scores. For all languages as targets, the *sif* initialization is consistently more effective than the *avg* initialization. A significant difference between the two types of retrofitting applied to both initialization strategies exists, our version resulting in higher AUC scores. By differentiating between the two relation types, equivalence and other relatedness, the music genre embeddings appear to encode more accurately their relations within and across languages.

4.4 Results on Multi-Source Genre Translation

In Table 4, we present the results of the English-language multi-source music genre annotation. We re-compute the baseline [6] results using Equation 7. Compared to the previously reported AUC scores [6], the re-computed ones are higher showing that the modified translation scoring function does not disadvantage the knowledge-based music genre embeddings derived with the baseline. In contrast to the baseline, our most effective method, using *sif* initialization and our version of retrofitting, yields consistently higher AUC scores. The increase in performance is of 11.3 percentage points for **Dc** as target, 5.9 points for **Lf** as target and 9.3 points for **Tt** as target.

The *sif* initialization of tag embeddings results in higher AUC scores than *avg* both when the embeddings are used directly as they are or retrofitted, in particular, when **Dc** is target. Also, let us notice that directly using the embeddings initialized with *sif* leads to an increased performance compared to the baseline for **Dc** and **Tt**. Retrofitting the embeddings significantly increases the AUC scores for all

	Baseline	$\hat{\mathbf{Q}}$ (*avg*)	$\mathbf{Q}$ (*avg*)	$\mathbf{Q}_{\overline{\beta}}$ (*avg*)	$\hat{\mathbf{Q}}$ (*sif*)	$\mathbf{Q}$ (*sif*)	$\mathbf{Q}_{\overline{\beta}}$ (*sif*)
En + Es $\Longrightarrow$ Fr	85.4 ± 0.4	73.7 ± 0.2	77.5 ± 0.1	87.0 ± 0.2	85.9 ± 0.1	87.7 ± 0.1	$\mathbf{92.3 \pm 0.1}$
En + Fr $\Longrightarrow$ Es	84.3 ± 0.2	73.6 ± 0.3	76.1 ± 0.2	84.9 ± 0.2	85.6 ± 0.0	86.3 ± 0.1	$\mathbf{91.3 \pm 0.1}$
Fr + Es $\Longrightarrow$ En	80.4 ± 0.1	76.7 ± 0.4	84.2 ± 0.2	87.0 ± 0.3	84.5 ± 0.2	88.4 ± 0.3	$\mathbf{90.2 \pm 0.2}$

Table 3. Macro-AUC (%) in cross-lingual music genre translation with standard deviation computed over 4 folds. Results are shown for different embedding initialization (*avg* and *sif*), used directly ($\hat{\mathbf{Q}}$) or retrofitted with the original retrofitting ($\mathbf{Q}$) or with our version ($\mathbf{Q}_{\overline{\beta}}$). The baseline is built on the shortest paths in the DBpedia-based multilingual graph.

	Baseline	$\hat{\mathbf{Q}}$ (*avg*)	$\mathbf{Q}$ (*avg*)	$\mathbf{Q}_{\overline{\beta}}$ (*avg*)	$\hat{\mathbf{Q}}$ (*sif*)	$\mathbf{Q}$ (*sif*)	$\mathbf{Q}_{\overline{\beta}}$ (*sif*)
Lf + Tt $\Longrightarrow$ Dc	76.2 ± 0.1	75.2 ± 0.2	82.0 ± 0.2	83.0 ± 0.2	81.3 ± 0.2	87.3 ± 0.1	$\mathbf{87.5 \pm 0.0}$
Dc + Tt $\Longrightarrow$ Lf	84.5 ± 0.2	81.6 ± 0.2	87.2 ± 0.1	88.0 ± 0.1	84.6 ± 0.1	90.1 ± 0.1	$\mathbf{90.4 \pm 0.1}$
Lf + Dc $\Longrightarrow$ Tt	82.5 ± 0.3	82.2 ± 0.3	87.8 ± 0.2	88.1 ± 0.2	86.4 ± 0.1	91.6 ± 0.2	$\mathbf{91.8 \pm 0.2}$

Table 4. Macro-AUC (%) in English multi-source music genre translation with standard deviation computed over 4 folds. Results are shown for different embedding initialization (*avg* and *sif*), used directly ($\hat{\mathbf{Q}}$) or retrofitted with the original retrofitting ($\mathbf{Q}$) or with our version ($\mathbf{Q}_{\overline{\beta}}$). The baseline consists in tag alignment against English DBpedia music genres [6].

tag systems as targets. Compared to the experiments reported in Section 4.3, this time, we observe only a marginal difference between the original retrofitting and our version.

Further, we give more details about the translations enabled by the baseline and our retrofitted *sif* embeddings. Often, we yield better music genre mappings (e.g. we map *Discogs:uk garage* on *Tagtraum:garagerock*, while the baseline maps it on *Tagtraum:dubstep*). However, there are also cases where the baseline leads to more accurate mappings (e.g. *Discogs:modal* is mapped on *Lastfm:cooljazz* compared to our best mapping on *Lastfm:jazz*). Finally, the baseline could not map at all some music genres, while we could (e.g. we map *Discogs:crunk* on *Tagtraum:gangstarap* and on *Lastfm:rap*).

To sum up, exploiting the semantics of the music genre graph edges leads to marginally improved results w.r.t. the original retrofitting in the English-language multi-source translation and significantly higher AUC scores in the cross-lingual translation. The *sif* initialization yields better translations than the *avg* initialization. Lastly, we outperform the baselines by large margins in both experiments.

5. CONCLUSION

In this paper, we presented a new multi-step method for multilingual music genre embeddings learning. This method combines pre-trained word embeddings, music genre graphs and a retrofitting method leveraging different types of music genre relations to adapt embeddings to the music domain and learn embeddings for music genres with unknown words in the pre-trained word embeddings vocabulary. Our experiments demonstrate the effectiveness of the proposed method, both in the English-language multi-source and the new cross-lingual translation tasks.

For future work, we plan to learn embeddings for each music genre relation type. Fang et al. [33] consider that each edge represents a linear translation from the embedding of one node to the embeddings of its neighbour. In a generalized setup, functional retrofitting proposed by Lengerich et al. [32] defines a linear relational penalty

function for each type of relation in the graph.

Then, we aim to address the incremental updates of the music genre graph in order to avoid re-applying retrofitting every time the graph is updated. Instead of relying on the constraints represented by the graph edges directly in retrofitting, Glavaš and Vulić [43] use them as training instances to learn an explicit retrofitting function, which can be after applied to new node embeddings.

Further, we want to apply our method to new languages, especially from other language families, as well as to investigate other pre-trained word embeddings and alternatives to embed multi-word concepts [26]. For this, the current music genre graph needs to be populated with new multilingual music genres and their relations, and a parallel corpus of music items covering new languages should be collected if further evaluation is required. Continuing to rely on the multilingual DBpedia is an option, though a limiting one, given that only some world languages are supported. Thus, music genre translation involving resource-poor languages remains a challenge. However, for the supported languages, our approach allows generating cross-lingual music genre annotations, which could be useful for other music information retrieval and recommendation tasks such as language-aware music genre autotagging, localized playlist captioning and music genre-driven recommendations, cross-cultural music genre perception modeling for user studies.

Finally, the multilingual data and the code to learn and evaluate music genre embeddings are made available to the community [5] . Also, a demo to visualize the music genre vector space and the cross-lingual translation results for DBpedia music items is available [44].

---

[5] https://github.com/deezer/MultilingualMusicGenreEmbedding

7. REFERENCES

[1] M. I. Mandel, D. Eck, and Y. Bengio, "Learning tags that vary within a song," in *Conference of the International Society for Music Information Retrieval*, 2010.

[2] M. Schedl and B. Ferwerda, "Large-scale analysis of group-specific music genre taste from collaborative tags," in *IEEE International Symposium on Multimedia*, 2017.

[3] A. J. Craft, G. A. Wiggins, and T. Crawford, "How many beans make five? The consensus problem in music-genre classification and a new evaluation method for single-genre categorisation systems," in *Conference of the International Society on Music Information Retrieval*, 2007.

[4] J. H. Lee, K. Choi, X. Hu, and J. Downie, "K-pop genres: A cross-cultural exploration," in *Conference of the International Society on Music Information Retrieval*, 2013.

[5] M. Sordo, O. Celma, M. Blech, and E. Guaus, "The Quest for Musical Genres: Do the Experts and the Wisdom of Crowds Agree?" in *Conference of the International Society on Music Information Retrieval*, 2008.

[6] E. V. Epure, A. Khlif, and R. Hennequin, "Leveraging knowledge bases and parallel annotations for music genre translation," in *Conference of the International Society for Music Information Retrieval*, 2019.

[7] R. Hennequin, J. Royo-letelier, and M. Moussallam, "Audio based disambiguation of music genre tags," in *Conference of the International Society of Music Information Retrieval*, 2018.

[8] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *International Conference on Language Resources and Evaluation*, 2018.

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[10] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *International Conference on Language Resources and Evaluation*, 2018.

[11] M. Artetxe, G. Labaka, and E. Agirre, "A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings," in *Annual Meeting of the Association for Computational Linguistics*, 2018.

[12] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv preprint arXiv:1309.4168*, 2013.

[13] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The Semantic Web*. Springer, 2007, pp. 722–735.

[14] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[15] S. Oramas, O. Nieto, F. Barbieri, and X. Serra, "Multi-label music genre classification from audio, text and images using deep features," in *Conference of the International Society on Music Information Retrieval*, 2017.

[16] E. Coviello, R. Miotto, and G. R. Lanckriet, "Combining content-based auto-taggers with decision-fusion," in *Conference of the International Society on Music Information Retrieval*, 2011.

[17] D. Bogdanov, A. Porter, J. Urbano, and H. Schreiber, "Mediaeval 2017 acousticbrainz genre task: content-based music genre recognition from multiple sources," in *MediaEval 2017 AcousticBrainz*, 2017.

[18] H. Schreiber, "Genre ontology learning: Comparing curated with crowd-sourced ontologies." in *Conference of the International Society for Music Information Retrieval*, 2019.

[19] M. Achichi, P. Lisena, K. Todorov, R. Troncy, and J. Delahousse, "Doremus: A graph of linked musical works," in *International Semantic Web Conference*, 2018.

[20] P. Lisena, K. Todorov, C. Cecconi, F. Leresche, I. Canno, F. Puyrenier, M. Voisin, T. Le Meur, and R. Troncy, "Controlled vocabularies for music metadata," in *Conference of the International Society on Music Information Retrieval*, 2018.

[21] R. D. Smith, "Distinct word length frequencies: distributions and symbol entropies," *Glottometrics*, vol. 23, pp. 7–22, 2012.

[22] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[23] J. Li and D. Jurafsky, "Do multi-sense embeddings improve natural language understanding?" in *Conference on Empirical Methods in Natural Language Processing*, 2015.

[24] C. V. Gysel, M. De Rijke, and E. Kanoulas, "Neural vector spaces for unsupervised information retrieval," *ACM Transactions on Information Systems (TOIS)*, vol. 36, no. 4, pp. 1–25, 2018.

[25] A. Joulin, P. Bojanowski, T. Mikolov, H. Jégou, and E. Grave, "Loss in translation: Learning bilingual word mapping with a retrieval criterion," in *Conference on Empirical Methods in Natural Language Processing*, 2018.

[26] V. Shwartz, "A systematic comparison of English noun compound representations," in *Joint Workshop on Multiword Expressions and WordNet*. Association for Computational Linguistics, 2019, pp. 92–103.

[27] B. Mandelbrot, "An informational theory of the statistical structure of language," *Communication theory*, vol. 84, pp. 486–502, 1953.

[28] G. K. Zipf, *Human behavior and the principle of least effort*. MA, Addison-Wesle: Cambridge, 1949.

[29] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *International Conference on Learning Representations*, 2017.

[30] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," in *Linear Algebra*. Springer, 1971, pp. 134–151.

[31] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[32] B. J. Lengerich, A. L. Maas, and C. Potts, "Retrofitting distributional embeddings to knowledge graphs with functional relations," in *International Conference on Computational Linguistics*, 2018.

[33] L. Fang, Y. Luo, K. Feng, K. Zhao, and A. Hu, "Knowledge-enhanced ensemble learning for word embeddings," in *World Wide Web Conference*, 2019.

[34] T. Scheepers, E. Kanoulas, and E. Gavves, "Improving word embedding compositionality using lexicographic definitions," in *World Wide Web Conference*, 2018.

[35] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[36] Y. Bengio, O. Delalleau, and N. Le Roux, "Label propagation and quadratic criterion," *Semi-Supervised Learning*, pp. 193–216, 2006.

[37] T. K. Saha, S. Joty, N. Hassan, and M. A. Hasan, "Diss2v: Discourse informed sen2vec," *arXiv preprint arXiv:1610.08078*, 2016.

[38] D. Hayes, "What just happened? Evaluating retrofitted distributional word vectors," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[39] R. Speer and J. Chin, "An ensemble method to produce high-quality word embeddings," *arXiv preprint arXiv:1604.01692*, 2016.

[40] K. Ibrahim, J. Royo-Letelier, E. Epure, G. Peeters, and G. Richard, "Audio-based auto-tagging with contextual tags for music," in *International Conference on Acoustics, Speech, and Signal Processing*, ser. ICASSP, 05 2020, pp. 16–20.

[41] K. Sechidis, G. Tsoumakas, and I. Vlahavas, "On the stratification of multi-label data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2011.

[42] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

[43] G. Glavaš and I. Vulić, "Explicit retrofitting of distributional word vectors," in *Annual Meeting of the Association for Computational Linguistics*, 2018.

[44] E. V. Epure, G. Salha, F. Voituret, M. Baranes, and R. Hennequin, "Muzeeglot: annotation multilingue et multi-sources d'entités musicales à partir de représentations de genres musicaux," in *6e conférence conjointe Journées d'Études sur la Parole (JEP, 31e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 4: Démonstrations et résumés d'articles internationaux*. ATALA, 2020, pp. 18–21.

MODELLING HIERARCHICAL KEY STRUCTURE WITH PITCH SCAPES

Robert Lieck
Digital and Cognitive Musicology Lab
EPFL, Switzerland
research@robert-lieck.com

Martin Rohrmeier
Digital and Cognitive Musicology Lab
EPFL, Switzerland
martin.rohrmeier@epfl.ch

ABSTRACT

Musical form and syntax in Western classical music are hierarchically organised on different timescales. One of the most important features of this structure is the organisation of modulations between different keys throughout a piece. Music theoretical research has established taxonomies of prototypical modulation plans for different modes and musical forms. However, these prototypes still require empirical validation based on quantitative statistical methods and cannot be retrieved automatically so far.

In this paper, we present a novel method to infer prototypical modulation plans from musical corpora. A modulation plan is formalised as a transposition-invariant probabilistic model over the underlying pitch class distributions based on a hierarchical *pitch scape* representation. Prototypical modulation plans can be learned in an unsupervised manner by training a mixture model (similar to a Gaussian mixture model) on the data, so that different prototypes appear as distinct clusters.

We evaluate our approach by performing hierarchical clustering on a corpus of more than 150 Baroque pieces, with the extracted clusters showing excellent agreement with the most common prototypes postulated in music theory. Our method bears a great potential for modelling, analysis and discovery of hierarchical key structure and prototypes in corpora across a broad range of musical styles. An accompanying library is available at: github.com/robert-lieck/pitchscapes.

1. INTRODUCTION

The hierarchical structure of a piece in Western classical music is strongly determined by musical form [1] and harmonic syntax [2, 3], based on different aspects, such as repetition and variation of the rhythmic, melodic and harmonic content and hierarchical relations between different harmonies.

A central aspect that links musical form and harmonic syntax is the modulation plan of a piece. Western musicology assumes a number of prototypical modulation plans that describe the overarching tonal structure of a piece,

such as I–V–I for pieces in major or i–III–i for pieces in minor [1]. These prototypes have a long-standing history in musicology and have emerged from inspection of numerous individual pieces and agreement among experts. However, a quantitative validation based on statistical methods constitutes an important supplement to confirm and refine the music theoretic findings. Furthermore, they cannot be automatically retrieved from musical data, which impedes large-scale investigations and the application to other styles and genres of music.

In this paper, we present a method to retrieve prototypical modulation plans from large corpora of musical pieces in an unsupervised manner. This is achieved by modelling the overall corpus as a mixture of multiple prototypes, similar to how Gaussian mixture models [4] can be applied to clustering in Euclidean space. A prototype is represented by a transposition-invariant Bayesian model that describes the pitch content of a piece (pitch class distributions) on multiple time scales. Modelling is based on a novel *pitch scape* representation of the musical content, which allows to account for the hierarchical structure inherent to both musical form and harmonic syntax. We evaluate our model on a corpus of more than 150 Baroque pieces, with the extracted clusters showing excellent agreement with the most common prototypes postulated in music theory.

By providing a solid statistical approach to modelling prototypical modulation plans, we make an important contribution to connecting music theory and empirical science. Our approach relies on minimal prior assumptions, works on simple pitch data, and learns prototypes in an unsupervised manner, which bears a great potential for modelling, analysis and discovery of hierarchical key structure and prototypes in corpora across a broad range of musical styles.
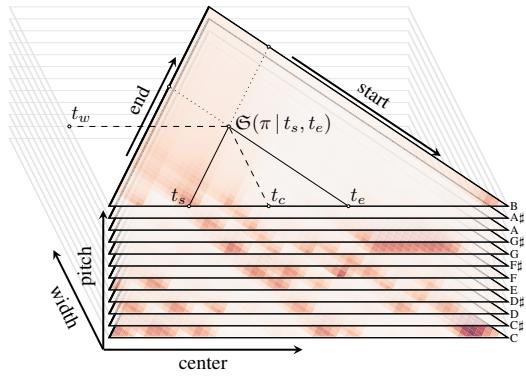
In the remainder of the paper, we describe the underlying *pitch scape* representation in Section 2, introduce the probabilistic Bayesian model that is used to learn prototypes and prototype mixtures from musical corpora in Section 3, and present and discuss the results of our evaluation in Section 4.

2. PITCH SCAPES

We model prototypical modulation plans based on a novel *pitch scape* representation of the musical content. Pitch scapes (see Figure 1 for an illustration) represent the pitch content of a piece on multiple time scales and can be for-

Figure 1. Pitch scape (Prelude in C major, BWV 846, Johann Sebastian Bach). The two time values can be specified in start-end-coordinates (t_s and t_e) or in center-width-coordinates (t_c and t_w).

mally defined as the conditional probability distribution of the pitch classes for a given section of the piece:

Definition 1 (Pitch Scape). *A pitch scape $\mathfrak{S}$ is a function that maps each proper time interval $[t_s, t_e]$ ($t_s < t_e$) to a pitch class distribution*

$$\mathfrak{S} : \mathbb{R} \times \mathbb{R} \to [0,1]^{12} , \quad \sum_{\pi=0}^{11} \mathfrak{S}(\pi \,|\, t_s, t_e) = 1 . \quad (1)$$

A pitch scape can equivalently be conceived as a conditional probability distribution $\mathfrak{S}(\pi \,|\, t_s, t_e)$ with three variables or a vector-valued function $\mathfrak{S}(t_s, t_e)$ in two variables.
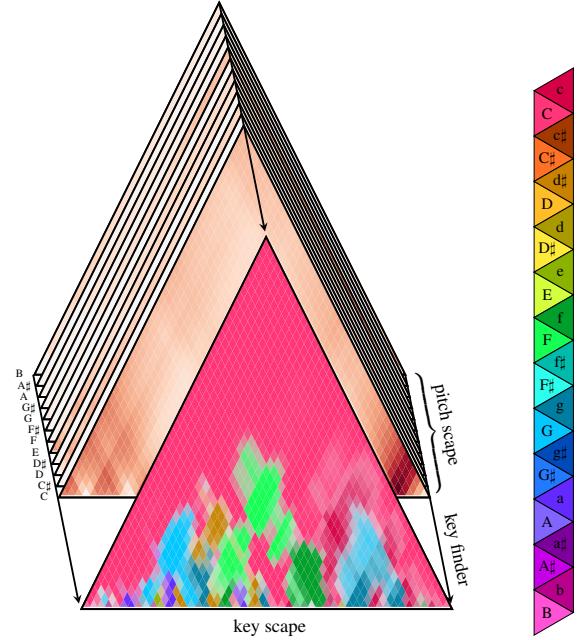
Pitch scapes are inspired by scape plot visualisations, to which we draw the connection in Section 2.1, while Section 2.2 describes how to compute pitch scape estimates for a given piece.

2.1 Pitch Scape Visualisation

Scape plot visualisations were introduced in [5,6] to depict key estimates for different sections of a piece in a hierarchical triangular plot and have since been used for a variety of visualisation tasks [7–11].

Visualising the entire information contained in a three-dimensional pitch scape in a single two-dimensional plot is difficult. However, there are two convenient ways to visualise the relevant information. First, the 12 components can be visualised separately by creating one scape plot per pitch class. This preserves the entire information but does not foster musical intuition because information about simultaneous events is scattered across multiple plots. Alternatively, a key finding algorithm can be employed [12–15] to map the pitch class distribution of each point to a colour value. This corresponds to a key-scape plot of the pitch scape. For illustration, we show in Figure 2 an overlay of the 12 separate pitch-scape plots and the corresponding key-scape plot.

The colour mapping for a key-scape plot can be realised in different ways. We use a template-based key finder that

Figure 2. Separate pitch-scape plots and resulting key-scape plot for the prelude in C major, BWV 846, Johann Sebastian Bach (colour legend for keys on the right).

provides a score value for each major and minor key. The scores can be transformed into a probability distribution $p(k)$ using a soft-max function. After choosing a unique colour for each key, $p(k)$ can be used to interpolate between colours by computing their weighted average. To define the colour for each key, we let the hue value vary either along the circle of fifths or chromatically, which has complementary advantages. Fifth-based hue maps related keys to similar colours, while chromatic hue allows to better distinguish them. We add a lightness offset to distinguish major and minor keys and map the entropy of $p(k)$ to saturation. Entropy-based saturation allows to indicate regions with uncertain key classification and avoids uninterpretable colour blends.

2.2 Pitch Scape Estimates

The pitch scape of a piece is computed from its musical content and reflects the probability of a certain pitch class to occur within the specified time interval. As we are working in a Bayesian framework, we model the pitch scape of a piece as a posterior estimate given a prior distribution and the observed notes. To formally define the pitch scape estimate of a piece, we first define its pitch class density:

Definition 2 (Pitch Class Density). *The pitch class density $\delta(\pi \,|\, t)$ for pitch class π at time t corresponds to the normalised pitch class counts over all tones that sound at time t*

$$\delta(\pi \,|\, t) := \frac{1}{\max\{1, |T_t|\}} \sum_{\tau \in T_t} [\![\tau \bmod 12 = \pi]\!] , \quad (2)$$

where T_t is the multiset of all tones (as integers in MIDI pitch representation) sounding at time t; $[\![\cdot]\!]$ is the Iverson

bracket, which equals 1 if its argument is true and 0 otherwise; and the max *avoids division by zero for silent parts where $T_t = \emptyset$ is the empty set.*

Using the pitch class density, we define the pitch scape estimate as follows:

Definition 3 (Pitch Scape Estimate). *The posterior estimate of the pitch scape $\mathfrak{S}(\pi \,|\, t_s, t_e)$ for pitch class π and time interval $[t_s, t_e]$ is*

$$\mathfrak{S}(\pi \,|\, t_s, t_e) := \underbrace{\frac{1}{t_e - t_s + 12c}}_{\text{normalisation}} \left[c + \overbrace{\underbrace{\int_{t_s}^{t_e} \delta(\pi \,|\, t) \, dt}_{\text{overall pitch class counts}}}^{\text{prior counts}} \right], \quad (3)$$

where the integral over the pitch class density computes the overall pitch class counts, $c \geq 0$ specifies the prior counts, and the leading term ensures proper normalisation.

Using zero prior counts $c = 0$ thus corresponds to using the average pitch class density as pitch scape estimate (in Bayesian terms this would be a maximum likelihood estimate). In contrast, using a prior count of $c = 1$, which is done throughout the paper, corresponds to a Bayesian maximum posterior estimate with a uniform prior over pitch classes ($c = 1$ corresponds to a uniform Dirichlet distribution, which is the appropriate conjugate prior for the categorical distribution over pitch classes). Note that choosing $c > 0$ also circumvents the zero-count problem for silent parts.

The relative weight of the overall pitch class counts, computed in the integral in (3), depends on the scale on which time is measured. Throughout the paper, we measure time in quarter notes, so that a time interval of one quarter note has the weight of a single observation. That means, for instance, a single pitch sustained for two quarter note adds two to the respective overall pitch class counts; two different pitches sustained for one quarter note add half a count each; and three pitches sustained for an eight note add one sixth count each. Thus, for small time intervals the prior counts c introduce a significant bias towards a uniform pitch class distribution, while for large time intervals they have a vanishingly small weight relative to the overall pitch class counts (e.g. a 32-bar piece in 4/4 yields a total of 128 pitch class counts, so that the prior counts do not cause a major change of the estimated pitch class distribution for the entire piece).

3. MODELLING KEY STRUCTURE

We define our model for mixtures of prototypes in two steps. First, we define a probabilistic pitch scape model of a transposition-invariant modulation plan (Section 3.1). Based on this model for single prototypes, we define a mixture model (Section 3.2) that incorporates explicit transposition and models a musical corpus as a mixture of multiple prototypes.

3.1 Prototypes

The idea of a prototype is to specify an object that represents a subset of the data. In probabilistic modelling this corresponds to defining a probability distribution for which a subset of the data has a high likelihood. Additionally, this distribution should be unimodal so that its mode can be taken as a representative of all data points belonging to that prototype. In n-dimensional Euclidean space, prototypes can for instance be defined using multivariate Gaussian distributions.

When defining prototypes for pitch scapes, we are facing some additional challenges that will be addressed in the following. First, the output of a pitch scape is a categorical distribution (over pitch classes), which has to be normalised. Second, time is continuous so that a pitch scape itself is an inherently continuous object. Third, the prototypes postulated in music theory are formulated in terms of scale degrees, which makes them transposition-invariant (i.e. transposing a piece does not affect its relation to a specific prototype). The first two points will be addressed in the following Section 3.1.1, the third point is resolved in Section 3.1.3 and incorporated in the mixture model in Section 3.2.

3.1.1 Definition

We address the first two points by defining a prototype as a point-wise Dirichlet distribution with a time-dependent parameter vector α. The Dirichlet distribution is the conjugate prior of a categorical distribution and acts as a likelihood function if the observations themselves are categorical distributions, as it is the case for individual points in a pitch scape (first point). Making its parameter vector time-dependent additionally allows it to vary continuously over the pitch scape (second point). Formally, a prototype is defined as follows:

Definition 4 (Prototype). *Given a function*

$$\alpha : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+^{12} \qquad (4)$$

that maps each proper time interval $[t_s, t_e]$ ($t_s < t_e$) to a vector with positive entries, a prototype is defined as the point-wise Dirichlet distribution with parameter vector α. The likelihood of observing a pitch class distribution Π for the interval $[t_s, t_e]$ given α is

$$p(\Pi \,|\, \alpha, t_s, t_e) = \text{Dir}(\Pi; \alpha(t_s, t_e)) \,. \qquad (5)$$

The log-likelihood of observing a full pitch scape $\mathfrak{S}$ given α is

$$\log p(\mathfrak{S} \,|\, \alpha) = \frac{2}{T^2} \iint_{0 \leq t_s < t_e \leq T} \log \text{Dir}\big(\mathfrak{S}(t_s, t_e); \alpha(t_s, t_e)\big) dt_s dt_e \,,$$
$$(6)$$

where T is the duration of the piece.

The definition of the log-likelihood in (6) is equivalent to the (negative) cross-entropy of an infinite number of uniform samples from the pitch scape. It differs from a simple integration of (5) only by the normalisation $\frac{2}{T^2}$, which rescales it to the magnitude of a single observation and makes it invariant to the duration of the piece. Note that both (5) and (6) are probability *density* functions with the usual implications (i.e. they can be greater than 1; their log can be positive; their cross-entropy can be negative).

3.1.2 Proxy Function

To learn prototypes from data, we define α via a three dimensional real-valued proxy function $\widetilde{\alpha}$ that has a set of adjustable parameters θ and an open parameter τ

$$\widetilde{\alpha}^{(\theta,\tau)} : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R} . \qquad (7)$$

The domain of interest is $[0, 1] \times [0, 1] \times \mathbb{Z}_{12}$ with the first two arguments specifying the time interval in normalised center-width-coordinates and the third specifying the pitch class. The πth component of α is then defined to be

$$\alpha_\pi^{(\theta,\tau)}(t_s, t_e) := e^{\widetilde{\alpha}^{(\theta,\tau)}(\bar{t}_c, \bar{t}_w, \pi)} \qquad (8)$$

with

$$\bar{t}_c = \frac{1}{2T}(t_s + t_e) \qquad \bar{t}_w = \frac{1}{T}(t_e - t_s) , \qquad (9)$$

where T is the duration of the piece that is to be modelled.

3.1.3 Fourier Representation

We parameterise $\widetilde{\alpha}$ as a Fourier series in three dimensions [16]

$$\widetilde{\alpha}^{(\theta,\tau)}(\mathbf{x}) = \sum_{\mathbf{n}} \theta_{\mathbf{n}} e^{2\pi i \mathbf{k_n} \cdot \mathbf{x}} , \qquad (10)$$

where π (only in this equation!) is the mathematical constant. The index vector $\mathbf{n}$, wave vector $\mathbf{k_n}$, and location vector $\mathbf{x}$ are

$$\mathbf{x} := (\bar{t}_c, \bar{t}_w, \pi) \qquad n_c \in \{-N_c, \dots, N_c\} \quad (11)$$

$$\mathbf{n} := (n_c, n_w, n_\pi) \qquad n_w \in \{-N_w, \dots, N_w\} \quad (12)$$
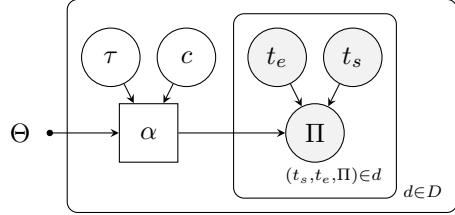
$$\mathbf{k_n} := \left(\sigma_c n_c, \sigma_w n_w, \frac{n_\pi + \tau}{12}\right) \quad n_\pi \in \{-6, \dots, 6\} . \quad (13)$$

N_c and N_w allow to independently control the smoothness (or bandwidth) of $\widetilde{\alpha}$ for the center and width dimension, respectively; $\tau \in \mathbb{Z}_{12}$ represents the transposition of the prototype (see below); and $\sigma = 1 - \frac{1}{2N}$ is a scaling factor. Scaling is required because we do not want $\widetilde{\alpha}$ to have periodic boundaries conditions in the time dimensions. The Nyquist frequency of the unscaled function is $2N$, the scaling factor thus stretches the function such that a critical fraction of $\frac{1}{2N}$ is moved out of the interval $[0, 1]$. This is not relevant for the pitch dimension because the space of pitch classes is inherently periodic and, moreover, we have a complete discrete Fourier series that allows to represent any function exactly. As $\widetilde{\alpha}$ is real-valued, $\theta_{\mathbf{n}}$ and $\theta_{-\mathbf{n}}$ are complex conjugates and (due to the properties of the discrete Fourier transform) all coefficients with $n_\pi = \pm 6$ are real-valued. We can thus store the parameters θ in a real-valued array of dimensions $(2N_c + 1, 2N_w + 1, 12)$.

As $\widetilde{\alpha}$ (and thus α) are periodic in the pitch dimension, the Fourier representation can be understood as a transposition-invariant formulation of a prototype. When creating a concrete instance of the prototype, τ needs to be specified and defines a specific transposition by inducing a corresponding phase shift through the cyclic pitch class space.

3.2 Mixture Model

In Section 3.1 we defined prototypes that have a point-wise Dirichlet distribution (Definition 4) and adjustable parameters θ. We will now build a transposition-invariant mixture

Figure 3. Graphical representation of our mixture model. Θ are the prototype parameters; c and τ the piece-specific cluster index and transposition; α the deterministically generated prototype instance; and $\Pi = \mathfrak{S}(t_s, t_e)$ the pitch scape values at intervals $[t_s, t_e]$ (see text for more details).

model using these prototypes. The overall structure of the model is shown in Figure 3 as a graphical model [17] and will be explained in detail below.

Our model is similar to classical topic models for corpora [18–20] with two nested levels. Each piece (or document) d in the data set D is generated independently from a specific prototype with parameters $\theta = \Theta_c$ and transposition τ (outer plate) and for a specific piece, each point Π in its pitch scape is generated independently (inner plate). [1]

3.2.1 Inference

We want to find parameters Θ^* that minimise the cross-entropy (i.e. maximise the likelihood) of our data D

$$\Theta^* = \text{argmin}_\Theta -\frac{1}{|D|} \log p(D \mid \Theta) , \qquad (14)$$

where

$$\log p(D \mid \Theta) = \sum_{d \in D} \log \sum_{c,\tau} p(d \mid \alpha^{(\Theta_c, \tau)}) \, p(c) \, p(\tau) , \quad (15)$$

is the data log-likelihood with the latent variables c and τ being marginalised out. The prior terms $p(c)$ and $p(\tau)$ are assumed to be constant so that a priori no specific prototype or transposition is preferred. We use

$$\log p(d \mid \alpha) = \frac{1}{|d|} \sum_{(t_s, t_e, \Pi) \in d} \log \text{Dir}\big(\Pi; \alpha(t_s, t_e)\big) \qquad (16)$$

to approximate the piece likelihood (6) based on a finite number of uniform samples. Marginalising out c and τ also readily yields the normalisation factor for the cluster and transposition probability for a piece

$$p(c, \tau \mid d) \propto p(d, c, \tau) = p(d \mid \alpha^{(\Theta_c, \tau)}) \, p(c) \, p(\tau) . \quad (17)$$

The optimal parameters Θ^* can be found by performing gradient descent on the cross-entropy (14).

---

[1] The assumption of different points in the pitch scape being generated independently is obviously incorrect, which is common to all topic models and the reason why they are not well suited to generate coherent data (e.g. text or music). In fact, in a pitch scape the values at different locations are highly correlated and would ideally be modelled as a single continuous latent function. One approach to achieve this are Gaussian processes (GPs) [21]. However, GPs are computationally expensive and GPs for multi-class classification have complex kernel functions and require approximations of the analytically intractable posterior distribution [21–23]. As we are primarily interested in extracting the mean pitch scape (corresponding to the GP prior), which represents a specific prototype, we therefore chose the simpler approach of defining prototypes as a point-wise Dirichlet distribution.

3.2.2 Hierarchical Clustering

Training the mixture model on a data set allows to perform unsupervised clustering with a fixed number of clusters. However, our motivation is a comparison with the prototypes described in the music theory literature. Instead of choosing a fixed number of clusters, we are rather interested in how clusters split hierarchically from more generic prototypes to more specific ones. We therefore take a hierarchical top-down clustering approach.

We start by training a single prototype on the whole corpus and perform a binary split of this cluster by using it to initialise a mixture of two prototypes, while adding minimal noise (10^{-8}) to the parameters θ to allow the clusters to properly split. This procedure is then recursively and *separately* applied to the resulting prototypes. To this end, the probability $p(d \,|\, c')$ of a piece d to fall into the parent cluster c' is used as a weight in (15) when training the two child clusters. This ensures a clear assignment between parent and child clusters and implies that only pieces that fell into the parent cluster influence the children.

After establishing a hierarchy of prototypes in this way, we perform a joint refinement of the resulting clusters. To this end, *all* final child clusters are combined in a single model while lifting the parent-specific piece weights. This serves a two-fold purpose. First, the prototypes may be sharpened as interactions between the clusters can now be exploited. Second, it acts as a sanity check for the established hierarchy: If the child clusters remain stable in the refinement phase, this indicates consistency of the hierarchical splitting.

4. EVALUATION

4.1 Experimental Setup

The model was implemented in `PyTorch` [24] and the parameters Θ were optimised via gradient descent using the Adam optimiser [25]. The "warm start" with pre-initialised clusters was realised by using a small initial learning rate (10^{-5}) to allow for the mean and variance estimators (internals of the Adam optimiser) to stabilise before reaching the normal learning rate (10^{-3}).

We trained our model on a corpus of 155 Baroque pieces in MIDI format by Johann Sebastian Bach (84%: WTK I/II; Brandenburgisches Konzert No. 5; Inventions and Sinfonias), Georg Friedrich Händel (4%: HWV 264, Movement 2, 4, 9, 10, 11, and 13; HWV 435), and Domenico Scarlatti (12%: Sonatas), see Appendix C for a complete list. As opposed to later periods with an increasing amount of chromaticism, the modulation plans of Baroque pieces are expected to more closely conform to the respective prototypes. Each piece was sampled by choosing interval start and end points on a uniform time grid of $n = 50$ points, resulting in $n(n-1)/2 = 1225$ samples per piece.

Hierarchical clustering was performed up to depth 3 (8 final clusters) with subsequent refinement (see Section 3.2.2). The hierarchy and final prototypes are shown in Figure 4 and discussed below.

4.2 Results and Discussion

The root cluster, which was trained on the entire corpus, represents a generic diatonic prototype. It does not unambiguously belong to a particular mode, being classified as minor based on 'Albrecht' profiles (as in Figure 4) and major based on 'Temperley' profiles. This is also reflected in the separate pitch-scape plots (Table 1 in Appendix B), which have strong weights for the entire pentatonic segment of the line of fifths (C–G–D–A–E). This can be interpreted as confirming the view that Baroque music is fundamentally diatonic.

The initial split results in a clear separation of major and minor keys, with cluster (0) and all its descendants being globally classified as minor pieces while (1) and its descendants are classified as major. From now on we can see a pronounced weight on the tonic pitch class (C for major, A for minor) at the beginning and end of a piece in the separate pitch-scape plots. This split into major and minor prototypes again is an important finding that confirms these two modes to be dominant in Baroque music.

4.2.1 Prototypes in Minor
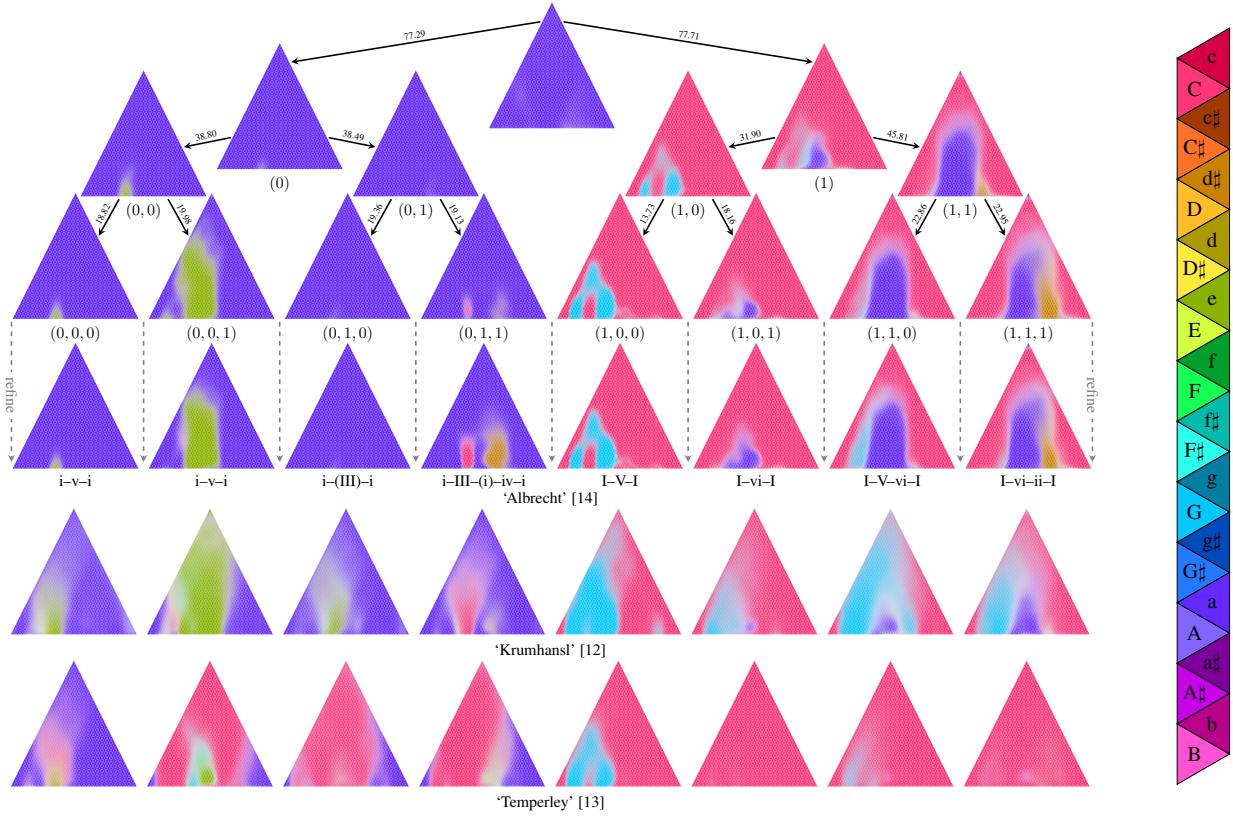
The next split of the minor cluster separates the two most common prototypes. Prototypical minor-mode pieces are assumed to either modulate to the key of v (the dominant) or to the key of III (relative major) before returning to i (tonic), which corresponds to (0,0) and (0,1) and their descendants, respectively. Note that the cluster (0,1,0) also has a strong tendency to modulate to III, which becomes more apparent when using 'Temperley' profiles.

The i–v–i modulation plan of cluster (0,0,0) and (0,0,1) is one of the two standard prototypes for minor pieces. In (0,0,1), the v is more pronounced and the middle section also has a certain tendency to modulate to III, possibly even including a short VII passage (see 'Temperley' profiles). This corresponds to a i–III–(VII)–v–III–i modulation plan, which is a common subtype of the i–v–i prototype that features two fifth-related, modally distinct key pairs: i–v and III–VII.

Cluster (0,1,0) and (0,1,1) both fall under the general i–III–i prototype. The (0,1,0) cluster has a less pronounced III, which may be partly due to the III being at different locations in the corresponding pieces, thus leading to smoothing/averaging. According to 'Krumhansl' profiles, there is a tendency for modulation to v in the middle section and 'Temperley' profiles classify larger parts of the middle section as III. Cluster (0,1,1) has an additional modulation to the iv after the III, possibly with a short return to the i in between (again this could also be an effect of averaging over multiple pieces), representing the common subtype i–III–(i)–iv–i.

4.2.2 Prototypes in Major

Major pieces are generally assumed to modulate to V before going back to I. However, this general prototype can be elaborated in different ways. For the split of the major cluster (1), we see a very pronounced I–V–I prototype on the left with (1,0) and its child (1,0,0).

Figure 4. Results of hierarchical clustering with subsequent refinement. To visualise the prototypes, the transposition parameter τ was fixed to minimise the accidentals of the diatonic root cluster. The corresponding absolute keys are shown in the chromatic colour scale (right). However, only relative keys (scale degrees) bear interpretable meaning as the prototypes are inherently transposition-invariant. Prototypes are labelled with a hierarchical index; the final prototypes (after refinement) are labelled with the corresponding modulation plan in Roman numeral notation; numbers on the arrows indicate the number of pieces falling into the respective cluster. Key estimates for colouring are computed using 'Albrecht' [14] templates; the final prototypes are repeated using 'Krumhansl' [12] and 'Temperley' [13] templates to improve interpretability. For better disambiguation, Figure 5 and Figure 6 in Appendix A show chromatic and fifth-based colouring in comparison.

The remaining three clusters all belong to one of the most common elaborations of the I–V–I prototype with an additional vi (relative minor) passage after the V. The V passage is most clearly pronounced in the (1,1,0) cluster, to a lesser extent in the (1,1,1) and even less in the (1,0,1) cluster (see especially the 'Krumhansl' profiles). Notably, (1,1,1) has an additional ii passage after the vi.

The I–vi–I and the I–vi–ii–I cluster taken separately do not contradict expectations from music theory, but due to the missing V they are less typical than the other prototypes so far. However, when being combined with the I–V–vi–I cluster, theses clusters form the very common subtype I–V–vi–ii–I [1]. This is typical in Baroque music but also in modern Pop music, where on the chord-level this sequence is known as "the four chord song" (optionally with a IV as an equivalent pre-dominant replacement of the ii). This combination of multiple prototypes suggests the existence *prototype sub-spaces*.

5. CONCLUSION

To address the problem of modelling and automatic retrieval of prototypical modulation plans from a corpus of musical pieces, a probabilistic Bayesian model of transposition-invariant prototypes was introduced. This model was based on a novel hierarchical *pitch scape* representation of the musical content. We learned prototypical modulation plans from a corpus of Baroque pieces, empirically confirming common prototypes postulated in music theory. Extending the conventional music theoretical concepts, we found that continuous *prototype sub-spaces* can be generated as the superposition of multiple prototypes.

Our approach relies on minimal prior assumptions, works on simple pitch data and delivers robust results while being scalable to large data sets. It can therefore be applied to model, analyse and discover hierarchical key structures and prototypes in a wide range of musical styles and genres, including diachronic studies of musical form and syntax in Western classical music, the influence of style- and composer-specific elements, and the investigation of modulation plans in other genres such as Jazz, Pop and Rock music. Therefore, our approach is suited for numerous applications and contributes a valuable method for music information retrieval.

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] E. Aldwell and C. Schachter, *Harmony and Voice Leading*. Thomson/Schirmer, 2003.

[2] M. Rohrmeier and M. Pearce, "Musical Syntax I: Theoretical Perspectives," in *Springer Handbook of Systematic Musicology*, R. Bader, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 473–486.

[3] M. Rohrmeier, "The Syntax of Jazz Harmony: Diatonic Tonality, Phrase Structure, and Form," *Music Theory and Analysis (MTA)*, vol. 7, no. 1, pp. 1–63, Apr. 2020.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, 2007.

[5] C. S. Sapp, "Harmonic Visualizations of Tonal Music." in *ICMC*, vol. 1. Citeseer, 2001, pp. 419–422.

[6] ——, "Visual hierarchical key analysis," *Computers in Entertainment*, vol. 3, no. 4, p. 3, Oct. 2005.

[7] M. Müller and N. Jiang, "A Scape Plot Representation for Visualizing Repetitive Structures of Music Recordings." in *ISMIR*. Citeseer, 2012, pp. 97–102.

[8] N. Jiang and M. Müller, "Towards efficient audio thumbnailing," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2014, pp. 5192–5196.

[9] C. Weiß and M. Müller, "Quantifying and visualizing tonal complexity," in *Proceedings of the Conference on Interdisciplinary Musicology (CIM)*, 2014, pp. 184–187.

[10] C. Vaquero, "A quantitative study of seven historically informed performances of Bach's bwv1007 Prelude," *Early Music*, vol. 43, no. 4, pp. 611–622, Nov. 2015.

[11] S. Park, T. Kwon, J. Lee, J. Kim, and J. Nam, "A Cross-Scape Plot Representation for Visualizing Symbolic Melodic Similarity," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, A. Flexer, G. Peeters, J. Urbano, and A. Volk, Eds., 2019, pp. 423–430.

[12] C. L. Krumhansl and E. J. Kessler, "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys." *Psychological review*, vol. 89, no. 4, p. 334, 1982.

[13] D. Temperley and E. W. Marvin, "Pitch-Class Distribution and the Identification of Key," *Music Perception*, vol. 25, no. 3, pp. 193–212, Feb. 2008.

[14] J. Albrecht and D. Shanahan, "The use of large corpora to train a new type of key-finding algorithm: An improved treatment of the minor mode," *Music Perception: An Interdisciplinary Journal*, vol. 31, no. 1, pp. 59–67, 2013.

[15] J. D. Albrecht and D. Huron, "A Statistical Approach to Tracing the Historical Development of Major and Minor Pitch Distributions, 1400-1750," *Music Perception*, vol. 31, no. 3, pp. 223–243, Feb. 2014.

[16] B. G. Osgood, *Lectures on the Fourier Transform and Its Applications*. American Mathematical Soc., 2019, vol. 33.

[17] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[18] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, pp. 993–1022, Mar. 2003.

[19] M. Steyvers and T. Griffiths, "Probabilistic Topic Models," *Handbook of latent semantic analysis*, vol. 427, no. 7, pp. 424–440, 2007.

[20] D. Hu and L. K. Saul, "A probabilistic topic model for unsupervised learning of musical key-profiles," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, K. Hirata, G. Tzanetakis, and K. Yoshii, Eds. International Society for Music Information Retrieval, 2009, pp. 441–446.

[21] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts / London, England: The MIT Press, 2006.

[22] M. A. Álvarez, L. Rosasco, and N. D. Lawrence, "Kernels for Vector-Valued Functions: A Review," *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

[23] D. Milios, R. Camoriano, P. Michiardi, L. Rosasco, and M. Filippone, "Dirichlet-based Gaussian processes for large-scale calibrated classification," in *Advances in Neural Information Processing Systems*, 2018, pp. 6005–6015.

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner,

L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds.   Curran Associates, Inc., 2019, pp. 8024–8035.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

CONTENT BASED SINGING VOICE SOURCE SEPARATION VIA STRONG CONDITIONING USING ALIGNED PHONEMES

Gabriel Meseguer-Brocal
STMS UMR9912, Ircam/CNRS/SU, Paris
gabriel.meseguerbrocal@ircam.fr

Geoffroy Peeters
LTCI, Institut Polytechnique de Paris
geoffroy.peeters@telecom-paris.fr

ABSTRACT

Informed source separation has recently gained renewed interest with the introduction of neural networks and the availability of large multitrack datasets containing both the mixture and the separated sources. These approaches use prior information about the target source to improve separation. Historically, Music Information Retrieval researchers have focused primarily on score-informed source separation, but more recent approaches explore lyrics-informed source separation. However, because of the lack of multitrack datasets with time-aligned lyrics, models use weak conditioning with non-aligned lyrics. In this paper, we present a multimodal multitrack dataset with lyrics aligned in time at the word level with phonetic information as well as explore strong conditioning using the aligned phonemes. Our model follows a U-Net architecture and takes as input both the magnitude spectrogram of a musical mixture and a matrix with aligned phonetic information. The phoneme matrix is embedded to obtain the parameters that control Feature-wise Linear Modulation (FiLM) layers. These layers condition the U-Net feature maps to adapt the separation process to the presence of different phonemes via affine transformations. We show that phoneme conditioning can be successfully applied to improve singing voice source separation.

1. INTRODUCTION

Music source separation aims to isolate the different instruments that appear in an audio mixture (a mixed music track), reversing the mixing process. Informed-source separation uses prior information about the target source to improve separation. Researchers have shown that deep neural architectures can be effectively adapted to this paradigm [1, 2]. Music source separation is a particularly challenging task. Instruments are usually correlated in time and frequency with many different harmonic instruments overlapping at several dynamics variations. Without additional knowledge about the sources the separation is often infeasible. To address this issue, Music Informa-

tion Retrieval (MIR) researchers have integrated into the source separation process prior knowledge about the different instruments presented in a mixture, or musical scores that indicate where sounds appear. This prior knowledge improves the performances [2–4]. Recently, conditioning learning has shown that neural networks architectures can be effectively controlled for performing different music source isolation tasks [5–10]

Various multimodal context information can be used. Although MIR researchers have historically focused on score-informed source separation to guide the separation process, lyrics-informed source separation has become an increasingly popular research area [10, 11]. Singing voice is one of the most important elements in a musical piece [12]. Singing voice tasks (e.g. lyric or note transcription) are particularly challenging given its variety of timbre and expressive versatility. Fortunately, recent data-driven machine learning techniques have boosted the quality and inspired many recent discoveries [13, 14]. Singing voice works as a musical instrument and at the same time conveys a semantic meaning through the use of language [14]. The relationship between sound and meaning is defined by a finite phonetic and semantic representations [15, 16]. Singing in popular music usually has a specific sound based on phonemes, which distinguishes it from the other musical instruments. This motivates researchers to use prior knowledge such as a text transcript of the utterance or linguistic features to improve the singing voice source sparation [10, 11]. However, the lack of multitrack datasets with time-aligned lyrics has limited them to develop their ideas and only weak conditioning scenarios have been studied, i.e. using the context information without explicitly informing where it occurs in the signal. Time-aligned lyrics provide abstract and high-level information about the phonetic characteristics of the singing signal. This prior knowledge can facilitate the separation and be beneficial to the final isolation.

Looking for combining the power of data-driven models with the adaptability of informed approaches, we propose a multitrack dataset with time-aligned lyrics. We explore then how we can use strong conditioning where the content information about the lyrics is available frame-wise to improve vocal sources separation. We investigate strong and weak conditioning using the aligned phonemes via Feature-wise Linear Modulation (FiLM) layer [17] in U-Net based architecture [18]. We show that phoneme conditioning can be successfully applied to improve standard

singing voice source separation and that simplest strong conditioning outperforms any other scenario.

2. RELATED WORK

Informed source separation use context information about the sources to improve the separation quality, introducing in models additional flexibility to adapt to observed signals. Researchers have explored different approaches for integrating different prior knowledge in the separation [19]. Most of the recent data-driven music source separation methods use weak conditioning with prior knowledge about the different instruments presented in a mixture [3, 5, 7–9]. Strong conditioning has been primarily used in score-informed source separation. In this section, we review works related to this topic as well as novel approaches that explore lyrics-informed source separation.

2.1 Score-informed music source separation

Scores provide prior knowledge for source separation in various ways. For each instrument (source), it defines which notes are played at which time, which can be linked to audio frames. This information can be used to guide the estimation of the harmonics of the sound source at each frame [2,4]. Pioneer approaches rely on non-negative matrix factorization (NMF) [20–23]. These methods assume that the audio is synchronized with the score and use different alignment techniques to achieve this. Nevertheless, alignment methods introduce errors. Local misalignments influence the quality of the separation [21, 24]. This is compensated by allowing a tolerance window around note onsets and offsets [20, 23] or with context-specific methods to refine the alignment [25]. Current approaches use deep neural network architectures and filtering spectrograms by the scores and generating masks for each source [2]. The score-filtered spectrum is used as input to an encoder-decoder convolutional neural network (CNN) architecture similar to [26]. [27] propose an unsupervised method where scores guide the representation learning to induce structure in the separation. They add class activity penalties and structured dropout extensions to the encoder-decoder architecture. Class activity penalties capture the uncertainty about the target label value and structured dropout uses labels to enforce a specific structure, canceling activity related to unwanted note.

2.2 Text-informed music source separation

Due to the importance of singing voice in a musical piece [12], it is one of the most useful source to separate in a music track. Researchers have integrated the vocal activity information to constrain a robust principal component analysis (RPCA) method, applying a vocal/non-vocal mask or ideal time-frequency binary mask [28]. [10] propose a bidirectional recurrent neural networks (BRNN) method that includes context information extracted from the text via attention mechanism. The method takes as input a whole audio track and its associated text information

and learn alignment between mixture and context information that enhance the separation. Recently, [11] extract a representation of the linguistic content related to cognitively relevant features such as phonemes (but they do not explicitly predict the phonemes) in the mixture. The linguistic content guide the synthetization of the vocals.

3. FORMALIZATION

We use the multimodal information as context to guide and improve the separation. We formalize our problem satisfying certain properties summarized as [29]:

How is the multimodal model constructed? We divide the model into two distinct parts [30]: a *generic* network that carries on the main computation and a *control mechanism* that conditions the computation regarding context information and adds additional flexibility. The *conditioning* itself is performed using FiLM layers [17]. FiLM can effectively modulate a generic source separation model by some external information, controlling a single model to perform different instrument source separations [3, 5]. With this strategy, we can explore the *control* and *conditioning* parts regardless of the *generic* network used.

Where is the context information used? at which place in the *generic* network we insert the context information, and defining how it affects the computation, i.e. weak (or strong) conditioning without (or with) explicitly informing where it occurs in the signal.

What context information? We explore here prior information about the phonetic evolution of the singing voice, aligned in time with the audio. To this end, we introduce a novel multitrack dataset with lyrics aligned in time.
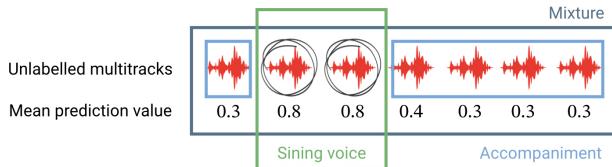
4. DATASET

The DALI (Dataset of Aligned Lyric Information) [31] dataset is a collection of songs described as a sequence of time-aligned lyrics. Time-aligned lyrics are described at four levels of granularity: **notes**, **words**, **lines** and **paragraphs**:

$$A_g = (a_{k,g})_{k=1}^{K_g} \text{ where } a_{k,g} = (t_k^0, t_k^1, f_k, l_k, i_k)_g \quad (1)$$

where g is the granularity level and K_g the number of elements of the aligned sequence, t_k^0 and t_k^1 being a text segment's start and end times (in seconds) with $t_k^0 < t_k^1$, f_k a tuple (f_{min}, f_{max}) with the frequency range (in Hz) covered by all the notes in the segment (at the note level $f_{min} = f_{max}$, a vocal note), l_k the actual lyric's information and $i_k = j$ the index that links an annotation $a_{k,g}$ with its corresponding upper granularity level annotation $a_{j,g+1}$. The text segment's events for a song are ordered and non-overlapping - that is, $t_k^1 \leq t_{k+1}^0 \forall k$.

There is a subset of DALI of 513 multitracks with the *mixture* and its separation in two sources, *vocals* and *accompaniment*. This subset comes from WASABI

Figure 1. Method used for creating the *vocals*, *accompaniment* and *mixture* version.

dataset [32]. The multitracks are distributed in 247 different artists and 32 different genres. The dataset contains 35.4 hours with music and 14.1 hours with vocals, with a mean average duration per song of 220.83s and 98.97s with vocals. All the songs are in English.

The original multitracks have the *mixture* decomposed in a set of unlabeled sources in the form track_1, track_2, ..., track_n. Depending of the songs, the files can be RAW (where each source is an instrument track e.g. a drum snare) or STEMS (where all the RAW files for an instrument are merged into a single file). In the following, we explain how the *vocals* and *accompaniment* tracks are automatically created from these unlabelled sources. The process is summarized in Figure 1.

For each track τ of a multitrack song, we compute a singing voice probability vector overtime, using a pretrained Singing Voice Detection (SVD) model [33]. We obtain then a global mean prediction value per tracks ϵ_τ. Assuming that there is at least one track with vocals, we create the *vocals* source by merging all the tracks with $\epsilon_\tau >= \max_\tau(\epsilon_\tau) \cdot \nu$ where ν is a tolerance value set to 0.98. All the remaining tracks are fused to define the *accompaniment*. We manually checked the resulting sources. The dataset is available at `https://zenodo.org/record/3970189`.
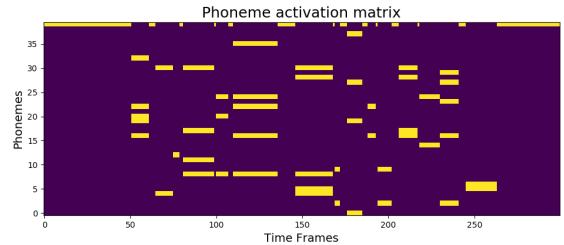
The second version of DALI adds the phonetic information computed for the word level [33]. This level has the words of the lyrics transcribed into a vocabulary of 39 different phoneme symbols as defined in the Carnegie Mellon Pronouncing Dictionary (CMUdict)[1]. After selecting the desired time resolution, we can derive a time frame based phoneme context activation matrix Z, which is a binary matrix that indicates the phoneme activation over time. We add an extra row with the 'non-phoneme' activation with 1 at time frames with no phoneme activation and 0 otherwise. Figure 2 illustrates the final activation matrix.

Although we work only with phonemes per word information, we can derive similar activation matrices for other context information such as notes or characters.

5. METHODOLOGY

Our method adapts the C-U-Net architecture [5] to the singing voice separation task, exploring how to use the prior knowledge defined by the phonemes to improve the vocals separation.

Input representations. Let $X \in \mathbb{R}^{T \times M}$ be the magnitude of the Short-Time Fourier Transform (STFT) with

---

Figure 2. Binary phoneme activation matrix. Note how words are represented as a bag of simultaneous phonemes.

$M = 512$ frequency bands and T time frames. We compute the STFT on an audio signal down-sampled at 8192 Hz using a window size of 1024 samples and a hop size of 768 samples. Let $Z \in \mathbb{R}^{T \times P}$ be the aligned phoneme activation matrix with $P = 40$ phoneme types and T the same time frames as in X. Our model takes as inputs two submatrix $x \in \mathbb{R}^{N \times M}$ and $z \in \mathbb{R}^{N \times P}$ of $N = 128$ frames (11 seconds) derived from X and Z.

Model. The C-U-Net model has two components (see [5] for a general overview of the architecture): a **conditioned network** that processes x and a **control mechanism** that conditions the computation with respect to z. We denote by $x_d \in \mathbb{R}^{W \times H \times C}$ the intermediate features of the **conditioned network**, at a particular depth d in the architecture. W and H represent the 'time' and 'frequency' dimension and C the number of feature channels (or feature maps). A *FiLM* layer conditions the network computation by applying an affine transformation to x_d:

$$FiLM(x_d) = \gamma_d(z) \odot x_d + \beta_d(z) \qquad (2)$$

where $\odot$ denotes the element-wise multiplication and $\gamma_d(z)$ and $\beta_d(z)$ are learnable parameters with respect to the input context z. A *FiLM* layer can be inserted at any depth of the original model and its output has the same dimension as the x_d input, i.e. $\in \mathbb{R}^{W \times H \times C}$. To perform Eqn (2), $\gamma_d(z)$ and $\beta_d(z)$ must have the same dimensionaly as x_d, i.e. $\in \mathbb{R}^{W \times H \times C}$. However, we can define them omitting some dimensions. This results in a non-matching dimensionality with x_d, solved by broadcasting (repeating) the existing information to the missing dimensions.

As in [5, 18, 34], we use the U-Net [18] as **conditioned network**, which has an encoder-decoder mirror architecture based on CNN blocks with skip connections between layers at the same hierarchical level in the encoder and decoder. Each convolutional block in the encoder halves the size of the input and doubles the number of channels. The decoder is made of a stack of transposed convolutional operation, its output has the same size as the input of the encoder. Following the original C-U-Net architecture, we insert the *FiLM* layers at each encoding block after the batch normalization and before the Leaky ReLU [5].

We explore now the different **control mechanism** we use for conditioning the U-Net.

5.1 Control mechanism for weak conditioning

Weak conditioning refers to the cases where

---

[1] https://github.com/cmusphinx/cmudict

Model	U-Net	W_{si}	W_{co}	S_a	S_{a*}	S_c	S_{c*}	S_f	S_{f*}	S_s	S_{s*}
θ	$9.83 \cdot 10^6$	$+14,060$	$+2.35 \cdot 10^6$	$+1.97 \cdot 10^6$	$+327,680$	$+80,640$	$+40,960$	$+40,320$	$+640$	$+480$	$+80$

Table 1. Number of parameters (θ) for the different configurations. We indicate increment to the U-Net architecture.

- $\gamma_d(z)$ and $\beta_d(z) \in \mathbb{R}^1$: they are scalar parameters applied independently of the times W, the frequencies H and the channel C dimensions. They depend only on the depth d of the layer within the network [5].

- $\gamma_d(z)$ and $\beta_d(z) \in \mathbb{R}^C$: this is the original configuration proposed by [17] with different parameters for each channel $c \in 1, ..., C$.

We call them *FiLM simple* (W_{si}) and *FiLM complex* (W_{co}) respectively. Note how they apply the same transformation without explicitly informing where it occurs in the signal (same value over the dimension W and H).

Starting from the context matrix $z \in \mathbb{R}^{N \times P}$, we define the **control mechanism** by first apply the autopool layer proposed by [35] [2] to reduce the input matrix to a timeless vector. We then fed this vector into a dense layer and two dense blocks each composed by a dense layer, 50% dropout and batch normalization. For *FiLM simple*, the number of units of the dense layers are 32, 64 and 128. For *FiLM simple*, they are 64, 256 and 1024. All neurons have ReLU activations. The output of the last block is then used to feed two parallel and independent dense layer with linear activation which outputs all the needed $\gamma_d(z)$ and $\beta_d(z)$. While for the *FiLM simple* configuration we only need 12 γ_d and β_d (one γ_d and β_d for each of the 6 different encoding blocks) for the *FiLM complex* we need 2016 (the encoding blocks feature channel dimensions are 16, 32, 64, 128, 256 and 512, which adds up to 1008).

### 5.2 Control mechanism for strong conditioning
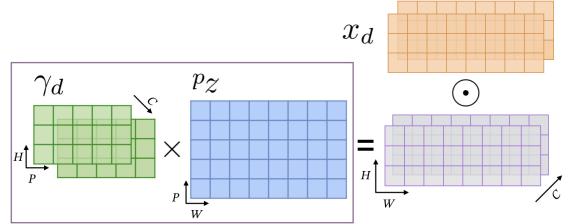
In this section, we extend the original *FiLM* layer mechanism to adapt it to the strong conditioning scenario.

The context information represented in the input matrix z describes the presence of the phonemes $p \in \{1, \ldots, P\}$ over time $n \in \{1, \ldots N\}$. As in the popular Non-Negative Matrix factorization [36] (but without the non-negativity constraint), our idea is to represent this information as the product of tensors: an activation and two basis tensors.

The **activation tensor** z_d indicates which phoneme occurs at which time: $z_d \in \mathbb{R}^{W \times P}$ where W is the dimension which represents the time at the current layer d (we therefore need to map the time range of z to the one of the layer d) and P the number of phonemes.

The **two basis tensors** γ_d and $\beta_d \in \mathbb{R}^{H \times C \times P}$ where H is the dimension which represents the frequencies at the current layer d, C the number of input channels and P the number of phonemes. In other words, each phoneme p is represented by a matrix in $\mathbb{R}^{H \times C}$ derived from Eqn (1). This matrix represents the specific conditioning to apply

Figure 3. Strong conditioning example with $(\gamma_d \times z_d) \odot x_d$. The phoneme activation z_d defines how the basis tensors (γ_d) are employed for performing the conditioning on x_d.

to x_d if the phoneme exists (see Figure 3). These matrices are learnable parameters (neurons with linear activations) but they do not depend on any particular input information (at a depth d they do not depend on x nor z), they are rather "activated" by z_d at specific times. As for the 'weak'conditionning, we can define different versions of the tensors

- the **all-version** (S_a) described so far with three dimensions: $\gamma_d, \beta_d \in \mathbb{R}^{H \times C \times P}$

- the **channel-version** (S_c): each phoneme is represented by a vector over input channels (therefore constant over frequencies): $\gamma_d, \beta_d \in \mathbb{R}^{C \times P}$

- the **frequency-version** (S_f): each phoneme is represented by a vector over input frequencies (therefore constant over channels): $\gamma_d, \beta_d \in \mathbb{R}^{H \times P}$

- the **scalar-version** (S_s): each phoneme is represented as a scalar (therefore constant over frequencies and channels): $\gamma_d, \beta_d \in \mathbb{R}^P$

The global conditioning mechanism can then be written as

$$FiLM(x_d, z_d) = (\gamma_d \times z_d) \odot x_d + (\beta_d \times z_d) \quad (3)$$

where $\odot$ is the element-wise multiplication and $\times$ the matrix multiplication. We broadcast γ_d and β_d for missing dimensions and transpose them properly to perform the matrix multiplication. We test two different configurations: inserting FiLM at each encoder block as suggested in [5] and inserting FiLM only at the last encoder block as proposed at [3]. We call the former 'complete' and the latter 'bottleneck' (denoted with $*$ after the model acronym). We resume the different configurations at Table 1.

6. EXPERIMENTS

DATA. We split DALI into three sets according to the normalized agreement score η presented in [31] (see Table 2). This score provides a global indication of the global alignment correlation between the annotations and the vocal activity.

---

[2] The auto-pool layer is a tuned soft-max pooling that automatically adapts the pooling behavior to interpolate between mean and max-pooling for each dimension

	Train	Val	Test
Threshold	$.88 > \eta >= .7$	$.89 > \eta >= .88$	$.89 > \eta$
Songs	357	30	101

Table 2. DALI split according to agreement score η.

Training	Test	Aug	SDR	SIR	SAR
Musdb18	Musdb18	False	4.27	13.17	5.17
(90)	(50)	True	4.46	12.62	5.29
	Musdb18	False	4.60	14.03	5.39
DALI	(50)	True	4.96	13.50	5.92
(357)	DALI	False	3.98	12.05	4.91
	(101)	True	4.05	11.40	5.32

Table 3. Data augmentation experiment.

Model	SDR	SIR	SAR	PES	EPS
U-Net	4.05	11.40	5.32	-42.44	-64.84
W_{si}	**4.24**	**11.78**	5.38	**-49.44**	-65.47
W_{co}	**4.24**	**12.72**	5.15	**-59.53**	-63.46
S_a	4.04	**12.14**	5.13	**-59.68**	-61.73
S_{a*}	**4.27**	**12.42**	5.26	**-54.16**	-64.56
S_c	**4.36**	**12.47**	5.34	**-57.11**	-65.48
S_{c*}	**4.32**	**12.86**	5.15	**-54.27**	**-66.35**
S_f	4.10	11.40	5.24	47.75	-62.76
S_{f*}	**4.21**	**13.13**	5.05	**-48.75**	**-72.40**
S_s	**4.45**	**11.87**	**5.52**	**-51.76**	-63.44
S_{s*}	**4.26**	**12.80**	5.25	**-57.37**	-65.62

Table 4. Median performance in dB of the different models on the DALI test set. In bold are the results that significantly improve over the U-Net (p < 0.001) and inside the circles the best results for each metric.

DETAILS. We train the model using batches of 128 spectrograms randomly drawn from the training set with 1024 batches per epoch. The loss function is the mean absolute error between the predicted vocals (masked input mixture) and the original vocals. We use a learning rate of 0.001 and the reduction on plateau and early stopping callbacks evaluated on the validation set, using patience of 15 or 30 respectively and a min delta variation for early stopping to $1e-5$. Our output is a Time/Frequency mask to be applied to the magnitude of the input STFT mixture. We use the phase of the input STFT mixture to reconstruct the waveform with the inverse STFT algorithm.

For the strong conditioning, we apply a `softmax` on the input phoneme matrix z over the phoneme dimension P to constrain the outputs to sum to 1, meaning it lies on a simplex, which helps in the optimization.

6.1 Evaluation metrics

We evaluate the performances of the separation using the mir evaltoolbox [37]. We compute three metrics: Source-to-Interference Ratios (SIR), Source-to-Artifact Ratios (SAR), and Source-to-Distortion Ratios (SDR) [38]. In practice, SIR measures the interference from other sources, SAR the algorithmic artifacts introduce in the process and SDR resumes the overall performance. We obtain them globally for the whole track. However, these metrics are ill-defined for silent sources and targets. Hence, we compute also the Predicted Energy at Silence (PES) and Energy at Predicted Silence (EPS) scores [10]. PES is the mean of the energy in the predictions at those frames with silent target and EPS is the opposite, the mean of the target energy of all frames with silent prediction and non-silent target. For numerical stability, in our implementation, we add a small constant $\epsilon = 10^{-9}$ which results in a lower boundary of the metrics to be -80 dB [3]. We consider as silent segments those that have a total sum of less than -25 dB of the maximum absolute in the audio. We report the **median** values of these metrics over the all tracks in the DALI test set. For SIR, SAR, and SDR larger values indicate better performance, for PES and EPS smaller values, mean better performance.

6.2 Data augmentation

Similarly as proposed in [39], we randomly created 'fake' input mixtures every 4 real mixtures. In non-augmented training, we employ the mixture as input and the vocals as a target. However, this does not make use of the accompaniment (which is only employed during evaluation). We can integrate it creating 'fake' inputs by automatically mixing (mixing meaning simply adding) the target vocals to a random sample accompaniment from our training set.
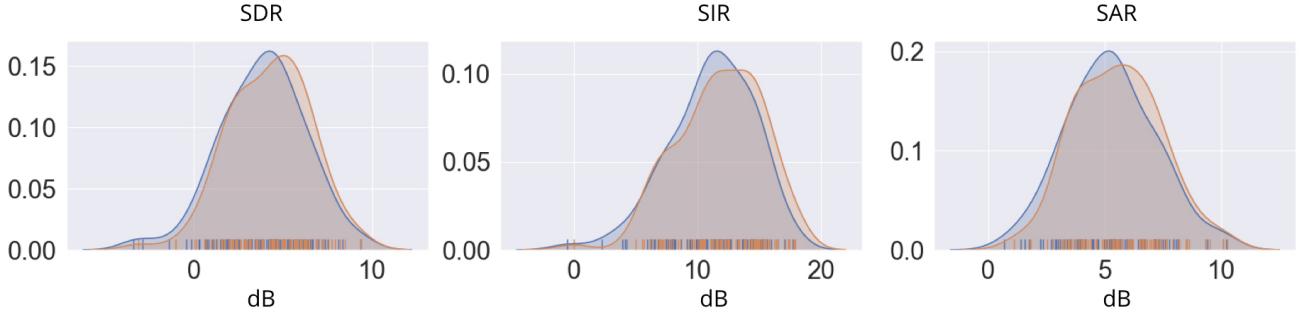
We test the data augmentation process using the standard U-Net architecture to see whether it improves the performance (see Table 3). We train two models on DALI and Musdb18 dataset [40] [3]. This data augmentation enables models to achieve better SDR and SAR but lower SIR. Our best results (4.96 db SDR) are not state-of-the-art where the best-performing models on Musdb18 achieve (approximately 6.60 db SDR) [41].

This technique does not reflect a large improvement when the model trained on DALI is tested on DALI. However, when this model is tested on Musdb18, it shows a better generalization (we have not seen any song of Musidb18 during training) than the model without data augmentation (we gain 0.36 dB). One possible explanation for not having a large improvement on DALI testset is the larger size of the test set. It also can be due to the fact that vocal targets in DALI still contain leaks such as low volume music accompaniment that come from the singer headphones. We adopt this technique for training all the following models.

Finally, we confirmed a common belief that training with a large dataset and clean separated sources improves the separation over a small dataset [42]. Both models trained on DALI (with and without augmentation) improve the results obtained with the models trained on Musdb18.

Since we cannot test the conditioning versions on Musdb (no aligned lyrics), the results on the DALI test (4.05 dB SDR) serves as a baseline to measure the contribution of the conditioning techniques (our main interest).

---

[3] We use 10 songs of the training set for the early stopping and reduction on plateau callbacks

Figure 4. Distribution of scores for the the standar U-Net (Blue) and S_s (Orange).

7. RESULTS

We report the median source separation metrics (SDR, SAR, SIR, PES, ESP) in Table 4. To measure the significance of the improvement differences, we performed a paired t-test between each conditioning model and the standard U-Net architecture, the baseline. This test measures (p-value) if the differences could have happened by chance. A low p-value indicates that data did not occur by chance. As expected, there is a marginal (but statistical significance) improvement over most of the proposed methods, with a generalized $p < 0.001$ for the SDR, SIR, and PES, except for the versions where the basis tensors have a 'frequency' H dimension. This is an expected result since when singing, the same phoneme can be sung at different frequencies (appearing at many frequency positions in the feature maps). Hence, these versions have difficulties to find generic basis tensors. This also explains why the 'bottleneck' versions (for both S_{f*} and S_{a*}) outperforms the 'complete' while this is not the case for the other versions. Most versions also improve the performance on silent vocal frames with a much lower PES. However, there is no difference in predicting silence at the right time (same EPS). The only metric that does not consistently improve is SAR, which measures the algorithmic artifacts introduced in the process. Our conditioning mechanisms can not reduce the artifacts that seem more dependent on the quality of the training examples (it is the metric with higher improvement in the data augmentation experiment Table 3). Figure 4 shows a comparison with the distribution of SDR, SIR, and SAR for the best model S_s and the U-Net. We can see how the distributions move toward higher values.

One relevant remark is the fact that we can effectively control the network with just a few parameters. S_s just adds 480 (or just 80 for S_{s*}) new learnable parameters and have significantly better performance than S_a that adds $1.97 \cdot 10^6$. We believe that the more complex control mechanisms tend to find complex basis tensors that do not generalize well. In our case, it is more effective to perform a simple global transformation. In the case of weak conditioning, both models behave similarly although W_{si} has $1.955 \cdot 10^6$ fewer parameters than W_{co}. This seems to indicate that controlling channels is not particularly relevant.

Regarding the different types of conditioning, when repeating the paired t-test between weak and strong models only S_s outperforms the weak systems. We believe that

strong conditioning can lead to higher improvements but several issues need to be addressed. First, there are misalignments in the annotations that force the system to perform unnecessary operations which damages the computation. This is one of the possible explanations of why models with fewer parameters perform better. They are forced to find more generic conditions. The weak conditioning models are robust to these problems since they process z and compute an optimal modification for a whole input patch (11s). We also need to "disambiguate" the phonemes inside words since they occur as a bag of phonemes at the same time (no individual onsets per phonemes inside one word, see Figure 2). This prevents strong conditioning models to properly learn the phonemes in isolation, instead, they consider them jointly with the other phonemes.

8. CONCLUSIONS

The goal of this paper is twofold. First, to introduce a new multimodal multitrack dataset with lyrics aligned in time. Second, to improve singing voice separation using the prior knowledge defined by the phonetic characteristics. We use the phoneme activation as side information and show that it helps in the separation.

In future works, we intend to use other prior aligned knowledge such as vocal notes or characters also defined in DALI . Regarding the conditioning approach and since it is transparent to the conditioned network, we are determined to explore recent state-of-the-art source separation methods such as Conv-Tasnet [43]. The current formalization of the two basis tensors γ_d and β_d does not depend on any external factor. A way to exploit a more complex control mechanisms is to make these basis tensors dependent on the input mixture x which may add additional flexibility. Finally, we plan to jointly learn how to infer the alignment and perform the separation [44, 45].

The general idea of lyrics-informed source separation leaves room for many possible extensions. The present formalization relies on time-aligned lyrics which is not the real-world scenario. Features similar to the phoneme activation [46, 47] can replace them or be used to align the lyrics as a pre-processing step. This two options adapts the current system to the real-world scenario. These features can also help in properly placing and disambiguating the phonemes of a word to improve the current annotations.

9. ACKNOWLEDGEMENT.

10. REFERENCES

[1] K. Kinoshita, M. Delcroix, A. Ogawa, and T. Nakatani, "Text-informed speech enhancement with deep neural networks," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[2] M. Miron, J. Janer Mestres, and E. Gómez Gutiérrez, "Monaural score-informed source separation for classical music using convolutional neural networks," in *Hu X, Cunningham SJ, Turnbull D, Duan Z. ISMIR 2017. 18th International Society for Music Information Retrieval Conference; 2017 Oct 23-27; Suzhou, China.[Canada]: ISMIR; 2017. p. 55-62.* International Society for Music Information Retrieval (ISMIR), 2017.

[3] O. Slizovskaia, G. Haro, and E. Gómez, "Conditioned source separation for music instrument performances," *arXiv preprint arXiv:2004.03873*, 2020.

[4] S. Ewert, B. Pardo, M. Müller, and M. D. Plumbley, "Score-informed source separation for musical audio recordings: An overview," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 116–124, 2014.

[5] G. Meseguer-Brocal and G. Peeters, "Conditioned-u-net: Introducing a control mechanism in the u-net for multiple source separations," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Delft, Netherlands, 2019.

[6] E. Tzinis, S. Wisdom, J. R. Hershey, A. Jansen, and D. P. Ellis, "Improving universal sound separation using sound classification," *arXiv preprint arXiv:1911.07951*, 2019.

[7] O. Slizovskaia, L. Kim, G. Haro, and E. Gomez, "End-to-end sound source separation conditioned on instrument labels," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 306–310.

[8] P. Seetharaman, G. Wichern, S. Venkataramani, and J. Le Roux, "Class-conditional embeddings for music source separation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 301–305.

[9] D. Samuel, A. Ganeshan, and J. Naradowsky, "Meta-learning extractors for music source separation," *arXiv preprint arXiv:2002.07016*, 2020.

[10] K. Schulze-Forster, C. Doire, G. Richard, and R. Badeau, "Weakly informed audio source separation," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 273–277.

[11] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "Content based singing voice extraction from a musical mixture," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 781–785.

[12] A. Demetriou, A. Jansson, A. Kumar, and R. M. Bittner, "Vocals in music matter: the relevance of vocals in the minds of listeners." in *In Proceedings of 19th International Society for Music Information Retrieval Conference*, Paris, France, September 2018.

[13] E. Gómez, M. Blaauw, J. Bonada, P. Chandna, and H. Cuesta, "Deep learning for singing processing: Achievements, challenges and impact on singers and listeners," *arXiv preprint arXiv:1807.03046*, 2018.

[14] E. J. Humphrey, S. Reddy, P. Seetharaman, A. Kumar, R. M. Bittner, A. Demetriou, S. Gulati, A. Jansson, T. Jehan, B. Lehner *et al.*, "An introduction to signal processing for singing-voice analysis: High notes in the effort to automate the understanding of vocals in music," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 82–94, 2018.

[15] J. Goldsmith, "Autosegmental phonology," Ph.D. dissertation, MIT Press London, 1976.

[16] D. R. Ladd, *Intonational phonology*. Cambridge University Press, 2008.

[17] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proc. of AAAI (Conference on Artificial Intelligence)*, New Orleans, LA, USA, 2018.

[18] A. Jansson, E. J. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Suzhou, China, 2017.

[19] A. Liutkus, J.-L. Durrieu, L. Daudet, and G. Richard, "An overview of informed audio source separation," in *2013 14th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*. IEEE, 2013, pp. 1–4.

[20] S. Ewert and M. Müller, "Using score-informed constraints for nmf-based source separation," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 129–132.

[21] Z. Duan and B. Pardo, "Soundprism: An online system for score-informed source separation of music audio," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1205–1215, 2011.

[22] F. J. Rodriguez-Serrano, Z. Duan, P. Vera-Candeas, B. Pardo, and J. J. Carabias-Orti, "Online score-informed source separation with adaptive instrument models," *Journal of New Music Research*, vol. 44, no. 2, pp. 83–96, 2015.

[23] J. Fritsch and M. D. Plumbley, "Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 888–891.

[24] M. Miron, J. J. Carabias Orti, and J. Janer Mestres, "Improving score-informed source separation for classical music through note refinement," in *Müller M, Wiering F, editors. Proceedings of the 16th International Society for Music Information Retrieval (ISMIR) Conference; 2015 Oct 26-30; Málaga, Spain. Canada: International Society for Music Information Retrieval; 2015.* International Society for Music Information Retrieval (ISMIR), 2015.

[25] M. Miron, J. J. Carabias-Orti, J. J. Bosch, E. Gómez, and J. Janer, "Score-informed source separation for multichannel orchestral recordings," *Journal of Electrical and Computer Engineering*, vol. 2016, 2016.

[26] P. Chandna, M. Miron, J. Janer, and E. Gómez, "Monoaural audio source separation using deep convolutional neural networks," in *Proc. of LVA/ICA (International Conference on Latent Variable Analysis and Signal Separation)*, Grenoble, France, 2017.

[27] S. Ewert and M. B. Sandler, "Structured dropout for weak label and multi-instance learning and its application to score-informed source separation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2277–2281.

[28] T.-S. Chan, T.-C. Yeh, Z.-C. Fan, H.-W. Chen, L. Su, Y.-H. Yang, and J.-S. R. Jang, "Vocal activity informed singing voice separation with the ikala dataset," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 718–722, 2015.

[29] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug 2013.

[30] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. d. Vries, A. Courville, and Y. Bengio, "Feature-wise transformations," *Distill*, 2018, https://distill.pub/2018/feature-wise-transformations.

[31] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Dali: a large dataset of synchronised audio, lyrics and notes, automatically created using teacher-student machine learning paradigm." in *In Proceedings of 19th International Society for Music Information Retrieval Conference*, Paris, France, September 2018.

[32] G. Meseguer-Brocal, G. Peeters, G. Pellerin, M. Buffa, E. Cabrio, C. F. Zucker, A. Giboin, I. Mirbel, R. Hennequin, M. Moussallam *et al.*, "Wasabi: A two million song database project with audio and cultural metadata plus webaudio enhanced client applications," in *Web Audio Conference 2017–Collaborative Audio# WAC2017*, 2017.

[33] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Creating dali, a large dataset of synchronized audio, lyrics, and notes," *Transactions of the International Society for Music Information Retrieval*, 2020.

[34] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Paris, France, 2018.

[35] B. McFee, J. Salamon, and J. P. Bello, "Adaptive pooling operators for weakly labeled sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2180–2193, 2018.

[36] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[37] C. Raffel, B. Mcfee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mir_eval: a transparent implementation of common mir metrics," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Porto, Portugal, 2014.

[38] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE/ACM TASLP (Transactions on Audio Speech and Language Processing)*, vol. 14, no. 4, 2006.

[39] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 261–265.

[40] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," 2017, https://zenodo.org/record/1117372.

[41] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2018, pp. 293–305.

[42] L. Prétet, R. Hennequin, J. Royo-Letelier, and A. Vaglio, "Singing voice separation: A study on training data," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 506–510.

[43] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[44] K. Schulze-Forster, C. S. Doire, G. Richard, and R. Badeau, "Joint phoneme alignment and text-informed speech separation on highly corrupted speech," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7274–7278.

[45] N. Takahashi, M. K. Singh, S. Basak, P. Sudarsanam, S. Ganapathy, and Y. Mitsufuji, "Improving voice separation by incorporating end-to-end speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 41–45.

[46] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. d'Alché Buc, "Audio-based detection of explicit content in music," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 526–530.

[47] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 181–185.

BEBOPNET: DEEP NEURAL MODELS FOR PERSONALIZED JAZZ IMPROVISATIONS

Shunit Haviv Hakimi*   **Nadav Bhonker***   **Ran El-Yaniv**

Computer Science Department

Technion – Israel Institute of Technology

`shunithaviv@gmail.com, nadavbh@gmail.com, rani@cs.technion.ac.il`

ABSTRACT

A major bottleneck in the evaluation of music generation is that music appreciation is a highly subjective matter. When considering an average appreciation as an evaluation metric, user studies can be helpful. The challenge of generating personalized content, however, has been examined only rarely in the literature. In this paper, we address generation of personalized music and propose a novel pipeline for music generation that learns and optimizes user-specific musical taste. We focus on the task of symbol-based, monophonic, harmony-constrained jazz improvisations. Our personalization pipeline begins with BebopNet, a music language model trained on a corpus of jazz improvisations by Bebop giants. BebopNet is able to generate improvisations based on any given chord progression[1] . We then assemble a personalized dataset, labeled by a specific user, and train a user-specific metric that reflects this user's unique musical taste. Finally, we employ a personalized variant of beam-search with BebopNet to optimize the generated jazz improvisations for that user. We present an extensive empirical study in which we apply this pipeline to extract individual models as implicitly defined by several human listeners. Our approach enables an objective examination of subjective personalized models whose performance is quantifiable. The results indicate that it is possible to model and optimize personal jazz preferences and offer a foundation for future research in personalized generation of art. We also briefly discuss opportunities, challenges, and questions that arise from our work, including issues related to creativity.

1. INTRODUCTION

Since the dawn of computers, researchers and artists have been interested in utilizing them for producing different forms of art, and notably for composing music [1]. The explosive growth of deep learning models over the past several years has expanded the possibilities for musical generation, leading to a line of work that pushed forward the state-of-the-art [2–6]. Another recent trend is the development and offerings of consumer services such as Spotify, Deezer and Pandora, aiming to provide personalized streams of existing music content. Perhaps the crowning achievement of such personalized services would be for the content itself to be generated explicitly to match each individual user's taste. In this work we focus on the task of generating user personalized, monophonic, symbolic jazz improvisations. To the best of our knowledge, this is the first work that aims at generating personalized jazz solos using deep learning techniques.

The common approach for generating music with neural networks is generally the same as for language modeling. Given a context of existing symbols (e.g., characters, words, music notes), the network is trained to predict the next symbol. Thus, once the network learns the distribution of sequences from the training set, it can generate novel sequences by sampling from the network output and feeding the result back into itself. The products of such models are sometimes evaluated through user studies (crowd-sourcing). Such studies assess the quality of generated music by asking users their opinion, and computing the mean opinion score (MOS). While these methods may measure the overall quality of the generated music, they tend to average-out evaluators' personal preferences. Another, more quantitative but rigid approach for evaluation of generated music is to compute a metric based on musical theory principles. While such metrics can, in principle, be defined for classical music, they are less suitable for jazz improvisation, which does not adhere to such strict rules.

To generate personalized jazz improvisations, we propose a framework consisting of the following elements: (a) BebopNet: jazz model learning; (b) user preference elicitation; (c) user preference metric learning; and (d) optimized music generation via planning.

As many jazz teachers would recommend, the key to attaining great improvisation skills is by studying and emulating great musicians. Following this advice, we train BebopNet, a harmony-conditioned jazz model that composes entire solos. We use a training dataset of hundreds of professionally transcribed jazz improvisations performed by saxophone giants such as Charlie Parker, Phil Woods and

---

[1] Supplementary material and numerous MP3 demonstrations of jazz improvisations of jazz standards and pop songs generated by BebopNet are provided in `https://shunithaviv.github.io/bebopnet`.

Figure 1. A short excerpt generated by BebopNet.

Cannonball Adderley (see details in Section 4.1.1). In this dataset, each solo is a monophonic note sequence given in symbolic form (MusicXML) accompanied by a synchronized harmony sequence. After training, BebopNet is capable of generating high fidelity improvisation phrases (this is a subjective impression of the authors). Figure 1 presents a short excerpt generated by BebopNet.

Considering that different people have different musical tastes, our goal in this paper is to go beyond straightforward generation by this model and optimize the generation toward personalized preferences. For this purpose, we determine a user's preference by measuring the level of their satisfaction throughout the solos using a digital variant of continuous response interface (CRDI) [7]. This is accomplished by playing, for the user, computer-generated solos (from the jazz model) and recording their good/bad feedback in real time throughout each solo. Once we have gathered sufficient data about the user's preferences, consisting of two aligned sequences (for the solos and feedback), we train a user preference metric in the form of a recurrent regression model to predict this user's preferences. A key feature of our technique is that the resulting model can be evaluated *objectively* using hold-out user preference sequences (along with their corresponding solos). A big hurdle in accomplishing this step is that the signal elicited from the user is inevitably extremely noisy. To reduce this noise, we apply selective prediction techniques [8, 9] to distill cleaner predictions from the user's preference model. Thus, we allow this model to abstain whenever it is not sufficiently confident. The fact that it is possible to extract a human continuous response preference signal on musical phrases and use it to train (and test) a model with non-trivial predictive capabilities is interesting in itself (and new, to the best of our knowledge).

Equipped with a personalized user preference metric (via the trained model), in the last stage we employ a variant of beam-search [10], to generate optimized jazz solos from BebopNet. For each user, we apply the last three stages of this process where the preference elicitation stage takes several hours of tagging per user. We applied the proposed pipeline on four users, all of whom are amateur jazz musicians. We present numerical analysis of the results showing that a personalized metric can be trained and then used to optimize solo generation.

To summarize, our contributions include: (1) a useful monophonic neural model for general jazz improvisation within any desired harmonic context; (2) a viable methodology for eliciting and learning high resolution human preferences for music; (3) a personalized optimization process of jazz solo generation; and (4) an objective evaluation method for subjective content and plagiarism analysis for the generated improvisations.

2. RELATED WORK

Many different techniques for algorithmic musical composition have been used over the years. For example, some are grammar-based [11], rule-based [1, 12], use Markov chains [13–15], evolutionary methods [16, 17] or neural networks [18–20]. For a comprehensive summary of this broad area, we refer the reader to [21]. Here we confine the discussion to closely related works that mainly concern jazz improvisation using deep learning techniques over symbolic data. In this narrower context, most works follow a generation by prediction paradigm, whereby a model trained to predict the next symbol is used to greedily generate sequences. The first work on blues improvisation [22] straightforwardly applied long short-term memory (LSTM) networks on a small training set. While their results may seem limited at a distance of nearly two decades [2], they were the first to demonstrate long-term structure captured by neural networks.

One approach to improving a naïve greedy generation from a jazz model is by using a mixture of experts. For example, Franklin et al. [23] trained an ensemble of neural networks were trained, one specialized for each melody, and then selected from among them at generation time using reinforcement learning (RL) utilizing a handcrafted reward function. Johnson et al. [24] generated improvisations by training a network consisting of two experts, each focusing on a different note representation. The experts were combined using the technique of product of experts [25] [3]. Other remotely related non-jazz works have attempted to produce context-dependent melodies [2, 3, 5, 26–30].

A common method for collecting continuous measurements from human subjects listening to music is the continuous response digital interface (CRDI), first reported by [7]. CRDI has been successful in measuring a variety of signals from humans such as emotional response [31], tone quality and intonation [32], beauty in a vocal performance [33], preference for music of other cultures [34] and appreciation of the aesthetics of jazz music [35]. Using CRDI, listeners are required to rate different elements of the music by adjusting a dial (which looks similar to a volume control dial present on amplifiers).

3. PROBLEM STATEMENT

We now state the problem in mathematical terms. We denote an input $x_t = (s_t, c_t)$ consisting of a note s_t and its context c_t. Each note $s_t \in \mathcal{S}$, in turn, consists of a pitch and a duration at index t and $\mathcal{S}$ represents a predefined set of pitch-duration combinations (i.e., notes). The context $c_t \in \mathcal{C}$ represents the chord that is played with note s_t, where $\mathcal{C}$ is the set of all possible chords. The context may contain additional information such as the offset of the note within a measure (see details in Section 4). Let $\mathcal{D}$ denote a training dataset consisting of M solos. Each

---

[2] Listen to their generated pieces at www.iro.umontreal.ca/~eckdoug/blues/index.html.

[3] Listen to the generated solos at www.cs.hmc.edu/~keller/jazz/improvisor/iccc2017/

solo is a sequence $X_\tau = x_1 \cdots x_\tau \in (\mathcal{S} \times \mathcal{C})^\tau$ of arbitrary length τ. In our work, these are the aforementioned jazz improvisations.

We define a context-dependent jazz model f_θ (Eq. 1), as the estimator of the probability of a note s_t given the sequence of previous inputs X_{t-1} and the current context c_t, where θ are the parameters of the model. This is similar to a human jazz improviser who is informed of the chord over which his next note will be played.

$$f_\theta(X_{t-1}, c_t) = Pr(s_t | X_{t-1}, c_t) \qquad (1)$$

For any solo X_τ, we also consider an associated sequence of annotation $Y_\tau = y_1 \cdots y_\tau \in \mathcal{Y}^\tau$. An annotation $y_t \in \mathcal{Y}$ represents the quality of the solo up to point t by some metric. In our case, y_t may be a measure of preference as indicated by a user or a score measuring harmonic compliance. Let $\widetilde{\mathcal{D}}$ denote a training dataset consisting of N solos. Each solo X_τ of arbitrary length τ is labeled with a sequence Y_τ. Given $\widetilde{\mathcal{D}}$, we define a metric g_ϕ (Eq. 2) to predict y_τ given a sequence of inputs X_τ. g_ϕ is the user-preference model and ϕ are the learned parameters.

$$\hat{y}_\tau = g_\phi(X_\tau) \qquad (2)$$

We denote by ψ a function that is used to sample notes from f_θ to generate solos. In our case, this will be our beam-search variant. The objective here is to train viable models, f_θ and g_ϕ, and then to use ψ to sample solos from f_θ while maximizing g_ϕ.

4. METHODS

In this section we describe the methods used and implementation details of our personalized generation pipeline.

4.1 BebopNet: Jazz Model Learning

In the first step of our pipeline, we use supervised learning to train BebopNet, a context-dependent jazz model f_θ from a given corpus of transcribed jazz solos.
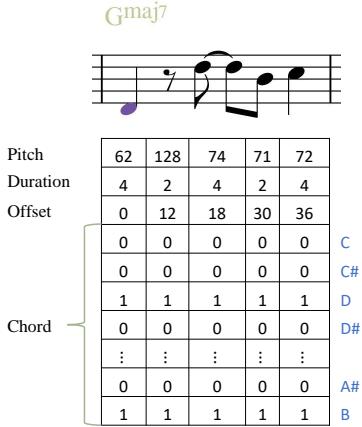
4.1.1 Dataset and music representation

Our corpus $\mathcal{D}$ consists of 284 professionally transcribed solos of (mostly) Bebop saxophone players of the early 20th century. These are Charlie Parker, Sonny Stitt, Cannonball Adderley, Dexter Gordon, Sonny Rollins, Stan Getz, Phil Woods and Gene Ammons. We consider only solos that are in 4/4 metre and include chords in their transcription. The solos are provided in musicXML format. As opposed to MIDI, this format allows the inclusion of chord symbols[4]. We represent notes using a representation method inspired by sheet music (see Figure 2).
Pitch The pitch is encoded as a one-hot vector of size 129. Indices 0—127 match the pitch range of the MIDI standard.[5] Index 128 corresponds to the *rest* symbol.

---

[4] The solos were purchased from SaxSolos.com [36]; we are thus unable to publish them. Nevertheless, in the supplementary material we provide a complete list of solos used for training, which are available from the above vendor.

[5] The notes appearing in the corpus all belong to a much smaller range; however, the MIDI range standard was maintained for simplicity.

Figure 2. An example of a measure in music notation and its vector representation. Integers are converted to one-hot representations.
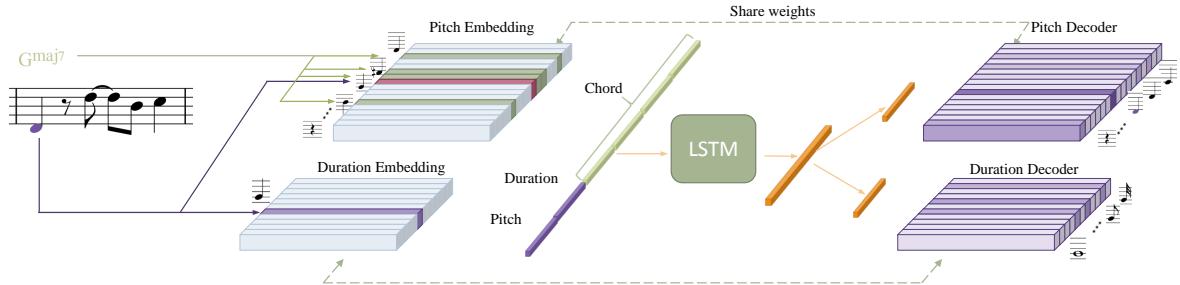
Duration The duration of each note is encoded using a one-hot vector consisting of all the existing durations in the dataset. Durations smaller than $1/24$ are removed.
Offset The offset of the note lies within the measure and is quantized to 48 "ticks" per (four-beat) measure. This corresponds to a duration of 1/12 of a beat. This is similar to the learned positional-encoding used in translation [37].
Chord The chord is represented by a four-hot vector of size 12, representing the 12 possible pitch classes to appear in a chord. As common in jazz music, unless otherwise noted, we assume that chords are played using their 7th form. Thus, the chord pitches are usually the 1st, 3rd, 5th, and 7th degrees of the root of the chord. This chord representation allows the flexibility of representing rare chords such as sixth, diminished and augmented chords.

4.1.2 Network Architecture

BebopNet, as many language models, can be implemented using different architectures such as recurrent neural networks (RNNs), convolutional networks (CNNs) [5, 26, 38] or attention-based models [39]. BebopNet contains a three-layer LSTM network [40]. Recent promising results with attention based models enabled us to improve BebopNet by replacing the LSTM with Transformer-XL [41]. The architecture of the network used to estimate f_θ is illustrated in Figure 3. The network's input x_t includes the note s_t (pitch and duration) and context c_t (offset and chord). The pitch, duration and offset are each represented by learned embedding layers. The chord is encoded by using the embedding of the pitches comprising it. While notes at different octaves have different embeddings, the chord pitch embeddings are always taken from the octave in which most notes in the dataset reside. This embedded vector is passed to the LSTM network. The LSTM output is then passed to two heads. Each head consists of two fully-connected layers with a sigmoid activation in-between. The output of the first layer is the same size as the embedding of the pitch (or duration), and the second output size is the number of possible pitches (or durations).

Figure 3. The BebopNet architecture for the next note prediction. Each note is represented by concatenating the embeddings of the pitch (red bar), the duration (purple bar) and the four pitches comprising the current chord (green bars). The output of the LSTM is passed to two heads (orange bars), one the size of the pitch embedding (top) and the other the size of the duration embedding (bottom).

Following [42, 43], we tie the weights of the final fully-connected layers to those of the embedding. Finally, the outputs of the two heads pass through a softmax layer and are trained to minimize the negative log-likelihood of the corpus. To enrich our dataset while encouraging harmonic context dependence, we augment our dataset by transposing to all 12 keys.

4.2 User Preference Elicitation

Using BebopNet, we created a dataset to be labeled by users, consisting of 124 improvisations. These solos were divided into three groups of roughly the same size: solos from the original corpus, solos generated by BebopNet over jazz standards present in the training set, and generated solos over jazz standards not present in the training set. The length of each solo is two choruses, or twice the length of the melody. For each standard, we generated a backing track in MP3 format that includes a rhythm section and a harmonic instrument to play along the improvisation using Band-in-a-Box [44]. This dataset amounts to approximately five hours of played music.

We created a system inspired by CRDI that is entirely digital, replacing the analog dial with strokes of a keyboard moving a digital dial. A figure of our dial is presented in the supplementary material. While the original CRDI had a range of 255 values, our initial experiments found that quantizing the values to five levels was easier for users. We recorded the location of the dial at every time step and aligned it to the note being played at the same moment.

4.3 User Preference Metric Learning

In the user preference metric learning stage we again use supervised learning to train a metric function g_ϕ. This function should predict user preference scores for any solo, given its harmonic context. During training, for each sequence X_τ we estimate y_τ, corresponding to the label the user provided for the last note in the sequence. We choose the last label of the sequence, rather than the mode or mean, because of delayed feedback. During the user elicitation step, we noticed that when a user decides to change the position of the dial, it is because he has just heard a sequence of notes that he considers to be more (or less)

pleasing than those he heard previously. Thus, the label indicates the preference of the past sequence. The labels are linearly scaled down to the range $[-1, 1]$. Since the data in $\widetilde{\mathcal{D}}$ is small and unbalanced, we use stratified sampling over solos to divide the dataset into training and validation sets. We then use bagging to create an ensemble of five models for the final estimate.

4.3.1 Network Architecture

We estimate the function g_ϕ using transfer learning from BebopNet. The user preference model consists of the same layers as BebopNet without the final fully-connected layers. Next, we apply scaled dot-product attention [45] over τ time steps followed by fully-connected and tanh layers. The transferred layers are initialized using the weights θ of BebopNet. Furthermore, the weights of the embedding layers are frozen during training.

4.3.2 Selective Prediction

To elevate the accuracy of g_ϕ, we utilize selective prediction whereby we ignore predictions whose confidence is too low. We use the prediction magnitude as a proxy for confidence. Given confidence threshold parameters, $\beta_1 < 0, \beta_2 > 0$, we define $g'_{\phi,\beta_1,\beta_2}(X_t^i)$ in Eq. 3.

$$g'_{\phi,\beta_1,\beta_2}(X_t^i) = \begin{cases} 0 & \text{if } \beta_1 < g_\phi(X_t^i) < \beta_2 \\ g_\phi(X_t^i) & \text{else} \end{cases} \quad (3)$$

The parameters β_1 and β_2 change our coverage rate and are determined by minimizing error (risk) on the risk-coverage plot along a predefined coverage contour. More details are given in Section 5.2.

4.4 Optimized Music Generation

To optimize generations from f_θ, we apply a variant of beam-search, ψ, whose objective scores are obtained from non-rejected predictions of g_ϕ. Pseudocode of the ψ procedure is presented in the supplementary material. We denote by $V_b = [X_t^1, X_t^2, ..., X_t^b]$ a running batch (beam) of size (beam-width) b containing the most promising candidate sequences found so far by the algorithm. The sequences are all initialized with the starting input sequence. In our

Name	Adderley	Gordon	Getz	Parker	Rollins	Stitt	Woods	Ammons	**BebopNet (Heard)**	**BebopNet (Unheard)**
Chord	0.50	0.54	0.53	0.52	0.52	0.53	0.50	0.54	**0.53**	**0.52**
Scale	0.78	0.83	0.81	0.80	0.81	0.83	0.78	0.83	**0.82**	**0.81**

Table 1. Harmonic coherence: The average chord and scale matches computed for artists in the dataset and for BebopNet. A higher number indicates a high coherency level. BebopNet is measured separately for harmonic progressions heard and not heard in the training dataset.

case, this is the melody of the jazz standard. At every time step t, we produce a probability distribution of the next note of every sequence in V_b by passing the b sequences through the network $f_\theta(X_t^i, c_{t+1}^i)$. As opposed to typical applications of beam-search, rather than choosing the most probable notes from $Pr(s_{t+1}|X_t^i, c_{t+1}^i)$, we independently and randomly sample them. We then calculate the score of the extended candidates using the preference metric, g_ϕ.

Every δ steps, we perform a beam update process. We choose the highest scoring k sequences calculated by g_ϕ. Then we duplicate these sequences b/k times to maintain a full beam of b sequences. Choosing different values of δ allows us to control a horizon parameter, which facilitates longer term predictions when extending candidate sequences in the beam. The use of larger horizons may lead to sub-optimal optimization but increases variability.

5. EXPERIMENTS
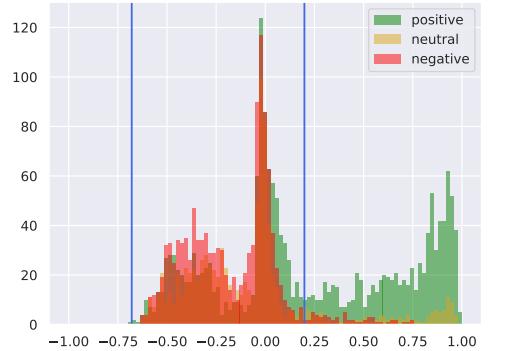
We start the experimental process by training BebopNet as described in Section 4. After training, we use BebopNet to generate multiple solos over different jazz standards[6]. To verify that BebopNet can generalize to harmonic progressions of different musical genres, we also generate improvisations over pop songs (see supplementary material).

This section has two sub-sections. First, we evaluate BebopNet in terms of harmonic coherence (5.1). Next, we present an analysis of our personalization process (5.2). All experiments were performed on desktop computers with a single Titan X GPU. Hyperparameters are provided in the supplementary material.
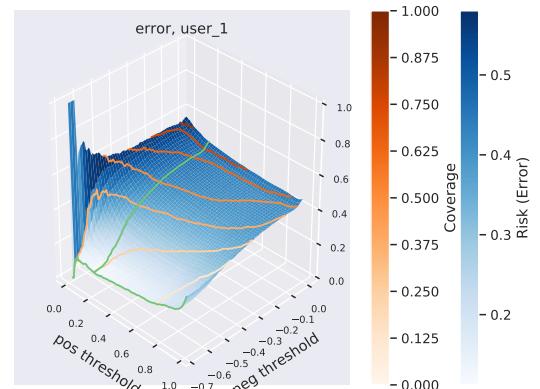
5.1 Harmonic Coherence

We begin by evaluating the extent to which BebopNet was able to capture the context of chords, which we term *harmonic coherence*. We define two harmonic coherence metrics using either scale match or chord match. These metrics are defined as the percent of time within a measure where notes match pitches of the scale or the chord being played, respectively. We rely on a standard definition of matching scales to chords using the chord-scale system [46]. While most notes in a solo should be harmonically coherent, some non-coherent notes are often incorporated. Common examples of their uses are chromatic lines, approach notes and enclosures [47]. Therefore, as we do not expect a perfect harmonic match according to pure music rules, we

take as a baseline the average matching statistics of these quantities for each jazz artist in our dataset. The harmonic coherence statistics of BebopNet are computed over the dataset used for the preference metric learning (generated by BebopNet), which also includes chord progressions not heard during the jazz modeling stage. The baselines and results are reported in Table 1. It is evident that our model exhibits harmonic coherence in the 'ballpark' of the jazz artists even on chord progressions not previously heard.

i Histogram of predictions

ii Risk-coverage plot

Figure 4. 4i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score ($y_\tau > 0$); yellow: neutral ($y_\tau = 0$); red: negative ($y_\tau < 0$). The blue vertical lines indicate thresholds β_1, β_2 used for selective prediction. 4ii Risk-coverage plot for the predictions of the preference model. β_1, β_2 (green lines) are defined to be the thresholds that yield a minimum error on the contour of 25% coverage.

---

[6] To appreciate the diversity of BebopNet, listen to seven solos generated for user-4 for the tune Recorda-Me in the supplementary material.

5.2 Analyzing Personalized Models

We applied the proposed pipeline to generate personalized models for each of the four users, all amateur jazz musicians. All users listened to the same training dataset of solos to create their personal metric (see Section 4). Each user provided continuous feedback for each solo using our CRDI variant. In this section, we describe our evaluation process for user-1. The evaluation results for the rest of the users are presented in the supplementary material.

We analyze the quality of our preference metric function g_ϕ by plotting a histogram of the network's predictions applied on a validation set. Consider Figure 4i. We can crudely divide the histogram into three areas: the right-hand side region corresponds to mostly positive sequences predicted with high accuracy; the center region corresponds to high confusion between positive and negative; and the left one, to mostly negative sequences predicted with some confusion. While the overall error of the preference model is high (0.4 MSE where the regression domain is [-1,1]), it is still useful since we are interested in its predictions in the positive (green) spectrum for the forthcoming optimization stage. While trading-off coverage, we increase prediction accuracy using selective prediction by allowing our classifier to abstain when it is not sufficiently confident. To this end, we ignore predictions whose magnitude is between two rejection thresholds (see Section 4.3.2). Based on preliminary observations, we fix the rejection thresholds to maintain 25% coverage over the validation set. In Figure 4ii we present a risk-coverage plot for user-1 (see definition in [8]). The risk surface is computed by moving two thresholds β_1 and β_2 across the histogram in Figure 4i, and at each point, for data not between the thresholds, we calculate the risk (error of classification to three categories: positive, neutral and negative) and the coverage (percent of data maintained).

We increase the diversity of generated samples by taking the score's sign rather than the exact score predicted by the preference model g_ϕ. Therefore, different positive samples are given equal score. For user-1, the average score predicted by g_ϕ for generated solos of Bebop-Net is **0.07**. As we introduce beam-search and increase the beam width, the performance increases up to an optimal point from which it decreases (see supplementary material). User-1's scores peaked at **0.8** with $b = 32, k = 8$. Anecdotally, there was one solo that user-1 felt was exceptionally good. For that solo, the model predicted the perfect score of 1. This indicates that the use of beam-search is indeed beneficial for optimizing the preference metric.

6. PLAGIARISM ANALYSIS

One major concern is the extent to which BebopNet plagiarizes. In our calculations, two sequences that are identical up to transposition are considered the same. To quantify plagiarism in a solo with respect to a set of source solos, we measure the percentage of n-grams in that solo that also appear in any other solo in the source. These statistics are also applied to any artist in our dataset to form a baseline for the typical amount of copying exhibited by humans.

Another plagiarism measurement we define is the largest common sub-sequence. For each solo, we consider the solos of other artists as the source set. Then, we average the results per artist. Also, for every artist, we compare every solo against the rest of his solos to measure self-plagiarism. For BebopNet, we quantify the plagiarism level with respect to the entire corpus. The average plagiarism level of BebopNet is 3.8. Interestingly, this value lies within the human plagiarism range found in the dataset. This indicates that BebopNet can be accused of plagiarism as much as some of the famous jazz giants. We present the extended results in the supplementary material.

7. CONCLUDING REMARKS

We presented a novel pipeline for generating personalized harmony-constrained jazz improvisations by learning and optimizing a user-specific musical preference model. To distill the noisy human preference models, we used a selective prediction approach. We introduced an objective evaluation method for subjective content and numerically analysed our proposed pipeline on four users.

Our work raises many questions and directions for future research. While our generated solos are locally coherent and often interesting/pleasing, they lack the qualities of professional jazz related to general structure such as motif development and variations. Preliminary models we have trained on smaller datasets were substantially weak. Can a much larger dataset generate a significantly better model? To acquire such a large corpus it might be necessary to abandon the symbolic approach and rely on raw audio.

Our work emphasizes the need to develop effective methodologies and techniques to extract and distill noisy human feedback that will be required for developing many personalized applications. Our proposed method raises many questions. To what extent does our metric express the specifics of one's musical taste? Can we extract precise properties from this metric? Additionally, our technique relies on a sufficiently large labeled sample to be provided by each user, a substantial effort on the user's part. We anticipate that the problem of eliciting user feedback will be solved in a completely different manner, for example, by monitoring user satisfaction unobtrusively, e.g., using a camera, EEG, or even direct brain-computer connections.

The challenge of evaluating neural networks that generate art remains a central issue in this research field. An ideal jazz solo should be creative, interesting and meaningful. Nevertheless, when evaluating jazz solos, there are no mathematical definitions for these properties—as yet. Previous works attempted to define and optimize creativity [48], but no one has yet delineated an explicit objective definition. Some of the main properties of creative performance are innovation and the generations of patterns that reside out-of-the-box— namely, the extrapolation of outlier patterns beyond the observed distribution. Present machine learning regimes, however, are mainly capable of handling interpolation tasks and not extrapolation. Is it at all possible to learn the patterns of outliers?

8. ACKNOWLEDGEMENTS

* Both authors contributed equally to this work.

9. REFERENCES

[1] L. A. Hiller Jr. and L. M. Isaacson, "Musical Composition with a High Speed Digital Computer," in *Audio Engineering Society Convention 9*. Audio Engineering Society, 1957.

[2] N. Jaques, S. Gu, R. E. Turner, and D. Eck, "Tuning Recurrent Neural Networks with Reinforcement Learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017. [Online]. Available: https://openreview.net/forum?id=Syyv2e-Kx

[3] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "Music Transformer: Generating Music with Long-Term Structure," *arXiv preprint arXiv:1809.04281*, 2018.

[4] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," *CoRR*, vol. abs/1810.12247, 2018. [Online]. Available: http://arxiv.org/abs/1810.12247

[5] K. Chen, W. Zhang, S. Dubnov, G. Xia, and W. Li, "The Effect of Explicit Structure Encoding of Deep Neural Networks for Symbolic Music Generation," in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*. IEEE, 2019, pp. 77–84.

[6] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[7] C. R. Robinson, "Differentiated Modes of Choral Performance Evaluation Using Traditional Procedures and a Continuous Response Digital Interface device," Ph.D. dissertation, Florida State University, 1988.

[8] Y. Geifman and R. El-Yaniv, "Selective Classification for Deep Neural Networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 4878–4887.

[9] Y. Geifman and R. El-Yaniv, "SelectiveNet: A Deep Neural Network with an Integrated Reject Option," in *International Conference on Machine Learning (ICML)*, 2019.

[10] P. Norvig, *Paradigms of Artificial Intelligence Programming: Case Studies in Common LISP*. Morgan Kaufmann, 1992.

[11] J. Gillick, K. Tang, and R. M. Keller, "Learning Jazz Grammars," *Proceedings of the SMC*, pp. 125–130, 2009.

[12] M. Löthe, "Knowledge Based Automatic Composition and Variation of Melodies for Minuets in Early Classical Style," in *Annual Conference on Artificial Intelligence*. Springer, 1999, pp. 159–170.

[13] F. Pachet, "The Continuator: Musical Interaction with Style," *Journal of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.

[14] R. Wooller and A. R. Brown, "Investigating Morphing Algorithms for Generative Music," in *Third Iteration: Third International Conference on Generative Systems in the Electronic Arts, Melbourne, Australia*, 2005.

[15] J. Sakellariou, F. Tria, V. Loreto, and F. Pachet, "Maximum Entropy Models Capture Melodic Styles," *Scientific reports*, vol. 7, no. 1, p. 9172, 2017.

[16] P. Laine and M. Kuuskankare, "Genetic Algorithms in Musical Style Oriented Generation," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. IEEE, 1994, pp. 858–862.

[17] G. Papadopoulos and G. Wiggins, "A Genetic Algorithm for the Generation of Jazz Melodies," *Proceedings of STEP*, vol. 98, 1998.

[18] P. Toiviainen, "Modeling the Target-Note Technique of Bebop-Style Jazz Improvisation: An Artificial Neural Network Approach," *Music Perception: An Interdisciplinary Journal*, vol. 12, no. 4, pp. 399–413, 1995.

[19] M. Nishijima and K. Watanabe, "Interactive Music Composer Based on Neural Networks," in *Proceedings of the 1992 International Computer Music Conference, ICMC 1992, San Jose, California, USA, October 14-18, 1992*, 1992. [Online]. Available: http://hdl.handle.net/2027/spo.bbp2372.1992.015

[20] J. Franklin, "Multi-Phase Learning for Jazz Improvisation and Interaction," in *In Proceedings of the Biennial Symposium on Arts and Technology*, 2001.

[21] J. D. Fernández and F. Vico, "AI Methods in Algorithmic Composition: A Comprehensive Survey," *Journal of Artificial Intelligence Research*, vol. 48, pp. 513–582, 2013.

[22] D. Eck and J. Schmidhuber, "Learning the Long-Term Structure of the Blues," in *International Conference on Artificial Neural Networks*. Springer, 2002, pp. 284–289.

[23] J. A. Franklin, "Jazz Melody Generation Using Recurrent Networks and Reinforcement Learning," *International Journal on Artificial Intelligence Tools*, vol. 15, no. 04, pp. 623–650, 2006.

[24] D. D. Johnson, R. M. Keller, and N. Weintraut, "Learning to Create Jazz Melodies Using a Product of Experts," in *Proceedings of the Eighth International Conference on Computational Creativity (ICCC'17), Atlanta, GA, 19*, 2017, p. 151.

[25] G. E. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[26] L. Yang, S. Chou, and Y. Yang, "MidiNet: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, 2017, pp. 324–331. [Online]. Available: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/226_Paper.pdf

[27] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1lYRjC9F7

[28] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a Steerable Model for Bach Chorales Generation," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 1362–1371. [Online]. Available: http://proceedings.mlr.press/v70/hadjeres17a.html

[29] H. H. Mao, T. Shin, and G. W. Cottrell, "DeepJ: Style-Specific Music Generation," in *12th IEEE International Conference on Semantic Computing, ICSC 2018, Laguna Hills, CA, USA, January 31 - February 2, 2018*, 2018, pp. 377–382. [Online]. Available: https://doi.org/10.1109/ICSC.2018.00077

[30] G. Hadjeres and F. Nielsen, "Interactive Music Generation with Positional Constraints using Anticipation-RNNs," *CoRR*, vol. abs/1709.06404, 2017. [Online]. Available: http://arxiv.org/abs/1709.06404

[31] E. Schubert, "Continuous Measurement of Self-Report Emotional Response to Music," *Music and Emotion: Theory and Research*, pp. 394–414, 2001.

[32] C. K. Madsen and J. M. Geringer, "Comparison of Good Versus Bad Tone Quality/Intonation of Vocal and String Performances: Issues Concerning Measurement and Reliability of the Continuous Response Digital Interface," *Bulletin of the Council for Research in Music education*, pp. 86–92, 1999.

[33] E. Himonides, "Mapping a Beautiful Voice: The Continuous Response Measurement Apparatus (CReMA)," *Journal of Music, Technology & Education*, vol. 4, no. 1, pp. 5–25, 2011.

[34] R. V. Brittin, "Listeners' Preference for Music of Other Cultures: Comparing Response Modes," *Journal of Research in Music Education*, vol. 44, no. 4, pp. 328–340, 1996.

[35] J. C. Coggiola, "The Effect of Conceptual Advancement in Jazz Music Selections and Jazz Experience on Musicians' Aesthetic Response," *Journal of Research in Music Education*, vol. 52, no. 1, pp. 29–42, 2004.

[36] "Sax solos," https://saxsolos.com/, accessed: 2019-05-16.

[37] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional Sequence to Sequence Learning," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 1243–1252. [Online]. Available: http://proceedings.mlr.press/v70/gehring17a.html

[38] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," in *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, 2016, p. 125. [Online]. Available: http://www.isca-speech.org/archive/SSW_2016/abstracts/ssw9_DS-4_van_den_Oord.html

[39] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-Level Language Modeling with Deeper Self-Attention," *CoRR*, vol. abs/1808.04444, 2018. [Online]. Available: http://arxiv.org/abs/1808.04444

[40] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[41] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[42] H. Inan, K. Khosravi, and R. Socher, "Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. [Online]. Available: https://openreview.net/forum?id=r1aPbsFle

[43] O. Press and L. Wolf, "Using the Output Embedding to Improve Language Models," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, 2017, pp. 157–163. [Online]. Available: https://aclanthology.info/papers/E17-2025/e17-2025

[44] PG Music Inc., "Band-in-a-box." [Online]. Available: https://www.pgmusic.com/

[45] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in*

Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, 2017, pp. 6000–6010. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need

[46] M. Cooke, D. Horn, and J. Cross, *The Cambridge Companion to Jazz.* Cambridge University Press, 2002.

[47] J. Cocker, "Elements of the Jazz Language for the Developing Improviser," *Miami: CPP Belwin*, 1991.

[48] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 230–247, 2010.

HOW MUSIC FANS SHAPE COMMERCIAL MUSIC SERVICES: A CASE STUDY OF BTS AND ARMY

Jin Ha Lee
University of Washington
jinhalee@uw.edu

Anh Thu Nguyen
University of Washington
atnd@uw.edu

ABSTRACT

Much of the existing research on user aspects in the music information retrieval field tends to focus on general user needs or behavior related to music information seeking, music listening and sharing, or other use of commercial music services. However, we have a limited understanding of the personal and social contexts of music fans who enthusiastically support musicians and are often avid users of commercial music services. In this study, we aim to better understand the contextual complexities surrounding music fans through a case study of the group BTS and its fan community, ARMY. In particular, we are interested in discovering factors that influence the interactions of music fans with music services, especially in the current environment where the prevalence of social media and other tools/technologies influences musical enjoyment. Through virtual ethnography and content analysis, we identified four factors that affect music fans' interactions with commercial music services: 1) perception of music genres, 2) participatory fandom, 3) desire for agency and transparency, and 4) importance of non-musical factors. The discussion of each aspect is followed by design implications for commercial music services to consider.

1. INTRODUCTION

Understanding users' motivations, needs, and behavior related to music is fundamental in designing commercial music services that will be well-received by users. Since the early 2000s, a steady stream of user studies has been conducted in the field of music information retrieval (MIR) [1]. These studies have investigated a variety of user aspects, such as their information needs and searching behavior [2], perception of music genres and moods [3], [4], and social music behavior [5], [6]. Some of these studies have focused on improving our understanding of MIR issues related to specific populations, such as youth [7], music creators [8], members of certain cultures [9], or people who use streaming, cloud, or recommender services [6], [10], [11].

One user group that has not been studied much in past MIR user studies are music fans. The term "fan," as an abbreviation for "fanatic," first appeared in a religious context in late 17th-century England, and became significant in the United States as it started to be used to describe passionate sports enthusiasts and later "dedicated audiences

for film and recorded music" [12, p.28]. Understanding what motivates and influences fans' behavior is important since they are often avid users of systems and services designed to provide access to media. Their digital touchpoints subsequently shape the design of these systems and services. While much of the current research involving user elements investigates user interactions with MIR systems and services, such as their usage of music and playlists [6], [10], [13], fewer works explore users more holistically or the characteristics of unique user groups. This work aims to fill this gap by 1) investigating the contextual factors that influence users' engagement with music services, and 2) attending more closely to music fans as an important subset of users. We conducted an empirical study of music fans to reveal the underlying motivations and reasons explaining why we see certain user behavior in commercial music services. The particular case we examined was a fandom called ARMY, consisting of supporters of the music group BTS and considered to be one of the largest pop music fandoms today [14]. BTS ARMY is an excellent case to study not only due to the sheer size of the fandom, but also because of its diversity and impact. ARMY is known to be an extremely dedicated fan base that actively participates in numerous initiatives to support BTS and relevant causes globally [15], [16].

In this paper, we aim to answer the following research questions: What does the case of BTS and ARMY tell us about the current landscape of music fans, specifically related to how they interact with commercial music services? Subsequently, what are the implications for designing and providing commercial music services for these fans?

2. RELATED WORK

2.1 Music Fans and Participatory Culture

Over the past two decades, user studies in MIR have shown that the way users interact with music has significantly changed. During this time, a majority of users moved from sharing actual music files and listening to personal collections to using streaming or subscription-based models and sharing YouTube links and music metadata [10]. Additionally, social media has become an important venue for people to share and discuss music [11]. These cultural and technological shifts have changed the ways people consume, create, and share music [17], and have altered the role of music users from passive listeners to more active participants, indicative of a larger trend in media use [18], [19]. Recent MIR user studies point out that users are not merely consumers of music but also shapers of music services. For instance, Lee and Price [20] observe that music users are getting increasingly savvy about the tools and

technologies available to them—rather than using only a single platform, music users very intentionally choose particular platforms that work well for specific purposes [10]. Contextual factors, sometimes unrelated to music itself, were also found to influence people's interaction with music services, such as their decisions about whether to listen to music recommendations [21]. These factors, of course, included reasons such as convenience or being in the mood to listen to recommendations, but interestingly, they also related to personal values. For instance, researchers learned that some participants actively "refused" to listen to certain songs based on how well their personal values aligned with those of the artist or their perception of the artist's ethical stance [21]. These findings imply the complexity of users' engagement with music services: they are often motivated by contextual factors going beyond preference for music based on musical attributes. In addition, many users have embraced participatory culture and contribute user-generated content related to music on social media, especially streaming venues like YouTube [22].

Music fans are at the center of this technological and cultural landscape and often drive trends taken up by other music users. Jenkins [18] discusses how in "convergence culture," fans play a central role in how culture operates, demonstrating the influence of an active audience in contemporary popular culture. While fandom suffered from stigma in earlier media studies, participatory culture has now become central in understanding fans and fan-based online communities in popular culture [18]. Users actively engage in online communities, produce creative works, and develop new knowledge [23], especially in the context of games [24], [25] and YouTube [26], [27]. Researchers also started investigating the participatory nature of user involvement related to music in certain contexts. For instance, Waldron [28] discusses how user-generated content in YouTube is used for music learning and teaching in online participatory communities. Schneider [22] uses qualitative media analysis to examine audience engagement with music videos on YouTube.

While it is important to understand how fans interact with music in commercial music services, it is also critical to consider the broader context in which users are situated to understand what factors may impact how they engage with commercial music services. Jenkins [23, p.7] emphasizes various aspects such as "the social, cultural, legal, political and economic institutions, practices, and protocols" that shape the communication technologies in media systems.

2.2 BTS and ARMY

Before Korean pop (Kpop) music became a global sensation, the Korean Wave, also known as "Hallyu," popularized Korean culture in media channels such as television dramas and digital games [15]. As it emerged alongside social media, Hallyu was able to reach audiences beyond East Asia including Latin America, Europe, and North America [29], [30]. While Hallyu became popular through various mediums, one of the driving forces behind its success is Kpop [15]. Artists such as Psy and BTS have collectively gained billions of views on YouTube, bringing Hallyu music fandom to North America.

BTS is a South Korean band with seven male musicians, managed by their entertainment agency, Big Hit, since June 2013. The abbreviation stands for "Bangtan Sonyeondan" or "Bulletproof Boy Scouts," which depicts the challenges that the younger generations face in modern social life [31]. A large part of BTS's success is its high engagement with fans on social media where band members share visual stories of their lives, aesthetic preferences, and commentary on their work. Tweets, Instagram posts, and other social media updates also enable fans from all over the world to connect with the band members [15].

Additionally, BTS engages with global campaigns to digitally connect with youth culture around the world. In November 2017, BTS launched a two-year anti-violence campaign called "Love Myself" in partnership with UNICEF that raised over $2,000,000 (USD) [32]. In 2018, the band delivered a speech at the United Nations General Assembly in New York to launch the "Generation Unlimited," a global partnership of UNICEF [32].

BTS's popularity has reached a global scale thanks to the unity of its fandom, ARMY, which has bonded through the band members' story of growth, authenticity, and determination to pursue musical careers. The influence of ARMY is massive in its own right [16], [33]. When the band's scheduled tour was cancelled in Korea due to COVID-19, ARMY followed the lead of one BTS member, Suga, by donating their refunds to disaster relief organization Hope Bridge—amounting to over $300,000 in just a few days [34]. In another instance, after ARMY learned that BTS and Big Hit Entertainment donated $1,000,000 to support Black Lives Matter, they organized a campaign to match the donation and raised another million dollars in a little over 24 hours [35]. The campaign is still ongoing on the One In An Army website (https://www.oneinanarmy.org/), along with many other campaigns for social good across the globe. These are just a few of numerous examples illustrating the power and impact the fandom has.

3. STUDY DESIGN AND METHODS

We employed a multimethod approach of virtual ethnography and content analysis. While traditional ethnography focuses on observing the interactions among individuals situated in a particular setting, as technologically mediated communications and online communities become ubiquitous, virtual ethnography is becoming increasingly common [36], [37]. Virtual ethnography explores social interactions taking place in virtual environments and emphasizes the immersion of the researcher in the setting for extended periods of time for a holistic understanding of the culture [38]. Beneito-Montagut [36] points out that applying physical boundaries in the context of the Internet during virtual ethnography provides limited understanding of everyday life in the Internet, as the various intersections between different sites are lost. For this reason, the researchers chose to examine multiple online websites and venues, including a swath of social media (such as Twitter, Facebook, Instagram, Reddit, and TikTok), YouTube, Daum BTS fan café, which used to be the official fan community, and Weverse, the current official channel for in-

teractions between BTS and ARMY. The first author observed the interactions among the fans for approximately a year from April 13, 2019, to March 30, 2020. During this phase, the researcher generated field notes documenting reflections and questions, while being cautious to minimize bias in collecting and interpreting the data.

In order to complement the ethnographic approach, the researchers also collected user data from Twitter and a subreddit, r/Bangtan. A total of 3,195 recent tweets (the maximum number allowed in the API policy) were pulled from the bts_twt timeline on Twitter. The tweets, along with the field notes generated from the observation phase, informed some of the prominent themes to be further investigated (e.g., streaming, radio play, award, donation). Additionally, discussion threads were scraped from r/Bangtan, and the second author, taking a deductive approach [39], qualitatively examined and coded the data into categories representing prominent themes identified in the observation phase. Here, we present selected user quotes that are helpful for discussing four aspects that commercial music services should consider to better cater to music fans.

This research investigates a single case study of BTS and its fandom, ARMY, to examine the current landscape of the music listening environment and the role of music fans in shaping it. Yin [40] describes a case study as "an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident" (p. 18). Single case studies and multiple case studies have their own merits. While studying multiple cases may help with generalizing the data, delving deeply into a single case can provide a richer understanding of the context that is being studied. In this paper, we selectively chose to study the case of BTS and ARMY for two reasons. First, BTS/ARMY is a unique case because it demonstrates an exemplary manifestation of the fandom phenomenon—ARMY could be seen as an outstanding success as it is often characterized as the most powerful, dedicated, and organized fandom in the world [14], [41]. Second, BTS/ARMY presents a longitudinal case where a single case is examined across time to demonstrate how situations and processes change. Indepth observation of user interactions over an extended period of time and examination of artifacts on various media channels enabled the researchers to have a better understanding of how ARMY might react to certain situations due to historical reasons. The case overall serves as an excellent candidate to examine the complexity of the user's context, which affects how fans engage with commercial music services.

4. FINDINGS AND DISCUSSION

4.1 Perception of Music Genres

Kpop is a prime example of how the boundaries of music genres are becoming increasingly blurry. The hybridity of Kpop music has been noted in prior research [15] and discussed in relation to the popularity of "idol" groups, often strategically created to consist of members with different musical strengths [42]. As a result, the music they produce tends to have elements accentuating the diverse strengths

of the members, embodying a fusion of multiple music styles [42]. This is also true in the case of BTS—in a recent interview about its latest album, *Map of the Soul: 7*, one BTS member actually stated that "the genre is BTS," and "it's less and less meaningful to divide music into genres now" [43]. Fans on the BTS subreddit also discuss this genre issue, as shown in the following quotes:

""bts is the genre" wasn't them saying they're not kpop, they were saying that their music is diverse and that putting it into a box is pointless."

"I love how he tried to highlight artistry and talent instead of focusing on artificial (and unrealistic) plastic subgenres like kpop. I am one of those firm believers that kpop (much like *pop*) is not a music genre but an industry with specific practices."

Beyond discussing the specific label "Kpop," which is already problematic given that it often refers to all music originating from Korea regardless of style, users also recognize the limitation of genre labels in general. They comment on the difficulty of applying one set of genre labels, commonly used in a certain culture, to all music.

"Yoongi [one of the BTS members] *said stop limiting south korea to a single genre when genre itself is meaningless as music continues to mix and evolve and we love to see it. Let's appreciate the artist no matter the genre!"*

"I don't know of one western equivalent in terms of the genre since BTS has a lot of different styles to their music."

Considering the wide range of artists who collaborate with BTS makes the band's music further challenging to label. For instance, its recent collaborators have ranged from vocalists such as Zara Larsson and Halsey to rappers like Desiigner and Nicki Minaj, who have very different music styles.

While BTS well exemplifies how different music genres and styles can be successfully mashed up to create hybrid music styles, this trend is also becoming increasingly popular among other Kpop groups and genres of music. Hybridity, fueled by increasing global collaboration among musicians and producers around the world, is becoming more prevalent in the music industry in general. "Genres have blended together so completely and seamlessly" that it is "almost impossible to label a lot of popular music as any one thing" [44].

Design Implication: Think of alternative means to show the connections of music, rather than relying on music genres, especially genres represented in a hierarchical structure.

For MIR researchers and designers of commercial music services, the blurred boundaries of genres raise an interesting question about how to best categorize music to support and improve users' access and discovery. The limitation of genre labels has been noted in previous MIR user research, especially with regards to users' perception of genres and the lack of consistency in their description in

cross-cultural contexts [42]. The trend of hybrid genres supports the importance of exploring other metadata elements that can be used to organize music, such as mood and suitable contexts (e.g., accompanying activities, users' biological rhythm). It also highlights the limitation of a hierarchical structure in organizing music, as it becomes increasingly difficult to find a single appropriate category for these kinds of music in a tree-like structure. Rather, a faceted approach may be more suitable, in which different musical styles can be described as a collection of organized tags, which avoids the pitfall of having to categorize the music with one *correct* label. In addition, a network-based approach (such as MusicLynx (http://www.semanticaudio. ac.uk/demonstrators/16-musicweb/) or Mora-Mcginity et al. [45]) could also complement a more traditional hierarchical organizational structure, as networks can represent the relationships and influences among artists that are often meaningful to the music fans. For instance, mapping the influences between BTS and other music groups based on their collaboration would probably generate a more interesting social network for users to explore than simply categorizing the music as Kpop.

4.2 Participatory Fandom

Our observation of fan activities on social media showed that fans engage in a wide variety of activities in addition to simply listening to and enjoying music from the artists they support. These additional activities include not only appreciating non-music content related to the artists (such as documentary films or books), but also interacting with the artists and other fans through social media and offline events. For example, a substantial number of user-generated videos, featuring reactions, theory (analytical videos examining music videos or lyrics), cover dances, unboxing, lyrics, and remixing, are regularly uploaded to YouTube [16], [46]. Many fans also donate a significant amount of time and resources to helping other fans or supporting various causes that the artists support [16]. For instance, much of the BTS content is translated by fans who volunteer their time—the massive followings on translation team and individual accounts on Twitter (e.g., @BTS_Trans with 1.5 million followers; @btstranslation7 with 340,700; @doolsetbangtan with 316,600; and many more) demonstrate their impact. Yoon [46] discusses how youths took on these roles as cultural translators and how they engage transnationally with digital media.

ARMY also showcases how to effectively use social media for participatory fandom, including Twitter, Facebook, Vlive, Reddit, etc. [46]. The various streaming, donation, and hashtag campaigns in which fans participate require them to utilize these apps and services to organize. While there is an app called Weverse, established by Big Hit as the official communication channel for BTS and fans, due to the volume of daily posts and the lack of organization, it is not an effective venue for systematically organizing these efforts. Instead, it primarily serves as a venue where the artists can directly communicate with fans by sharing posts and responding to fans' posts.

While globally there is no central coordinating organization within ARMY, the fan group is still extremely successful in coordinating to support the artists despite its loose structure—for instance, there are various accounts that provide streaming guides (e.g., @AllForARMY, @BTS_graphs, @BTS_Billboard, @ARMY52Hz) that recommend strategically curated streaming lists in specific music streaming services to enhance the band's rankings in various charts [15], [16]. The big BTS Fan Twitter accounts (Twitter accounts with a lot of followers) often follow these streaming guide accounts and retweet these messages so they can quickly and efficiently reach a large number of fans. Lee [16] and Lee [33] also discuss in depth how international ARMY, coordinating its efforts on Twitter and other social media, worked systematically to translate and distribute BTS content to get BTS to debut on US radio and television by providing ARMY members with instructions for requesting BTS's songs and actions to take if a request is rejected.

Sometimes fans actively use commercial music services to make a statement—in the case of BTS, when the People's Choice Awards awarded another band Group of 2019, Music Video of 2019, and Concert of 2019 despite BTS receiving the largest number of Twitter votes in all three categories (according to data collected and tracked by accounts like @ResearchBTS), fans protested by banding together on social media and re-charting BTS's entire discography (20 albums including solo mixtapes and two Japanese repackage albums) in iTunes in less than 24 hours [47]. This happened again when BTS did not receive any Grammy nominations despite having a huge success with its *Persona: Map of the Soul* album in 2019, with the hashtag #ThisIsBTS trending in Twitter [48]. The purpose of these organized movements is partly to show support and appreciation for the artists, but also to express listeners' discontent and disapproval of outdated customs and systems, such as the management of music awards.

Design Implication: Consider ways to promote artist and fan interactions in a two-way communication model and features that can support fans' activities to achieve group goals.

The active, participatory nature of fans should prompt designers of commercial music services to think about features they might employ to support fans' activities and maintain strong communities. One idea might be to explore features within commercial music services where fans and artists can have interactions. While existing social media platforms already offer venues for artist-fan interactions and community discussions, commercial music services can offer other unique types of interactions. For instance, Spotify for Artists allows artists to put together and share their own playlist through their profile page. This enables fans to feel closer to artists they like, as they can better understand artists' tastes and know what they are currently listening to. Furthermore, this feature has an additional benefit of encouraging listeners to venture out of their typical music realm and discover new songs and artists, as prior research suggests people tend to be more receptive to music recommendations coming from people who are experts or those they trust to have good music taste [21]. One suggestion is to expand this feature by turning it into a two-way interaction, in which fans can also suggest

music to artists. We observed that this occasionally happens in social media such as Twitter or Weverse, especially when fans want the artist to collaborate with another artist. A voting mechanism could be implemented so that fans could collectively put together a playlist for the artists, which would increase user engagement and provide a sense of bonding. Another idea is to consider features that better support group activities among fans. While sharing music and playlists is already a feature in many commercial music services, tweaking these features so that they can support group goals may appeal to fans. For instance, the streaming guide account's use of playlists is unique compared with other use scenarios in that the goal is not to enable group members to collaboratively create or modify the queue. Rather, it is to effectively distribute a specific playlist—created by a small team of users—that has been deemed most efficient to support the artists. Gifting music or codes for streaming to fans who cannot afford access to music also frequently occurs within the fan community, which can potentially be supported in commercial music services.

4.3 Desire for Agency and Transparency

During our observation of ARMY's interactions in online communities and our examination of tweets and reddit posts, we noticed a strong desire among users to have more agency in how they interact with music. For many BTS fans, the act of streaming is not only to simply listen to and enjoy music but also to strategically support the artist. As discussed above, sometimes streaming can also be an act of resistance against the existing social structure and systems that fans feel are unjust. These fans' selection of music services is often driven by their desire to maximize the impact of their streaming on the rankings of artists' albums in various charts.

A recent event that well illustrates this desire for agency among fans concerns the lack of radio play BTS received in Western radio stations after the release of its latest album, *Map of the Soul: 7*. Despite the group's past success in album sales, sold-out stadium tours worldwide, and participation in various promotional events in the US organized by radio stations, BTS received almost no radio play for its latest album [49], [50]. This led to many fans speculating why, resulting in discussion of multiple factors including payola (the illegal practice of payment from record companies in exchange for more radio exposure for their artists, discussed extensively in Leight [51]), DJs' perception that ARMY consists exclusively of teenage girls, the public's perception of BTS as "foreign" artists, and xenophobia [49], [50]. While we do not have space for a full discussion of all of these factors here, what is clear is that many fans expressed frustration and actively boycotted radio, opting to use music streaming services as a result. Many believed that their participation would matter and that they would have more control over how they interacted with music on streaming services, as opposed to radio, in which gatekeepers control what is played. Fans on Bangtan subreddit share:

"Ugh it's 2020. Traditional radio just needs to die and make way for streaming. I know radio matters for charting

and GP exposure, etc but ever since we proved BTS can chart well without it, the tiny bit of interest I have in it has all but evaporated. If anything, I would rather they continue to succeed without it. Especially if getting on radio still means they have to play their ridiculous games (interviews, payola)."

"People in radio here who don't play BTS, that's why I use Spotify AND buy BTS albums. Living in the past, these radio people are. That's what Freddie Mercurys "Radio Gaga" was about. Barely anything has changed. Look, most of the world, their first language isn't English anyway. You know? Thanks to internet, people are waking up. These people in the industry continuing to sleep screams hello, boomer. They do their thing, I'll continue to support BTS."

"why is radio so important still in terms of charting, i feel my generation doesn't even listen to it in the car or whatever, we all just use our spotify etc. bc it's more personalized playlist and u don't have to sit thru annoying ads every 2 or 3 songs."

Hertweck [50] points out that "streaming's prevalence creates an avenue to success in the music industry, and artists are no longer required to rely on Big Radio to reach mainstream." In addition to streaming services' access and convenience, the desire for user agency is a critical reason why certain streaming services have become increasingly relevant and supported by users. Fans are now much more aware of potential issues that traditional media carry due to abundant information shared on social media. Furthermore, listeners now have a legitimate alternative option to support their artists and still make the impact they want by focusing "efforts on streaming and purchasing digital/physical albums to make sure that BTS charts well" [49]. Indeed, BTS's latest album debuted as number one in the US Billboard 200, with the lead single "On" debuting at number four on the Billboard Hot 100 even with almost no radio play [52]. Some fans, however, noted that Billboard also potentially made mistakes in its calculation and wanted clarification on how it arrived at the final numbers. Transparency is extremely important for these fans, as shown in the following tweets:

"I think the ranking of no. 4 is amazing, but the calculation seems very suspect. At very least, Billboard should clarify how it arrived at those numbers because they are not adding up, and it doesn't seem like they counted other streams outside of Spotify and YT. #billboardrecalculate."

"You tweeted the wrong date representing an incorrect tracking week, then made a new tweet, still claiming 18.3M US streams when the true figures show 18.9M streams in the US on Spotify and YouTube ALONE. Please hear us and recalculate @billboardcharts."

Design Implication: Recognize that users want more agency and provide transparency in how usage and popularity are measured.

Streaming services in fact benefit to some degree from fans' desire to have more agency, as fans intentionally choose streaming services over traditional models of music access like radio. Interestingly, streaming services are

also attempting to adopt a model similar to radio's as they seek to expand their revenue sources by asking artists to pay to advertise their songs within the app. Spotify's Marquee, for example, notifies listeners about new songs/albums for artists who pay $5,000 or more, but Shaw warns that "the effort is controversial because it's complicating wider talks over long-term music rights between Spotify and the record companies" [53]. As music services try to balance meeting user needs and ensuring revenues, it is difficult to say which model is in the best interests of all stakeholders. Regardless of the final decisions made by music services, at the minimum, it is important to recognize fans will continue to demand more agency, the ability to see the impact of their efforts, and transparency in how various statistics are collected and shared.

4.4 Importance of Non-musical Factors

One of the reasons for BTS's massive success is often attributed to the group's abundant visual content, such as high-quality music videos and choreography videos [31]. Music listeners' consumption and appreciation of music is changing as they consider more non-musical factors such as visual elements or information about the artists or labels when deciding what to listen to [37].

In addition to publishing various types of non-music materials, the transmedia storytelling BTS does through these different works is impressive. In order to truly understand the whole narrative created by the band, fans not only have to listen to the music and decipher the lyrics, but also connect the clues hidden in the music video, choreography, performances from concerts and award shows, and printed books and webtoons containing episodes from the fictional narrative. In fact, there are numerous "theory" videos on YouTube analyzing the music of BTS in depth, connecting the symbols embedded in various creative works and explaining the ideas behind them [31].

The visual elements that accompany music do not merely serve to enhance individuals' experiences. Prior research also shows that users watching the music videos with friends and family and engaging with other people around music videos or user-generated videos (such as reaction or theory videos) are important and memorable social experiences [44]. This community connection is also evident for many reaction videos related to BTS, as researchers have observed these videos generate substantial numbers of views and user comments.

Jenkin [24] also discusses the importance of recognizing the interrelationship among different technologies and thinking beyond the affordances of individual technologies or tools, stating:

> Rather than dealing with each new technology in isolation, we would do better to take an ecological approach, thinking about the interrelationship among different communication technologies, the cultural communities that grow up around them, and the activities they support. (p. 8)

This assertion hints at the importance of exploring how tools and technologies that were not necessarily designed to support music listening or sharing music information could be used to complement the activities in commercial music services.

Design Implication: Consider incorporating more visual and non-musical content, along with metadata pointing to related works, to promote fans' engagement.

In MIR user studies, YouTube consistently ranks very highly as the most used music service, often above services that are specifically designed for music [10], [20], [54]. While providing album art or sharing links to relevant music videos are already basic features in many existing commercial music services, we can envision more features that creatively incorporate visual elements. Spotify's Enhanced Albums (which BTS used for its *Map of the Soul: 7* album) or Pandora Stories, which allow artists to add video messages explaining the album concept, other visual materials, or voice commentaries, are good examples. These kinds of features not only enrich the music listening experience by adding a visual layer, but also provide more information about the music and the artist, which helps fans understand the context of creation and the creator's intent. One suggestion for expanding this feature is to allow users to interpret the music and related materials and share their perspectives, or share stories about what the music means to them individually, which would contribute to the sense of bonding. This currently happens mainly in Twitter, YouTube, or online forums for fans, but it could be interesting to foster the conversation within commercial music services, alongside the artist's expression of the original intention. Additionally, providing metadata that links users to related non-music materials would allow fans to more easily access a variety of materials to assist their analysis and interpretation of creative works.

5. CONCLUSION AND FUTURE WORK

In this paper, we investigated a case study of BTS and ARMY to better understand the contextual complexities that drive music fans' perceptions and behavior in commercial music services. Through a case study of BTS and its fandom, ARMY, we derived and described four design implications for commercial music services to consider.

This study examines only a single case study of BTS and ARMY; thus, there could potentially be additional factors that are missing in our discussion. As previously stated, this is an intentional methodological choice, given that the aim of the study is not to produce generalized findings, but to gain insights from a deep investigation of a highly impactful single case study to help us think about design ideas related to commercial music services.

The bonding experience created in these fan communities is increasingly important in our current situation, where most people are practicing some degree of social distancing due to COVID-19. In our future work, we plan to continue our research on fan communities and investigate the underlying social structure and practices of the ARMY fandom. Additionally, we are interested in exploring how fans are finding ways to connect with one another through music in this unprecedented situation by co-listening, participating in streaming events, and collaboratively creating and sharing playlists.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] D. M. Weigl and C. Gustavino, "User studies in the music information retrieval literature," in *Proc. ISMIR 2011,* pp. 335–340.

[2] J. H. Lee, "Analysis of user needs and information features in natural language queries seeking music information," *Journal of the American Society for Information Science and Technology,* vol. 61, no. 5, pp. 1025–1045, 2010.

[3] X. Hu, "Music and mood: Where theory and reality meet," in *Proc. iConference,* 2010.

[4] M. Sordo, O. Celma, M. Blech, and E. Guaus, "The quest for musical genres: Do the experts and the wisdom of crowds agree?" in *Proc. ISMIR 2008,* pp. 255–260.

[5] S. J. Cunningham and D. M. Nichols, "Exploring social music behaviour: An investigation of music selection at parties," in *Proc. ISMIR 2009*, Kobe, Japan, Oct. 26–30, 2009, pp. 747–752.

[6] S. Y. Park, A. Laplante, J. H. Lee, and B. Kaneshiro, "Tunes together: Perception and experience of collaborative playlists," in *Proc. ISMIR 2019.*

[7] A. Laplante and J. S. Downie, "Everyday life music information-seeking behaviour of young adults," in *Proc. 7th International Conf. on Music Information Retrieval,* pp. 381–382, 2006.

[8] C. Inskip, A. MacFarlane, and P. Rafferty, "Creative professional users' musical relevance criteria," *Journal of Information Science,* vol. 36, no. 4, pp. 517–529, 2010, doi: 10.1177/0165551510374006.

[9] X. Hu, J. H. Lee, K. Choi, and J. S. Downie, "A cross-cultural study of mood in K-pop songs," in *Proc. ISMIR 2014*, pp. 385–390

[10] J. H. Lee, Y.-S. Kim, and C. Hubbles, "A look at the cloud from both sides now: An analysis of cloud music service usage," in *Proc. ISMIR 2016,* pp. 299–305.

[11] L. Spinelli, J. Lau, and J. H. Lee, "Influences on social practices surrounding commercial music services: A model for rich interactions," in *Proc. ISMIR 2018,* pp. 671–677.

[12] M. Duffett, *Understanding Fandom: An Introduction to the Study of Media Fan Culture.* New York, NY, USA: Bloomsbury, 2013.

[13] A. Hagen, "The playlist experience: Personal playlists in music streaming services," *Popular Music and Society,* vol. 38, no. 5, pp. 625–645, 2015, doi: 10.1080/03007766.2015.1021174.

[14] Y. Seo and J. Hollingsworth. BTS' army of admirers: Inside one of the world's most powerful fandoms. CNN. https://www.cnn.com/2019/10/12/asia/bts-fandom-army-intl-hnk/index.html (accessed May 13, 2020).

[15] W. Chang and S.E. Park, "The fandom of Hallyu, a tribe in the digital network era: The case of ARMY of BTS," *Kritika Kultura,* vol. 32, pp. 260–287, 2019, doi: 10.13185/KK2019.03213.

[16] J. Lee, *BTS and ARMY Culture.* Seoul, South Korea: CommunicationBooks, 2019.

[17] C. Cayari, "The YouTube effect: How YouTube has created new ways to consume, create, and share music," *International Journal of Education and the Arts,* vol. 12, no. 6, pp. 1–28, 2011.

[18] H. Jenkins, *Convergence Culture: Where Old and New Media Collide.* Cambridge, MA, USA: MIT Press, 2006.

[19] J. van Dijck, "Users like you? Theorizing agency in user-generated content," *Media, Culture & Society,* vol. 31, no. 1, pp. 41–58, 2009, doi: 10.1177/0163443708098245.

[20] J. H. Lee and R. Price, "Understanding users of commercial music services through personas: Design implications," in *Proc. ISMIR 2015*, pp. 476–482.

[21] J. H. Lee, L. Pritchard, and C. Hubbles, "Can we listen to it together?: Factors influencing reception of music recommendations and post-recommendation behavior," in *Proc. ISMIR 2019,* pp. 663–669.

[22] C. J. Schneider, "Music videos on YouTube: Exploring participatory culture on social media," in *Symbolic Interactionist Takes on Music* (Studies in Symbolic Interaction 47). Emerald Group Publishing Limited, 2016, pp. 97–117.

[23] H. Jenkins, *Confronting the Challenges of Participatory Culture Media Education for the 21st Century.* Cambridge, MA, USA: The MIT Press, 2009.

[24] A. Delwiche and J. J. Henderson, *The Participatory Cultures Handbook.* Taylor & Francis Group, 2012 [Online]. Available: ProQuest Ebook Central.

[25] J. Gee, *What Video Games Have to Teach Us About Learning and Literacy,* 1st ed. New York: Palgrave Macmillan, 2003.

[26] J. Burgess, J. Green, H. Jenkins, and J. Hartley, *YouTube: Online Video and Participatory Culture* (Digital Media and Society Series). Cambridge, England; Malden, MA, USA: Polity, 2009.

[27] C. Chau and M. U. Bers, "YouTube as a participatory culture," *New Directions for Youth Development,* vol. 2010, no. 128, pp. 65–74, 2010, doi: 10.1002/yd.376.

[28] J. Waldron, "User-generated content, YouTube and participatory culture on the Web: Music learning and teaching in two contrasting online communities,"

Music Education Research, vol. 15, no. 3, pp. 257–274, 2013.

[29] D. Jin, "An analysis of the Korean wave as transnational popular culture: North American youth engage through social media as TV becomes obsolete," *International Journal of Communication,* vol. 12, pp. 404–422, 2018.

[30] N. Ko, S. No, J. Kim, and R. Simões, "Landing of the wave: Hallyu in Peru and Brazil," *Development and Society,* vol. 43, no. 2, pp. 297–350, 2014, doi: 10.21588/dns.2014.43.2.008.

[31] C. Abbes, "Online digital media practices on Twitter by Korean pop idol BTS and fans: A case on BTS (방탄소년단) and their fans," M.A. thesis, Dept. Informatics and Media, Uppsala Univ., Uppsala, Sweden, 2019. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1334673/FULLTEXT01.pdf

[32] K. Jun. (2018). We have learned to love ourselves, so now I urge you to "speak yourself" [Speech]. Available: https://www.unicef.org/press-releases/we-have-learned-love-ourselves-so-now-i-urge-you-speak-yourself

[33] J. Lee, *BTS, Art Revolution: BTS Meets Deleuze.* Seoul, South Korea: Parrhesia Publishers, 2018.

[34] Y-R. Kwon. "BTS fans flock to donate for coronavirus fight." The Korea Herald. http://www.koreaherald.com/view.php?ud=20200302000697&np=1&mp=1 (accessed May 13, 2020).

[35] K. Kwak. "BTS' fan ARMY matches group's $1 million Black Lives Matter donation within 24 hours." Variety. https://variety.com/2020/music/news/bts-army-matches-black-lives-matter-million-dollar-donation-1234627455/ (accessed July 22, 2020).

[36] R. Beneito-Montagut, "Ethnography goes online: Towards a user-centred methodology to research interpersonal communication on the internet, *Qualitative Research,* vol. 11, no. 6, pp. 716–735, 2011, doi: 10.1177/1468794111413368.

[37] A. C. Garcia, A. I. Standlee, J. Bechkoff, and Y. Cui, "Ethnographic approaches to the internet and computer-mediated communication," *Journal of Contemporary Ethnography,* vol. 38, no. 1, pp. 52–84, 2009, doi: 10.1177/0891241607310839.

[38] L. M. Given, "Virtual Ethnography," in *The SAGE Encyclopedia of Qualitative Research Methods.* Thousand Oaks, CA, USA: SAGE Publications, 2008, doi: 10.4135/9781412963909.

[39] S. Elo and H. Kyngäs, "The qualitative content analysis process," *Journal of Advanced Nursing,* vol. 62, pp. 107–115, 2008, doi:10.1111/j.1365-2648.2007.04569.x

[40] R. Yin, *Case Study Research: Design and Methods.* 4th ed. Thousand Oaks, CA, USA: Sage Publications, 2009.

[41] L. M. Foong. "'The ARMY still holds the power': Inside the organised chaos of the BTS fandom." Medium. https://medium.com/@deconreconasia/the-army-still-holds-the-power-inside-the-organised-chaos-of-the-bts-fandom-9a54f4864ef2 (accessed May 13, 2020).

[42] J. H. Lee, K. Choi, X. Hu, and J. S. Downie, "K-pop genres: A cross-cultural exploration," in *Proc. ISMIR 2013.*

[43] J. Ochoa. "BTS talk new album 'Map of The Soul: 7': 'The Genre is BTS.'" Recording Academy Grammy Awards. https://www.grammy.com/grammys/news/bts-talk-new-album-map-soul-7-genre-bts (accessed May 13, 2020).

[44] R. Hilton, A. Tsioulcas, E. Hu, and S. Thompson. "The 2010s: The globalization of music." NPR. https://www.npr.org/2019/10/07/767904453/the-2010s-the-globalization-of-music (accessed May 13, 2020).

[45] M. Mora-Mcginity, A. Allik, G. Fazekas, M. Sandler, "MusicWeb: music discovery with open linked semantic metadata," in *Metadata and Semantics Research* in Communications in Computer and Information Science, vol. 672, Nov. 2016, pp. 291–296.

[46] K. Yoon, *Digital Mediascapes of Transnational Korean Youth Culture.* New York, NY: Routledge, 2020.

[47] S. Smiles. "BTS' entire discography re-enters iTunes Charts worldwide." The ONE. https://www.callthe-one.com/en/musicians-singers/bts-entire-discography-re-enters-itunes-charts-worldwide (accessed May 13, 2020).

[48] F.-T. Farha. "With or without a Grammy nomination, BTS have made a global impact with their music." USA Today. https://www.usatoday.com/story/entertainment/music/2019/11/20/bts-dont-get-grammy-nomination-fans-react-in-support/2581434001/ (accessed May 13, 2020).

[49] S. Field. "American radio continues to expose themselves by largely ignoring BTS." Hypable. https://www.hypable.com/american-radio-expose-themselves-by-ignoring-bts (accessed May 13, 2020).

[50] N. Hertweck. "No radio, no problem: How BTS scored a No. 1 hit without radio's help." Recording Academy Grammy Awards. https://www.grammy.com/advocacy/news/no-radio-no-problem-how-bts-scored-no-1-hit-without-radios-help (accessed May 13, 2020).

[51] E. Leight. "Want to get on the radio? Have $50,000?" Rolling Stone. https://www.rollingstone.com/music/music-features/radio-stations-hit-pay-for-play-867825/ (accessed May 13, 2020).

[52] B. Rolli, "BTS just broke a major YouTube record formerly held by Psy." Forbes. https://www.forbes.com/sites/bryanrolli/2020/04/07/bts-dna-youtube-

record-psy-gangnam-style/#12b42fb67e2f (accessed May 13, 2020).

[53] L. Shaw. Spotify's newest pitch to labels and musicians: Now you pay us. https://www.bloomberg.com/news/articles/2020-03-02/spotify-s-newest-pitch-to-labels-and-musicians-now-you-pay-us (accessed May 13, 2020).

[54] L. Liikkanen and P. Aman, Shuffling services: Current trends in interacting with digital music. Interacting with Computers, vol. 28, no. 3, pp. 352–371, 2016, doi: 10.1093/iwc/iwv004.

THE MIDI DEGRADATION TOOLKIT:
SYMBOLIC MUSIC AUGMENTATION AND CORRECTION

Andrew McLeod
EPFL
andrew.mcleod@epfl.ch

James Owers
University of Edinburgh
james.owers@ed.ac.uk

Kazuyoshi Yoshii
Kyoto University
yoshii@kuis.kyoto-u.ac.jp

ABSTRACT

In this paper, we introduce the MIDI Degradation Toolkit (MDTK), containing functions which take as input a musical excerpt (a set of notes with pitch, onset time, and duration), and return a "degraded" version of that excerpt with some error (or errors) introduced. Using the toolkit, we create the Altered and Corrupted MIDI Excerpts dataset version 1.0 (ACME v1.0), and propose four tasks of increasing difficulty to detect, classify, locate, and correct the degradations. We hypothesize that models trained for these tasks can be useful in (for example) improving automatic music transcription performance if applied as a post-processing step. To that end, MDTK includes a script that measures the distribution of different types of errors in a transcription, and creates a degraded dataset with similar properties. MDTK's degradations can also be applied dynamically to a dataset during training (with or without the above script), generating novel degraded excerpts each epoch. MDTK could also be used to test the robustness of any system designed to take MIDI (or similar) data as input (e.g. systems designed for voice separation, metrical alignment, or chord detection) to such transcription errors or otherwise noisy data. The toolkit and dataset are both publicly available online, and we encourage contribution and feedback from the community.

1. INTRODUCTION

Music language models (MLMs) have been the subject of much research in recent years. In the most general terms, their goal is to learn the structure of a typical piece of music, usually in symbolic form, as either a piano roll or a (monophonic or polyphonic) sequence of notes. Such models can be designed either as a stand-alone system (i.e. to perform a specific task such as voice separation, metrical alignment, or chord detection), or as part of an automatic music transcription (AMT) system along with an acoustic model.

In AMT systems, MLMs have thus far led to only small increases in performance compared to state-of-the-art acoustic models by themselves [9]. One possible reason

is that such MLMs are typically run at the frame-level [1], rather than at the note-level or the beat-level [20]. Regardless, even beat- or note-level MLMs have not led to very large improvements by themselves (e.g. [19,21]). One approach to solving this issue has been proposed in [22], where a separate "blending model" is used to combine the acoustic model with the MLM. The blending model leads to a small but significant increase in performance over using the acoustic model only.

Another possible reason for their minimal improvement is that such MLMs are not directly trained to solve the task at hand—to correct errors produced by the acoustic model. That is, they are not *discriminative* models taking data with errors as input and producing the correct transcription as output. Instead, they are typically trained to model the distribution of clean (usually MIDI) data, and used to alter the probabilistic predictions of the acoustic model. The integration of such an MLM into an AMT system usually involves searching through a large space of possible output transcriptions. One potential solution to this problem (at least when using an RNN-based MLM), is to train the model with scheduled sampling [3], which uses its own (noisy) outputs during training, teaching it to recover from such mistakes. In fact, the MLM from [22] is trained using scheduled sampling. However, this training strategy is only designed to allow the MLM to recover after a mistake, rather than to recognize and correct a mistake directly.

Training a discriminative model which "cleans" the output of an acoustic model is only feasible in the presence of a dataset mapping degraded data to clean data. Whilst this dataset could be produced by running an acoustic model on a dataset mapping audio to the correct transcription, such datasets are small relative to the amount of clean MIDI data available elsewhere. Our MDTK package allows the user to take any clean data and degrade it to have musical errors of their choosing. The pool of clean MIDI data is many orders of magnitude larger than that which maps audio to transcription data. For example, MAESTRO [7] has aligned MIDI and audio data of ~1 300 performances totalling ~200 hours. In comparison, the Lakh MIDI Dataset [14] comprises ~175 000 MIDI files [2] totalling ~9 000 hours. This is over 40 times the size, and

---

[1] It is debatable as to whether frame-based models should be called "language" models, since they do not work at a step related to the language (e.g. musical notes or beats), but rather the frames of the acoustic model. However, such a distinction is not the focus of this work.

[2] The true number of files is slightly smaller than this as it is known that some of these MIDI files are corrupted.

additionally spans diverse genres. Using a dataset such Lakh MIDI, MDTK allows for the creation of datasets large enough to make the direct discriminative task feasible. In addition, as we will discuss later in Section 2.2, there is no need to restrict learning capability by explicitly creating a degraded dataset: MDTK's `Degrader` objects can be used to degrade clean input dynamically when loading it into the model, thus providing on-the-fly data augmentation, enabling the model to be trained on a degraded dataset which is essentially unlimited in size.

This process is analogous to performing learned data augmentation—MDTK makes the discriminative task of correcting errors feasible by increasing the effective size of the dataset. Data augmentation has proved essential in other fields. In [5], the authors advocate the automated application of data augmentation for the ImageNet task [6], a classification task for image data. They find that by automatically tuning the type of data augmentation they apply for each task, they can attain a significant improvement over the state-of-the-art. In [1], the authors explicitly investigate the effects of generating augmented data in low-data regimes, advocating the use of learned generators—essentially what MDTK's `Degrader` objects are—using GANs. Finally, in [17], the authors solve their low-data regime issue for environmental sound classification by using data augmentation, finding that performing augmentations such as pitch shifting and time stretching leads to a 6 percentage point boost in classification accuracy. MDTK enables similar such data augmentation techniques to be performed easily for AMT.

For non-AMT tasks, standalone MLMs typically take as input MIDI files and output some alignment or label, depending on the task. To our knowledge, the robustness of these MLMs to noisy or incorrect data is rarely if ever analysed. This is not necessarily an important factor when clean MIDI files are used as input, but when such a MIDI file is the result of noisy process such as AMT or Optical Music Recognition (OMR; e.g. [18]), a model's robustness to noise becomes an important piece of information.

We propose that both of these shortcomings—poor AMT post-processing, and that MLMs' robustness to noise has not been analysed—can be addressed using excerpts of music to which noise is added. In an AMT system, a post-processing model which is trained directly to identify and correct similar noise should be better able to correct noisy acoustic model outputs than a generic MLM. Likewise, the robustness of a standalone MLM to noisy input can be analyzed with such noisy data, allowing the MLM to be evaluated for its potential usefulness in downstream tasks such as those involved in creating a complete piece of sheet music given an audio signal.

In this paper, we introduce the MIDI Degradation Toolkit (MDTK), a set of tools to easily introduce controlled noise into excerpts automatically extracted from a set of MIDI files. MDTK is similar to the Audio Degradation Toolbox [11] for audio, but to the authors' knowledge, ours is the first toolkit of its kind for MIDI data. The controlled noise includes (1) shifting the pitch of a note; (2) lengthening, shortening, or shifting a note in time;

(3) adding or removing a note; and (4) splitting or joining notes.

We also introduce the Altered and Corrupted MIDI Excerpts dataset version 1.0 (ACME v1.0), containing MIDI excerpts which have been degraded (and some which have not been degraded) using the toolkit, and four new tasks of increasing difficulty: to (1) detect whether each excerpt has been degraded; (2) if so, classify what degradation has been applied and (3) locate where a degradation has taken place; and (4) recover the original excerpt.

We present a simple baseline model for each task and analyse its performance. These baselines are provided as an easy starting point for researchers wanting to attempt our proposed tasks or post-process their own AMT data. We provide evaluation metrics for assessment and postulate that, if high performance were achieved, we would be able to improve AMT output using models trained for these tasks. We can easily swap out ACME v1.0 for a dataset matching the errors for a specific AMT system using a provided script.

2. THE TOOLKIT

The MIDI Degradation Toolkit (MDTK) is a python package, installable with pip, which can be used to introduce errors to MIDI excerpts. The code is released open source under an MIT License, and is available online. We encourage feedback and contribution from the community in its continued development.

Internally, MDTK stores each excerpt as a set of notes in a Pandas [12] DataFrame with columns pitch (MIDI pitch, with C4=60), onset (the onset time of the note, in milliseconds), track, and dur (the duration of the note, in milliseconds), all integers. It contains functionality to load an excerpt from a MIDI file (using pretty_midi [15]), as well as to read from and write to a CSV file.

2.1 Degradations

Each degradation provided in MDTK takes as input a pandas DataFrame of an excerpt of music, and returns a DataFrame with the given degradation. Some degradations (e.g. removing a note from an empty excerpt) are not always possible. In such cases, a warning is printed and None is returned. Care is also taken to ensure that no overlaps on the same pitch are introduced by a degradation. There are a total of 8 degradations in MDTK, each of which is described below.

The `pitch_shift` degradation changes the pitch of a random note. By default, the new pitch is chosen uniformly at random from all possible pitches (a minimum and maximum pitch can be given, and the valid range defaults to 21–108 inclusive). It can also be drawn from a weighted distribution of intervals around the original pitch, for example to emphasize octave errors from overtones. We also include a flag to force the new pitch to align with the pitch of some other note in the excerpt, to reduce out-of-key shifts, if desired.

Three degradations shift a random note in time in some way: `onset_shift` changes the note's onset time, leaving its offset time unchanged; `offset_shift` changes

the note's offset time, leaving its onset time unchanged; and `time_shift` changes the note's onset and offset times by the same amount, leaving its duration unchanged. For all of these degradations, care is taken to ensure that the shifted note does not lie outside the excerpt's initial time range. A minimum and maximum resulting duration can be specified, as well as a minimum and maximum shift amount. We also include flags to align some combination of the shifted note's onset or duration with those of other notes from the excerpt, ensuring the note lies on some metrical grid, if desired.

Two degradations can be used to either add a random note to an excerpt (`add_note`), or remove a random note from an excerpt (`remove_note`). Flags to align an added note's pitch, onset, or duration to those of existing notes are included.

Two degradations can be used either to split a note into multiple shorter consecutive notes or to combine consecutive notes at the same pitch into a single longer note. Specifically, `split_note` will cut a random note into some number of consecutive notes of shorter duration (the first of which begins at the original note's onset time and the last of which ends at the original note's offset time). By default the note is split into two shorter notes, but this—as well as a minimum allowable duration for the resulting notes—can be set with a parameter. Similarly, `join_notes` takes two or more consecutive notes at the same pitch (with a maximum allowable gap—set with a parameter—allowed between them), and joins them into a single note with onset time equal to that of the first note and offset time equal to that of the last.

2.2 Other Tools

2.2.1 Dynamically degrading clean data

MDTK includes the `Degrader` class, which can be used to degrade excerpts dynamically. When instantiating a `Degrader` object, the proportion of excerpts that should remain undegraded is set with a parameter (which can be 0). The probability of each degradation being performed on an excerpt (if it is to be degraded) can also be set at this time. Then, each time `Degrader.degrade(excerpt)` is called, a randomly degraded version of the input excerpt is generated according to the proportions set during object creation. The `Degrader` class can be easily inserted into any model training procedure in order to dynamically create new degraded excerpts during each epoch, dramatically increasing the amount of data available for training.

2.2.2 Automatically matching model errors

MDTK includes a `measure_errors.py` script, which can be used to estimate the types of errors (specifically, as degradations) typical to a particular AMT system, given a set of transcriptions and ground truths from that system. Note that there is no unique set of degradations which reproduce the errors that a transcription system has made (e.g., any `shift` degradation can be trivially replaced by a `remove_note` and an `add_note`).

We make no claim that the degradations found by the script correspond to the exact causes of the errors made by the AMT system. Rather, only that the distribution of degradations produces excerpts with similar properties to those transcribed by that system. Nonetheless, the script finds what we believe are a *reasonable* set of degradations to have produced those errors using a simple heuristic-based approach. Notes are first matched as correct if possible (same pitch, and onset and offset within a changeable threshold), and the remaining notes are checked for the various degradations in the following order: (1) `join_notes` and `split_note`, either of which may include an additional `offset_shift` or `onset_shift`; (2) `offset_shift`, if the pitch and onset time match; (3) `onset_shift`, if the pitch and offset time match; (4) `time_shift`, but only if the transcribed note overlaps the position of the corresponding ground truth note; and (5) `pitch_shift`, which must match in onset time, although an additional `offset_shift` can be added. Finally, any remaining unmatched notes are counted as `add_note` and `remove_note`.

The output of the script is a json file containing the estimated proportion of each degradation in the given set of transcriptions. It does not yet include values for the various degradation parameters (though this is planned for a future update to MDTK). This output file can be used, for example, to create a custom-tuned, static, degraded dataset for training a model. However, the two tools can also be combined in powerful ways. By passing this json file to the `Degrader` constructor, a `Degrader` object can be instantiated that generates degradations exactly matching the estimated proportions. This could then be used to train a model to correct the errors of that specific AMT system using a relatively small amount of raw data.

3. THE DATASET

3.1 ACME version 1.0

The Altered and Corrupted MIDI Excerpts dataset v1.0 (ACME v1.0) is a dataset of 5 second excerpts with degradations implemented by MDTK. It is not intended to emulate the errors of any specific AMT system, but rather serve as a starting point for the modelling tasks we introduce below.

The dataset is taken from two sources: (1) the piano-midi dataset [3] , which contains 328 MIDI files of pseudo-live performance [4] piano pieces of various styles (generally classical); (2) the 22194 primes from the small, medium, and large sections of the monophonic and polyphonic Patterns for Prediction Development Datasets (PPDD-Sep2018) [5] , which contain excerpts drawn randomly from the Lakh MIDI Dataset (LMD) [14]. We remove track information, flattening each excerpt to a single track, simplifying the modelling tasks [6] ; analysis

---

[3] http://www.piano-midi.de

[4] The files are quantized and beat-aligned, but their tempo curves were manually edited by their creator to sound more like live performance.

[5] https://www.music-ir.org/mirex/wiki/2019: Patterns_for_Prediction

[6] The use of tracks is not standard our different data sources.

of multi-track MIDI files will be addressed in future work. We then fix any pair of overlapping notes of the same pitch by cutting the first note at the onset time of the second. We additionally set the offset time of the second note to the maximum of the original offset times of the two notes, such that no sustain is removed.

Once this pre-processing is complete, we select a ~5 second excerpt from each piece by choosing a random note and all notes beginning in the subsequent 5 seconds, but require that at least 10 notes be present. The excerpt ends when the last held note is released. This duration is approximately 2 bars for most songs so is small enough for the models proposed in section 4 to train quickly. We degrade $\frac{8}{9}$ of the excerpts, selecting the degradation uniformly at random from the set of 8 defined degradations, and leave the remaining $\frac{1}{9}$ undegraded. For ACME v1.0, we use default parameter settings for all degradations, although we intend to investigate the effect of different settings in future work (and future releases of ACME datasets).

The excerpts and degraded excerpts are split randomly into training, validation, and test sets of proportion 0.8, 0.1, and 0.1, creating the official splits for ACME v1.0. The canonical form is available online as a set of CSV files. Additionally, the MDTK package includes the `make_dataset.py` script which we used to create the dataset from scratch—including the automatic downloading of the raw data—and thus serves as a record of how it was created.

3.2 Custom Dataset Creation
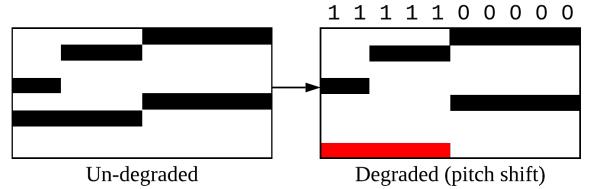
The `make_dataset.py` script can also be used to generate an ACME-style dataset from a user-provided set of MIDI or CSV files. The user can specify custom sizes for the excerpts, a custom distribution of the various degradations, as well as custom parameters for each. The script can be given the json output of the `measure_errors.py` script in order to match the properties of the generated dataset with those measured from an AMT result. Alternatively, a user can simply choose to degrade individual excerpts from their own training set by calling MDTK during the training process, either manually or randomly using the `Degrader` class.

4. PROPOSED TASKS

4.1 Motivation

These tasks are performed on ACME v1.0, and proposed in lieu of taking existing AMT systems and measuring their improvement when trained with the assistance of MDTK. It is proposed that the *output* of arbitrary AMT systems could be improved with models that can solve these tasks.

For instance, we could use a model trained to classify the error contained within a given excerpt to call out for human intervention. We could also train models to perform the actual fix; however, we show that, with the models we have chosen for our baseline, this problem is far from solved.

Figure 1. Example piano rolls of a clean excerpt (left) being degraded with pitch_shift (right), including the labels for Error Location (top right).

4.2 Tasks

We propose four tasks of increasing difficulty. Figure 1 shows a simple toy data point which has been pitch shifted (changed note in red). We will use it as an example when necessary throughout this section. We should note that the tasks we introduce here are not in any way trivial, but represent significant steps towards successful AMT postprocessing.

1. Error Detection: detect whether a given excerpt has been degraded. This is a binary classification task with a skewed distribution: $\frac{8}{9}$ excerpts are degraded (the positive class), and $\frac{1}{9}$ are not degraded (the negative class). We evaluate performance using F-measure but, since the negative class is the minority, for the purposes of F-measure evaluation, we treat those as positives. Thus, a model which always outputs "degraded" achieves a "reverse F-measure" of 0.00 (with precision and recall both 0) rather than its F-measure of 0.94 (with precision $\frac{8}{9}$ and recall 1).

2. Error Classification: specify which degradation (if any) was performed on each excerpt. This is a multi-class classification problem, and since ACME v1.0 contains a uniform distribution of each class, we evaluate performance using accuracy and a confusion matrix to show specific error tendencies for each degradation.

3. Error Location: assign a binary label to each (40 ms) frame of input identifying whether it contains an error i.e. whether this frame contains a degradation. We evaluate performance using the standard F-measure. The labels for this task are shown in the top right of Figure 1.

4. Error Correction: output the original, un-degraded version of each excerpt. In Figure 1, a model is given the degraded excerpt (right) and expected to output the original excerpt (left). For this task, we define our own metric, helpfulness (H), based on two F-measures proposed by [2]: frame-based F-measure with 40ms frames, and note-based onset-only F-measure. We use the `mir_eval` [16] implementation of note-based F-measure (with 50ms onset tolerance) to evaluate both the given excerpt and the system's output compared to the original excerpt. We take the average between the two F-measures for each excerpt, which we denote F_g (for the given excerpt) and F_c (for the system's corrected output). If $F_g = 1$ (the given excerpt was not degraded), $H = F_c$. If the given excerpt was degraded, however ($F_g < 1$), H is calculated as in Equation (1). An intuition for this calculation is as follows: $H = 0.5$ represents an output which is exactly as accurate as the given excerpt (the error correction system has neither helped nor hurt), and H scales linearly up to 1

and down to 0 from there. For example, $H = 0.75$ represents an output which is in some sense twice as accurate as the given excerpt (its error, $1 - F_c$, is half of the given excerpt's error, $1 - F_g$). Similarly, $H = 0.25$ represents an output which has twice as many errors as the input.

$$H = \begin{cases} 1 - \frac{1}{2}\frac{1-F_c}{1-F_g} & F_c \geq F_g \\ \frac{1}{2}\frac{F_c}{F_g} & F_c < F_g \end{cases} \quad (1)$$

4.3 Baseline Models

4.3.1 Data Formats

For input into our baseline models, we first quantize each excerpt onto 40 ms frames, rounding note onsets and offsets to the nearest frame. We use two different input formats for our baseline models, and provide data conversion and loading functions for each of them.

The *command format* is based on the one designed by [13]. Each excerpt is converted into a sequence of one-hot vectors representing commands from a pre-defined vocabulary of 356 commands: $note\_on(p)$, $note\_off(p)$, and $shift(t)$ ($p \in [0, 127]$, $t \in [1, 100]$). The note on and off commands represent note onsets and offsets at the current frame, and the shift command skips t frames. Longer shifts are represented by multiple shift commands.

The *piano roll format* represents each excerpt as two binary piano rolls: one representing pitch presence in each frame, and another represent pitch onsets in each frame. These two piano rolls are concatenated together frame-wise to form the model's input.

4.3.2 Model details

The details for the models provided in this paper are brief. For code which fully defines the models and the code used to train and evaluate them, see the repo[7]. Our choice of models was relatively arbitrary; they are easy to implement with existing open source packages and easy to improve upon.

Our baseline for **Error Detection** uses the *command format* as input. It consists of an embedding layer of size 128, followed by a basic Long Short-Term Memory (LSTM) [8]. A dropout of 0.1 is applied to the final LSTM state's output, which is then passed to a fully-connected layer of size 2 with softmax activation, resulting in a single output for each input sequence.

Our **Error Classification** baseline uses the same design, but with output dimensionality 9 for the final layer (one for each degradation plus one for no degradation).

For **Error Location**, we use the *piano roll format*. We first feed the input frames into a bi-directional LSTM (Bi-LSTM), and send the output of each Bi-LSTM state (with dropout 0.1) into 3 linear layers, each with dropout 0.1 and ELU activation. These are each fed into a final fully-connected layer of size 2 with softmax activation, resulting in one output per input frame.

For **Error Correction**, we use the *piano roll format*, and base our network on a basic Encoder-Decoder structure [4], where both the encoder and the decoder are Bi-LSTMs. The input is passed directly into the encoder Bi-LSTM, and the output at each frame is passed through a

---

[7] https://www.github.com/JamesOwers/midi_degradation_toolkit

Task	Model	Loss	Metric
Error Detection	Rule-based	0.466	0.000
	Baseline	**0.344**	0.000
Error Classification	Rule-based	2.197	0.113
	Baseline	**2.130**	**0.189**
Error Location	Rule-based	0.404	0.000
	Baseline	**0.109**	**0.525**
Error Correction	Rule-based	**0.690**	**0.590**
	Baseline	0.693	0.000

Table 1. Loss and evaluation metric for the baseline and rule-based models for each task on the ACME v1.0 test set. Each task's metric is different, as explained in the text.

single fully connected layer with dropout 0.1. This sequence is input into the decoder Bi-LSTM, each output of which is fed into 4 linear layers which output a vector of the same length as the input.

The models were trained using the Adam optimizer [10], and a grid search was performed for weight decay, learning rate, LSTM hidden-unit size, and linear layer sizes (for full details, see the code). The model with the lowest validation loss on each task is used as the baseline.
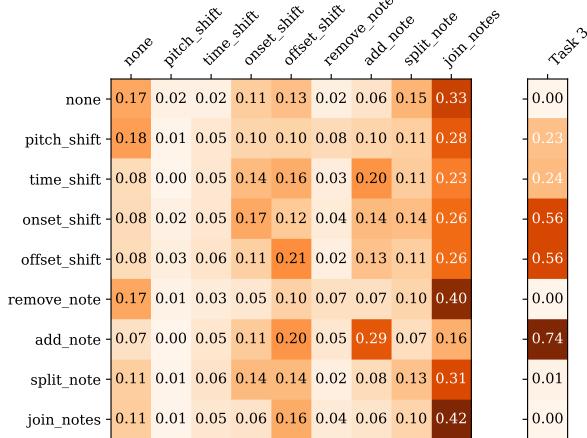
4.4 Analysis

To gauge the difficulty of each task, we compare each of the baseline models to a simple rule-based approach. Like our baseline models, the rule-based systems output probability values $\in [0, 1]$. For Error Detection, the rule-based system returns an $\frac{8}{9}$ probability of each data point being degraded. For Error Classification, the rule-based system outputs a $\frac{1}{9}$ probability for each class. For Error Location, the rule-based system outputs a 0.06 probability that each frame has been degraded (the proportion of frames that are degraded in the training set is 0.06). Finally, for Error Correction, we calculate $p(1|0) = 0.01$ and $p(1|1) = 0.96$ from the training set[8] and have the system output these values for each cell in a given piano roll.

The results for each task on the ACME v1.0 test set are shown in Table 1. From the losses, it is clear that the baseline models have learned something, since all of their losses are lower than the rule-based losses except for in Error Correction. However, from the metrics, it is also clear that there is much room for improvement on each of the proposed tasks (as we would hope).

For Error Detection, the baseline predicts 1 (degraded) for every data point, just like the rule-based system, likely because of the skew of the training data. As a simple attempt to overcome this tendency, we trained another model identical to the baseline which weights the loss of each data point inversely proportional to that label's frequency in the training set. This results in a model with greater overall loss (as expected), but which outputs some 0s, achieving a

---

[8] That is, for the degraded piano rolls from the training set, 1% of cells with a 0 and 96% of cells with a 1 map to a value of 1 in the corresponding cell of the clean piano roll.

Figure 2. Left: Confusion matrix showing the distribution of the baseline Error Classification model's classifications, normalized by true label. Rows show the true label, and columns show the predicted label. Right: The baseline Error Location model's F-measure for each degradation type.

reverse F-measure of 0.155. Overcoming the skew of the dataset may prove to be a challenge for this task.

For Error Classification, the baseline achieves an accuracy of greater than that of the rule-based system. The baseline's confusion matrix is shown in Figure 2 (left), where rows represent the ground truth label and columns represent its output. This shows error tendencies, and (more importantly) gives an idea of the general difficulty of detecting each degradation. Here, it can be seen that the maximum point in each column is always on the diagonal, showing that the model does seem to have learned something sensible. It performs well on the add note degradation, classifying 32% of those data points correctly. Pitch shift, time shift, and remove note seem to be the most difficult, while join notes is a common target for false positives. We are interested to see whether the above trends continue in future work on Error Classification, and intend to further investigate their causes.

The Error Location baseline outperforms the rule-based system in terms of both loss and F-measure by wide margins. It achieves this F-measure with a precision of 0.844 and a recall of 0.381, so although it rarely guesses that a frame has been degraded, it is usually correct when it does. Figure 2 (right) presents the baseline's F-measure split by degradation type, which shows the model performing best on add_note, but also well for onset and offset shifts (precision is over 0.9 for all three). It is slightly worse with pitch and time shifts (precision over 0.6 for both), and performs poorly on the other degradations (the value for "none" will always be 0 since it has no positives). Given the relative success of this model compared to the other tasks' baselines, pre-training a model for this task before continuing to train it for another task might be an avenue for improved performance. Another strategy could be to use a model trained for this task as an attention mechanism for some of the other tasks.

Error Correction is clearly the most difficult task of the four, and the baseline model's performance reflects this.

Although its loss is similar to that of the rule-based system, its helpfulness lags clearly behind. The rule-based model's strategy of (essentially) reproducing the input turns out to be a strong baseline. Our baseline, on the other hand, almost always outputs empty piano rolls, no matter the input. The difficulty of this task might require a more modular approach than the presented end-to-end baseline, perhaps combining the results of models from tasks 2 and 3 with a system designed to correct a specific degradation affecting a specific set of frames.

5. CONCLUSION

In this paper, we have introduced the MIDI Degradation Toolkit (MDTK), which contains tools to "degrade" (introduce errors to) MIDI excerpts. The toolkit is publicly available online [9] under an MIT License, and we encourage contributions and feedback from the community. Using MDTK, we have created the Altered and Corrupted MIDI Excerpts v1.0 (ACME v1.0) dataset [10] and include in MDTK a tool to create custom ACME-style datasets with different settings or data. We have proposed a set of four new tasks of increasing difficulty involving such datasets: Error Detection, Classification, Location, and Correction, and designed evaluation metrics and scripts for each of them. We also designed and presented simple models to be used as a baseline for each, which show that the proposed tasks are non-trivial, and may require innovative solutions.

The toolkit is ready to be used for improving Automatic Music Transcription (AMT). To do so, a user can:

1. use `measure_errors.py` to analyse the types of errors made by an AMT system or acoustic model.
2. instantiate a `Degrader` with the configuration produced by `measure_errors.py`—this can generate unlimited data matching the errors made by the system from step (1).
3. train a discriminative model using data generated by the `Degrader`.
4. apply that model to the output of the model from step (1) and evaluate the difference in performance.

As performance on the proposed tasks modelling ACME v1.0 improves, we intend to introduce ACME v2.0 with additional features such as multi-track excerpts, a track-based degradation, longer excerpts, multiple degradations per excerpt, and various parameter settings for the degradations. We also intend to analyze the effect of adding noise on MLM performance.

6. ACKNOWLEDGEMENTS

---

[9] https://www.github.com/JamesOwers/midi_degradation_toolkit

[10] https://www.github.com/JamesOwers/acme

7. REFERENCES

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks, 2017.

[2] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, 2009.

[3] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[5] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. *International Society for Music Information Retrieval Conference (ISMIR)*, pages 475–481, 2016.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015.

[11] Matthias Mauch and Sebastian Ewert. The audio degradation toolbox and its application to robustness evaluation. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2013.

[12] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[13] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 32(4):955–967, 2020.

[14] Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. PhD thesis, Columbia University, 2016.

[15] Colin Raffel and Daniel PW Ellis. Intuitive analysis, creation and manipulation of midi data with pretty_-midi. In *International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, pages 84–93, 2014.

[16] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Dan P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[17] Justin Salamon and Juan P. Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[18] Eelco van der Wel and Karen Ullrich. Optical music recognition with convolutional sequence-to-sequence models. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 731–737, 2017.

[19] Qi Wang, Ruohua Zhou, and Yonghong Yan. Polyphonic piano transcription with a note-based music language model. *Applied Sciences*, 8(3), 2018.

[20] Adrien Ycart and Emmanouil Benetos. A study on LSTM networks for polyphonic music sequence modelling. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 421–427, 2017.

[21] Adrien Ycart and Emmanouil Benetos. Polyphonic music sequence transduction with meter-constrained LSTM networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 386–390, 2018.

[22] Adrien Ycart, Andrew McLeod, Emmanouil Benetos, and Kazuyoshi Yoshii. Blending acoustic and language model predictions for automatic music transcription. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

JOYFUL FOR YOU AND TENDER FOR US: THE INFLUENCE OF INDIVIDUAL CHARACTERISTICS AND LANGUAGE ON EMOTION LABELING AND CLASSIFICATION

Juan Sebastián Gómez-Cañón[1]   **Estefanía Cano** [2]
Perfecto Herrera [1]   **Emilia Gómez**[3,1]
[1] Music Technology Group, Universitat Pompeu Fabra, Spain
[2] Songquito UG, Erlangen, Germany
[3] European Commission, Joint Research Centre, Seville, Spain

`juansebastian.gomez@upf.edu`

ABSTRACT

Tagging a musical excerpt with an emotion label may result in a vague and ambivalent exercise. This subjectivity entangles several high-level music description tasks when the computational models built to address them produce predictions on the basis of a "ground truth". In this study, we investigate the relationship between emotions perceived in pop and rock music (mainly in Euro-American styles) and personal characteristics from the listener, using language as a key feature. Our goal is to understand the influence of lyrics comprehension on music emotion perception and use this knowledge to improve Music Emotion Recognition (MER) models. We systematically analyze over 30K annotations of 22 musical fragments to assess the impact of individual differences on agreement, as defined by Krippendorff's α coefficient. We employ personal characteristics to form group-based annotations by assembling ratings with respect to listeners' familiarity, preference, lyrics comprehension, and music sophistication. Finally, we study our group-based annotations in a two-fold approach: (1) assessing the similarity within annotations using manifold learning algorithms and unsupervised clustering, and (2) analyzing their performance by training classification models with diverse "ground truths". Our results suggest that a) applying a broader categorization of taxonomies and b) using multi-label, group-based annotations based on language, can be beneficial for MER models.

1. INTRODUCTION

Several studies suggest that the main reason people engage with music is its emotional effect [1–3]. This makes the idea of computational algorithms that can "predict" the

emotions in music particularly intriguing and provocative. These algorithms evaluate emotionally relevant acoustic features from the audio signals, and correlate them with certain emotions that the music could convey, express or induce. Recently, deep learning approaches have been used to further improve emotion recognition [4–6]. However, the performance of these algorithms may have reached a "glass ceiling" possibly due to the subjective nature of the perception of emotions [7,8], the limited agreement in the annotations of datasets [9–11], the lack of an agreed methodology for annotation gathering [2], the generalized confusion between perceived and induced emotions [3, 12], amongst other reasons. In particular, the low agreement problem extends to other high-level description tasks in Music Information Retrieval (MIR) such as music auto-tagging [13], music genre recognition [14, 15], music similarity [9, 11], and even computationally well-defined tasks like automatic chord estimation [16] and beat tracking [17]. Given the importance of annotation collection and in an attempt to improve their quality, we address two research questions in this paper: *RQ1* - Do personal characteristics and mother tongue have an influence on the annotation of perceived emotions for listeners? *RQ2* - Can this information be used to improve MER algorithms? The rest of the paper is structured as follows: Section 2 reviews basic definitions and previous work, in Section 3 we detail the methodology of our study, including annotation gathering, clustering, and classification schemes. Section 4 provides results of our study which are later discussed in Section 5.

2. RELATED WORK

In this study, we focus on the *perception* of emotions as a key factor in the "musical communication" between music itself and a listener. *Perceived* emotions refer to those recognized by the listener through the interpretation of musical properties [3]. In contrast, *induced* emotions concern the arousal of psycho-physiological responses [18]. [1] The relation between musical properties and emotion perception has been widely researched in the literature [3, 12]:

---

[1] For a review of the emotivist-cognitivist argument, refer to [19,20].

happiness is linked to fast mean tempo, bright timbre, sharp duration contrasts; *sadness* is related to slow mean tempo, low sound level, dull timbre, slow vibrato; *fear* is linked with very low sound level, large sound level variability, large timing variations.

From music psychology, two dominant views or taxonomies for emotion representation prevail [21]:

- *Categorical approach*: emotions are represented as categories, distinct from each other - such as happiness, sadness, anger, and fear [22]. The major drawbacks of this approach are: (1) the amount of categories results too small compared to the richness of human emotion, and (2) the ambiguity of using language during self-reports [7].

- *Dimensional approach*: emotions are conceptualized based on their positions on a few dimensions, mainly arousal and valence (AV). Russell popularized the two-dimensional circumplex model, where the valence dimension describes the pleasantness or positiveness of the emotion, and the arousal dimension describes the activation or energy (i.e., *happiness* would have positive arousal and valence) [23]. However, the major flaw is that categories are not mutually exclusive and tend to overlap (i.e., rage-anger), making the mapping of categories on the dimensional space vague and unreliable [7].

In the particular case of instrumental classical music, Schedl et al. explored the relationship between listeners' characteristics and nine categories of emotion on segments from the 3rd Symphony *Eroica* by L.V. Beethoven [10]. Their results suggest that: (1) the perception of transcendence and power correlates significantly with affinity to classical music; (2) participants trained on classical music tend to disagree more on perceived emotions of peacefulness, tension, sadness, anger, disgust, and fear; (3) the agreement among perceived emotions decreases with increasing familiarity with the piece. Our work is based on this study, however we focus on music with lyrics of pop and rock style. Very few studies have explored different styles of music and researchers report that 50% of music and emotion studies focus on classical music [2,24]. When it comes to annotation reliability, researchers have studied ways of increasing inter- and intra-rater agreement for music similarity. Flexer and Lallai found evidence that upper bounds for inter-rater agreement (i.e., measured between different subjects) cannot be increased for this task, while the intra-rater case can be improved (i.e., measured on ratings from the same subject at different time) [11]. We base our research on increasing inter-rater agreement by analyzing listeners with similar characteristics and assembling group-based annotations based on listeners characteristics. Group-based MER has been attempted by Yang et al. by assembling annotations according to cultural factors, music experience, and personality traits [7, 25]. Their results suggest insignificant improvement for the regression task as compared to using general averaged annotations. However, and to the extent of our knowledge, this is the first work that uses language and self-reported lyrics comprehension to group annotations of perceived emotion.

3. METHODOLOGY

The main contribution of our work is to address open questions from previous studies by focusing on rock and pop music. We use these musical styles since they appear to be similar across different cultures and are musically homogeneous, even when sung in different languages. Contrary to most studies, we focus on a small set of songs with existing emotion annotations and gather large-scale, diverse annotations per song from participants of different mother tongues. The goal of our work is to study the relationship between listeners' demographics, preference, familiarity, musical knowledge, and native language with agreement of perceived emotions in music. In order to achieve this, we use these personal characteristics to form group-based annotations and analyze the agreement of participants belonging to these groups. We then perform manifold learning and K-means clustering to study if group-based annotations yield representations that are more similar amongst them. Finally, we compare the performance of well-known classifiers trained using these annotations, in order to analyze the impact of grouping variants on the MER task.

3.1 Emotion annotation gathering

Our pool of annotators were presented with surveys designed with PsyToolkit [26] in four languages: Spanish, English, German and Mandarin. The survey was structured as follows: (1) collection of general demographic information (age, gender, country of origin and formation, and language), (2) volume adjustment task, (3) explanation of the difference between *perceived* and *induced* emotions, (4) random presentation and annotation of excerpts with a 5-point Likert response format per emotion, and (5) the Music Sophistication Index self-report inventory [27]. In (2), each user was asked to set the volume w.r.t. a 1 KHz sinusoid making it barely audible. In (4), we used synonyms for each emotion for clarity, which were validated by native speakers from each language, following [10]. For each excerpt, we collected information about the listeners' preference, familiarity, and understanding of the lyrics. [2]

We selected a set of excerpts from the 4Q emotion dataset [28], which was previously annotated with categories in the four arousal-valence (AV) quadrants: Q1 (positive valence and arousal, A+V+), Q2 (positive arousal and negative valence, A+V-), Q3 (negative valence and arousal, A-V-), Q4 (negative arousal and positive valence, A-V+). In order to gather annotations on a larger scale, we asked participants to rate the excerpts with the following emotion categories: Q1 - *joyful activation*, *power*, *surprise*, Q2 - *anger*, *fear*, *tension*, Q3 - *bitterness*, *sadness*, Q4 - *peace*, *tenderness*, *transcendence*. These emotion adjectives were selected from the Geneva Emotion Music Scale (GEMS) [29] and a subset of basic emotions [30]. To select the target songs, we queried for these adjectives in

---

[2] Refer to Figure 1 - supp. mat. for the annotation interface.

English on the datasets' metadata. Since some words were not found, synonyms were used for certain emotions (e.g., fear - anguished). [3] From the resulting excerpts, we randomly selected two excerpts per emotion for a total of 22 fragments. All audio excerpts were normalized from -1 to 1 in amplitude, in order to balance the volume during playback. We collected additional audio features from Spotify API [31]: beats per minute, energy [0,1], and valence [0,1].

3.2 Emotion agreement analysis

Following [4, 10, 16], we used inter-rater reliability statistics to assess the agreement of the annotated data with respect to personal characteristics [32]. Researchers have found that Krippendorff's α is used increasingly to assess reliability in content analysis methodologies [33]. We employed Krippendorff's coefficient α defined as:

$$\alpha = 1 - \frac{D_o}{D_e} \qquad (1)$$

where D_o is the measure of observed disagreement:

$$D_o = \frac{1}{n} \sum_c \sum_k o_{ck} \cdot {}_{metric}\delta_{ck}^2 \qquad (2)$$

and D_e is a measure of the expected disagreement given chance:

$$D_e = \frac{1}{n(n-1)} \sum_c \sum_k n_c \cdot n_k \cdot {}_{metric}\delta_{ck}^2 \qquad (3)$$

The variables o_{ck}, n_c, n_k are the frequencies of values of observed coincidences of c and k values or ranks, and n is the total amount of paired $c - k$ values or ranks. Advantages of using α are: the suitability for any number of observers, handling of any type of metric (nominal, interval, ordinal), handling incomplete or missing data, and not requiring a minimum of sample size. When disagreement is absent ($D_o = 0$), there is perfect reliability ($\alpha = 1$). Conversely, when agreement and disagreement are a matter of chance ($D_e = D_o$), there is absence of reliability ($\alpha = 0$). Nevertheless, α could be smaller than zero if the sample size is too small or agreement below what would be expected by chance. According to [32], data with $\alpha \geq 0.8$ is considered to have good agreement and $0.4 \leq \alpha \leq 0.667$ shows fair agreement. Finally, ${}_{metric}\delta_{ck}$ represents the difference function: the squared difference between any two values or ranks c and k, depending on the data gathering approach (in our case, we use an ordinal metric).

To obtain annotation groupings, we defined positive and negative filters to classify ratings based on different user responses and characteristics. Considering the 5-point Likert response format, we define a positive filter by keeping ratings higher than 3 (neither agree or disagree) and a negative filter by keeping those less than 3. These filters were used to form groups using the users' response of preference, familiarity, and understanding for each excerpt. In the case of behavioral factors of music sophistication, we

specified positive and negative filters by grouping the annotations of participants with higher and lower scores than the population mean, respectively. We collected six behavioral factors, yet used the ones in **bold** to group ratings: Active Engagement, Perceptual Abilities, **Musical Training**, **Emotion Perception**, Singing Abilities, and **General sophistication**. [4]

In order to evaluate the collected annotations, we clustered group-based ratings in 2D and 3D spaces. Our intuition is that group-based annotations are more similar amongst them, and that clusters obtained with these annotations show less variance than those obtained with the original "ground truth". We generated a low-dimensional representation of all annotations with manifold learning, used one of the proposed filters to keep a group, clustered the resulting embeddings, and compared the resulting clusters with the original "ground truth" (i.e., four quadrants of emotion). We use Adjusted Rand Index (ARI) and Adjusted Mutual Information (AMI) for all filters as measures of quality of the clusters. We proceeded as follows: (1) standarization of the annotations, (2) categorical Principal Component Analysis (CATPCA), (3) manifold learning for dimensionality reduction, and (4) clustering of embeddings into the four quadrants of emotion using K-means (k-means++ initialization [34]). In (2), we use 10 components retaining 97.4% from variance. [5] In (3), we used the following algorithms: Multi-dimensional Scaling (MDS) [35], t-distributed Stochastic Neighbor Embedding (t-SNE) [36], and Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [37]. [6]

3.3 Emotion classification approach

Despite the low number of data instances (22), we trained support vector machine (SVM) classifiers to study how group-based annotations compare to annotations from all participants. Following relevant literature [4, 38, 39], we extracted the IS13 ComParE feature set with the openS-MILE toolbox [40]: 260 low-level features (mean and standard deviation of 65 emotionally-relevant acoustic descriptors, and their first order derivatives) with a frame size of 60 ms and a 10 ms hop size. These features are widely used as a benchmark for speech and music emotion recognition tasks [4]. Each feature vector was aggregated in segments of 5 seconds with a 75% overlap, resulting in 24 feature vectors per excerpt and 528 samples in total. We performed standarization over each feature and PCA for dimensionality reduction. After a Scree test, we selected 8 components that retained 65.7% of the variance. We performed a grid search for parameter optimization with the following settings (final parameters in bold): regularization parameter C [0.001, 0.01, 0.1, 1, **10**] and radial ba-

---

[4] Refer to Table 2 - supp. mat. for music sophistication results.

[5] Refer to Figure 2 - supp. mat. for an example of CATPCA.

[6] We used different settings for each algorithm (final parameters in bold): (1) MDS - **metric** and non-metric, iterations [300, 1000, **3000**], epsilon [**1e-1**, 1e-3, 1e-9, 1e-12], (2) t-SNE - perplexity [**3**, 5, 10, 30, 50, 100], learning rate [**200**, 500, 1000], number of iterations [**1000**, 3000], (3) UMAP - minimum distance [0.1, 0.25, 0.5, **0.8**, 0.99], number of neighbors [**10**, 30, 100, 200], metric [**Euclidean**, Cosine and Chebyshev].

| Configuration | Ratings | % | Q1 | | | | Q2 | | Q3 | | | Q4 | | |
			joy	surp.	pow.	ang.	fear	tens.	sad	bit.	pea.	tend.	tran.
All	27720	100.00%	0.393	0.064	0.273	0.335	0.169	0.253	0.340	0.215	0.357	0.396	0.062
By Preference (>3)	11275	40.67%	0.402	0.074	0.275	*0.264*	0.142	*0.171*	0.367	0.204	0.363	0.414	0.066
By Preference (<3)	8976	32.38%	*0.325*	0.036	0.267	0.368	0.158	**0.309**	0.308	0.191	0.352	0.384	0.049
By Familiarity (>3)	4477	16.15%	**0.445**	0.065	*0.199*	0.314	**0.223**	*0.158*	0.376	**0.284**	*0.266*	*0.297*	0.038
By Familiarity (<3)	21439	77.34%	0.319	0.047	0.275	0.329	0.145	0.270	0.304	0.173	0.385	0.414	0.070
By Understanding (>3)	13827	49.88%	0.428	0.074	0.276	*0.269*	0.160	*0.195*	0.361	0.250	0.324	0.365	0.046
By Understanding (<3)	9977	35.99%	*0.328*	0.044	0.263	0.361	0.156	0.299	0.308	*0.159*	0.361	0.391	0.070
By Music Training (>μ)	12320	44.44%	0.406	0.074	**0.327**	**0.407**	0.190	0.268	0.364	0.235	**0.424**	**0.456**	0.067
By Music Training (<μ)	15400	55.56%	0.381	0.052	0.231	*0.283*	0.151	0.236	0.318	0.194	*0.306*	0.348	0.054
By Emotion (>μ)	16500	59.52%	0.430	0.058	0.299	0.381	0.212	0.288	0.378	0.260	**0.411**	0.426	0.068
By Emotion (<μ)	11220	40.48%	0.343	0.067	0.230	*0.273*	*0.114*	0.206	*0.287*	*0.149*	*0.289*	0.351	0.046
By General Sophistication (>μ)	13640	49.21%	0.415	0.083	0.319	**0.400**	0.195	0.274	0.357	0.251	**0.441**	**0.466**	0.091
By General Sophistication (<μ)	14080	50.79%	0.371	0.043	0.225	*0.275*	0.147	0.226	0.327	0.180	*0.283*	*0.331*	0.038

Table 1. Krippendorff's α for each emotion for all participants filtered by preference, familiarity, lyrics comprehension, and music sophistication (positive and negative) using only music with lyrics (17 in English and 3 in Spanish).

sis function kernel with coefficient gamma [**0.001**, 0.01, 0.1, 1]. We report precision, recall, and F1-score using 5-fold cross-validation to evaluate the models. We test three possible "ground truths" per excerpt: (1) single-label annotations from the original metadata (MD), (2) multi-label annotations from all participants (All), and (3) multi-label annotations from participants belonging to a group (Filtered). In cases (2) and (3), we summarized ratings by taking the statistical mode of each emotion rating across participants. We employ the mode as some ratings showed a bimodal distribution and our annotations were categorical. Since the mode may result in multiple maximum values, we created multi-label annotations (i.e., an excerpt can have a mode of 4 for both *anger* and *tension*). Anonymized data and evaluation code are available online. [7]

4. RESULTS

4.1 Emotion annotation and agreement analysis

The participation was unbalanced regarding languages: a total of 126 (65 Male, 61 Female, $M = 34.12$ years, $SD = 11.75$) participants completed all tasks in our experiment from English ($n = 26$), Spanish ($n = 56$), Mandarin ($n = 27$), and German ($n = 17$) surveys. Listeners that wanted to participate in the survey but were not native to any of the languages were asked to take the English

[7] https://github.com/juansgomez87/agreement-emotion

Emotions	Eng. (26)	Spa. (56)	Man. (27)	Ger. (17)	All (126)
anger	**0.429**	*0.311*	0.367	**0.482**	0.364
bitter	**0.278**	0.209	0.155	**0.278**	0.202
fear	**0.241**	0.175	*0.091*	0.207	0.171
joy	*0.304*	**0.437**	*0.311*	**0.476**	0.372
peace	0.401	0.332	0.401	**0.438**	0.371
power	**0.379**	0.287	0.296	0.325	0.289
sad	0.330	0.343	0.279	**0.378**	0.326
surprise	0.041	0.055	0.068	**0.218**	0.075
tender	0.444	*0.314*	**0.452**	**0.581**	0.396
tension	0.264	0.324	0.282	0.323	0.296
transc.	0.080	0.049	0.083	*-0.012*	0.057

Table 2. Krippendorff's α for each emotion across different languages.

version ($n = 15$). Since the surveys were made available through different channels, listeners were asked to state the country in which they spent the formative years of childhood and youth [8]. We evaluated outliers for every musical fragment, finding that the participants that systematically annotated outside the interquartile range (Q1-Q3) did so at most 10.33% of the time. Hence, we decided to keep all ratings for analysis. Median and Kruskal-Wallis H tests showed that there was a statistically significant difference in the ratings of emotions between raters from the surveys of each language ($p < 0.01$). [9] Concretely, the ratings of *anger, bitterness, fear, sadness, surprise, tenderness, tension*, and *transcendence* have different distributions across the surveys. This suggests that emotion significance varies across cultures and languages as hypothesized in our study. Emotion adjectives might have varied meanings and associations across different languages and cultures as studied by [41], but results should be validated with a higher amount of participants and excerpts.

Results from the agreement analysis are presented in Tables 1 and 2, and appear **bold** when the agreement of the emotion is higher than 0.05 as to the agreement measured across all participants. Conversely, the text in *italic* indicates a difference less than -0.05, following [10]. Table 1 shows agreement over all participants that belong to a certain group (see Configuration column). Hence, it contains information about the number of ratings for the corresponding filters (groups). The music selection contained 20 songs with lyrics (17 in English and 3 in Spanish), thus agreement was analyzed only for this subset for lyrics comprehension to be meaningful. We find two tendencies for groups assembled with preference, familiarity, and lyrics comprehension: (1) positive filters will result in higher agreement with respect to all ratings for emotions such as *joy, surprise, power, sadness*, and *bitterness*; (2) positive filters will result in lower agreement for emotions such as *anger, fear, tension, peace, tenderness* and *transcendence*. Interestingly, emotions in (1) belong to

[8] Subjects participated from the following countries: (1) *Spanish* - Bolivia, Colombia, Ecuador, Perú, Spain, Uruguay, (2) *Mandarin* - Mainland China, Taiwan, (3) *German* - Austria, Germany, Switzerland, and (4) *English* - Australia, Bulgaria, Belgium, Brazil, France, Greece, India, Italy, Portugal, Romania, United Kingdom, United States.

[9] Refer to Table 3 - supp. mat. for multiple pairwise test results.

quadrants Q1 (A+V+) and Q3 (A-V-), while emotions in (2) belong to Q2 (A+V-) and Q4 (A-V+). Regarding musical sophistication, we find a consistent trend in groups assembled with musical training, emotion perception and general sophistication: positive filters will result in higher agreement in all filters with respect to all ratings. Table 2 shows agreement over all participants from each survey and exposes low agreement over certain emotions: *bitterness*, *fear*, *power*, *surprise*, and *transcendence*. On the other hand, a higher agreement is reached for emotions such as *anger*, *joy*, *peace*, *sadness*, and *tenderness*. The results confirm that participants of different surveys show significant differences in the emotions perceived from music. Additionally, we found positive linear correlations, as obtained by Pearson's coefficient, between *anger*, *bitterness*, *fear*, and *tension*; *peace* and *tenderness*; *joy*, *power*, and *surprise*; *sadness* and *bitterness*. [10]

4.2 Implications for MER models
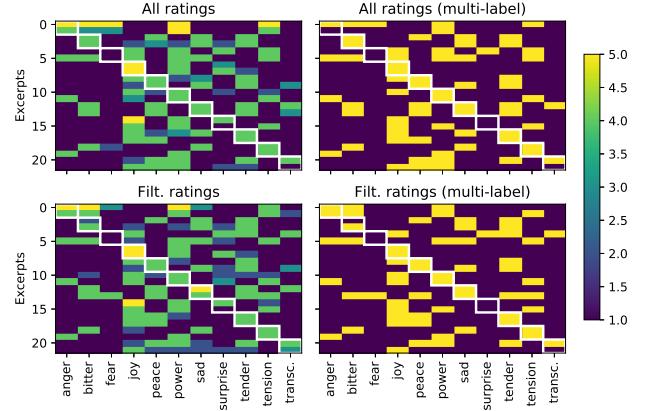
Table 3 shows the clustering evaluation when comparing clusters from manifold learning representations with the original "ground truth". Scores are reported in **bold** when they are higher than the scores obtained without filters (All). In every case, positive filters result in improved clusterability (particularly when selecting participants with high scores for music sophistication). MDS and UMAP appear to separate the data better than t-SNE before performing K-means. As a baseline, applying K-means on the raw data shows that manifold learning techniques can be useful to find similarities between group-based ratings. We argue that using manifold learning previous to clustering extracts possible similarities across annotations that belong to a given AV quadrant, yielding annotation embeddings that are easier to cluster. [11]

	MDS + K-Means		t-SNE + K-Means		UMAP + K-Means		K-Means Raw data	
	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI
All	0.248	0.255	0.191	0.214	0.244	0.257	0.221	0.229
Pref. (>3)	**0.267**	**0.282**	**0.195**	**0.230**	**0.252**	**0.271**	0.197	0.223
Pref. (<3)	0.234	0.234	0.184	0.206	0.244	0.248	0.215	0.234
Fam. (>3)	**0.301**	**0.319**	**0.252**	**0.264**	**0.295**	**0.305**	**0.223**	**0.251**
Fam. (<3)	0.231	0.241	0.183	0.202	0.236	0.245	0.218	0.220
Und. (>3)	**0.253**	**0.260**	**0.225**	**0.239**	**0.250**	**0.263**	0.213	0.230
Und. (<3)	0.235	0.250	0.187	0.210	0.244	0.243	0.209	0.219
MT (>μ)	**0.283**	**0.290**	**0.210**	**0.242**	**0.302**	**0.309**	**0.286**	**0.289**
MT (<μ)	0.214	0.227	0.180	0.203	0.207	0.220	0.172	0.185
Emo. (>μ)	**0.303**	**0.305**	**0.216**	**0.242**	**0.297**	**0.309**	**0.262**	**0.271**
Emo. (<μ)	0.186	0.199	0.159	0.176	0.179	0.196	0.151	0.162
GS (>μ)	**0.303**	**0.303**	**0.221**	**0.245**	**0.291**	**0.303**	**0.307**	**0.308**
GS (<μ)	0.202	0.217	0.169	0.193	0.202	0.219	0.155	0.165

Table 3. Clustering metrics for all filters and manifold learning algorithms. MT refers to Musical Training, Emo. to Emotion Perception, GS to General Sophistication. The last column shows clustering results on the raw data.

An example of the group-based, multi-label annotations produced by using the positive understanding filter can be seen in Figure 1. This plot compares different "ground

truths" for the data according to the collected ratings and a given filter that we used to train our classifiers. For example, excerpt 0 (originally labeled as *anger*) is also labeled with *bitterness*, *fear*, *power*, and *tension* when considering our annotations (top-right plot). In contrast, excerpt 1 is labeled as *power* when considering all ratings, but labeled as *anger*, *bitter*, and *power* when considering the filter (comparison of top- and bottom-right plots).

Figure 1. Example of annotations using *All* ratings (top row) and the *positive understanding* filter (bottom row). White rectangles highlight the original annotation from the metadata (MD - single-label). For every plot, rows represent one of the 22 excerpts and columns represent an emotion. The plots on the left show the mode over the participants for each emotion. The plots on the right show selected multi-labels for each excerpt used for classification. The color bar represents the 5-point Likert scale.

Classification experiments are reported in Table 4, including precision (P), recall (R) and F1-Score (F) of classifiers trained on annotations with different filters and 5-fold cross-validation. We compare three settings: original labels from the metadata (MD - single-label), annotations collected from all raters (All - multi-label), and annotations from selected raters with respect to the defined groups (Filter - multi-label). Comparisons are presented by subtracting the mean performance scores of classifiers trained on two annotation scenarios. For example, All - MD refers to the comparison between a classifier trained on all annotations and one trained on the original metadata. Scores are reported in **bold** when the difference is greater than 0.05 and in *italic* when it is less than -0.05. We also report the Jaccard coefficient (JC), bounded from [0,1], to estimate the similarity of the compared ratings in each case [42]. JC shows that the similarity from the original "ground truth" and the collected annotations is low (i.e., All - MD and Filt. - MD), which is expected since we compare single- and multi-label annotations. Interestingly in these two cases, classification results from the collected annotations (multi-label) provide consistent mean improvements of 15.01 percent points in precision and 11.8 in F1-scores with respect to classifiers trained on the original "ground truth" (single-label). We argue that improvements are due to the correlations between tags that are exploited in the multi-label

---

[10] Refer to Table 4 - supp. mat. for full correlation analysis.
[11] Refer to Figures 3-5 - supp. mat. for visual sample embeddings.

case. In the case of comparing filtered and all collected annotations (i.e., Filt. - All), we find that a classifier trained on the group-based annotations generated from the positive understanding filter consistently results in better classification for this dataset. In this case, both classifiers were trained on multi-label annotations. In contrast, other filtered group-based annotations result in very similar performance as with all annotations, confirming previous findings from Yang [7].

Comparison	Filter	JC	Δ P	Δ R	Δ F
All - MD	-	0.287	**0.171**	**0.056**	**0.124**
Filt. - MD	Pref. (>3)	0.286	**0.210**	**0.061**	**0.134**
	Pref. (<3)	0.326	**0.055**	-0.086	-0.008
	Fam. (>3)	0.236	**0.171**	**0.069**	**0.150**
	Fam. (<3)	0.297	**0.159**	-0.026	0.038
	Und. (>3)	0.322	**0.228**	**0.130**	**0.179**
	Und. (<3)	0.284	**0.051**	-0.093	-0.017
	MT (>μ)	0.253	**0.198**	0.008	**0.100**
	MT (<μ)	0.314	**0.084**	-0.056	0.019
	Emo. (>μ)	0.314	**0.144**	0.015	**0.091**
	Emo. (<μ)	0.255	**0.121**	-0.025	0.041
	GS (>μ)	0.308	**0.133**	-0.037	0.037
	GS (<μ)	0.282	**0.078**	-0.002	**0.053**
Filt. - All	Pref. (>3)	0.718	0.039	0.005	0.010
	Pref. (<3)	0.639	-0.116	-0.142	-0.132
	Fam. (>3)	0.547	0.001	0.013	0.026
	Fam. (<3)	0.851	-0.011	-0.082	-0.086
	Und. (>3)	0.783	**0.057**	**0.074**	**0.056**
	Und. (<3)	0.697	-0.120	-0.149	-0.141
	MT (>μ)	0.861	0.027	-0.048	-0.024
	MT (<μ)	0.794	-0.087	-0.113	-0.105
	Emo. (>μ)	0.898	-0.026	-0.041	-0.033
	Emo. (<μ)	0.767	-0.050	-0.081	-0.083
	GS (>μ)	0.842	-0.038	-0.093	-0.087
	GS (<μ)	0.844	-0.093	-0.058	-0.070

Table 4. Performance comparison of models trained with different "ground truths". We report the difference of mean Precision (P), Recall (R), and F1-Score (F1). MD refers to metadata (single-label) and JC refers to Jaccard coefficient.

5. DISCUSSION AND CONCLUSION

In this paper, we systematically evaluated agreement of categorical annotations of emotions in 22 fragments of music. We characterized listeners by language, preference, familiarity, lyrics comprehension, and music sophistication.

With respect to *RQ1* - Do personal characteristics and mother tongue have an influence on the the annotation of perceived emotions for listeners? - our main finding is that there are substantial differences in the annotations of our surveys. In fact, the collected annotations show different distributions in the majority of emotions, and only the distributions of *joy* and *peace* appeared to be similar across languages. This relates to recent research on "colexification" of semantically related emotion concepts, where researchers found evidence that the relationship between emotion words varies significantly across languages [41]. Our results have also confirmed that certain basic emotions have higher agreement, while complex ones show the opposite. However, agreement in our experiment ap-

pears to be lower than values reported in [4] and similar to [10]. Our results advocate for taking into account diverse languages while gathering annotations and reducing the number of categories when dealing with cross-cultural MER models (i.e., four quadrants in AV space). Our findings suggest that preference, familiarity, and lyrics comprehension increase agreement for emotions corresponding to quadrants Q1 and Q3, and decreases it for quadrants Q2 and Q4. Regarding music sophistication, positive filters result in higher agreement for all emotions, conflicting with results from Schedl et al. [10]. We argue that in the case of classical music, music experts could tend to disagree more on subtle musical expression cues, while pop and rock music have stronger indicators for emotion (tempo, musical instruments, and meaning of lyrics). This has given us new understanding of the effect of language and lyrics comprehension: in the case of Q1 (A+V+) and Q3 (A-V-) higher agreement is found, contrasted to Q2 (A+V-) and Q4 (A-V+) where dimensions have opposite signs.

As to *RQ2* - Can this information be used to improve MER algorithms? - we find that models trained on multi-label annotations (all and filtered) will consistently show higher precision and F1-score than models trained with the original annotations from metadata (single-label) for this particular dataset. Our results show increments up to 18 percentage points in F1-Scores, when comparing single- and multi-labeled "ground truths". As to models trained with all collected annotations and our proposed filters (Filt. - All), we only find consistent gains for the case of positive lyrics comprehension. Nonetheless, further research is needed in order to confirm these findings. We propose four recommendations when creating datasets for MER algorithms with cross-cultural applications: (1) previous selection of listeners' population and music style have a deep impact on the agreement of annotations - good understanding of the population of annotators is required; (2) inter-rater reliability is crucial to define categories - agreement should be reported and analyzed; (3) group-based annotations can lead to improved agreement - models should be evaluated with both average ratings and group-based ratings; and (4) selecting annotators that are proficient in the language sung in the music may result advantageous - understanding the semantic content of lyrics could help increase the agreement in annotations and possibly lead to improving models. As future work, we consider balancing the styles with respect to different languages. It is also arguable that pop and rock are in fact musically homogeneous, since several variations across the world show different ways of conveying emotions (e.g., Hindi pop). Lastly, the experiment could have biased responses when asking for lyrics comprehension, forcing the participants' attention on lyrics and compromising ecological validity. Different studies regarding lyrics intelligibility should be taken into account in future research, such as [24, 43, 44]. Nevertheless, our study attempts to dispute Henry Wadsworth Longfellow's famous quote - is in fact music the universal language of mankind?

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] P. Juslin, "Music and emotion: Seven questions, seven answers," *Music and the mind: Essays in honour of John Sloboda*, pp. 113–35, 2011.

[2] T. Eerola and J. K. Vuoskoski, "A Review of Music and Emotion Studies: Approaches, Emotion Models, and Stimuli," *Music Perception: An Interdisciplinary Journal*, vol. 30, no. 3, pp. 307–340, 2013.

[3] P. N. Juslin, *Musical Emotions Explained*, 1st ed. Oxford: Oxford University Press, 2019.

[4] A. Aljanaki, Y.-H. Yang, and M. Soleymani, "Developing a benchmark for emotional analysis of music," *PLoS 1*, pp. 1–22, 2017.

[5] S. Chowdhury, A. Vall, V. Haunschmid, and G. Widmer, "Towards Explainable Music Emotion Recognition: The Route via Mid-level Features," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 237–243.

[6] K. W. Cheuk, Y.-J. Luo, G. Roig, and D. Herremans, "Regression-based Music Emotion Prediction using Triplet Neural Networks," in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, 2020.

[7] Y.-H. Yang and H. H. Chen, *Music Emotion Recognition*. CRC Press, 2011.

[8] E. B. Lange and K. Frieler, "Challenges and Opportunities of Predicting Musical Emotions with Perceptual and Automatized Features," *Music Perception: An Interdisciplinary Journal*, vol. 36, no. 2, pp. 217–242, 2018.

[9] A. Flexer and T. Grill, "The Problem of Limited Inter-rater Agreement in Modelling Music Similarity," *Journal of New Music Research*, vol. 45, no. 3, pp. 239–251, 2016.

[10] M. Schedl, E. Gómez, E. S. Trent, M. Tkalcic, H. Eghbal-Zadeh, and A. Martorell, "On the Interrelation Between Listener Characteristics and the Perception of Emotions in Classical Orchestra Music," *IEEE TRANSACTIONS ON AFFECTIVE COMPUTING*, vol. 9, no. 4, pp. 507–525, 2018.

[11] A. Flexer and T. Lallai, "Can we increase inter- and intra-rater agreement in modeling general music similarity?" in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 494–500. [Online]. Available: https://www.music-ir.org/mirex/wiki/2006:

[12] P. N. Juslin, *Handbook of Music and Emotion: Theory, Research, Applications*. Oxford: Oxford University Press, 2010.

[13] E. Bigand and J.-J. Aucouturier, "Seven problems that keep MIR from attracting the interest of cognition and neuroscience," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 483–497, 2013.

[14] B. L. Sturm, "Evaluating music emotion recognition: Lessons from music genre recognition?" in *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, San Jose, USA, 2013, pp. 1–6.

[15] ——, "A Simple Method to Determine if a Music Information Retrieval System is a Horse," *IEEE TRANSACTIONS ON MULTIMEDIA*, vol. 16, no. 6, 2014.

[16] H. V. Koops, W. Bas De Haas, J. A. Burgoyne, J. Bransen, A. Kent-Muller, and A. Volk, "Annotator subjectivity in harmony annotations of popular music," *Journal of New Music Research*, vol. 48, no. 3, pp. 232–252, 2019. [Online]. Available: https://www.tandfonline.com/action/journalInformation?journalCode=nnmr20

[17] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon, "Selective sampling for beat tracking evaluation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 9, pp. 2539–2548, 2012.

[18] C. L. Krumhansl, "An exploratory study of musical emotions and psychophysiology," *Canadian Journal of Experimental Psychology*, vol. 51, no. 4, pp. 336–353, 1997.

[19] L. B. Meyer, *Emotion and meaning in music*. Chicago University Press, 1956.

[20] P. Kivy, *Music alone: Reflections on a purely musical experience*. Cornell University Press, 1990.

[21] M. Zentner, D. Grandjean, and K. R. Scherer, "Emotions Evoked by the Sound of Music: Characterization, Classification, and Measurement," *Emotion*, vol. 8, no. 4, pp. 494–521, 2008.

[22] K. Hevner, "Experimental studies of the elements of expression in music," *American Journal of Psychology*, vol. 48, no. 2, pp. 246–268, 1936.

[23] J. A. Russell, "A Circumplex Model of Affect," *Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[24] D. Vidas, R. Calligeros, N. L. Nelson, and G. A. Dingle, "Development of emotion recognition in popular music and vocal bursts," *Cognition and Emotion*, pp. 1–14, 2019.

[25] Y.-H. Yang, Y.-F. Su, Y.-C. Lin, and H. H. Chen, "Music Emotion Recognition: The Role of Individuality," National Taiwan University, Tech. Rep., 2007.

[26] G. Stoet, "Psytoolkit: A novel web-based method for running online questionnaires and reaction-time experiments," *Teaching of Psychology*, vol. 44, no. 1, pp. 24–31, 2017.

[27] D. Müllensiefen, B. Gingras, J. Musil, and L. Stewart, "The Musicality of Non-Musicians: An Index for Assessing Musical Sophistication in the General Population," *PLoS ONE*, vol. 9, no. 2, p. 89642, 2014.

[28] R. Panda, R. M. Rui, and P. Paiva, "Musical texture and expressivity features for music emotion recognition," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[29] K. R. Scherer, "Expression of Emotion in Voice and Music," *Journal of Voice*, vol. 9, no. 3, pp. 235–248, 1995.

[30] P. Ekman, "An argument for basic emotions," *Cognition and Emotion*, vol. 6, no. 3-4, pp. 169–200, 1992.

[31] Spotify, "Get audio features for a track," 2020, https://developer.spotify.com/console/get-audio-features-track/.

[32] K. H. Krippendorff, *Content Analysis: An Introduction to Its Methodology*, 2nd ed. SAGE Publications, 2004.

[33] J. Lovejoy, B. R. Watson, S. Lacy, and D. Riffe, "Three Decades of Reliability in Communication Content Analyses," *Journalism and Mass Communication Quarterly*, vol. 93, no. 4, pp. 1135–1159, 2016.

[34] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. USA: Society for Industrial and Applied Mathematics, 2007, p. 1027–1035.

[35] I. Borg and P. J. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. Springer, 1997.

[36] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[37] L. McInnes, J. Healy, N. Saul, and L. Großberger, "Umap: Uniform manifold approximation and projection." *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[38] F. Weninger, F. Eyben, B. W. Schuller, M. Mortillaro, K. R. Scherer, and J. Krajewski, "On the acoustics of emotion in audio: what speech, music, and sound have in common," *Frontiers in Psychology*, vol. 4, pp. 1–12, 2013.

[39] E. Coutinho and B. Schuller, "Shared acoustic codes underlie emotional communication in music and speech - evidence from deep transfer learning," *PLoS ONE*, vol. 12, no. 6, 2017.

[40] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: The munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM International Conference on Multimedia*, ser. MM '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1459–1462. [Online]. Available: https://doi.org/10.1145/1873951.1874246

[41] J. C. Jackson, J. Watts, T. R. Henry, J.-M. List, R. Forkel, P. J. Mucha, S. J. Greenhill, R. D. Gray, and K. A. Lindquist, "Emotion semantics show both cultural variation and universal structure," *Science*, vol. 1522, no. December, pp. 1517–1522, 2019.

[42] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[43] N. Condit-Schultz and D. Huron, "Catching the lyrics: intelligibility in twelve song genres," *Music Perception*, vol. 32, no. 5, pp. 470–483, 2015.

[44] ——, "Word Intelligibility in Multi-voice Singing: The Influence of Chorus Size," *Journal of Voice*, vol. 31, no. 1, pp. 121.e1–121.e8, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.jvoice.2016.02.011

"BUTTER LYRICS OVER HOMINY GRIT"[†]: COMPARING AUDIO AND PSYCHOLOGY-BASED TEXT FEATURES IN MIR TASKS

Jaehun Kim[1] [*]   **Andrew M. Demetriou**[1][*]   **Sandy Manolios**[1]
M. Stella Tavella[2]   **Cynthia C. S. Liem**[1]

[1] Delft University of Technology, Netherlands

[2] Musixmatch, Bologna, Italy

`J.H.Kim@tudelft.nl`

`A.M.Demetriou@tudelft.nl`

ABSTRACT

Psychology research has shown that song lyrics are a rich source of data, yet they are often overlooked in the field of MIR compared to audio. In this paper, we provide an initial assessment of the usefulness of features drawn from lyrics for various fields, such as MIR and Music Psychology. To do so, we assess the performance of lyric-based text features on 3 MIR tasks, in comparison to audio features. Specifically, we draw sets of text features from the field of Natural Language Processing and Psychology. Further, we estimate their effect on performance while statistically controlling for the effect of audio features, by using a hierarchical regression statistical model. Lyric-based features show a small but statistically significant effect, that anticipates further research. Implications and directions for future studies are discussed.

1. INTRODUCTION

Popular Western music very often contains lyrics. Social science research has shown informative relationships between popular songs and their lyrical content: e.g., country music lyrics rarely include political concepts [1], songs with more typical [2] and more negative [3] lyrics appear to be more successful, and the psychological content of song lyrics appears to correlate with cultural changes in psychological traits [4]. As for music consumption, lyrics have also been shown to be a salient component of music in the minds of listeners [5]. Furthermore, [6] showed that patients are more likely to choose music with lyrics

---

[*] Authors contributed equally to the work.

[†] Quoted words are lyrics written by Clifford Smith, from the song "The What", by the Notorious B.I.G. featuring Methodman, on the album "Ready to Die", released in 1994.

when participating in music-based pain reduction interventions; [7] showed that lyrics enhance self reported emotional responses to music, although melody had an overall larger effect, and [8] showed a number of additional brain regions were active during the listening of sad music with lyrics, vs. sad music without lyrics.

In the Music Information Retrieval (MIR) field, some interest for lyrics and how they can be used to improve MIR tasks has been shown. Popular uses of lyrics for MIR tasks consider mood classification [9–12], genre classification [13, 14] and topic detection for indexing and browsing [15, 16]. [17] also proposed a metric to assess the novelty of lyrics, and suggested that novelty can play a role in music preference.

From these findings, one can conclude that lyrics are a rich data source. Although MIR interests have historically focused more on audio, lyrics information may fruitfully be leveraged for various MIR tasks. Still, there are many possible ways to extract information from lyrics text, and it is an open question what information extraction procedure will turn out most fruitful. To gain more insight into this, we present a study investigating several textual feature sets. In shaping these sets—acknowledging potential value of the topic for social science research—we are inspired by the way text analysis has been performed in the Psychology domain, and draw several of our extractors from prior work in that field. We will assess the performance of these textual feature sets on 3 common MIR tasks, and will statistically control for the effect of each chosen feature set, including an audio feature set for comparison. Our analysis will be performed on a large dataset from the online Musixmatch lyrics catalogue.
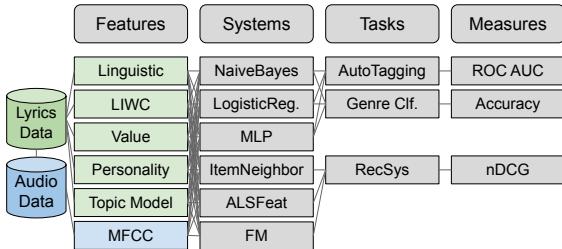
In the remainder of the paper, in Section 2, we discuss relevant previous work on text information extraction in the Psychology literature. Section 3 will subsequently explain our research design, after which Section 4 discusses the feature sets we used. Section 5 describes the data collection and pre-processing procedures, after which Section 6 details the experimental design. Section 7 justifies our chosen analytical strategy, followed by a presentation of results in Section 8 and the conclusion in Section 10.

Figure 1. Overview of the experimental pipeline.

2. RELATED WORK

The field of Psychology has long pondered the importance of the words people choose to use, and how this reflects their individual differences [18]. The features we use in present work are primarily inspired by two prior lines of work in which Natural Language Processing (NLP) techniques were applied in psychology research: one employing closed-vocabulary lexicon approaches, the other employing open vocabulary approaches. Firstly, [19] used NLP techniques to derive estimates of personality for music genres. Specifically, they created a lexicon (a meaningful group of words) from psychology research that described personality dimensions, as well as a corpus of lyrics, separated into music genres. They then computed the similarity between the lyrics of music genres and the groups of personality dimension words, and considered this result to be an estimate of the personality dimension represented in the lyrics of each genre. Lexicon-based approaches have generally been popular, also thanks to the release of the Linguistic Inquiry Word Count (LIWC) lexicon-based software [20]; e.g., in the context of lyrics, [21] used it to examine psychological distress in the lyrics of musicians that committed suicide vs. those who had not.

Secondly, [22] demonstrated the usefulness of an open vocabulary approach vs. a lexicon approach while examining personality in the context of online social networks. Although lexicons are carefully curated and meaningful, they are also time-consuming to create and context-specific. In contrast, data-driven techniques can automatically estimate latent topics from groups of words that tend do appear together. [22] showed relationships between personality scores and automatically extracted latent topics. Further, they showed that the open vocabulary approach may have stronger correlations to self-reported personality scores than the closed-vocabulary lexicon approaches.

3. RESEARCH DESIGN

In this study, we seek to examine the relative importance of lyric-based text features—especially features drawn from psychology research— for various popular MIR tasks. We wish to compare this importance to that of conventional audio based features.

An overview of our experimental pipeline is given in Figure 1. Various feature sets will feed into various systems, that are appropriate for various MIR machine learn-

ing tasks. We employ a full-factorial experimental design for feature sets, tasks, and the systems attached to each task, which means we research all the possible combinations of those factors. For each combination, we will employ the traditional train-validation-test machine learning setup. Performance results on the test sets will feed into our statistical analysis, where we will explicitly control for the effect of each of the feature sets.

4. FEATURE SETS

In this work, we will consider 5 lyric-based text feature sets and an audio-based feature set. More details are given in the following subsections; a summary of the dimensionalities of all feature sets is given in Table 1.

4.1 Linguistic Features

As baseline textual features for this study, we first extract several simple *linguistic* features:

- *NumWords:* the number of words included in the lyrics text.
- *NumUniqueWords:* the number of unique words in the lyrics text.
- *NumStopWords:* the number of stop words in the lyrics text [1].
- *NumRareWords* the number of words that appeared in less than 5 lyrics.
- *NumCommonWords* the number of words extremely commonly used within a lyrics corpus. We set the threshold as the 30% percentile of the document frequency of words.

Along with the absolute number, we also compute the ratio over the total number of words for each lyrics text.

4.2 Topic Modeling

As a more advanced feature extraction technique, we employ probabilistic Latent Semantic Analysis (pLSA) [24] for *topic* modeling. We treat each of the lyric texts as a document, and will take the found topic distribution for a given document as the document feature. We chose the number of topics $K = 25$, which maximizes validation log-perplexity. Taking advantage of the unsupervised learning setup, we use the total pool of songs to setup the training-validation-test split.

4.3 LIWC

Linguistic Inquiry Word Count (LIWC) is a software package built on a lexicon that has been validated for text analysis in psychological studies [20]. It uses a curated lexicon, separated into 73 categories (e.g., the category 'Social Processes' includes references to family and friends). The software outputs the counts of words in a given text for each of the 73 categories. We employ the latest LIWC, released in 2015.

---

[1] As we will focus on English lyrics in this study, we used the English stop words corpus from the Natural Language Toolkit (NLTK) [23]

4.4 Psychology Inventory Scores

We will consider two more feature sets, inspired by psychology inventory scores: a feature set focusing on *personality* and a feature set focusing on *values*. In both cases, we will use lexicons from literature. However, rather than performing a word count as was done in LIWC, we will use more contemporary NLP techniques based on word embeddings.

Contemporary personality theory is derived from lexical studies: it has been suggested that meaningful individual psychological differences between people are captured in the adjectives that describe people [25]. Although the number of meaningful clusters of adjectives (called Personality Dimensions) is under debate, the OCEAN or Big-Five model is often used. It is composed of 5 traits : Openness to Experience, Conscientiousness, Extroversion, Agreeableness and Neuroticism [25]. Our *personality* feature set consists of 2 word clusters per dimension, comprised of words representing positive and negative aspects for each personality dimension, derived from prior research [26].

Personal *values* are another important component of identity, though less studied. They are stable over time and represent who people want to be, targeting the most important things for them in life at the most abstract level. The traditional way to obtain people's personal values is through questionnaires, but recent works focused on NLP techniques to extract them from text [27–29]. In our work, we used the value inventory and lexicon from [28].

Both for the *personality* and *values* feature sets, we will exploit the `word2vec` model [30] to approximate distances between lyrics and the various inventory categories in the feature sets. For this, we use the model pre-trained on the Google News dataset [2]. The average distance score $s_{d,c}$ for each lyric text d, and category c is computed by taking the average cosine distance between the words belonging to the lyrics and the categories, respectively:

$$s_{d,c} = \frac{1}{|\mathcal{W}_d||\mathcal{W}_c|} \sum_{n \in \mathcal{W}_d} \sum_{m \in \mathcal{W}_c} \frac{\langle \mathbf{v}_n, \mathbf{v}_m \rangle}{||\mathbf{v}_n|| \cdot ||\mathbf{v}_m||} \quad (1)$$

where $\mathcal{W}_d$ and $\mathcal{W}_c$ represent the set of words belonging to the lyrics text d and the category c. v_n and v_m denote the pre-trained word vectors corresponding to word n in the lyrics and word m in the category, respectively.

4.5 MFCC

Finally, we employ a set of audio features based on the Mel-Frequency Cepstral Coefficients (MFCC). We include these, such that the effect of the lyric-based text features can be compared to a commonly used feature set from the primary modality of interest in many MIR tasks. Specifically, we adopt the feature computation introduced in [31] with 40 mel bins.

Feature Set	Dimensions
Audio	240
LIWC	73
Values	49
Topics	25
Personality	10
Linguistic	9

Table 1. Number of dimensions per feature set

5. DATA COLLECTION

We analyzed the lyrics contained in the Musixmatch dataset [3], which is the official lyrics metadata selection integrated in the Million Song Dataset (MSD) [32], a collection of relevant data and metadata for one million popular contemporary songs. Musixmatch is a lyrics and music language platform. The Musixmatch community drives the content production by adding, correcting, syncing and translating lyrics of songs. The process of lyrics quality verification involves several steps, including spam detection, formatting, spelling and translation checking. These steps are accomplished by the use of both artificial intelligence and machine learning models. In addition, they are manually verified by more than 2000 Curators worldwide, and a local team of Musixmatch Editors, who are native speakers in different languages.

The data used for the purpose of this project consists of $182,808$ lyrics, plus relevant metadata such as the unique identifier, artist and title. The data encompasses $20,219$ unique artists over various genres of music.

5.1 Preprocessing

For the given lyrics dataset, we consider the following preprocessing steps: the sentence strings are 1) tokenized and 2) lemmatized, followed by 3) stop-words filtering and 4) filtering extremely rare and extremely common words (see Section 4.1). Finally, we filter out non-English lyrics by a filtering process using the topic modeling. More precisely, we fit the topic model to detect whether the topics contain non-English words above a certain threshold. Songs that mostly load on non-English topics are removed.

6. EXPERIMENT

6.1 Tasks & Systems

As shown in Figure 1, to assess the lyrics feature set, we consider 3 popular MIR machine learning tasks; for each of these, we use 3 different commonly used types of systems, and a task-specific performance measure is considered, as detailed below.

6.1.1 Music Genre Classification

Music Genre Classification (MGC) is a multi-class classification problem. Typically, a set of music genres is given as the classes, and music audio content or features are used as the observations. In this study, we examine 3 machine

---

[2] https://code.google.com/archive/p/word2vec/

[3] http://millionsongdataset.com/musixmatch/

learning based systems: *Gaussian Naive Bayes* (GNV), *Logistic Regression* (LR) and the *Multi-Layer Perceptron* (MLP). For performance quantification, we opt for *classification accuracy*.

For this task, we use the data in the intersection between our lyrics database and the part of the MSD for which the music genre mapping introduced in [33] can be made. By choosing the intersection with the MSD, our audio features can be extracted from the MSD preview audio excerpts. Due to genre label availability, this leads to $67,719$ songs being used in this task.

6.1.2 Music Auto-Tagging

Music Auto-Tagging (MAT) is often formulated as a multi-label classification problem in which multiple positive labels may exist for one input music observation. We used the same set of systems as in the MGC task [4]. Again, we cross-match to the MSD, now also considering MSD's LastFM social tags. Similarly to [31], we choose to focus on the 50 most frequent tags from the dataset. The *Area Under Curve - Receiver Operating Characteristic* (AUC-ROC) is used as the performance measure, which will be referred to as AUC^{song} for the rest of this paper [5]. Due to tagging label availability, $137,095$ songs are used under this task.

6.1.3 Music Recommendation

Finally, Music Recommendation (MR) is considered for a user-related retrieval task. In particular, we consider a cold-start scenario, in which a batch of songs is newly introduced to the market, and required to be recommended to users. Due to the lack of previous interaction history, in such a scenario, a model will be maximally dependent on item attributes. As this is a substantially different type of task than the previous classification tasks, a different set of the systems common to the recommender systems field is used. *Item Nearest Neighbor* (INN) is a memory-based collaborative filtering method, which recommends the items closest to those that the user had consumed. We employ the feature vector introduced in Section 4 to compute the distance between entities using the cosine distance. We also use the *Feature-augmented Matrix Factorization* (FMF) [34] method, as well as the *Factorization Machine* [35] (FM). These models are more sophisticated collaborative recommenders, which also are capable of exploiting item attributes. The systems are developed and evaluated using the MSD-Echonest dataset [32]. Due to limits on available computational resources, we exploit a densified subset with $96,551$ users and $66,850$ songs from the initial song pool with the lyrics [6]. Finally, the binarized *normalized Discounted Cumulative Gain* (nDCG) is

considered as performance measure, for the top-100 songs recommended.

6.2 Task Simulation Setup

All MIR tasks above are machine learning tasks, but the systems and data we choose to use for them did not yet exist in a real-life system. Therefore, we ran the machine learning procedures to initiate them. For this, for each task, we randomly split the available song data into *training/validation/test* subsets by a ratio of $8 : 1 : 1$. Each model is trained using the *training* set and evaluated on the *validation* set to tune the hyper-parameters. Once the optimal hyper-parameters are found, final performance is measured on the the the *test* set.

For MLP and FMF, which have more than one hyper-parameter, automatic hyper-parameter tuning is conducted through a Bayesian approach, using the Gaussian Process [7] [8]. Every search process iterates through 50 training-validation procedures to reach the optimal point. For the MGC and MAT tasks, the hyper-parameters are searched at every trial, while in the MR task, the search process runs only once and is used for all the other trials.

7. ANALYTIC STRATEGY

We wish to assess the usefulness of each of the feature sets for the 3 MIR tasks. Therefore, the resulting performance score from each trial run in our experimental setup (see Section 3) forms the measurement that is our outcome variable of interest. We seek to estimate the relative contribution of each feature set, while statistically controlling for the contribution of all other variables in the analysis. In addition, we assess whether feature sets perform better or worse, depending on the task.

Our data has a nested structure. Specifically, we might say that our systems are nested within the tasks: each task is likely to influence the score, as will the underlying systems that were used for each task. Further, not all systems were used in all tasks. To account for this structure, we employed hierarchical regression models which allow for the modeling of variances of nested data.

The typical example for this category of models is the task of modeling the standardized test scores of various students within various schools. Test scores may be due to the performance of the student, but the school itself may also influence the scores. In this case, the students are said to be nested within the school. If we wanted to accurately assess the effect of e.g. a specific teaching technique on the scores of the students, we would want to statistically control for the effect of the nested structure. A hierarchical regression allows for us to estimate the variance in both intercept and slope of the school, to more accurately assess the effect of the teaching technique on the score of the student. For example, the following equations allow us to model the varying intercepts and slopes of each school:

---

[4] We employ a one-vs-rest strategy for the LR and GNV, which transforms a multi-label classification problem to multiple binary classification problems.

[5] We employed song-wise aggregation for this study

[6] We initially matched the original Echonest dataset to our initial song pool and 30% of randomly sampled users. Consequentially, we apply a filter, such that users who interacted with more than 5 songs remain, and vice versa for songs.

[7] We use the implementation from the *scikit-optimize* package.

[8] We do not search the hyper-parameters for FM and use a manually tuned setup, mostly due to the computational complexity required for this specific model.

$$y_i = a_{j[i]} + \beta x_i + \epsilon_i \qquad (2)$$

$$\alpha_j = a_0 + b_0 u_j + \eta_{j1} \qquad (3)$$

$$\beta_j = a_1 + b_1 u_j + \eta_{j2} \qquad (4)$$

where i refers to the individual students, and $j[i]$ refers to the school that student i attends. The first line is similar to a classic regression, where the x represents a predictor at the level of student, the teaching technique in our example, and the ϵ represents the error term of the main regression. However, equations (3) and (4) allow for the modeling of the intercept and slope respectively, where the u and η expressions are the predictors and error terms at the school levels.
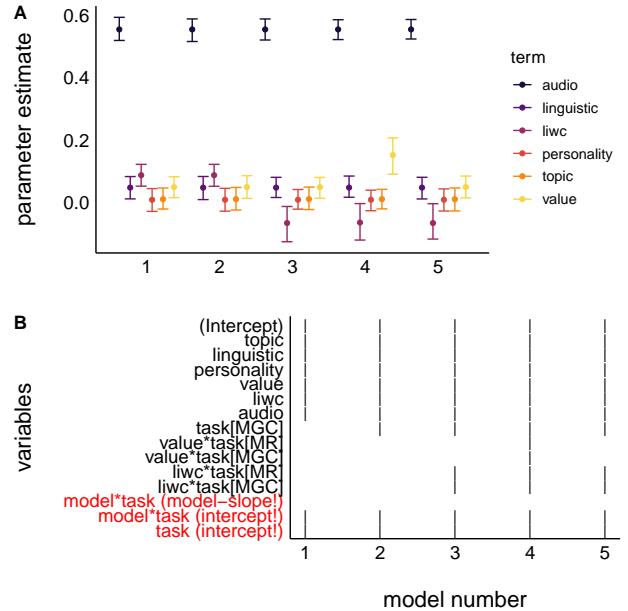
By statistically controlling for these additional variances, hierarchical modeling allows for a more precise estimate of the variables of interest. A more complete discussion can be found in [36].

In our study, we treat the task similarly to the school in our example, and the systems similarly to the students. By controlling for these variances, we estimate the effect of each feature set. From the resulting parameter estimates, we extract 95 % confidence intervals, which we then interpret for our results.

This approach also allows for the comparison of models containing different specifications, where the specifications refer to which specific parameter estimates are computed. As some parameters may not meaningfully contribute to the variance, their effects will be estimated at very close to 0, and may be removed to improve model fit. Indices of fitness, i.e. Akaike and Bayesian Information Criteria (AIC and BIC respectively) give an estimate of model fit, which is penalized by the number of terms. We can therefore arrive at the best-fitting model with the fewest parameters estimated, by systematically removing poorly performing parameter estimates, comparing successive fit indices e.g. with a Likelihood Ratio Test.

Following from our strategy, we examined the usefulness of the inclusion of the various features sets on the 3 considered MIR tasks. Our variables of interest are 1) binary indicators for the inclusion of each of the feature sets: *linguistic*, *topic*, *LIWC*, *personality*, and *values*, as well as the set of audio features, where (0 = not included, 1 = included), 2) a categorical variable representing each of the MIR tasks, 3) a categorical variable representing the systems implemented within each task, and 4) the resulting Measurement scores which were standardized within each task for comparability. We further estimate whether feature sets perform better or worse for certain tasks, by examining interactions between each feature set, and our task variable. Feature sets had differing numbers of sub-dimensions which were not individually analyzed (see Table 1) [9].

We ran multiple models and compared the results of our feature sets across specifications (see Figure 2). Model specifications varied based on 1) how we accounted for the nested structure (i.e. task and systems), as we can estimate

---

[9] Analyses were conducted on two servers running R 3.6.3. and 3.4.4.
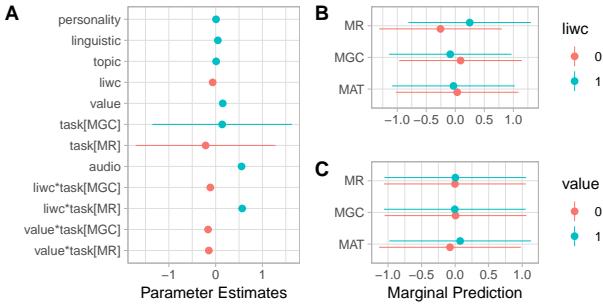
Figure 2. A: Parameter estimates of 5 hierarchical regression models. Error bars are 95% confidence intervals, bootstrapped 500 times. B: Specific parameters that are estimated in the each of the models. Parameters that form the structure of the model are denoted both in red and with a "!" symbol, feature sets of primary interest are denoted in black, and variables for which two terms separated by a "*" are interaction terms.

intercepts for task, for system, for system within task, as well as as slopes for tasks, for systems, and for systems within task, etc., and 2) the interaction terms we specified, i.e. whether we estimated an interaction term for a given feature set and our task variable.

8. RESULTS

We assessed models with two nested structures specified, where the parameters estimated are referred to as "random effects". The first included intercepts for each task, and the system used within task. The second estimated the same intercepts, and additionally estimated a slope for each system. For each of these two random effects structures, we then determined which parameters to estimate, referred to as "fixed effects". Specifically, we estimated parameters for each feature set, and interactions between all feature sets and the tasks. We first specified a "maximal" model, with all features and the task variable, and all two-way interactions among these variables. To remove unnecessary parameters, we ran a protocol which iteratively removed parameter estimates, retaining only those that either 1) significantly decrease model fit if not included, or 2) do not significantly decrease model fit if excluded. The Step function in the *lmerTest* package, was used for this phase [37]. What remained were two interaction terms: the interaction between *values* and task, and between *LIWC* and task. As such, we estimated models with no interaction terms, as well as models with and without each of those inter-

Figure 3. A: Parameter estimates of model 4. Error bars are 95% confidence intervals. Interaction terms are denoted with the "*" symbol. B: Predicted scores for the inclusion of *LIWC* on each of three MIR tasks, where 1 indicates that it was included and 0 indicates that it was not. C: Predicted scores for the inclusion of *values* on each of three MIR tasks, where 1 indicates that it was included and 0 indicates that it was not.

action terms. When we assessed the interaction term, we also included the main effect of task. Thus, we also ran models with and without task included. The 5 models included for interpretation were those that converged without error. Parameter estimates are shown in Figure 2A, and Figure 2B shows which parameters were estimated in each model. For the full specification of our models, we refer the readers to the reproducibility package [10] accompanying this paper.

As is shown in Figure 2A, we observe a consistent, large, positive effect of *audio features* on the score, and no meaningful effects of *topic* and *personality* feature sets. Further, we observe a consistent, small, positive effect of *values* across our specifications. This effect size increases in model 4, where the interaction between values and task was included. Similarly, *LIWC* shows a small but positive effect, that appears to decrease when the interaction term of *LIWC* and task is included. This suggests that *LIWC* and *values* may perform differently, depending on the task.

To clarify if this is the case, we examined the parameter estimates of model 4, which included interaction terms for both *LIWC* and *values* (see Figure 3A). Although both interaction terms were statistically significant, we observe that the confidence intervals for the main effect of task are very wide. This was expected, as 1) we were assessing an interaction effect which might increase the width of a the confidence interval, and 2) we were largely accounting for this variance by standardizing the score within each task, and by including task in the random effect structure. Figures 3A and 3B show the predicted values for both *LIWC* and *values* across tasks. Although the score was higher when *LIWC* was included in the MR task and when *values* was included in the MAT task, the predicted estimates are imprecise, as evidenced by the wide confidence intervals. As such, a more sensitive study design is likely required to obtain estimates of these interaction effects, e.g. analyses

on individual dimensions of feature sets, to establish the most informative features, and/or more systems and more MIR tasks. Thus, we conclude that *linguistic* and *values* feature sets show the most consistent positive effects, and that *LIWC* and *values* may vary in performance based on task.

9. LIMITATIONS AND FUTURE WORKS

Several limitations are still present in our current study. Firstly, although our feature sets did show promising yet small effect sizes, we did not assess the performance of individual dimensions. Given that the feature sets vary greatly in both in terms of the number and content of sub-dimensions (see Table 1), reducing the overall set may result in a more sensitive set of features to examine.

Secondly, we did not consider subgroups of users, or of groups of songs. It may be possible that some users are more sensitive to the content of lyrics than others, and that lyric-sensitive users would benefit far more from lyric features than others. Further, it may be the case that lyrics are very important in some groups of songs vs. others (e.g Hip-Hop music vs. electronic dance music). Further research could examine the potential existence of a lyric-sensitive sub-group of users, lyric-sensitive songs, and how these two may interact.

Thirdly, aspects of our experimental design can be elaborated in future work: 1) Although we strategically sampled a limited number of MIR tasks and a limited number of systems, we did not fully address all possibilities. For instance, future work can include more contemporary systems such as deep learning, thereby increasing generalizability of our results. 2) Certain task metrics could be improved, although we strategically designed our experiment to prevent local noise from skewing our conclusions: e.g. a different performance measure for the genre classification (i.e. AUC-ROC) could deliver a more accurate experimental result, given its skewed class distribution.

Lastly, the reliability of all of our feature sets could be better assessed in the future. This is particularly true of our *personality* features: they contain words that have been shown to describe individuals that have or lack in personality traits, but it is not clear that individuals with those traits use the specific words that describe them.

10. CONCLUSION

Although the *audio* features in our analysis most positively affected performance on various MIR tasks, our lyric-based text features did show some promise. More specifically, *linguistic* and *values* feature sets showed consistent, small effect sizes. Given that the interactions between *LIWC* and task were significant, it may be the case that *LIWC* features are also useful. We can conclude that text-based features drawn from Psychology literature anticipate further research, and that further investigations addressing the current limitations will lead to better data-driven understanding of the role lyrics play in music consumption.

11. ACKNOWLEDGEMENT

12. REFERENCES

[1] R. W. Van Sickel, "A world without citizenship: On (the absence of) politics and ideology in country music lyrics, 1960–2000," *Popular music and society*, vol. 28, no. 3, pp. 313–331, 2005.

[2] A. C. North, A. E. Krause, and D. Ritchie, "The relationship between pop music and lyrics: A computerized content analysis of the United Kingdom's weekly top five singles, 1999–2013," *Psychology of Music*, pp. 1–24, 2020.

[3] C. O. Brand, A. Acerbi, and A. Mesoudi, "Cultural evolution of emotional expression in 50 years of song lyrics," *Evolutionary Human Sciences*, vol. 1, pp. 1–14, 2019.

[4] C. N. DeWall, R. S. Pond Jr, W. K. Campbell, and J. M. Twenge, "Tuning in to psychological change: Linguistic markers of psychological traits and emotions over time in popular us song lyrics." *Psychology of Aesthetics, Creativity, and the Arts*, vol. 5, no. 3, pp. 200–207, 2011.

[5] A. Demetriou, A. Jansson, A. Kumar, and R. M. Bittner, "Vocals in music matter: the relevance of vocals in the minds of listeners," in *Proceedings of the 19th International Society for Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, E. Gómez, X. Hu, E. Humphrey, and E. Benetos, Eds., 2018, pp. 514–520.

[6] C. Howlin and B. Rooney, "Patients choose music with high energy, danceability, and lyrics in analgesic music listening interventions," *Psychology of Music*, vol. 0, no. 0, pp. 1–14, 2020.

[7] S. O. Ali and Z. F. Peynircioğlu, "Songs and emotions: are lyrics and melodies equal partners?" *Psychology of music*, vol. 34, no. 4, pp. 511–534, 2006.

[8] E. Brattico, V. Alluri, B. Bogert, T. Jacobsen, N. Vartiainen, S. K. Nieminen, and M. Tervaniemi, "A functional MRI study of happy and sad emotions in music with and without lyrics," *Frontiers in psychology*, vol. 2, pp. 1–16, 2011.

[9] X. Hu and J. S. Downie, "When lyrics outperform audio for music mood classification: A feature analysis," in *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, J. S. Downie and R. C. Veltkamp, Eds. International Society for Music Information Retrieval, 2010, pp. 619–624.

[10] M. McVicar, T. Freeman, and T. D. Bie, "Mining the correlation between lyrical and audio features and the emergence of mood," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, A. Klapuri and C. Leider, Eds. University of Miami, 2011, pp. 783–788.

[11] Y. Hu, X. Chen, and D. Yang, "Lyric-based song emotion detection with affective lexicon and fuzzy clustering method," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, K. Hirata, G. Tzanetakis, and K. Yoshii, Eds. International Society for Music Information Retrieval, 2009, pp. 123–128.

[12] X. Wang, X. Chen, D. Yang, and Y. Wu, "Music emotion classification of Chinese songs based on lyrics using TF*IDF and rhyme," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, A. Klapuri and C. Leider, Eds. University of Miami, 2011, pp. 765–770.

[13] R. Mayer, R. Neumayer, and A. Rauber, "Rhyme and style features for musical genre classification by song lyrics," in *Proceedings of the 9th International Society for Music Information Retrieval Conference, ISMIR 2008, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, J. P. Bello, E. Chew, and D. Turnbull, Eds., 2008, pp. 337–342.

[14] A. Tsaptsinos, "Lyrics-based music genre classification using a hierarchical attention network," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull, Eds., 2017, pp. 694–701.

[15] F. Kleedorfer, P. Knees, and T. Pohle, "Oh oh oh whoah! Towards automatic topic detection in song lyrics," in *Proceedings of the 9th International Society for Music Information Retrieval Conference, ISMIR 2008, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, J. P. Bello, E. Chew, and D. Turnbull, Eds., 2008, pp. 287–292.

[16] S. Sasaki, K. Yoshii, T. Nakano, M. Goto, and S. Morishima, "Lyricsradar: A lyrics retrieval system based on latent topics of lyrics," in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, H. Wang, Y. Yang, and J. H. Lee, Eds., 2014, pp. 585–590.

[17] R. J. Ellis, Z. Xing, J. Fang, and Y. Wang, "Quantifying lexical novelty in song lyrics." in *ISMIR*, 2015, pp. 694–700.

[18] Y. R. Tausczik and J. W. Pennebaker, "The psychological meaning of words: LIWC and computerized text analysis methods," *Journal of language and social psychology*, vol. 29, no. 1, pp. 24–54, 2010.

[19] Y. Neuman, L. Perlovsky, Y. Cohen, and D. Livshits, "The personality of music genres," *Psychology of Music*, vol. 44, no. 5, pp. 1044–1057, 2016.

[20] J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn, "The development and psychometric properties of LIWC2015," Tech. Rep., 2015.

[21] D. M. Markowitz and J. T. Hancock, "The 27 Club: Music lyrics reflect psychological distress," *Communication Reports*, vol. 30, no. 1, pp. 1–13, 2017.

[22] H. A. Schwartz, J. C. Eichstaedt, M. L. Kern, L. Dziurzynski, S. M. Ramones, M. Agrawal, A. Shah, M. Kosinski, D. Stillwell, M. E. Seligman *et al.*, "Personality, gender, and age in the language of social media: The open-vocabulary approach," *PloS one*, vol. 8, no. 9, pp. 1–16, 2013.

[23] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly, 2009.

[24] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, vol. 42, no. 1/2, pp. 177–196, 2001.

[25] L. R. Goldberg, "An alternative "description of personality": the Big-Five factor structure." *Journal of personality and social psychology*, vol. 59, no. 6, pp. 1216–1229, 1990.

[26] G. Saucier and L. R. Goldberg, "Evidence for the Big Five in analyses of familiar english personality adjectives," *European Journal of Personality*, vol. 10, no. 1, pp. 61–77, 1996.

[27] S. Wilson, R. Mihalcea, R. Boyd, and J. Pennebaker, "Disentangling topic models: A cross-cultural analysis of personal values through words," in *Proceedings of the First Workshop on NLP and Computational Social Science*, 2016, pp. 143–152.

[28] S. R. Wilson, Y. Shen, and R. Mihalcea, "Building and validating hierarchical lexicons with a case study on personal values," in *International Conference on Social Informatics*. Springer, 2018, pp. 455–470.

[29] H. Liu, Y. Huang, Z. Wang, K. Liu, X. Hu, and W. Wang, "Personality or value: A comparative study of psychographic segmentation based on an online review enhanced recommender system," *Applied Sciences*, vol. 9, no. 10, pp. 1–28, 2019.

[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing*

Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 3111–3119.

[31] K. Choi, G. Fazekas, M. B. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull, Eds., 2017, pp. 141–149.

[32] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28,2011*, A. Klapuri and C. Leider, Eds. University of Miami, 2011, pp. 591–596.

[33] H. Schreiber, "Improving genre annotations for the Million Song Dataset," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, M. Müller and F. Wiering, Eds., 2015, pp. 241–247.

[34] D. Liang, M. Zhan, and D. P. W. Ellis, "Content-aware collaborative music recommendation using pre-trained neural networks," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, M. Müller and F. Wiering, Eds., 2015, pp. 295–301.

[35] S. Rendle, "Factorization machines," in *Proceedings of the 10th IEEE International Conference on Data Mining, ICDM 2010, Sydney, Australia, 14-17 December 2010*, G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, Eds. IEEE Computer Society, 2010, pp. 995–1000.

[36] A. Gelman and J. Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge university press, 2006.

[37] A. Kuznetsova, P. B. Brockhoff, and R. H. B. Christensen, "lmerTest package: tests in linear mixed effects models," *Journal of statistical software*, vol. 82, no. 13, 2017.

MODE CLASSIFICATION AND NATURAL UNITS IN PLAINCHANT

Bas Cornelissen   **Willem Zuidema**   **John Ashley Burgoyne**

Institute for Logic, Language and Computation, University of Amsterdam

`mail@bascornelissen.nl`, `zuidema@uva.nl`, `j.a.burgoyne@uva.nl`

ABSTRACT

Many musics across the world are structured around multiple *modes*, which hold a middle ground between scales and melodies. We study whether we can classify mode in a corpus of 20,865 medieval plainchant melodies from the Cantus database. We revisit the traditional 'textbook' classification approach (using the final, the range and initial note) as well as the only prior computational study we are aware of, which uses pitch profiles. Both approaches work well, but largely reduce modes to scales and ignore their melodic character. Our main contribution is a model that reaches 93–95% F_1 score on mode classification, compared to 86–90% using traditional pitch-based musicological methods. Importantly, it reaches 81–83% even when we discard all absolute pitch information and reduce a melody to its contour. The model uses tf–idf vectors and strongly depends on the choice of units: i.e., how the melody is segmented. If we borrow the syllable or word structure from the lyrics, the model outperforms all of our baselines. This suggests that, like language, music is made up of 'natural' units, in our case between the level of notes and complete phrases, a finding that may well be useful in other musics.

1. INTRODUCTION

In his seminal Grove entry, Harold Powers [1] points out a remarkable cross-cultural generalisation: many musics are structured around multiple *modes*. Modes are often associated with the major–minor distinction in Western music, but there are much richer systems of modes: examples include Indian *raga*, Arabic *makam*, Persian *dastgah*, *pathet* in Javanese gamelan music and the *modes* of Gregorian chant. The specifics obviously vary, but all these phenomena share properties with both scales and melodies, and are perhaps best thought of as occupying the continuum in between [1]. On the one hand, a mode is more than a scale: it might imply a hierarchy of pitch relations or favour the use of characteristic motifs. On the other hand, it is not as specific as a particular tune: a mode rather describes a melody *type*. Modes are of central importance to their musical tradition, both as means to classify the repertoire, and

as practical guides for composition and improvisation [1]. Characterising modes computationally is therefore an important problem for *computational ethnomusicology*.

Several MIR studies have investigated automatic mode classification in Indian *raga* [2, 3], Turkish *makam* [4, 5] and Persian *dastgah* [6, 7]. These studies can roughly be divided in two groups. First, studies emphasising the scalar aspect of mode usually look at pitch distributions [2, 5, 7], similar to key detection in Western music. Second, studies emphasising the melodic aspect often use sequential models or melodic motifs [3, 4]. For example, [4] trains *n*-gram models for 13 Turkish makams, and then classifies melodies by their perplexity under these models. Going beyond *n*-grams, [3] uses motifs, characteristic phrases, extracted from raga recordings to represent every recording as a vector of motif-frequencies. They weigh counts amongst others by the *inverse document frequency* (see section 3.4), which balances highly frequent motifs, and favours specific ones.

In this paper, we focus on automatic mode classification in Medieval plainchant. This has only rarely been studied computationally, even though the term (if not the phenomenon) 'mode' originates there. At first glance, mode in plainchant is relatively clear, though certainly not entirely unambiguous. With a second glance, it has a musicological and historical depth that inspired a vast body of scholarship going back over one thousand years. The music is indeed sufficiently distant in time from most other musics, including Western classical and pop music, to provide an interesting cross-cultural comparison. And for once, data is abundant, thanks to the immense efforts of chant scholars.

Chant has mostly figured in MIR studies in optical music recognition of medieval manuscripts: the SIMSSA project, for example, has used such systems to transcribe plainchant from the Cantus database [8]. Recent ISMIR conferences have also included analyses of Byzantine plainchant [9] and Jewish Torah tropes [10], and a comparison of five Christian chant traditions using interval *n*-grams [11]. But, to the best of our knowledge, Huron and Veltman's study [12] is the only computational study addressing mode classification in chant. They took a scalar perspective on mode by using pitch class profiles, an approach which was later criticised, partly for ignoring mode's melodic character [13].

We aim to revisit this work on a larger dataset, and also to model the melodic aspect of mode. Concretely, we compare three approaches to mode classification:

1. **Classical approach:** based the range, final, and initial note of a chant.
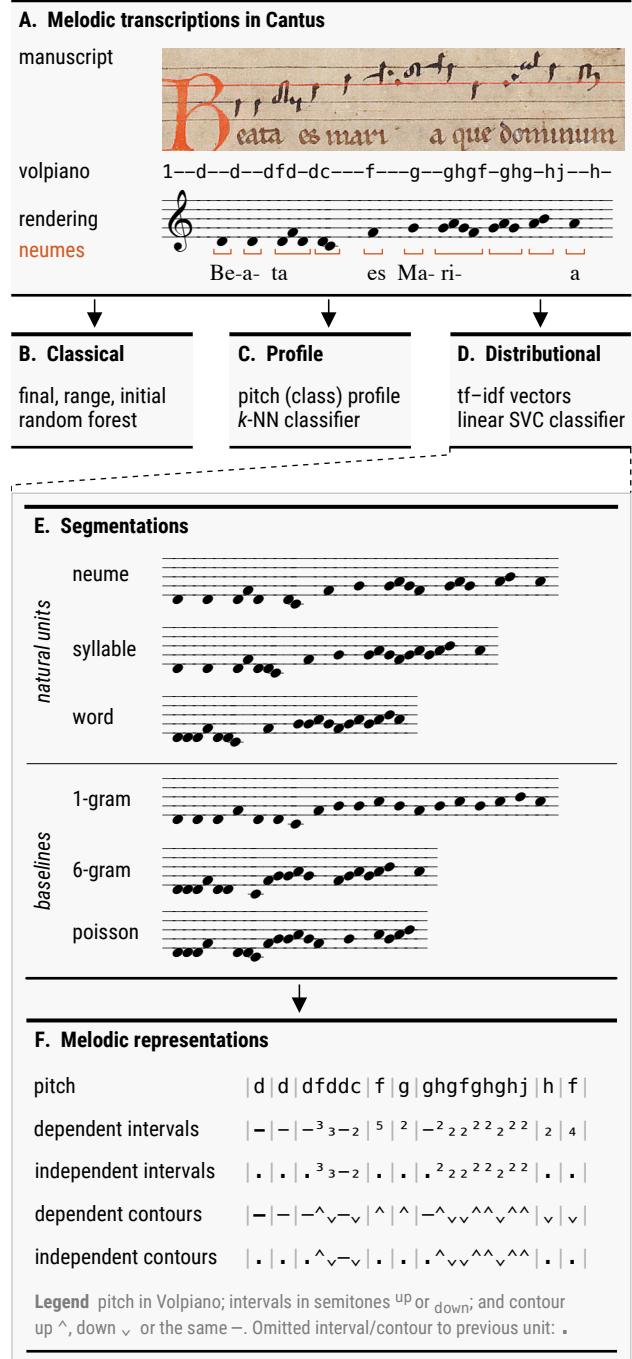
2. **Profile approach:** uses pitch, pitch class and repetition profiles (cf. [12]).

3. **Distributional approach:** uses tf–idf vectors based on various segmentations and representation of the melody.

2. GREGORIAN CHANT

Gregorian chant is the monophonic, Latin chant sung during services in the Roman church. It started out as an oral tradition, coexisting with several others in late Antiquity. Although the specifics are debated [14, ch. 2], from the 9th century onwards it gradually turned into a (partly) written tradition, displacing other chant traditions. Initially, only the texts of the chants were written down, as singers would know the melodies by heart. Chant is rooted in recitation, and the music and text are intimately related: "the basic unit of music-writing [was] not the note, but the syllable" [15], the smallest singable unit of text. Accordingly, the earliest notation lived between the lines of text: signs, called *neumes*, reminding the singer of the contour of the melody: perhaps how many notes and their direction, but not *which* exact pitches. The earliest melodies are therefore unknown, but later manuscripts use a pitch-specific notation by placing neumes on staff lines, preserving those melodies to the present day (see Figure 1A).

There are different chant genres for different parts of the liturgy, each with own musical characteristics [16]. Some genres consist of recitations of a sacred text mostly on a fixed pitch, with common starting and ending formulae, while others use elaborate melodies and few repeated notes. Genres also differ in their *melismaticness*: the number of notes per syllable (see Figure S5). In *syllabic* genres like *antiphons*, every syllable of text aligns with roughly one note. More melismatic genres like *responsories* align single syllables to long melismas of ten notes or more. In this paper, we focus on antiphons and responsories, two melodic and common genres.

Gregorian chant uses a distinct tonal system of eight modes, usually numbered 1–8, but sometimes named like church scales. Modes come in pairs that share the same scale (*Dorian*, *Phrygian*, *Lydian* or *Mixolydian*), but have a different range or *ambitus*: *authentic* modes moves mostly above the tonal center or the *final*, *plagal* ones mostly around it. Mode 3 is for example also called *Phrygian authentic*, and melodies in this mode rarely go below the final note E. The standard way of determining the mode is to first determine the final, and then the range [16]. For the majority of the chants this will be sufficient, but one might further consider the initial note, characteristic phrases or circumstantial evidence (e.g. psalm tones). Nevertheless, the mode of some chants will remain ambiguous: the *theory* of eight modes was borrowed from Byzantine theory in the 8th century, and applied to an already existing chant repertoire (with its own modalities [13]). The fit between theory and practice was reasonable, but not perfect [1]. This also suggests that perfect classification accuracy is likely out of reach.

Figure 1. **Overview of this study** which compares three approaches to mode classification in a corpus of Gregorian chant. Cantus contributors have transcribed a vast number of melodies from medieval manuscripts (**A**). We classify mode based on the final, range and initial in the *classical approach* (**B**), and based on pitch (class) and repetition profiles in the *profile approach* (**C**). Finally in the *distributional approach* (**D**), we use tf–idf vectors where we tweak two parameters: the *segmentation*, or which melodic units we use (**E**), and the *representation* (**F**), where we gradually discard information about the scale when we move from pitches to contours. In this way we aim to capture the melodic, rather than scalar, aspect of mode.

3. METHODS

The design of this study is visualized in Figure 1.

3.1 Data: the Cantus Database

We use chant transcriptions from the Cantus database [17]. This is primarily a digital index of medieval chant manuscripts, recording the chant location in the manuscript, its full text, and properties like the mode, the liturgical feast, but also links to manuscript images. Cantus currently consists of almost 150 manuscripts, containing over 450,000 chants, contributed by chant scholars from all over the world. Over 60,000 chants also contain melodic transcriptions written in Volpiano.[1] It sets plain text as musical notes on a five-line staff, as illustrated in Figure 1a. Volpiano also supports some accidentals, clefs, liquescents, barlines and strokes. All submissions to Cantus are subject to strict guidelines and manually checked by the Cantus editors (see also [18]). This ensures the quality and consistency of database, making it a valuable resource for computational research.

We scraped the entire database of 497,071 chants via its REST API and we have released this as the CantusCorpus.[2] We here only consider chants that have a Volpiano transcription (63,628 chants) and further filter out chants with incomplete or non-standard transcriptions, without a complete melody, without 'simple' mode annotation, and exact duplicates (see section S1). This resulted in 7031 responsories (966,871 notes, avg. length 138 notes) and 13,865 antiphons (825,143 notes, avg. length 60 notes). We fixed a 70/30 train/test split for all datasets and only used training data in exploratory analyses. Cantus often contains multiple variants of any particular melody, transcribed from different manuscripts (see Figure S11). One may wonder whether the simple train/test split is sufficient, or whether even more care is needed to avoid overlap between such melodic variants in the train and test sets. This is a difficult issue that also applies to other musical corpora (e.g., the Essen folk-song corpus), and for which there is no perfect solution. We tried repeating our experiments on a subset without variants and return to this issue in section 4.4.
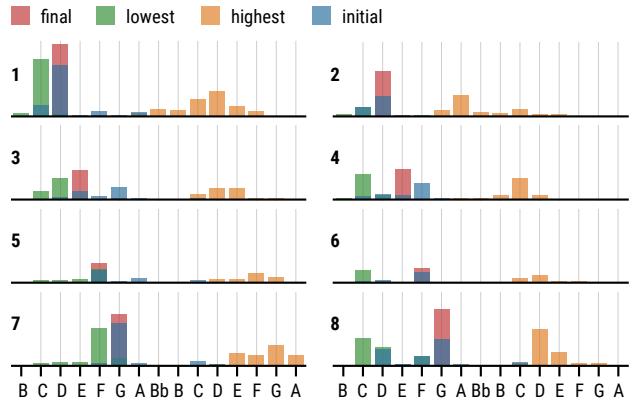
According to the transcription guidelines, flat symbols are transcribed only once, directly before the first flattened note. We replace the first and later flattened notes by the corresponding accidental, a Volpiano character that sits at a specific staff line. In this way, flat notes are also encoded by a single Volpiano character. We discard characters like clefs and pausas, and only retain the notes, accidentals and boundaries (hyphens). The resulting string is used in our three classification experiments, which we now discuss.

3.2 Classical Approach: Final, Range, Initial
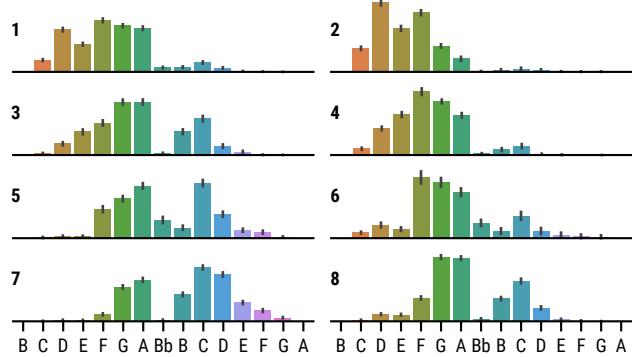
The first approach is motivated by the classical procedure for mode classification. We extract three features from every chant: the final pitch, the range (lowest and highest pitches)
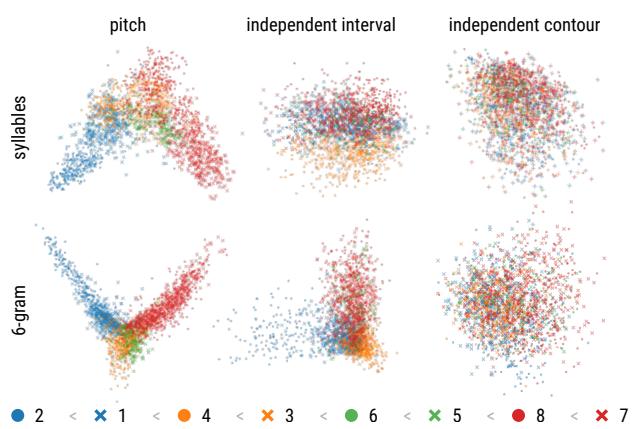
Figure 2. Classical features. The classical approach uses the final, range and initial to determine the mode. The overall distribution for each of the modes (1–8) is clearly different, although not entirely without ambiguity.

Figure 3. Pitch profiles showing the relative frequency of every pitch in each of the 8 modes. Again, although the distribution of individual modes are clearly distinct, some residual ambiguity remains.

Figure 4. PCA of tf–idf vectors. Principal component projection of the tf–idf vectors of responsories in several conditions. The figure suggests that classification gets harder when moving from a pitch to a contour representation. The legend shows a theoretical ordering of the modes based on their range. See Figure S9 and Figure S10 for larger plots.

and the initial pitch. Theory suggests that the final alone should give an accuracy of roughly 50%, and adding the range should further increase that by roughly 50%, if there is no ambiguity. Figure 2 shows the feature distributions for all modes. It suggests that there is some ambiguity, and so numbers will be a little lower. For this task we use random forest classifiers [19], which aggregate multiple decision trees. Training details of all models are discussed below.

3.3 Profile Approach: Pitch (Class) Profiles

The second approach is inspired by Huron & Veltman [12]. Using 97 chants from the Liber Usualis, they compute average *pitch class profiles* (the relative frequency of each pitch class) for each of the modes and then classified chants to closest profile. We take a similar approach and use k-nearest neighbour classification, where k is tuned (see section 3.5). In a commentary, Wiering [13] argued for using actual pitches rather than pitch classes, as the pitches an octave above the final have a very different role than those an octave below it. We follow that suggestion by also computing *pitch profiles* (Figure 3). Finally, we propose a *repetition profile* aiming to describe which notes function like a recitation tone. For every Volpiano pitch q we compute a repetition score $r(q)$, which is the relative frequency of direct repetitions, and collect these to get a repetition profile. Formally, if a chant has pitches $p_1, \ldots, p_N$, then $r(q) = \#\{i : p_i = q \text{ and } p_{i+1} = q\}/(N-1)$ since there are $N-1$ possible repetitions.

3.4 Distributional Approach: tf–idf Vectors

Our third approach aims to capture the melodic aspect of mode. In short, we use a bag of 'words' model (cf. [3]) and tweak two parameters: the segmentation (which melodic units to use as 'words') and the representation (pitches, intervals and contours). The idea is to discard more and more information about the scale, and see if we can nevertheless determine the mode.

First, the units. For chant, three natural segmentations suggest themselves: one can segment the melody (1) at neume boundaries, but also wherever we find (2) a syllable or (3) a word boundary in the lyrics. Given the close relation between text and music in chant, there is some reason to believe that these are meaningful units. Conveniently, all of these boundaries are explicitly encoded in Volpiano, by a single, double and triple dash respectively. Note that these natural units are nested: neumes never cross syllable boundaries. We compare the natural units to two types of baselines. The first is an *n*-gram baseline where we slice the melody after every *n* notes, for $n = 1, \ldots, 16$. The second is a random, variable-length baseline. Here the melody is segmented randomly, but in such a way that the segment length is approximately Poisson distributed with a mean length of 3, 5, or 7. We stress that all these units are proper segmentations: units do not overlap. In particular, we choose not to use a higher-order model (using *n*-grams of units), because we are only interested in comparing different segmentations.

Second, the representation. We represent melodies in three ways: as a sequence of *pitches*, *intervals* (the number of semitones between successive notes) and *contours* (the contour between successive notes: up, down or level). There is one complication when segmenting sequences of intervals or contours: we introduce dependencies between the units. All units would, for example, start with the interval from the previous unit. We call this a *dependent* segmentation. Alternatively, you could discard the intervals between units to obtain an *independent* version. This effectively makes every unit one interval shorter. We analyse both independent and dependent versions, but in the independent one we found it convenient to start all units (including the first) with a dot to keep the segmentation identical across representations. You can think of the dot as marking the omitted interval to the previous unit.

Third, the model. Given a segmentation, we represent every chant by a vector of unit frequencies, but weighted to favour frequent, yet *specific* units: units that do not occur in too many chants. A standard way of doing this in textual information retrieval is using *term-frequency inverse-document-frequency* (tf–idf) scores, which multiply the frequency of a term in a document (tf) by the inverse document frequency (idf): the inverse of the number of documents containing the term. We use +1 smoothing for the idf, at most 5000 features, and found it was important *not* to set a minimum or maximum document frequency. We train a linear support vector machine to classify mode using the resulting tf–idf vectors.

In sum, we analyse 22 segmentations (3 natural ones, 16 *n*-grams, 3 random) and 5 representations (pitch and dependent/independent interval/contour), giving a total of 110 conditions.

3.5 Training
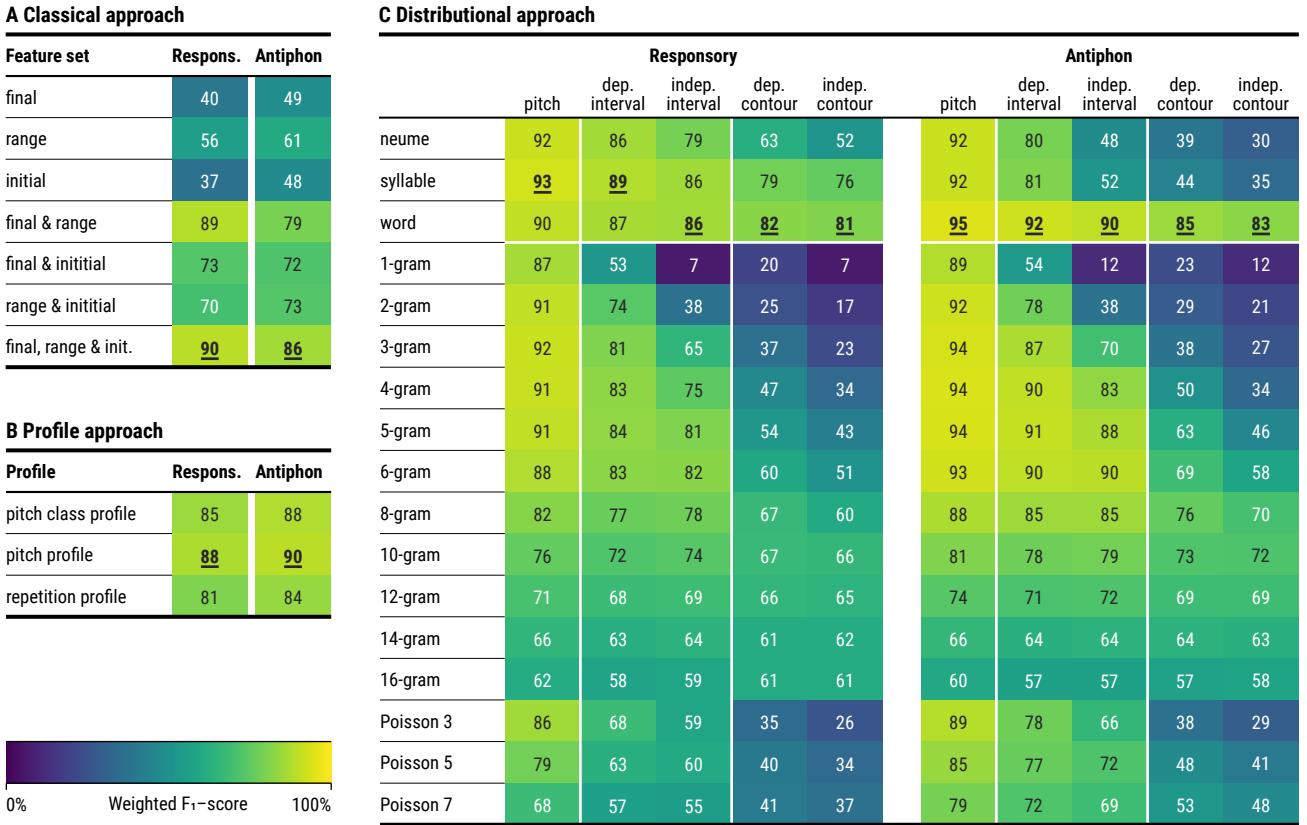
We tune every model using a randomised hyperparameter search with 5-fold stratified cross-validation. That is to say that we randomly sample hyperparameters from a suitable grid (determined by extensive manual analyses) and determine their performance using 5-fold cross-validation on the training set, where we ensure the class frequencies are similar in all folds. We use the hyperparameters yielding the highest cross-validation test accuracy to train the final model. All models were implemented in Python using scikit-learn [20] and data and code are available online. [3]

4. RESULTS

Figure 5 gives support-weighted [4] averages of F_1-scores obtained on the full test sets for all three approaches. The scores are averages of five independent runs of the experiment, using different train/test-splits. Standard deviations were small and are included in figure S12. We now compare the three approaches and then discuss the effect of representation and segmentation on the distributional approach.

---

[3] See github.com/bacor/ismir2020

[4] The retrieval scores for all classes (modes) are averaged, weighted by the number of instances in each class.

A Classical approach

Feature set	Respons.	Antiphon
final	40	49
range	56	61
initial	37	48
final & range	89	79
final & inititial	73	72
range & inititial	70	73
final, range & init.	**90**	**86**

B Profile approach

Profile	Respons.	Antiphon
pitch class profile	85	88
pitch profile	**88**	**90**
repetition profile	81	84

0%　Weighted F_1-score　100%

C Distributional approach

	Responsory					Antiphon				
	pitch	dep. interval	indep. interval	dep. contour	indep. contour	pitch	dep. interval	indep. interval	dep. contour	indep. contour
neume	92	86	79	63	52	92	80	48	39	30
syllable	93	**89**	86	79	76	92	81	52	44	35
word	90	87	**86**	**82**	**81**	**95**	**92**	**90**	**85**	**83**
1-gram	87	53	7	20	7	89	54	12	23	12
2-gram	91	74	38	25	17	92	78	38	29	21
3-gram	92	81	65	37	23	94	87	70	38	27
4-gram	91	83	75	47	34	94	90	83	50	34
5-gram	91	84	81	54	43	94	91	88	63	46
6-gram	88	83	82	60	51	93	90	90	69	58
8-gram	82	77	78	67	60	88	85	85	76	70
10-gram	76	72	74	67	66	81	78	79	73	72
12-gram	71	68	69	66	65	74	71	72	69	69
14-gram	66	63	64	61	62	66	64	64	64	63
16-gram	62	58	59	61	61	60	57	57	57	58
Poisson 3	86	68	59	35	26	89	78	66	38	29
Poisson 5	79	63	60	40	34	85	77	72	48	41
Poisson 7	68	57	55	41	37	79	72	69	53	48

Figure 5. **Classification results.** Weighted F_1-score for three approaches to mode classification, using two chant genres: responsories and antiphons. Scores are averages of five independent runs of the experiment. The classical approach (**A**) using the final, range and initial reaches F_1-scores of 90% and 86%. The profile approach (**B**) works better for antiphons (90% vs. 86%) and somewhat worse for responsories (88% vs. 90%). As [13] suspected, pitch profiles outperform pitch *class* profiles by a small margin. The distributional approach (**C**) reaches the highest F_1 scores of 95% on both responsories and antiphons. The choice of segmentation (vertically) is crucial: classification is improved by using 'natural' units, word-based units in particular, rather than *n*-grams. As the representation (horizontally) becomes cruder, from pitches to intervals and finally to contours, the task becomes much harder. But, when using word-based segmentation, performance remains high.

4.1 Approaches: Distributional Approach Works Best

First of all, we report the highest classification scores with our distributional approach using pitch representations: an F_1-score of 93% for responsories and 95% for antiphons. This corresponds of an error reduction of 30–60% compared to the classical approach (90% and 86%). The classical approach confirms the rule of thumb: the range and final are very informative features. Using only these, we obtain F_1-scores of 89% and 79%, which are further increased by also adding the initial. The profile approach outperforms the classical approach for antiphons (90% vs. 86%), but is outperformed for responsories (88% vs. 90%). Our results support Wiering's [13] intuition that pitch profiles more accurately describe mode than pitch *class* profiles, but the effect is small: it increases F_1 scores by 2–3%. Repetition profiles appear to be less useful for both genres.

In broad strokes, our results validate the classical and profile approach, both of which peak around a 90% F_1-score, using simple features. The distributional approach improves this, up to 95% using complex features. Importantly, we now show that the distributional approach maintains high performance when using interval or contour representations.

4.2 Representations: Contours are Sufficient

We find that the classification task gets harder when the representation gets cruder, from those based on pitch, to intervals and finally to contours (figure 5C, horizontally). This was anticipated: cruder representations are obtained by discarding information from every unit. Shorter units are impacted more by this information loss. For example, the performance with 1-grams drops by over 75% when moving from pitch to independent contour representation. At that point it performs at majority baseline (a 7% F_1-score for responsories and 12% for antiphons). [5] For longer units such as 10-grams, the drop is not as dramatic (around 10%). However, this comes at the cost of a comparatively low performance using the pitch-representation, presumably because of increasing sparsity.

Natural units, however, escape this trade-off. Word-based segmentations perform consistently well, dropping

---

[5] For 1-grams in independent interval *and* contour representation, every unit is identical: a dot representing the omitted contour to the previous note. The majority class for both responsories and antiphons is mode 8, taking up 21% and 28% of the test data respectively (see table S3). This is precisely the accuracy of the model in those conditions.

only 3% below the classical baseline using the highly impoverished independent contour representation. In contrast to the other representations, the contours do not carry any information about the scale: the same contour can be reproduced in any scale. Apparently, we can discard the scalar aspect of mode, and still classify it: contours alone contain sufficient information for mode classification. The success of pitch-based methods might obscure the fact that mode is as much a melodic phenomenon as a scalar one.

It is interesting to note that the earliest chant notation used *unpitched* neumes that mainly described the contour of the melody—not the exact pitches. Our results reinforce the idea that contour is highly informative—so informative that given a mode, text and contour, an experienced singer could reconstruct the chant melody.

4.3 Segmentations: Natural Units Work Best.

Our most important result is that among all the representations we considered, natural units (neume, syllables, and words) yield the highest classification performance. The 4- and 6-gram baselines also reach top F_1-scores in antiphons, but only when we use representations that include information about pitch. Furthermore, the success of natural units cannot be explained solely by their length. In responsories, neumes, syllables and words are on average 2.3, 3.0 and 7.1 notes long, respectively (see table S6), and yet the performance of these natural units is consistently higher than n-grams of comparable length. The performance of the natural units is also consistently higher than that of the variable-length Poisson baselines, which are intended to mimic the overall distribution of natural lengths but ignore musical and textual semantics.

A few other observations merit discussion. Firstly, although neume and syllable segmentations behave differently for responsories, they behave similarly to each other for antiphons. The reason may be that in antiphons, neumes and syllables more often coincide. Antiphons are less *melismatic* than responsories (i.e., they use fewer notes per syllable, 1.5 to be precise). Secondly, both the n-grams and the Poisson baseline perform better on antiphons than on responsories, possibly because the n-grams are more likely to end up being coincidentally aligned with the natural units the less melismatic the genre.

4.4 Controlling for Melodic Variants

We repeated all experiments on a subset of the data from which we removed melody variants (see supplement S13 for details). In terms of the number of notes, this meant a 75% and 66% reduction in data size for responsories and antiphons respectively. The performance of all models decreased on this subset, and for responsories more than for antiphons. Our main findings that contours are sufficient and that natural units work best across representations stand. We do observe some reorderings: some already high-performing n-grams in antiphons now for example slightly overtake word segmentations, although only for pitch and dependent interval representations. The distributional approach works best for antiphons regardless of including or excluding chant variants, but for responsories, the distributional approach drops slightly below the classical approach on the subset (where the profile approach is worst). These findings might be explained by increased sparsity in the smaller dataset: natural units in responsories are, after all, longer. Exploring these issues further is left for future work.

5. DISCUSSION AND CONCLUSION

In this paper, we analyzed three approaches to mode classification in a large corpus of plainchant: (1) the classical approach using the final, range and initial; (2) the profile approach using pitch (class) profiles and (3) the distributional approach using a tf–idf vector model and various segmentations and representations. We found that the distributional approach performs best, and that it can maintain high performance on contour representations if using the right segmentation: at word boundaries, in this case. The main findings were largely upheld when we removed melody variants, but the handling of variants is an issue that deserves further investigation and that has implications beyond this study.

Although our results are specific to one corpus of medieval music and one classification task, we believe our conclusions are of wider relevance. We often fall back on n-grams because they are well understood and easy to use. A more natural segmentation may be harder to obtain, but if finding them can have such a large effect on a relatively simple task like mode classification, their advantages may be even stronger for more complex tasks.

A first next step could be to explore whether lyrics yield equally useful units in other vocal musics. As noted, the link between text and music in plainchant is particularly tight. This at least suggests that the text may be useful in other types of chant, like Byzantine chant or Torah trope. For folk melodies designed to standard poetic meters, it is not as obvious whether lyrics would help or hinder the identification of useful units. This is worth investigating, as characteristic motifs and repeated pattern are commonly used in computational folk-song studies, in particular for tune family identification [21, 22].

Our results raise another question: is chant indeed composed by stringing together certain melodic units, much like a sentence is composed of words? It has been suggested (and disputed) that Gregorian chant is composed in a process of *centonization*, and that a chant is a patchwork of existing melodic chunks called *centos*. A recent study used the tf–idf weighting to discover centos in Arab-Andalusian music [23]. This raises the possibility that classification using natural units may have been successful because they indeed are the building blocks, the centos.

Chant is not yet commonly studied in the MIR community, but we hope that this study shows that chant is an interesting repertoire that can yield insights of broader relevance. The immense efforts of chant scholars mean that data are abundant. In short, we think chant can aid the development of models that apply beyond Western classical and pop music, and embrace the true diversity of musics around the world.

6. REFERENCES

[1] H. S. Powers, F. Wiering, J. Porter, J. Cowdery, R. Widdess, R. Davis, M. Perlman, S. Jones, and A. Marett, "Mode," in *Grove Music Online*. Oxford University Press, 2001. [Online]. Available: https://doi.org/10.1093/gmo/9781561592630.article.43718

[2] P. Chordia and A. Rae, "Raag recognition using pitch-class and pitch-class dyad distributions," in *Proceedings of the 8th International Society for Music Information Retrieval Conference*, 2007, pp. 431–436.

[3] S. Gulati, J. Serra, V. Ishwar, S. Senturk, and X. Serra, "Phrase-based rãga recognition using vector space modeling," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, 2016, pp. 66–70.

[4] E. Ünal, B. Bozkurt, and M. K. Karaosmanoğlu, "N-gram based statistical makam detection on makam music in Turkey using symbolic data," in *Proceedings of the 13th International Conference on Music Information Retrieval*, 2012, p. 43–48.

[5] N. B. Atalay and S. Yöre, "Pitch distribution, melodic contour or both? modeling makam schema with multidimensional scaling and self-organizing maps," *New Ideas in Psychology*, vol. 56, p. 100746, Jan. 2020.

[6] S. Abdoli, "Iranian traditional music dastgah classification," in *Proceedings of the 12th International Conference on Music Information Retrieval*, 2011, pp. 275–280.

[7] P. Heydarian and D. Bainbridge, "Dastgàh recognition in iranian music: Different features and optimized parameters," in *Proceedings of the 6th International Conference on Digital Libraries for Musicology*. The Hague, the Netherlands: ACM Press, 2019, pp. 53–57.

[8] K. Helsen, J. Bain, I. Fujinaga, A. Hankinson, and D. Lacoste, "Optical music recognition and manuscript chant sources," *Early Music*, vol. 42, no. 4, pp. 555–558, 2014.

[9] M. Panteli and H. Purwins, "A computational comparison of theory and practice of scale intonation in Byzantine chant," in *Proceedings of the 14th International Conference on Music Information Retrieval*, 2013, pp. 169–174.

[10] P. van Kranenburg, D. P. Biro, S. Ness, and G. Tzanetakis, "A computational investigation of melodic contour stability in Jewish Torah trope performance traditions," in *Proceedings of the 12th International Conference on Music Information Retrieval*, 2011, pp. 163–168.

[11] P. van Kranenburg and G. Maessen, "Comparing offertory melodies of five medieval Christian chant traditions," in *Proceedings of the 18th International Conference on Music Information Retrieval*, 2017, pp. 163–168.

[12] D. Huron and J. Veltman, "A cognitive approach to medieval mode: Evidence for an historical antecedent to the major/minor system," *Empirical Musicology Review*, vol. 1, no. 1, pp. 33–55, 2006.

[13] F. Wiering, "Comment on Huron and Veltman: Does a cognitive approach to medieval mode make sense?" *Empirical Musicology Review*, vol. 1, no. 1, pp. 56–60, 2006.

[14] P. Jeffery, *Re-Envisioning Past Musical Cultures: Ethnomusicology in the Study of Gregorian Chant*, ser. Chicago Studies in Ethnomusicology. University of Chicago Press, 1992.

[15] T. F. Kelly, "Notation I," in *The Cambridge History of Medieval Music*. Cambridge University Press, 2018, vol. 1, pp. 236–262.

[16] D. Hiley, *Gregorian Chant*. Cambridge University Press, 2009.

[17] D. Lacoste, T. Bailey, R. Steiner, and J. Koláček, "Cantus: A database for Latin ecclesiastical chant," http://cantus.uwaterloo.ca/, 1987-2019, directed by Debra Lacoste (2011-), Terence Bailey (1997-2010), and Ruth Steiner (1987-1996). Web developer, Jan Koláček (2011-).

[18] K. Helsen and D. Lacoste, "A report on the encoding of melodic incipits in the Cantus database with the music font 'Volpiano'," *Plainsong and Medieval Music*, vol. 20, no. 01, pp. 51–65, Apr. 2011.

[19] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] A. Volk and P. van Kranenburg, "Melodic similarity among folk songs: An annotation study on similarity-based categorization in music," *Musicæ Scientiæ*, vol. 16, no. 3, pp. 317–339, 2012.

[22] B. Janssen, P. van Kranenburg, and A. Volk, "Finding occurrences of melodic segments in folk songs employing symbolic similarity measures," *Journal of New Music Research*, vol. 46, no. 2, pp. 118–134, 2017.

[23] T. Nuttall, M. G. Casado, V. N. Tarifa, R. C. Repetto, and X. Serra, "Contributing to new musicological theories with computational methods: The case of centonization in Arab-Andalusian music," in *Proceedings of the 20th International Conference on Music Information Retrieval*, 2019, pp. 223–228.

CONLON: A PSEUDO-SONG GENERATOR BASED ON A NEW PIANOROLL, WASSERSTEIN AUTOENCODERS, AND OPTIMAL INTERPOLATIONS

Luca Angioloni[1]   **Tijn Borghuis**[2,3]   **Lorenzo Brusci**[3]   **Paolo Frasconi**[1]

[1] DINFO, Università di Firenze, Italy   [2] Eindhoven University of Technology, The Netherlands
[3] Musi-co, Eindhoven, The Netherlands

`first.last@unifi.it, first.last@musi-co.com`

ABSTRACT

We introduce CONLON, a pattern-based MIDI generation method that employs a new lossless pianoroll-like data description in which velocities and durations are stored in separate channels. CONLON uses Wasserstein autoencoders as the underlying generative model. Its generation strategy is similar to interpolation, where MIDI pseudo-songs are obtained by concatenating patterns decoded from smooth trajectories in the embedding space, but aims to produce a smooth result in the pattern space by computing optimal trajectories as the solution of a widest-path problem. A set of surveys enrolling 69 professional musicians shows that our system, when trained on datasets of carefully selected and coherent patterns, is able to produce pseudo-songs that are musically consistent and potentially useful for professional musicians. Additional materials can be found at `https://paolo-f.github.io/CONLON/`.

1. INTRODUCTION

Algorithmic music generation has attracted the interest of musicians and practictioners for long time, starting from early works by Guttman, Hiller and Isaacson [1] and Xenakis [2] in the 1950's. Significant progress has recently resulted from the widespread application of new and powerful methods based on deep generative models, letting this class of data-driven approaches gradually take over more traditional rule-based or probabilistic techniques [3, 4]. This thriving line of research spans several dimensions of the generation process, including different types of data: audio signal [5, 6] vs. symbolic MIDI data; musical textures: monophonic [7] vs. polyphonic [8, 9]; ensembles: single-instrument [9] vs. multi-instrument [7]; goals: e.g., continuation [10], accompaniment [11], style transfer [12–14], or interpolation [7, 14].

We are particularly interested in developing a tool for a context where the automatic generation and proliferation of new music material (not necessarily finished pieces) is useful to assist musicians in music and media production. Usefulness in this context may have different facets: supporting and accelerating personal explorations of unknown or semi-known music areas, collecting intra-genre or cross-genre ideas of various complexity and abstraction, augmenting the composer's ability to explore the combinatorial space of rhythmic, melodic, and harmonic variations. In other words, what Boden calls "exploring conceptual spaces" [15]. In spite of the impressive amount of recent advancements in music generation with machine learning approaches, the musical quality of the results is still not always sufficient to enable a widespread adoption in realistic professional scenarios such as studio production using standard Digital Audio Workstations (DAWs) or live performance of electronic music.

In this paper, we focus on the autonomous generation of polyphonic and multi-instrument MIDI partitures, aiming at producing relatively long *pseudo-songs* (i.e. tracks that have the duration of a song but whose temporal structure is not controlled by a compositional intent) in mainstream genres such as Acid Jazz, Soul or High Pop, that are effectively *usable* in a professional music production context. We argue that achieving this goal requires not only the effective exploitation of algorithmic ideas but also a careful selection of coherent musical materials to be used as training data. Unlike the case of image data, where there exist large scale high quality coherent datasets (for example CelebA [16] focusing on human faces), existing symbolic music datasets for mainstream music contain large variations in genre, style, and track/instrument role, that make it more difficult to learn to generate musically coherent pseudo-songs. In the attempt to verify the impact of dataset quality on the results, we introduce in this paper (perhaps for the first time in this research area) two new datasets that were not extracted from existing collections but that have been especially composed and edited by two musicians made aware of creating training sets for generative models. One dataset, ASF-4, is in Acid Jazz, Soul and Funk; the other one, HP-10, in High Pop. Compared to datasets used in other experiments (e.g. LPD-5 [17]) they are small, i.e. of a size that individual musicians would be able to compose or curate by themselves. This opens up the

possibility of personalized generators, assisting musicians in producing their own music.

From the algorithmic point of view, possible solutions to the music generation problem can be characterized across several separate dimensions (see [3] for a thorough taxonomy). The most important ones for the goals of this paper are (1) the type of data structures that are used to describe MIDI patterns (2) the nature of the generative learning models, and (3) the strategy used to produce a whole musical piece. The system presented in this paper introduces novelties across all these three dimensions, whose combination allows us to generate meaningful and professionally usable streams of music. In terms of data description, we introduce a novel pianoroll-like pattern description that stores velocities and durations in two separate channels. The description is lossless (i.e., it can be inverted to recover the original MIDI data exactly) and perceptually robust to reconstruction errors (i.e., it does not suffer the note shattering problem associated with binary pianorolls that store consecutive high bits to represent note durations). As a generative model, we experiment with Wasserstein autoencoders (WAE) [18], a type of autoencoder that is less subject to the "blurriness" problem typically associated with variational autoencoders (VAE) [19]. To the best of our knowledge, WAEs have not been applied to music generation before. Third, our generation strategy is based on interpolation as in previous works [7,14] but we formulate it as an optimization problem for exploring the autoencoder latent space in a way that prevents abrupt transitions between consecutively generated patterns, as well as regions with little variation.
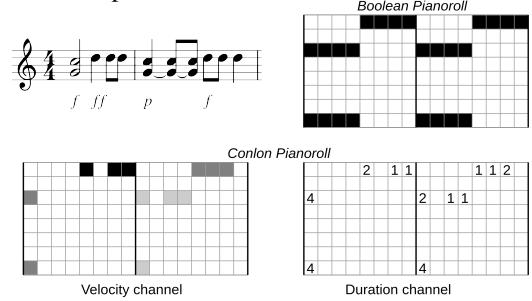
We call our system CONLON, for Channeled Onset of Notes and Length Of Notes, and in honor of Conlon Nancarrow (1912–1997), a pioneer of piano roll compositions. A thorough evaluation with human experts suggest that CONLON is able to produce pseudo-songs that are truly exploitable by professional musicians in mainstream genres.

2. A NEW PIANOROLL WITH EXPLICIT DURATIONS

Each track in a MIDI stream consists of a sequence of time stamped events. We consider here only two types of note events: $\mathrm{ON}(t, n, v)$, and $\mathrm{OFF}(t, n)$. Here t denotes the time at which a note with pitch n begins or ends, measured in MIDI clock pulses. In general, n takes values between 0 (C-1) and 127 (G9). The MIDI velocity, v, takes values in the integer range $[0, 127]$. Roughly, velocity is associated with the note intensity (allowing to represent dynamic expression elements, such as *pianissimo*, *forte*, or accents in percussive instruments) but depending on the instrument attached to the track, it can also affect timbre (for example, a "clicked" Hammond organ sound could be selected in the sound bank when the velocity exceed a given threshold). Note that although n and v take values in the same range, v should be regarded as a continuous variable and n as a categorical one.

Before this data can be fed into a learning algorithm, it needs to be arranged in a proper description format.

Both variable-lenght and fixed-lenght descriptions have been studied in the literature. Variable-length descriptions are typically used in conjunction with various types of recurrent neural networks (RNNs) [20]. Fixed-length description include the pianorolls (PR) introduced in [21] for melodies and later extended to the multitrack polyphonic case [17, 22], and are suitable for modules based on convolutional neural networks (CNNs) [23]. Pianorolls, however, are a *lossy* description of MIDI data in at least two ways. First, they do not include note velocities, which are important for dynamic expression in many musical genres. Second, they make it impossible to distinguish between long notes and repeated occurrences of the same notes (see Fig. 1). The latter limitation can be mitigated by using a finer quantization step or fixed by adding a replay matrix [9]. Still, the PR description may suffer a fundamental problem when there are imperfections in the reconstructions generated by a trained model. In facts, false negatives in the reconstruction may shatter a long note into several shorter ones, which may produce a musically obsessive and unpleasant result.

Figure 1. A short phrase described as PR (top right) and as PR$^\mathrm{C}$ (bottom). To construct a simple example, here we set quantization at 1/8.

The solution proposed in this paper uses a second channel as in [9] but explictly represents note durations as continuous variables. More precisely, in our PR$^\mathrm{C}$ description, the tensor associated with a fixed-length music pattern is constructed as follows. First, we introduce a time quantization function q that maps fine-grained timestamps into coarse-grained temporal positions in the range $1, \ldots, T$. For example $T = 128$ if we quantize four $\frac{4}{4}$ bars at 1/32th. Assuming for simplicity a single instrument and denoting by N the number of pitches in the used range, we create a tensor x of shape $T \times N \times 2$. In the first channel, $x_{q(t),n,1} = v$ if there occurs an event $\mathrm{ON}(t, n, v)$ and $x_{q(t),n,1} = 0$ otherwise. In the second channel, $x_{q(t),n,2} = d$ if there occurs an event $\mathrm{ON}(t, n, v)$ whose duration (expressed in quantized steps) is d, and $x_{q(t),n,2} = 0$ otherwise. The construction is illustrated in Fig. 1. Polyphony is handled naturally in this description and multi-instrument patterns with K tracks can be easily described by allocating two channels for each track as above, resulting in a $T \times N \times 2K$ tensor. Our PR$^\mathrm{C}$ description does not suffer the ambiguity between long notes and repeated occurrences of the same note and, except for time quantization, is completely lossless (i.e., a quantized MIDI pattern transformed into the corresponding PR$^\mathrm{C}$ tensor can be recovered exactly). Ad-

ditionally, it can be perceptually more robust to reconstruction errors. A further advantage is that all the information about a note is local, whereas in the case of PR, a convolutional network requires a wide receptive field to infer the note duration.

3. GENERATING PATTERNS WITH WASSERSTEIN AUTOENCODERS

Generative models that have been applied to music include various types of encoder/decoder architectures, different variants of generative adversarial networks (GAN) [24], as well as transformer based architectures [25]. Variational autoencoders (VAE) [19] have been used in systems like VRAE [26], GLSR-VAE [27] and MusicVAE [7] while GANs have been used in systems like C-RNN-GAN [28] MidiNet [21] and MuseGAN [17, 22].

Both with autoencoders and GANs, a network $G(z)$ (called either decoder or generator) is trained to map a latent or noise vector $z \in \mathbb{R}^{d_z}$ into a pattern. Here we are interested in autoencoder based approaches (see also Section 4.1 for a motivation). Among these approaches, VAEs are theoretically elegant and applicable to music generation. They are regularized by penalizing the expected KL divergence between the posterior $q(z|x)$ and a zero-mean Gaussian prior p_z, with the expectation being taken over training points. However, as nothing prevents different patterns being mapped to close latent codes, the decoder is sometimes asked to reconstruct different patterns from similar codes, resulting into a well known blurriness phenomenon in the case of images [29]. In the case of music patterns described as tensors, we observed that a form of "blurriness" also occurs, resulting in large clusters of notes being played together and sometimes in swarms of short notes that are never present in the training data. WAEs [18] avoid this problem by pushing the expectation inside the divergence, i.e., penalizing a divergence $\mathcal{D}$ between the prior q_z and the *aggregated* posterior $q_z(z) = \mathbb{E}_p \, q(z|x)$, where p is the data distribution. WAEs thus minimize, with respect to the parameters of the decoder, the quantity

$$\min_{q(z|x)} \mathbb{E}_p \, \mathbb{E}_{q(z|x)} \, c(x, G(z)) + \lambda \mathcal{D}(q_z, p_z) \qquad (1)$$

where c is a reconstruction loss and λ a hyperparameter to be fixed. In all our experiments we employed the Maximum Mean Discrepancy (MMD) [30] for $\mathcal{D}$ and a Gaussian prior for p_q, and we structured the encoder and the decoder as in the DCGAN [31] architecture. Note that unlike MuseGAN, which stacks bars over an additional tensor axis and uses 3D convolutional layers, tensors in PR^C can be processed by 2D convolutional layers. The input tensors are normalized in the $(-1, 1)$ range. The final layer of all our decoders has hyperbolic tangent output units. The mean squared error between reconstructions and input patterns was used in the optimization criterion during training.

The latent vector size, d_z, was adjusted with a trial-and-error approach, trying to find the smallest possible value yielding good quality interpolations. If d_z is too small the validation error is large but if d_z is too large, interpolations tend to create many patterns that are too close to those in the training set, in spite of a small validation error (which is therefore not an useful metric). Additionally, nearly empty patterns tend to appear in the middle of interpolations. We found $d_z = 3$ for ASF-4 and $d_z = 5$ for HP-10 and LPD-5 to be a reasonable compromise. The remaining hyperparameters were tuned using random search [32] guided by the validation set reconstruction error. We focused in particular on the learning rate, η, and the number of epochs T used in conjunction with the Adam algorithm; the number of layers n_l; the number of filters n_f and the size of filters, k, used in the the DCGAN encoder and decoder (strides were fixed to 2). Interestingly, large filter sizes $k = 8$ were found to perform better than standard smaller sizes as compact filters are not able to capture musical patterns and distant relationships between notes.

3.1 Converting Generated Tensors to MIDI

In the case of PR^C descriptions, "decoding" a MIDI pattern from the output tensor is almost straightforward (unlike the case of PR, where specialized GAN architectures have been introduced to avoid post-processing based on either hard thresholding or Bernoulli sampling [22]). First, predicted velocities and durations for each time t, pitch n, and instrument i, are rescaled in the range 0–127. Events whose rescaled predicted velocity $v(t, n, i) < 1$ or duration $d(t, n, i) < 1$ (i.e., non-audible notes) are discarded. Finally, we apply the following transformation (similar to a gamma correction) to obtain corrected velocities $v_c(t, n, i)$ as follows:

$$v_c(t, n, i) = \left\lfloor 127 \left(\frac{v(t, n, i)}{127} \right)^{\frac{1}{\gamma_i}} \right\rfloor \qquad (2)$$

where γ_i is an instrument specific correction factor ranging from 2.9 for drums to 4.0 for Rhodes. Durations are directly retrieved from the (rescaled) duration channel.

4. PSEUDO-SONGS

The following approach assumes that a generative model $G : z \in \mathbb{R}^{d_z} \mapsto x \in \mathbb{R}^{m \times q \times 2}$ from embeddings to patterns is available. Function G can be either the decoder of an autoencoder or the generator of a GAN. A pseudo-song is then generated by creating a trajectory of length T, $z_1, \ldots, z_T$, and applying G to each latent vector to produce a corresponding sequence of patterns.

4.1 Interpolations

When using autoencoders, we have the choice of picking a start pattern x_s and a goal pattern x_g (both from the test set) and use the encoder E to obtain $z_1 = E(x_s)$, $z_T = E(x_g)$. This for example allows users to produce a pseudo-song that moves smoothly from one genre to another. Musicians could even create ex-novo start and goal patterns with a particular purpose in mind. When using GANs to generate, this option is not available but z_1 and z_T can be sampled from $p(z)$ (with a less intentional result) or, alternatively, users may be given a set of pre-generated patterns

from which to pick the endpoints from the pre-images of the generator.

Two options are common for creating trajectories, linear interpolation: $z_t = \frac{t-T}{1-T}z_1 + \frac{1-t}{1-T}z_T$, $t = 2, \ldots, T-1$, and spherical interpolation:

$$z_t = \frac{\sin\left(\frac{1-t}{\theta(1-T)}\right)z_1 + \sin\left(\theta\frac{t-T}{1-T}\right)z_T}{\sin(\theta)} \qquad (3)$$

where $\theta = \arccos(z_1/\|z_1\|, z_T/\|z_T\|)$. The latter is preferable when $p(z)$ is Gaussian and d_z is large [33].

4.2 Swirls

In this approach (also applicable to GANs), latent trajectories are produced by taking real and imaginary parts of periodic complex-valued parametric functions of the form

$$f(t; a_l, b_l, c_l, d_l) = e^{ja_l t} - e^{jb_l t}/2 + je^{jc_l t}/3 + e^{jd_l t}/4$$

for random choices of (a_l, b_l, c_l, d_l), i.e., using $z_{2l,t} = \Re(f(t; a_l, b_l, c_l, d_l))$ and $z_{2l+1,t} = \Im(f(t; a_l, b_l, c_l, d_l))$, for $l = 1, \ldots, \lfloor d_z/2 \rfloor$, $t = 1, \ldots, T$.

4.3 Trajectory Smoothing

Equally spaced points in the embedding space do not necessarily correspond to equally spaced reconstructions in the pattern space. This essentially depends on how generative models allocate points in the pattern space to points in the embedding space. When creating pseudo-songs with either interpolations or swirls, this fact may lead in some cases to abrupt transitions and in some other cases to repetitive regions that might be musically uninteresting. To address this issue, we suggest to increase T beyond the desired length and then subsample the trajectory. Smoothness can be achieved by maximizing the minimum distance between consecutive reconstructions and constraining the final length to a desired integer L:

$$\max_{t_1, \ldots, t_L} \min_{i=1, \ldots, L-1} \delta(G(z_{t_i}), G(z_{t_{i+1}})) \qquad (4)$$
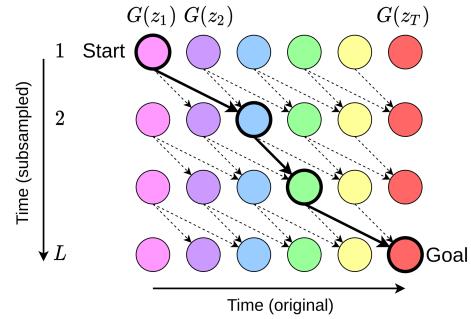
$$\text{s.t.} \quad 1 \le t_i < t_{i+1} \le T \quad i = 1, \ldots L-1 \qquad (5)$$

$$t_{i+1} - t_i \le H \quad i = 1, \ldots L-1 \qquad (6)$$

where δ is a distance function on patterns (e.g., the Euclidean distance on PR^{C} tensors) and H a lookahead horizon, i.e., the maximum allowed number of positions that may be skipped. Problem (4) can be reduced to an instance of the bottleck shortest path problem [34] on the $T \times L$ trellis with vertex set $\{(t, \tau), t = 1, \ldots, T; \tau = 1, \ldots, L\}$, edge set $\{((t, \tau), (t+1, \tau+s)), t = 1, \ldots, T; \tau = 1, \ldots, L, s = 1, \ldots, H\}$, and edge weights $w((t, \tau), (t+1, \tau+s)) = \delta(G(z_t), G(z_{t+s}))$ (see Fig. 2). The problem is solvable by a slightly modified version of the Dijkstra algorithm (where nodes in the frontier are labeled by their maximum step cost rather than the sum of the step costs) or by a faster algorithm based on bucketing [34].

5. DATASETS

We tested CONLON on three dataset. ASF-4 is a set of 910 patterns of four bars in three genres: *acid jazz*, *soul*

Figure 2. Trellis for trajectory smoothing. The horizon H is 2 in this example. Among all paths from Start to Goal, the highlighted path is the one whose smallest edge weight is maximum.

and *funk*. Each pattern has $K = 4$ tracks associated with a simple electro-acoustic quartet: drums, bass, Rhodes piano, and Hammond organ. HP-10 is a set of 968 patterns of four bars in two genres: *high-pop* and *progressive trance*. Each pattern has $K = 10$ tracks associated with the following instrument set: drums, bass, Rhodes, brass-synth, choir, dark-pad, guitar, lead, pad, and strings. Both ASF-4 and HP-10 have been especially composed by two professional musicians for this study. In both cases, composers were instructed to create coherent 120bpm patterns of four bars. All patterns in these datasets were subsequently quantized at 1/32th resolution, manually curated for mistakes (including fixing errors due to quantization), and finally transposed to either Cmaj or Amin to prevent tonality variations. The resulting PR^{C} tensors have size $128 \times N \times 2K$, where $N = 55$ for ASF-4 and $N = 60$ for HP-10. LPD-5 (cleansed version) was derived from the Lakh MIDI dataset [35] by Dong *et al.* [17] by retaining only songs with high matching scores to the Million Song Dataset. It contains 21,425 multitrack MIDI songs with $K = 5$ tracks and $N = 108$ pitches, where original tracks/instruments were merged into instrument families and cut by the authors of [17] into patterns consisting of two 4/4 bars. Automatic quantization at 1/48 was applied yielding tensors of size $192 \times 108 \times 10$.

Being manually curated, ASF-4 and HP-10 are much more *coherent* than LPD-5 in at least three ways. First, music style and genre is highly constrained, whereas LPD-5 contains a wide assortment of different genres. Second, instrument-role is well defined, i.e., the instrument that plays in a certain track always maintains its role across the whole dataset. In LPD-5, some heuristics have been applied by the authors of [17] to map instruments to the five tracks but in some cases very different instruments may collide on the same track. Third, the endpoints that demarcate patterns always cover homogeneous phrases, i.e., the phrase always begins on the first bar. In LPD-5, patterns are extracted by an automatic segmentation technique that cannot be equally reliable.

6. LISTENING EXPERIMENTS

To validate the CONLON approach, we conducted three listening experiments with a group of 69 musicians, re-

cruited by the authors with the help of colleagues teaching in several music educational institutes. Subjects mainly identified themselves as composers and producers (often in combination with instrumentalist/performer), working mostly in three (non mutually exclusive) genres: Classical, Contemporary and Electronic Dance Music. Over half the participants had more than 10 years of professional experience in music, only one less than 3 years.

Participants were told that the survey was part of a research project on the generation of music with machine learning techniques, with the long-term aim of developing new tools for music production and performance. They participated through an online survey service [1] that allows for questions with audio materials. The experiments consisted of two types of tasks.

Comparison Subjects listen to a pair of short music tracks (64 bars, 130s), and indicate which of the two tracks is most usable in the context of mainstream music production (forced choice). The scenario they are asked to keep in mind is that they would receive the tracks as a midi file for inclusion and editing in the production of a mainstream song in their own DAW.

Analysis Subjects listen to a single track, an excerpt (64 bars) from a longer piece and assess the musical development of the composition over time, with regards to four aspects: Harmony, Rhythm, Melody, and Interplay of instruments, each judged on a 5-point Likert scale (from "very incoherent" to "very coherent"). To obtain further feedback, we asked the subjects to comment on good points of the composition and points for improvement.

Answers in the comparison task were converted to ranks for the tracks in a pair (i.e. rank is 1 for the preferred track and 2 for the other). We then computed the mean rank of the tracks across subjects. Following [36], we employed Kendall's Coefficient of Concordance (W) [37], to analyse the level of agreement among subjects, along with a commonly used significance test against the null hypothesis of no agreement [38].

In the following we describe three experiments aiming at testing specific hypotheses relating CONLON to MuseGAN, PR to PR^C, and the usability of pseudo-songs. For all models we set the interpolation length $T=64$, the desired length $L=16$, and the widest-path horizon $H=20$.

---

[1] http://www.surveygizmo.com

Method	HP-10	LPD-5
CONLON	1.17	1.45
MuseGAN	1.83	1.55
Concordance	0.64	0.01
Significance	$p<0.0005$	ns

Table 1. Mean ranks assigned by subjects to the usability of interpolations generated with our system (CONLON) and MuseGAN [22]. $m = 75$ pairs were ranked.

Description	ASF-4	HP-10	LPD-5
PR^C	1.08	1.31	1.5
PR	1.92	1.69	1.5
Concordance	0.72	0.15	0
Significance	$p<0.0005$	$p<0.001$	ns

Table 2. Mean ranks assigned by subjects to the usability of pseudo-songs generated with PR^C and PR descriptions. $m = 78$ pairs were ranked.

	Aspect			
	Harmony	Rhythm	Melody	Interplay
Coherent	49%	67%	42%	51%
Neutral	35%	20%	29%	20%
Incoherent	16%	13%	29%	29%
Significance	$p<.005$	$p<.0005$	$p<.005$	$p<.0005$

Table 3. Coherence of CONLON pseudo-songs as judged by subjects, with respect to harmony, rhythm, melody, interplay of instruments. $m = 69$ judgements were collected.

6.1 Comparing CONLON and MuseGAN

To substantiate that CONLON is more usable in music production than previous approaches, we tested *Hypothesis 1: Musicians find pseudo-songs generated with WAEs and PR^C descriptions more useable in music production than pseudo-songs generated with the MuseGAN model and PR (other factors being equal).* A WAE-model was trained on PR^C representations of the datasets HP-10 and LPD-5, and a MuseGAN model on PR representations of datasets HP-10 and LPD-5 to generate interpolations. On the same data, we trained a MuseGAN model with binary neurons [22] using the implementation at https://github.com/salu133445/musegan. Subjects were given pairs of matching interpolations to compare, differing in the approach used, but with the same start, goal and length. Six pairs (three for each dataset) were presented for comparison to 25 subjects, producing a total of 75 observations per dataset.

Subjects generally judged pseudo-songs generated by CONLON to be more usable than pseudo-songs generated with the MuseGAN approach (for 5 out of the 6 pairs). But whereas the difference in ranking is clear and concordance among participants is significant for the three pairs on the HP-10 dataset, differences were small and concordance not significant among subjects for pseudo-songs generated from LPD-5. Table 1 shows the aggregated mean ranking per dataset.

6.2 Comparing PR and PR^C

To investigate whether part of the improvement over previous approaches is due to the representation, we tested *Hypothesis 2: Musicians find pseudo-songs generated with PR^C descriptions more useable in music production than pseudo-songs generated with PR descriptions (other factors being equal).* A WAE-model was trained on PR

	ASF-4				HP-10				LPD-5			
	P	R	V	D	P	R	V	D	P	R	V	D
PR	6.1	51.1	32.2	2.5	4.1	58.9	28.8	3.1	1.4	89.7	41.0	5.5
PR^C	32.1	53.9	24.3	1.0	37.0	58.0	23.4	1.8	35.5	60.2	19.5	2.6

Table 4. Test set precision (P), recall (R), mean absolute errors on velocity (V) and duration (D) for PR and PR^C .

and PR^C representations of the datasets ASF-4, HP-10 and LPD-5 to generate interpolations. 26 subjects were given nine pairs of matching interpolations to compare (differing only in representation used), three for each dataset, resulting in 78 observations per dataset.

Subjects generally judged pseudo-songs generated with PR^C representations to be more usable than pseudo-songs generated with PR representations (for 8 out of the 9 pairs). The difference in ranking is clear and concordance among participants is significant for the 6 pairs on the ASF-4 and HP-10 datasets, but differences were small and concordance not significant for the 3 pairs generated on different representations of the LPD-5 dataset. Table 2 shows the aggregated mean ranking and cross-subject concordance per dataset.

6.3 Analyzing Development over Time

To validate the way patterns are chained together by CONLON, we tested *Hypothesis 3: Musicians find the development over time of pseudo-songs generated with WAEs and PR^C description coherent rather than incoherent (in terms of harmony, rhythm, melody and interplay between instruments).* A WAE-model was trained on a PR^C representation of datasets ASF-4 and HP-10 to generate swirls. All subjects were given the analysis task for a swirl, resulting in 69 observations per aspect.

Subjects generally judged the coherence of pseudo-songs on the positive side of the scale. For the three swirls presented in the experiment, each with four aspects, the median for all aspects lies at "somewhat coherent" (8 out of 12) or "neutral" (4 out of 12). The rating "very coherent" is reached for all aspects, the rating "very incoherent" in 10 out of 12. For all swirls, rhythm is the aspect with the highest coherence rating. Table 3 shows the aggregated answers for the three swirls, recoded to a 3-point scale.

7. QUANTITATIVE EVALUATIONS

When only a limited amount of human expert time is available for surveys, it becomes difficult to cover all different dimensions on which alternative methods can be compared. Rather that allowing non experts in our surveys, it may be preferable to complement human evaluation with a number of automatically computed metrics [39]. Here we consider reconstruction error and note shattering.

7.1 Reconstruction error

Here we complement human evaluations with some automatically computed metrics that are derived from test set reconstructions and are applicable to autoencoder-based methods. In particular, we aim to compare WAEs fed by

PR vs WAEs fed by PR^C . Precision and recall are defined on the binary classification problem where the ground truth consists of Bernoulli variables $y(t, n, i) = 1$ if there is a note-on event at time t for note n and instrument i. For these metrics we considered as predictions the binary quantities $\hat{y}(t, n, i) = 1$ if the reconstructed value of the velocity at position (t, n, i) is above the smallest velocity encountered in the training set. In the case of PR description, the predicted note-on event was the first element in the merged row of consecutive predictions. We further considered the mean absolute errors in predicting velocities (in the range $[0 - 127]$) and durations (in units of 1/32ths of bar). Test set results comparing PR and PR^C (everything else being equal) are reported in Table 4.

7.2 Note Shattering

Results in Table 4 indicate that PR yields good recall but very low precision, and has a higher error on both velocity and duration. This can be partially explained by the presence of a high number of shattered notes. To verify this hypothesis we computed the note number growth due to shattering as follows. For each note in the ground truth, identified by the triplet (n, i, T), being n the pitch, i the instrument, and $T = [t_{\text{ON}}, t_{\text{OFF}}]$ the temporal interval, we counted the number of notes in the reconstruction that have the same pitch n and instrument i, and whose note-ON time falls within T. We then summed these counts over all notes in the test set. In the absence of shattering, the total count equals the original number of notes. We found that WAE-PR increased the number of notes by 19%, 12%, 38% on ASF-4, HP-10, and LPD-5, respectively. By comparison, the increase factors were only 5%, 3%, and 10% in the case of WAE-PR^C .

8. CONCLUSIONS

CONLON combines the new PR^C data description with Wasserstein autoencoders and generation strategies based on optimized interpolation and swirling to produce pattern-based pseudo-songs. When trained on coherent datasets, the generated material is musically coherent and potentially useful in music production by professional musicians.

Pseudo-songs can sound like directed musical flows, but this is entirely due to the properties of the embedding space, longer-term structure is not considered. In that sense, interpolating and swirling are closer to improvisation than to composition. A natural next step is to label dataset patterns with structural categories (e.g. verse, chorus) and introduce form via mechanisms of conditioning.

9. REFERENCES

[1] L. A. Hiller and L. M. Isaacson, *Experimental Music: Composition with an Electronic Computer*. McGraw-Hill, 1959.

[2] I. Xenakis, "Musiques formelles: nouveaux principes formels de composition musicale," *La Revue musicale*, no. 253–254, 1963, paris: Editions Richard-Masse.

[3] J.-P. Briot, G. Hadjeres, and F. Pachet, *Deep Learning Techniques for Music Generation, Computational Synthesis and Creative Systems*. Springer Nature, 2019.

[4] F. Carnovalini and A. Rodà, "Computational creativity and music generation systems: An introduction to the state of the art," *Frontiers in Artificial Intelligence*, vol. 3, Apr 2020.

[5] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016.

[6] S. Dieleman, A. van den Oord, and K. Simonyan, "The challenge of realistic music generation: modelling raw audio at scale," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 8000–8010.

[7] A. Roberts, J. H. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 4361–4370.

[8] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[9] H. H. Mao, T. Shin, and G. W. Cottrell, "Deepj: Style-specific music generation," in *12th IEEE International Conference on Semantic Computing*, 2018, pp. 377–382.

[10] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 685–692.

[11] H. Lim, S. Rhyu, and K. Lee, "Chord generation from symbolic melody using BLSTM networks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017, pp. 621–627.

[12] I. Malik and C. H. Ek, "Neural translation of musical style," *CoRR*, vol. abs/1708.03535, 2017.

[13] S. Dai, Z. Zhang, and G. Xia, "Music style transfer issues: A position paper," *CoRR*, vol. abs/1803.06841, 2018.

[14] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: modeling dynamics and instrumentation of music with applications to style transfer," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018, pp. 747–754.

[15] M. A. Boden, *The Creative Mind: Myths and Mechanisms*. Routledge, 2004.

[16] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[17] H. Dong, W. Hsiao, L. Yang, and Y. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 34–41.

[18] I. O. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf, "Wasserstein auto-encoders," in *6th International Conference on Learning Representations*, 2018.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations*, 2014.

[20] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, 2017.

[21] L. Yang, S. Chou, and Y. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017, pp. 324–331.

[22] H. Dong and Y. Yang, "Convolutional generative adversarial networks with binary neurons for polyphonic music generation," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018, pp. 190–196.

[23] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Back-propagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[24] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2672–2680.

[25] C. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *7th International Conference on Learning Representations*, 2019.

[26] O. Fabius, J. R. van Amersfoort, and D. P. Kingma, "Variational recurrent auto-encoders," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[27] G. Hadjeres, F. Nielsen, and F. Pachet, "GLSR-VAE: geodesic latent space regularization for variational autoencoder architectures," in *IEEE Symposium Series on Computational Intelligence*, 2017, pp. 1–7.

[28] O. Mogren, "C-RNN-GAN: continuous recurrent neural networks with adversarial training," *CoRR*, vol. abs/1611.09904, 2016, constructive Machine Learning Workshop at NIPS 2016, Barcelona.

[29] A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 658–666.

[30] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD gans," in *6th International Conference on Learning Representations*, 2018.

[31] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *4th International Conference on Learning Representations*, 2016.

[32] J. Bergstra and Y. Bengio, "Random search for hyperparameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[33] T. White, "Sampling generative networks: Notes on a few effective techniques," *CoRR*, vol. abs/1609.04468, 2016.

[34] V. Kaibel and M. Peinhardt, "On the bottleneck shortest path problem," Otto-von-Guericke-Univ. Magdeburg, Magdeburg, Germany, Tech. Rep., 2006.

[35] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-Midi alignment and matching," Ph.D. dissertation, Columbia University, 2016.

[36] A. Novello, M. F. McKinney, and A. Kohlrausch, "Perceptual evaluation of music similarity," in *Proceedings of the 7th International Conference on Music Information Retrieval*, 2006, pp. 246–249.

[37] M. Kendall, *Rank Correlation Methods (5th ed.)*. London: Charles Griffin, 1975.

[38] M. Kendall and J. D. Gibbons, *Rank Correlation Methods (5th ed.)*. New York, NY: Oxford University Press, 1990.

[39] L. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.

LESS IS MORE: FASTER AND BETTER MUSIC VERSION IDENTIFICATION WITH EMBEDDING DISTILLATION

Furkan Yesiler[1]   **Joan Serrà**[2]   **Emilia Gómez**[3,1]

[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
[2] Dolby Laboratories, Barcelona, Spain
[3] European Commission, Joint Research Centre, Seville, Spain

`furkan.yesiler@upf.edu`

ABSTRACT

Version identification systems aim to detect different renditions of the same underlying musical composition (loosely called cover songs). By learning to encode entire recordings into plain vector embeddings, recent systems have made significant progress in bridging the gap between accuracy and scalability, which has been a key challenge for nearly two decades. In this work, we propose to further narrow this gap by employing a set of data distillation techniques that reduce the embedding dimensionality of a pre-trained state-of-the-art model. We compare a wide range of techniques and propose new ones, from classical dimensionality reduction to more sophisticated distillation schemes. With those, we obtain 99% smaller embeddings that, moreover, yield up to a 3% accuracy increase. Such small embeddings can have an important impact in retrieval time, up to the point of making a real-world system practical on a standalone laptop.

1. INTRODUCTION

The concept of music versions is as old as the concept of music itself. Before the existence of recorded music, listening to a piece mostly meant listening to a version of it. Nowadays, with the advancements in recording technologies, most music we listen to comes in recorded form. Nevertheless, musicians keep creating their own versions of existing songs for various reasons, including commercial ones (for example, to attract new audiences), political ones (to connect people or make a stance), and artistic ones (to re-imagine a song with a personal touch).

Version identification (VI) is the task of automatically detecting different renditions of the same underlying musical composition. VI systems are mainly focused on retrieval, aiming to find all renditions of a query song in a reference database. Although creating new versions is common practice, defining the characteristics that enable us to perceive the links connecting different renditions of the same piece is not a straightforward task [1, 2]. Based on quantitative evidence, many successful systems exploit tonal and melodic descriptors that are invariant to the typical differences among versions, including the differences in timbre, tempo, structure, lyrics, and so on [3–5]. By further processing such descriptors, the ultimate goal is to obtain representations that allow inferring links among versions.

The accuracy-scalability trade-off stands as a key challenge in version retrieval. The early, alignment-based systems [6, 7] incorporated musical know-how to capture the similarities among versions, resulting in strong performances. However, due to the scarcity of data, and their dependence on complex representations and computationally-intensive algorithms, they ended up in limited evaluation environments and, ultimately, not being suitable for industrial-size databases. With the release of the Million Song Dataset [8], researchers were further encouraged to address the scalability issue by exploring embedding-based systems that encode songs into more compact vectors. Although offering significant improvements for the scalability aspect, the performance of such systems failed to match their predecessors [9]. Recent embedding-based systems that use deep learning techniques pave the way to encapsulate the similarities among versions in ways that are both efficient and accurate [5, 10–12].

The main use case of version retrieval in commercial settings is to detect copyright infringement cases in media streaming platforms and live performance venues or events. Such application scenarios require having fast and scalable solutions. For example, more than 500 h of video content are uploaded to YouTube every minute [1], and handling the music licensing aspect of that requires having accurate and scalable systems that can identify the cases where a video includes a copyrighted piece of music.

In this paper, we investigate a number of ways to improve the scalability of existing embedding-based VI systems that use neural networks as encoders. Specifically, our goal is to reduce the size of embedding vectors without compromising the accuracy of the systems. Since embeddings can be pre-computed, reducing their size is crucial to improve data storage and, more importantly, retrieval

---

[1] https://www.cnbc.com/2018/03/14/with-over-1-billion-users-heres-how-youtube-is-keeping-pace-with-change.html

time. For this purpose, we consider three core state-of-the-art strategies, namely unsupervised dimensionality reduction, neural network pruning, and knowledge distillation. Apart from introducing a number of techniques from other fields to VI research, we also consider a novel knowledge distillation loss for metric learning, which aims to optimize a clustering evaluation metric. Moreover, inspired by transfer learning applications, we propose a technique called latent space reconfiguration, to show that learning a compact and efficient latent space is facilitated by using a pre-trained feature extractor due to its stronger priors, compared with using a randomly-initialized one. Our experiments suggest that the performance of a pre-trained network can be preserved, or even improved, while shrinking the embedding vectors down to less than 1% of their original sizes. We evaluate our approach on a publicly-available test set, and share our code, instructions for using a newly-contributed training dataset and supplementary materials (SM) on Github [2].

2. RELATED WORK

2.1 Version identification

Like many other systems in music information retrieval (MIR), VI systems extract audio descriptors to obtain relevant information from signals, including mid-level ones, such as pitch class profiles (PCP) [7, 13–16] and predominant melody [4, 10, 17], or low-level ones, such as the constant-Q transform (CQT) [18–20]. To achieve invariance against the changes in musical characteristics, further processing steps have been proposed, including beat-synchronous features for handling tempo variations [13,15, 21], and the optimal transposition index for handling pitch transpositions [6, 7, 14, 15]. Many rule-based VI systems use alignment algorithms to then compare these representations, resulting in long retrieval times.

Embedding-based VI systems are designed to obtain compact representations that speed up the retrieval phase. Compact embeddings reduce the required storage and facilitate similarity estimation through the use of efficient nearest-neighbor libraries implementing common metrics like Euclidean or cosine distances. Early attempts of such systems use techniques like the 2D Fourier transform, principal component analysis, and linear discriminant analysis for encoding and dimensionality reduction operations [21, 22].

Current deep learning-based systems learn non-linear transformations that map the feature representations into embedding vectors of various sizes, ranging from 300 to 16,000. Xu et al. [23] and Yu et al. [11] train their convolutional networks with a classification loss, but obtain the embedding vectors to use in the retrieval phase from the penultimate layer of their networks. Doras and Peeters [10], and Yesiler et al. [5] formulate the network training as a metric learning setting, in which they use the triplet loss for optimizing distances among training samples.

[2] https://github.com/furkanyesiler/re-move

2.2 Metric learning

This line of research is concerned with learning functions that produce low distance values between semantically similar data points, and high values otherwise. The early approaches include learning Mahalanobis distances [24] with linear [25, 26] or non-linear [27] transformations. Parametrizing such transformations with neural networks was pioneered by Chopra et al. [28] and Salakhutdinov and Hinton [29], and the research domain combining these two concepts is often called deep metric learning.

Deep metric learning methods offer new solutions regarding how to exploit the semantic relations among data, often by formulating or revising loss functions. The triplet loss [30], similar to LMNN [26], manipulates the distances between genuine and impostor pairs with an energy-based approach. The ProxyNCA loss [31], similar to NCA [25], and NormalizedSoftmax loss [32], similar to cross entropy loss, maximize the likelihoods of samples being close to particular class proxies. The group loss [33] incorporates the idea that similar elements should belong in the same class by using replicator dynamics [34]. Although there is a variety of deep metric learning losses, each one with distinctive advantages, the triplet loss variants remain the popular (and almost unique) choice in MIR research.
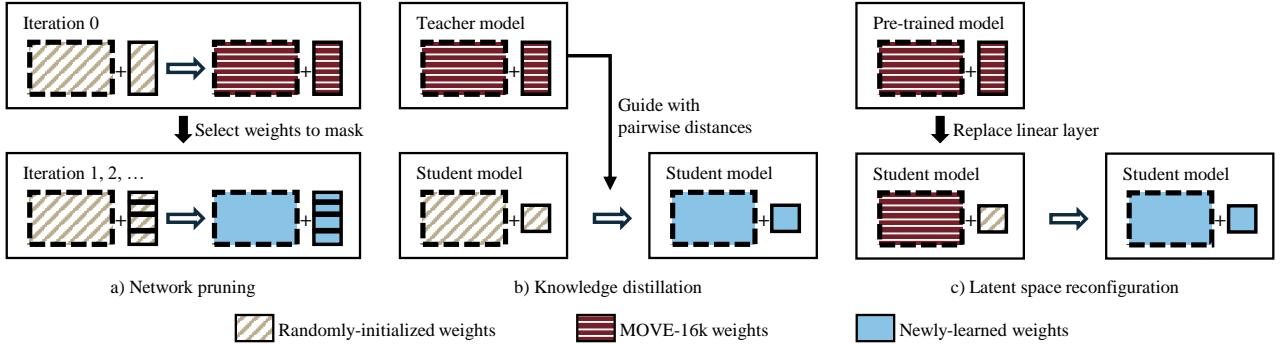
2.3 Model reduction

2.3.1 Neural network pruning

Pruning a large neural network can preserve the original performance while eliminating more than 90% of its weights [35–38]. The main challenge is to identify the importance of connections and weights, and previous techniques explored the use of absolute weight magnitudes [36–38] and the Hessian of the loss function [35]. Pruning operations can be performed layer- or network-wise, in a one-shot or an iterative fashion, and combined with quantization or clustering. To the best of our knowledge, network pruning has not been considered for VI, nor further explored in MIR systems in general.

2.3.2 Knowledge distillation

Bucilă et al. [39], and later Hinton et al. [40], explored the idea where a small neural network (the student model) is trained with the guidance from a wide, deep, and better-performing network (the teacher model). In the metric learning context, some works explored this idea with a slightly changed formulation: Classical knowledge distillation methods use teacher networks to guide the students on individual examples, but metric learning methods exploit similarity relations among samples. For this, researchers proposed methods that match a number of properties between the embeddings obtained from the teacher and the student models, including the ranks of retrieved samples [41], distances between samples [42], class likelihood distributions [43], and absolute positions of embeddings in the latent space [44]. With few exceptions [45,46], distillation techniques are largely under-explored in MIR, and we believe that no attempt has been done within VI.

Figure 1: An overview of neural network-based embedding distillation methods. The hollow arrows denote training process, the boxes with dashed and with solid outlines denote feature extractors and fully-connected layers, respectively.

3. METHODOLOGY

3.1 Embedding distillation

Our study focuses on a set of techniques for improving the scalability of existing VI systems in the retrieval phase by reducing the size of the embeddings, rather than building a novel network architecture.[3] We hypothesize that a high-capacity encoder is still needed to extract the essential information from complex and noisy signals such as current tonal representations. However, once a reliable encoder is obtained, it can be used for training a second model that outputs embeddings with a lower dimensionality, ideally without compromising accuracy. Due to the goal of having smaller embeddings that yield a similar performance, we call this set of methods embedding distillation techniques.

3.2 Data

Our models are developed with the Da-TACOS training set that we make available under Creative Commons BY-NC-SA 4.0 license together with this publication. It features a training partition of 83,904 songs in 14,499 cliques (or unique works), and a validation partition of 14,000 songs in 3,500 cliques. The annotations for the songs are obtained using the API of `secondhandsongs.com`. We share the crema-PCP features [47] and the related metadata. Further detail on the contributed dataset is available in SM.

3.3 Model architecture and training details

Our methods require to start from a pre-trained and sufficiently reliable model. For this, we take advantage of the publicly-available MOVE model [5], together with its pre-trained weights. Nonetheless, we believe all methods introduced here can be applied to other embedding-based systems using neural networks (initial results are available in SM).

MOVE uses crema-PCP features $\mathbf{X} \in [0,1]^{12 \times T}$ as input, where T is the number of frames, pre-computed with non-overlapping windows of 93 ms. The model outputs embedding vectors $\mathbf{v} = f(\mathbf{X}) \in \mathbb{R}^d$, where d is the embedding size. The original work reports results for d between 128 and 32 k, and shows a clear accuracy drop for $d < 2048$ (the final model employs a rather high dimensionality $d = 16$ k). In contrast, the dimensionalities we consider in this work are $d = \{128, 256, 512, 2048\}$.

To find a suitable learning rate and an optimizer for each experiment setting, we perform a grid search over both stochastic gradient descent and Ranger[4] optimizers and initial learning rates in $\{0.0001, 0.001, 0.01, 0.1\}$, using our validation set. The full training lasts for 70 epochs, and we decrease the learning rate by a factor of 10 at epochs 50 and 60. We save the model weights that result in the best performance on the validation set. The remaining training details and design decisions follow the ones made by the MOVE authors [5]. All neural network models are trained using PyTorch [48], and the hyper-parameter values used for each experiment can be found at our repository.

3.4 Methods for embedding distillation

3.4.1 Classical unsupervised techniques

Before going into complex solutions, we investigate the benefits of using classical dimensionality reduction techniques for embedding distillation. For this, we use principal component analysis (PCA), independent component analysis (ICA), and Gaussian random projection (GRP) techniques. Each model is fit using the training set embeddings obtained with MOVE-16k, and applied to the evaluation set embeddings. We use the implementations from the scikit-learn library [49] and change only the number of target components.

3.4.2 Pruning

Based on the approach of Han et al. [37], we study whether we can prune the dimensions of the latent space constructed by MOVE-16k in an iterative way. Although pruning the weights of all layers throughout the network is the most common practice, the underlying idea can be applied

---

[3] To illustrate the benefits of using smaller embeddings, consider computing distances between a query and a reference database with 10 M embeddings. This takes us (with a simple brute-force, double-loop Euclidean distance function) 0.32 s using $d = 256$, but the elapsed time increases up to 4.75 s for $d = 4k$, and to 18.43 s for $d = 16k$ (the embedding size of MOVE [5]). Although the absolute values are subject to change based on computational resources, for real-world applications on portable devices, such differences in magnitude for the retrieval time (from 0.32 to 18.43 s) may drastically affect user experience and product appeal.

[4] https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer

to only the final linear layer of the model in order to obtain embeddings with fewer dimensions. Denoting the weights of the linear layer of MOVE-16k as $\mathbf{W} \in \mathbb{R}^{d \times f}$, where d is the size of the embeddings and f is the number of input connections to the linear layer, we compute the mean of the absolute values per row for $\mathbf{W}$ and sort the rows based on these mean values. At the end of each iteration m ($m \in \{0, 1, ...\}$), the weights of the top 50% rows are restored to their initial values from Iteration 0 and retrained. The weights of the bottom 50% rows are zeroed-out and not considered for the next iterations (Figure 1(a)) .

3.4.3 Knowledge distillation

This set of experiments consider MOVE-16k as a teacher model, and our goal is to train from scratch a student model of the same size but with a lower embedding dimensionality. Our approach is formulated in a deep metric learning setting where the guidance of the teacher model is shaped by the distances among samples (Figure 1(b)). In the experiments described next, the weights of the teacher model are frozen, and the weights of the student model are initialized randomly.

Distance matching — Perhaps the most intuitive way of guiding the student model is to match the distances obtained from the student with the ones from the teacher, allowing the two models to have different embedding sizes. In our implementation, we pass the samples in each mini-batch to both models, compute in-batch pairwise distances, and use the mean absolute error between the pairwise distance matrices from the teacher model and the student model to train the latter:

$$L_i^{\mathrm{DM}} = \sum_j \left| D(\mathbf{v}_i^{\mathrm{s}}, \mathbf{v}_j^{\mathrm{s}}) - D(\mathbf{v}_i^{\mathrm{t}}, \mathbf{v}_j^{\mathrm{t}}) \right|, \qquad (1)$$

using

$$D(\mathbf{v}_i, \mathbf{v}_j) = \frac{1}{d} \left\| \mathbf{v}_i - \mathbf{v}_j \right\|^2, \qquad (2)$$

where $\| \ \|$ represents the Euclidean norm, and $\mathbf{v}_i^{\mathrm{s}}$ and $\mathbf{v}_i^{\mathrm{t}}$ the embeddings of song i obtained with the student and teacher models, respectively.

Cluster matching — Our second knowledge distillation scheme aims to obtain a student model that constructs clusters with both low intra-class and high inter-class distances. Assuming the teacher model holds this desired property, we take advantage of this information to guide the student model. To the best of our knowledge, this distillation criterion has not been explored in previous deep metric learning research.

Our criterion exploits internal cluster evaluation metrics [50]. In the experiments reported here, we use the Davies-Bouldin (DB) index [51], but other cluster evaluation metrics can be used:

$$L_i^{\mathrm{DB}} = \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{D(c_i, c_j)} \right), \qquad (3)$$

where σ_i denotes the average intra-class distance, computed with a suitable distance measure D, and c_i denotes the centroid for class i. The DB index yields low values for configurations that have low intra-class and high inter-class distances.

In our implementation, we pre-compute the class centroids using the MOVE-16k embeddings from the entire training set. To match the dimensions of the centroids with the student model embeddings, we train a linear projection simultaneously with the student model. The intra-class and inter-centroid distances are computed using only the samples present in the mini-batch and their respective centroids. After computing DB scores for each class in the mini-batch, we average them to obtain the final loss value.

3.4.4 Latent space reconfiguration

Transfer learning applications take advantage of the strong priors learned by the feature extractor part of successful, high-capacity models that are trained on large datasets. Inspired by this idea, we hypothesize that, by using the feature extractor of a pre-trained model, we can obtain a better-structured and lower-dimensional latent space that cannot be obtained by training a randomly-initialized model from scratch.

To test this idea, we use the pre-trained convolutional layers of MOVE-16k as the feature extractor, remove the final linear layer, and learn a new latent space with a randomly-initialized linear layer using a metric learning loss function (Figure 1(c)). Note that the original MOVE-16k model is trained with a triplet loss, meaning that it learned a distance metric parametrized by a neural network. Our approach uses the non-linear part of that metric, and 'reconfigures' the latent space and the distance metric by optimizing a second loss function (hence the name latent space reconfiguration). Our motivation is based on two assumptions: (1) training losses play an important role in shaping the latent space where the embeddings lie, and (2) embeddings with lower dimensionalities may be sufficient to successfully represent semantically meaningful information, as long as the dimensions are effectively utilized. Note that although this technique follows the same procedure as transfer learning, the latter requires, by definition, distinct source and target tasks (or datasets), which is not the case for the proposed technique. Focusing on metric learning schemes, the term latent space reconfiguration denotes the process of starting with an already learned distance metric and modifying it to represent the semantic relations in a more compact embedding space.

In our experiments, we consider 4 loss functions which are explained below. The weights of the feature extractor are frozen during the first epoch and updated with a lower learning rate during the rest of the training. Batch normalization is applied after the linear layer as in MOVE-16k. Apart from using the loss functions below for latent space reconfiguration, we also use them individually and train models from scratch with the same settings to set baseline models.

Triplet loss — We follow the triplet loss formulation used by Yesiler et al. [5]. Distances among vectors are computed using D as specified in Eqn (2):

$$L_i^{\mathrm{T}} = \max \left(D \left(\mathbf{v}_i, \mathbf{v}_+ \right) - D \left(\mathbf{v}_i, \mathbf{v}_- \right) + m, 0 \right), \quad (4)$$

where $\mathbf{v}_i$ corresponds to the anchor, $\mathbf{v}_+$ to the positive sample, $\mathbf{v}_-$ to the negative sample, and $m = 1$ is a margin hyper-parameter. For selecting which triplets to use, we follow the hard-positive, hard-negative mining strategy used by the authors.

ProxyNCA loss — Our implementation of ProxyNCA loss [31] also uses the normalized squared Euclidean distance metric from Eqn (2). Every class in our training set is represented with one proxy vector that is initialized randomly and trained simultaneously with the model parameters. In mathematical notation, the ProxyNCA loss can be expressed as:

$$L_i^{\mathrm{P}} = -\log\left(\frac{\exp(-D(\mathbf{v}_i, y))}{\sum_{z \in Z} \exp(-D(\mathbf{v}_i, z))}\right), \qquad (5)$$

where $y \in \mathbb{R}^d$ denotes the proxy vector for the class of $\mathbf{v}_i$ and Z denotes the set of proxies for all classes different than the one of $\mathbf{v}_i$.

NormalizedSoftmax loss — As proposed in [32], we implement this function using the cosine distance. We randomly initialize one proxy per class and update their parameters at each training step. We use

$$L_i^{\mathrm{N}} = -\log\left(\frac{\exp(\langle \mathbf{v}_i, y \rangle / \tau)}{\sum_{z \in Z} \exp(\langle \mathbf{v}_i, z \rangle / \tau)}\right), \qquad (6)$$

where $\langle\ \rangle$ denotes cosine similarity, $y \in \mathbb{R}^d$ the proxy for the positive class, Z the set of proxies for all classes, and $\tau = 0.05$ the temperature parameter.

Group loss — Following the approach of [33], we use Pearson's correlation coefficient as the similarity metric and replace the negative values with 0. We perform three iterations for refining the class probabilities and, unlike the original implementation, we select one anchor per class in the mini-batch. The main loss is regular cross-entropy:

$$L_i^{\mathrm{G}} = -\log\left(\frac{\exp(l_i^c)}{\sum_{t \in C} \exp(l_i^t)}\right), \qquad (7)$$

where l_i^c denotes the logit of sample i with respect to its positive class c, and C denotes the set of all classes in the training set. However, in group loss, logits are updated with replicator dynamics using pairwise similarities [33].

4. RESULTS

4.1 Evaluation methodology

For development, we use the newly available dataset mentioned in Section 3.2 and detailed in SM. Results are then evaluated on Da-TACOS benchmark subset [9], which contains a non-intersecting set of cliques with respect to our training and validation data. Da-TACOS contains 1,000 cliques with 13 songs each and 2,000 noise songs that do not belong to any other clique and are not queried. Following common practice, we report the performance of our models using mean average precision (MAP) and mean rank of the first relevant item (MR1) metrics.

Method	d			
	128	256	512	2048
Baselines (no reduction, training from scratch)				
Triplet	0.459	0.469	0.478	0.487
ProxyNCA	0.168	0.185	0.212	0.206
NormalizedSoftmax	0.445	0.470	0.475	0.422
Group	0.265	0.271	0.269	0.271
Unsupervised				
PCA	0.494	**0.507**	**0.507**	**0.507**
ICA	0.456	0.425	n/a	n/a
GRP	0.429	0.465	0.485	**0.502**
Knowledge distillation				
Distance matching + Triplet	0.492	**0.499**	**0.503**	**0.500**
Cluster matching + Triplet	0.424	0.471	0.465	0.455
Latent space reconfiguration				
Triplet	0.485	0.491	0.494	**0.506**
ProxyNCA	0.424	0.465	0.485	**0.502**
NormalizedSoftmax	**0.513**	**0.524**	**0.525**	**0.525**
Group	0.465	0.483	**0.495**	**0.511**

Table 1: MAP for different embedding sizes d when training from scratch (top) and when using pre-trained models and embedding distillation (middle-bottom). MAPs for the original MOVE-4k and MOVE-16k baselines are 0.495 and 0.507, respectively (values equal to or above MOVE-4k are highlighted in bold).

4.2 Embedding distillation experiments

Table 1 presents the exhaustive list of results for the methods described in Section 3. The baseline results (top block) show that, when training from scratch, changing the loss function of a network causes significant accuracy differences. It should be noted that all alternative losses we consider achieve state-of-the-art performances in computer vision datasets. Nevertheless, our results suggest that they may not generalize across other types of data or tasks, or that they may be oversensitive to hyper-parameters or specific architectural decisions.

For unsupervised dimensionality reduction (second block of Table 1), we find that PCA successfully projects the information contained in MOVE-16k embeddings, even when using 256 dimensions. This suggests that, although achieving state-of-the-art performance, MOVE-16k embeddings contain redundant information that can be drastically compressed. GRP reaches a similar performance as PCA with $d = 2048$, but the resulting performance decreases when using lower-dimensional embeddings.

The initial experiments on pruning reached the same performance as MOVE-16k after one iteration, that is, after reducing the dimensionality by 50%. However, further pruning iterations drastically decreased MAP, up to the point of yielding non-useful embeddings. Therefore, we decided to stop iterating and not to report the corresponding results.

Among the considered knowledge distillation techniques (third block, Table 1), the additional distance matching loss clearly increases the model performance

	d	MAP	MR1
2DFTM [21]	50	0.275	155
Dmax [16]	5.5 k	0.322	132
SiMPle [14]	2.2 k	0.332	142
Qmax [7]	5.5 k	0.365	113
Qmax* [52]	5.5 k	0.373	104
EarlyFusion [15]	8.5 k	0.426	116
LateFusion [16]	5.5 k	0.454	177
MOVE [5]	4 k	0.495	42
MOVE [5]	16 k	0.507	40
Re-MOVE	**256**	**0.524**	**38**

Table 2: Comparison with existing VI systems using Da-TACOS (taken from [9]). When not explicit, embedding sizes d are estimated for a song duration of 3.5 min (see text). Results for the proposed methodology are highlighted in bold.

Figure 2: MAP with respect to embedding dimensionality d for Re-MOVE (red stars), MOVE (blue squares), and other existing approaches (blue circles). Notice the logarithmic axis.
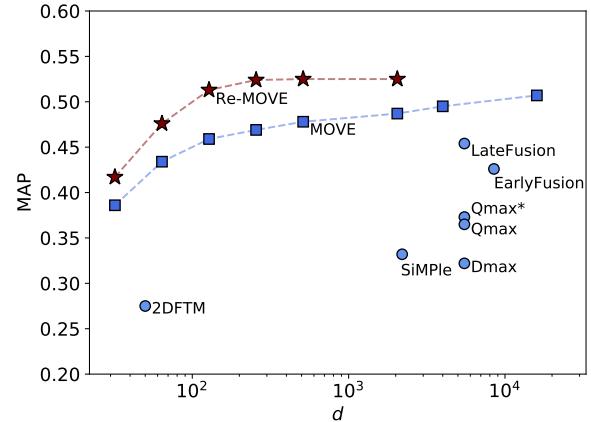
compared with the case where only the triplet loss is optimized. However, the same advantage is not observed with cluster matching using DB loss. We hypothesize that this may be related to training an extra linear projection for compressing the centroid embeddings to match the size of the embeddings obtained with the student model.

Latent space reconfiguration results seem to justify our hypothesis regarding the use of strong priors of a pre-trained feature extractor (last block, Table 1). All considered alternatives outperform their baseline counterparts. Moreover, we find that using probabilistic losses such as NormalizedSoftmax and Group for latent space reconfiguration even outperforms the original model while reducing the embedding size by a large margin ($128/16000 = 0.8\%$). Notice that, in addition to these advantages, latent space reconfiguration does not suffer from the setbacks of network pruning and knowledge distillation methods, namely training a model for multiple iterations and using two models simultaneously during training, respectively.

4.3 Comparison with the state of the art

Lastly, Table 2 compares our best result with state-of-the-art methods. The second column, d, shows the size of the smallest representation (per song) required for each method to estimate pairwise similarities (equivalent to the embedding dimensionality). As the results for the first 7 methods are computed with the publicly-available acoss library [9], we use those implementations for estimating the embedding sizes [5]. As the sizes of some representations depend on the song duration (SiMPle, Qmax, Dmax, LateFusion) or tempo (EarlyFusion), we use 3.5 min and 102 bpm estimates, which correspond to average song duration and bpm of the songs in Da-TACOS, respectively.

Re-MOVE, which stands for 'reduced MOVE', denotes the model trained with latent space reconfiguration using NormalizedSoftmax. With $d = 256$, it demonstrates rel-

ative performance increases of 3%, 6%, and 15% when compared with MOVE-16k, MOVE-4k, and LateFusion systems, respectively (Table 2). We also find that Re-MOVE improves over MOVE for a wide range of dimensionalities $d \in [32, 2048]$ (Figure 2). Along with its state-of-the-art performance, Re-MOVE provides a crucial advantage in terms of scalability, which positions it as the most viable system from a practical point of view.

5. CONCLUSION

In this work, we have studied a set of techniques for reducing the embedding sizes of existing VI systems, which we consider under the name embedding distillation. We have claimed that by using a pre-trained and high-capacity network, we can train a second network that yields smaller embedding vectors without a decrease in performance. To investigate this idea, we have studied a wide range of techniques, including classical dimensionality reduction, neural network pruning, and knowledge distillation methods. Moreover, we have introduced latent space reconfiguration, which is a technique that builds upon the non-linear part of a distance metric learned by a pre-trained network to construct a compact latent space with fewer dimensions. Our results show that it is possible to reduce the embedding dimensionality of a model while maintaining, or even surpassing, its performance.

As future work, we plan to investigate further techniques for compressing entire networks rather than just embedding vectors. We emphasized the importance of having smaller embeddings for real-world applications, and we plan to demonstrate it further in carefully-designed version retrieval scenarios that mimic real-world use cases. Lastly, we believe that optimizing the existing methods to make them applicable in industrial scenarios is a valuable research direction, and we hope to facilitate bridging the gap between academy and industry in MIR research.

---

[5] For 2DFTM, the acoss library uses a 450-dimensional embedding while the authors apply PCA to reduce the dimensionality to 50.

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] D. R. Madoery, "Los procedimientos de producción musical en música popular," *Revista del ISM*, vol. 1, no. 7, pp. 76–93, 2000.

[2] K. Mosser, "Cover songs: ambiguity, multivalence, polysemy," *Philosophy Faculty Publications*, 2008.

[3] J. Serrà, "Identification of versions of the same musical composition by processing audio descriptions," Ph.D. dissertation, Universitat Pompeu Fabra, Spain, 2011.

[4] J. Salamon, J. Serrà, and E. Gómez, "Melody, bass line, and harmony representations for music version identification," in *Proc. of the Int. World Wide Web Conf. (WWW): Int. Workshop on Advances in Music Information Research (AdMIRe)*, 2012, pp. 887–894.

[5] F. Yesiler, J. Serrà, and E. Gómez, "Accurate and scalable version identification using musically-motivated embeddings," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 21–25.

[6] J. Serrà, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 16, no. 6, pp. 1138–1151, 2008.

[7] J. Serrà, X. Serra, and R. G. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, p. 093017, 2009.

[8] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2011, pp. 628–634.

[9] F. Yesiler, C. Tralie, A. Correya, D. F. Silva, P. Tovstogan, E. Gómez, and X. Serra, "Da-TACOS: A dataset for cover song identification and understanding," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 327–334.

[10] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 107–114.

[11] Z. Yu, X. Xu, X. Chen, and D. Yang, "Temporal pyramid pooling convolutional neural network for cover song identification," in *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2019, pp. 4846–4852.

[12] F. Zalkow and M. Müller, "Learning low-dimensional embeddings of audio shingles for cross-version retrieval of classical music," *Applied Sciences*, vol. 10, no. 1, p. 19, 2020.

[13] D. P. W. Ellis and G. E. Poliner, "Identifying 'cover songs' with chroma features and dynamic programming beat tracking," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. IV, 2007, pp. 1429–1432.

[14] D. F. Silva, M. Y. Chin-Chia, G. E. A. P. A. Batista, and E. J. Keogh, "SiMPle: assessing music similarity using subsequences joins," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2016, pp. 23–29.

[15] C. J. Tralie, "Early MFCC and HPCP fusion for robust cover song identification," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2017, pp. 294–301.

[16] N. Chen, W. Li, and H. Xiao, "Fusing similarity functions for cover song identification," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 2629–2652, 2018.

[17] R. Foucard, J. Durrieu, M. Lagrange, and G. Richard, "Multimodal similarity between musical streams for cover version detection," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2010, pp. 5514–5517.

[18] Z. Rafii, B. Coover, and J. Han, "An audio fingerprinting system for live version identification using image processing techniques," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 644–648.

[19] T. J. Tsai, T. Prätzlich, and M. Müller, "Known-artist live song ID: a hashprint approach," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2016, pp. 427–433.

[20] P. Seetharaman and Z. Rafii, "Cover song identification with 2D Fourier transform sequences," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 616–620.

[21] T. Bertin-Mahieux and D. P. W. Ellis, "Large-scale cover song recognition using the 2D Fourier Transform magnitude," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2012, pp. 241–246.

[22] E. J. Humphrey, O. Nieto, and J. P. Bello, "Data driven and discriminative projections for large-scale cover song identification," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2013, pp. 149–154.

[23] X. Xu, X. Chen, and D. Yang, "Key-invariant convolutional neural network toward efficient cover song identification," in *Proc. of the IEEE Int. Conf. on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[24] P. C. Mahalanobis, "On the generalized distance in statistics," *Proc. of the National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936.

[25] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems 17 (NeurIPS)*, 2004, pp. 513—-520.

[26] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems 18 (NeurIPS)*, 2005, pp. 1473–1480.

[27] D. Kedem, S. Tyree, F. Sha, G. R. Lanckriet, and K. Q. Weinberger, "Non-linear metric learning," in *Advances in Neural Information Processing Systems 25 (NeurIPS)*, 2012, pp. 2573–2581.

[28] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 539—546.

[29] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Proc. of the Int. Conf. on Artificial Intelligence and Statistics*, vol. 2, 2007, pp. 412–419.

[30] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: a unified embedding for face recognition and clustering," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[31] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 360–368.

[32] A. Zhai and H. Wu, "Classification is a strong baseline for deep metric learning," in *Proc. of the British Machine Vision Conference (BMVC)*, 2019, p. 91.

[33] I. Elezi, S. Vascon, A. Torcinovich, M. Pelillo, and L. Leal-Taixe, "The group loss for deep metric learning," *ArXiv: 1912.00385*, 2019.

[34] J. Weibull, *Evolutionary Game Theory*. Cambridge, MA: The M.I.T. Press, 1995.

[35] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems 2 (NeurIPS)*, 1989, pp. 598—605.

[36] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Advances in Neural Information Processing Systems 1 (NeurIPS)*, 1988, pp. 177—-185.

[37] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural networks," in *Advances in Neural Information Processing Systems 28 (NeurIPS)*, 2015, pp. 1135—1143.

[38] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2019.

[39] C. Bucilă, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2006, pp. 535—-541.

[40] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *ArXiv: 1503.02531*, 2015.

[41] Y. Chen, N. Wang, and Z. Zhang, "Darkrank: Accelerating deep metric learning via cross sample similarities transfer," in *Proc. of the AAAI conference on Artificial Intelligence*, 2018, pp. 2852–2859.

[42] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3967–3976.

[43] J. Han, T. Zhao, and C. Zhang, "Deep distillation metric learning," in *Proc. of the ACM Multimedia Asia*, 2019.

[44] L. Yu, V. O. Yazici, X. Liu, J. van de Weijer, Y. Cheng, and A. Ramisa, "Learning metrics from teachers: Compact networks for image embedding," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2907–2916.

[45] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "DALI: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2018, pp. 431–437.

[46] C.-W. Wu and A. Lerch, "Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2017, pp. 613–620.

[47] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2017, pp. 188–194.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019, pp. 8024–8035.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[50] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *IEEE Int. Conf. on Data Mining*, 2010, pp. 911–916.

[51] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[52] J. Serrà, M. Zanin, C. Laurier, and M. Sordo, "Unsupervised detection of cover song sets: Accuracy improvement and original identification," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2009, pp. 225–230.

GENERATING MUSIC WITH A SELF-CORRECTING NON-CHRONOLOGICAL AUTOREGRESSIVE MODEL

Wayne Chi   **Prachi Kumar**   **Suri Yaddanapudi**
Rahul Suresh   **Umut Isik**

Amazon Web Services

waynchi@amazon.com, kumprach@amazon.com, yaddas@amazon.com

surerahu@amazon.com, umutisik@amazon.com

ABSTRACT

We describe a novel approach for generating music using a self-correcting, non-chronological, autoregressive model. We represent music as a sequence of edit events, each of which denotes either the addition or removal of a note— even a note previously generated by the model. During inference, we generate one edit event at a time using direct ancestral sampling. Our approach allows the model to fix previous mistakes such as incorrectly sampled notes and prevent accumulation of errors which autoregressive models are prone to have. Another benefit is a finer, note-by-note control during human and AI collaborative composition. We show through quantitative metrics and human survey evaluation that our approach generates better results than orderless NADE and Gibbs sampling approaches.

1. INTRODUCTION

There have been two primary approaches to generating music with deep neural network-based generative models. In the first class, music generation is essentially treated as an image generation problem [1, 2]. In the second class, music generation is treated as a musical time series generation problem, analogous to autoregressive language modeling [3–7]. The human process of music composition, however, is often non-chronological. Notes can be filled in anytime throughout the music piece to create new chords and melodies, add harmony, or embellish existing motifs.

In this work, we propose ES-Net [1], a method that uses elements from both the image-based and time series generation techniques. Our method operates on piano roll images with a 2D convolutional neural network, but autoregressively adds or removes notes one at a time in an arbitrary, non-chronological order. We model the condi-

---

[1] Code: https://git.io/esnet
Samples: https://git.io/esnet-samples

tional distribution of note add or remove events given pre-existing notes. After sampling from the distribution, we re-input the resulting piano roll into the model to get the distribution of the next add and remove events. From a probabilistic point of view, this corresponds to considering each piano roll as obtained from a randomly ordered sequence of add and remove events and autoregressively modeling the distribution of such sequences of events.

Poor samples due to accumulation of errors is a well-documented problem with autoregressive models [8–11], especially when directly sampling from the conditional distribution (i.e. direct ancestral sampling). While other sampling techniques such as Gibbs sampling [12] can be used to bypass this problem, we show that direct ancestral sampling is sufficient if the data representation includes removal of past samples. This allows the model to detect previous mistakes and fix them.

Our primary use case is melody assistance for users generating musical compositions. Users can feed in a melody as a conditional input and have the model generate musical accompaniments as well as fix any off-beat or out of tune inputs. One distinct advantage of our approach is that it allows note-by-note control for users. A user can undo and redo the generation of individual notes or explicitly add and remove individual notes to collaborate with the model and guide the music composition process. Thus, this approach allows users to have a finer degree of control during sampling and better promotes human and AI collaboration.

The remainder of the paper is organized as follows. In Section 2 we discuss related works. In Section 3 we show how to model a distribution of musical pieces using a new representation of music. In Section 4 we discuss our training procedure. In Section 5 we discuss our sampling procedure. In Section 6 we provide empirical results in the form of quantitative metrics and human evaluation compared against other approaches. Finally, in Section 7 and 8 we describe future work and conclusions.

2. RELATED WORK

Following the introduction of NADE [13, 14] and orderless NADE [15] there have been several works built upon the concept of ordered and unordered autoregressive models. Coconet—the algorithm behind Google's Bach Doo-

dle [2] —is a machine learning model that also uses a convolutional model to generate music by adding counterpoints to existing user input [12]. The difference with this work is that Coconet's inference uses Gibbs sampling rather than direct ancestral sampling. DeepBach [16] generates Bach style chorales using pseudo-Gibbs sampling. PixelCNN [17] models an image autoregressively and generates pixels one by one in a pre-specified order while our generation is unordered. In a NLP setting, recent works also explore non left-to-right ordering [18, 19] and deletion [20].

In general, there is a rich history of using deep learning to generate music [21]. Many of them use autoregressive based approaches. RNN-RBM models temporal dependencies to generate polyphonic music in a single track [22]. Hierararchical RNNs have been used to encode different features of pop music [23]. LSTMs were able to successfully model and generate music as well [24]. Music Transformer is able to capture and generate music with long term structure and motifs [3]. So far, these approaches have been mostly chronological while ours is non-chronological. While GAN-based approaches clearly differ from ours, these methods have shown the ability to generate high quality music. MuseGAN is a GAN-based approach for multi-track piano roll generation [1]. MidiNet uses a CNN-based GAN to generate music [2]. C-RNN-GAN generates music using a RNN based architecture with adversarial training [25]. SeqGAN use GANs for sequence generation and apply it to music generation [26]
.
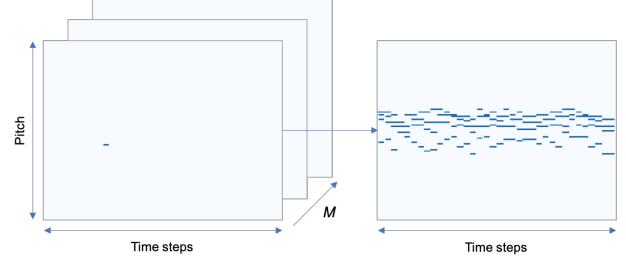
3. PROBLEM DEFINITION

We consider a musical piece $x \in X$ as a point in $\{0,1\}^{T \times P}$ where T is the number of time steps and P is the number of note pitches. This represents a simplified piano roll (PR)—a discrete representation of music as an image matrix across pitch and time. There exists a probability density function $p^{\mathrm{PR}}(x)$ on $\{0,1\}^{T \times P}$ of musical pieces. Note in particular that this does not model velocity and that notes adjacent in time are treated as one continuous held note; we discuss ways to represent velocity and repeated notes in Section 7. Instead of modeling $p^{\mathrm{PR}}(x)$ on $\{0,1\}^{T \times P}$ directly, we model the distributions as $p^{\mathrm{ES}}(s)$ on the set of **edit sequences** (ES). An **edit sequence** of length M is a tuple of M-many **edit events** where an **edit event** is a matrix $e^{(t,p)} \in \{0,1\}^{T \times P}$ that has one entry equal to one, and all other entries equal to zero (i.e. a one-hot matrix). We denote the set of all edit events by $\mathcal{E}$ and of edit sequences of length M by $\mathcal{E}^M$. The following maps edit sequences to piano rolls:

$$\pi : \bigcup_{M=1}^{\infty} \mathcal{E}^M \to \{0,1\}^{T \times P} \qquad (1)$$

$$\pi(e_1, \ldots, e_M) = \sum_{i=1}^{M} e_i \pmod{2}. \qquad (2)$$

---

where (2) allows edit events to handle either note addition or removal depending on if a previous edit event exists at the same time and pitch.

Figure 1. Mapping from an edit sequence (left) of length M to a piano roll (right). Each slice in an edit sequence is the addition or removal of a note.

The mapping between the two joint probability distributions is as follows:

$$\begin{aligned} p^{\mathrm{PR}}(x) &= p^{\mathrm{PR}}(\{(t_1, p_1), \ldots, (t_N, p_N)\}) \\ &= \sum_{s \in \pi^{-1}(x)}^{\infty} p^{\mathrm{ES}}(s) \end{aligned} \qquad (3)$$

where N is the number of notes in the piano roll, (t_i, p_i) is the time and pitch of a note or edit event, $\pi^{-1}(x)$ is the inverse image set of $\pi(x)$, and s is a sequence of edit events $(t_1, p_1) \ldots (t_M, p_M)$ where $M \geq N$. We can further factorize $p^{\mathrm{ES}}(s)$ as:

$$\begin{aligned} p^{\mathrm{ES}}(s) &= p^{\mathrm{ES}}\big((t_1, p_1), \ldots, (t_M, p_M)\big) \\ &= \prod_{i=1}^{M} p^{\mathrm{ES}}\big((t_i, p_i)|(t_1, p_1), \ldots, (t_{i-1}, p_{i-1})\big) \end{aligned} \qquad (4)$$

We assume that $p^{\mathrm{ES}}((t_i, p_i)|(t_1, p_1), \ldots, (t_{i-1}, p_{i-1}))$ is ordering invariant (i.e. the ordering of edit events in an edit sequence does not affect the resulting piano roll).

Our goal is to train a model to map the distribution of edit sequences $p^{\mathrm{ES}}(s)$. By sampling autoregressively from $p^{\mathrm{ES}}(s)$, we will generate a sequence of edit events that can be mapped back into a piano roll representation and then converted to MIDI.

3.1 Orderless NADE

We compare our approach to orderless NADE which generates music by randomly choosing an ordering and sampling notes one by one until termination. We can represent the iterative notewise addition of orderless NADE as a special case of edit sequences where edit events can only represent note addition. Let us call this distribution $p^{\mathrm{O\text{-}NADE}}(x)$. Since notes are only added, $M = N$ for unconditioned generation; thus, there is a finite set of orderings and we can factorize $p^{\mathrm{O\text{-}NADE}}(x)$ as:

$$\sum_{\sigma \in S_N} \prod_{i=1}^{N} p^{\mathrm{O\text{-}NADE}}\big((t_{\sigma(i)}, p_{\sigma(i)})| $$

$$(t_{\sigma(1)}, p_{\sigma(1)}), \ldots, (t_{\sigma(i-1)}, p_{\sigma(i-1)})\big)$$

where S_N is the set of all permutations $\{1, 2, ..., N\} \rightarrow \{1, 2, ..., N\}$. This factorization is equivalent to orderless NADE [15]. In practice, the orderless NADE approach leads to poorer musical samples due to accumulation of errors which we confirm in Section 6.

4. TRAINING

Given an input piano roll, $\mathcal{I}$, and target piano roll, $\mathcal{T}$, we train our model to output the conditional probabilities $p^{\text{ES}}((t_i, p_i)|(t_1, p_1), \ldots, (t_{i-1}, p_{i-1}))$ of the next edit event in edit sequences that can recreate the target from the input piano roll. For each piano roll in the training set, we generate the input piano roll by (a) masking existing random notes and (b) adding extraneous random notes to the target piano roll. We train the model to recreate $\mathcal{T}$ from $\mathcal{I}$; augmentation (a) trains the model to add notes and (b) trains the model to remove notes. For each target piano roll, we generate multiple augmented inputs with varying number of notes masked and added, in order to train the model to handle varying number of differences between the input and target. We find that masking between 0 to 100 percent of all existing notes and adding 0 to 1.5 percent of all possible extraneous notes gives us the best results.

Our goal is to have the model output the conditional probabilities for the next edit event. Since we assume ordering invariance in (4), we can also assume that every note difference between $\mathcal{I}$ and $\mathcal{T}$—whether it requires the addition or removal of a note—is equally likely to be the next edit event. Thus, we model the distribution of edit events for the next step as the uniform distribution U supported on the symmetric difference $\mathcal{I}\Delta\mathcal{T}$ between $\mathcal{I}$ and $\mathcal{T}$ (i.e the exclusive or of each note between $\mathcal{I}$ and $\mathcal{T}$).

We use the Kullback-Liebler divergence between U and the model's output distribution as the loss function:

$$\mathcal{L}(\mathcal{I}, \mathcal{T}, P) = D_{KL}(P \parallel U), \tag{5}$$

where P is the softmax over the model's logits at each time and pitch. Normally, binary cross-entropy loss—where the label is the next note—would be used, but since we assume ordering invariance in (4), the next note is equally likely to be any of the future notes. Therefore, training with (5) is equivalent to training many times where the label is randomly chosen from future notes.

4.1 Model

We train a model based on the U-Net architecture [27]. This choice is not critical as our approach should generalize to other CNN architectures. We describe our approach for reproducibility. Our U-Net contains five downsampling blocks and five upsampling blocks. In each block there contains a batch normalization layer, two 2D convolutional layers each with a 3x3 kernel, a max pooling layer, and a drop out layer with a 0.5 dropout rate. We begin with 32 filters. We double the number of filters after each downsampling block and halve the number of filters after each upsampling block. We use the Adam optimizer [28] with a learning rate of 0.001. We use RELU for our activation
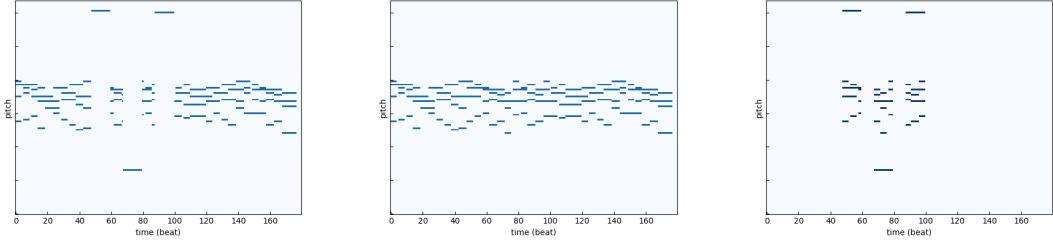
function, except for the final layer where we output a linear activation at each time and pitch. Finally, we apply softmax over the logits when calculating the loss and during sampling.

5. INFERENCE

We sample from the model's output probabilities through direct ancestral sampling. We feed the input melody to the model, sample from the softmax over all times and pitches to determine the next edit event, modify the input melody based on that edit event, and then feed that melody back into the model. We repeat this over multiple iterations and condition each time on our previous predictions. Since we do not differentiate between adding and removing notes during training, the sampling process is the same for any type of edit event. We allow users to restrict the number of notes to remove; this prevents the model from completely overwriting the original input. We also allow users to control how many sampling iterations are performed. Lastly, we allow the user to change the temperature during sampling. By changing the shape of the distribution, users can make compositions more or less "creative" at the risk of lowering quality. We surface these hyperparameters to allow users to more freedom and customizability when generating music compositions.

6. EMPIRICAL EVALUATION

We compare our approach against orderless NADE and Gibbs sampling using quantitative metrics and human survey evaluation. We also describe a notewise approximate log likelihood calculation for our approach and explain why log likelihood is not a good metric for comparing our approach to orderless NADE. We build an orderless NADE model using the approach described in Section 3.1 and training with only masked notes. We use Coconet [12] to represent Gibbs sampling. While our main focus is to only use Coconet for sampling technique comparisons, there are a few notable differences between Coconet and our approach. First, Coconet does not explicitly train the model to remove notes, but notes—including the input—may be removed during the Gibbs sampling masking process; our approach explicitly models note removal. Second, Coconet assumes that there are four instruments and that "each instrument plays exactly one pitch a time" [12]; our approach has no such constraint and can generate music across all times and pitches. Third, Coconet trains a CNN that preserves the same size for each layer; we train a model based on the U-Net architecture. Since Coconet is trained on the JSB Chorales dataset, we evaluate our results and Orderless NADE's results using the same dataset and the same train-val-test split in order to provide a fair comparison. For all other parameters (e.g. temperature), we maintain identical settings for each approach in order to benchmark fairly.

Figure 2. Input piano roll (left), target piano roll (middle), and symmetric difference between the input and target piano rolls (right)

6.1 Data

We use the Infinite Bach dataset[3] and the JSB Chorales dataset[4]. The JSB Chorales and Infinite Bach datasets contain MIDI files—382 for JSB Chorales and 498 for Infinite Bach—of chorales harmonized by J.S. Bach. The MIDI files in Infinite Bach dataset are generally longer in duration allowing for approximately three times more samples overall compared to the JSB Chorales dataset. Since the Gibbs sampling model is trained on JSB Chorales, we use the JSB Chorales dataset for benchmarking.

For both datasets, we preprocess the data by: 1) mapping MIDI to its piano roll representation using a sixteenth-note quantization, 2) converting multi-track inputs into a single track by merging all tracks, and 3) splitting each MIDI into multiple 2 or 8 bar samples.

6.2 Log Likelihood

We calculate log likelihood using equation (3). Since for each piano roll x the inverse image $\pi^{-1}(x)$ is infinite, the sum cannot be calculated exactly; thus, we calculate an approximate log likelihood for a subset of all possible edit sequences in $\pi^{-1}(x)$. This value lower bounds the true log likelihood value. We compare this lower bound to the log likelihood for orderless NADE. Since our method removes notes as well, the proposed model is modeling a distribution with larger support so we do not expect the likelihood value of our method to be better than orderless NADE's. Our likelihood values show that—in the toy case when the sum can be sufficiently expanded—the likelihood lower bound value approaches that of orderless NADE.

Consider a graph where each vertex corresponds to a piano roll state and each edge corresponds to an edit event. A path in the graph corresponds to an edit sequence described in equation (2). As we traverse over a path, we calculate the log likelihood of the edit sequence corresponding to that path.

For each input $\mathcal{I}$ and target $\mathcal{T}$ pairing, we calculate our log likelihood over multiple levels, traversing over edit sequences of length $K + 2d$ at level d. K is the minimum number of edit events needed to reach the target from the input. All K edit events are unique along time and pitch. For level $d = 0$, there exist $K!$ different edit sequences.

We calculate the average log likelihood over a randomly chosen subset of these edit sequences and approximate it over all $K!$ edit sequences. During the traversal we keep track of the most probable (time, pitch) predictions that do not occur in the edit sequences, and add them to a pool Q. We keep these predictions as they will appear in the most probable edit sequence paths at level $d = 1$. For level $d = 1$, we traverse down the same paths, but we add two edit events with the same time and pitch chosen from Q to the path. This increases the path length to $K + 2$ and results in the same target pianoroll since the two new edit events cancel out. We approximate the log likelihood sum over all possible edit sequences. We repeat this for each (time, pitch) pair in Q. This process can be repeated until level $d = D$ expanding our coverage of the edit sequence graph along the most probable paths.

We calculate the approximate log likelihood as:

$$\frac{1}{K} \log \sum_{d=0}^{D} \sum_{Q} \frac{(K+2d)!}{2^d} \frac{1}{|S|} \sum_{s \in S} p^{\text{ES}}(s)$$

where S is a random subset of $K + 2d$ length edit sequences that can transform $\mathcal{I}$ to $\mathcal{T}$.[5] As we increase the levels of our approximation, our log likelihood will converge towards orderless NADE which we see in Table 1 at $d = 1$.

Since music completion is a task with high uncertainty, the large number of low probability predictions leads to underflow issues, which we avoid by using the log-sum-exp trick. Also, since log likelihood in this case is highly dependent on the number of notes in a piece, we compute an approximate *notewise* log likelihood by dividing the approximate log likelihood by the minimum number of note additions and removals needed to reconstruct the target pianoroll. We do not use log likelihood to compare our approach with Gibbs sampling used in Coconet as they use framewise log likelihood, which is different than our calculation [12].

6.3 Quantitative Metrics

We calculate several quantitative metrics to compare the quality of generated music using our approach, orderless NADE, and Gibbs sampling. For each approach, we generate 3405 bars of music—the same number of bars in the

---

3 https://github.com/jamesrobertlloyd/infinite-bach
4 https://github.com/czhuang/JSB-Chorales-dataset

5 We divide the $K + 2d$ factorial by 2^d as we cannot "remove" before we "add" a note.

Approach	Notewise Approximate Log Likelihood
ES-Net	-0.635
Orderless NADE	-0.558

Table 1. Notewise approximate log likelihood for reconstructing 10 missing notes from each test sample.

training data—and compare them to the training data. We generate the music by conditioning on 150 8-bar monophonic inputs. We evaluate on the following metrics designed in [1, 29]:

- PC - Number of unique pitch classes used. Notes whole octaves apart from each other (e.g. C4 and C5) belong to the same pitch class.

- P - Number of unique pitches used.

- ISR - In-scale rate which is the proportion of all notes that lie in C Major [6].

- PR - Polyphonic rate which is the proportion of timesteps where the number of pitches being played is greater than or equal to 4.

We use pypianoroll [29] to calculate these values.

	PC	P	ISR	PR
Training Data	6	46	0.541	0.917
ES-Net	**6**	**46**	**0.540**	**0.930**
Gibbs Sampling	6	46	0.535	0.898
Orderless NADE	8	55	0.559	0.759
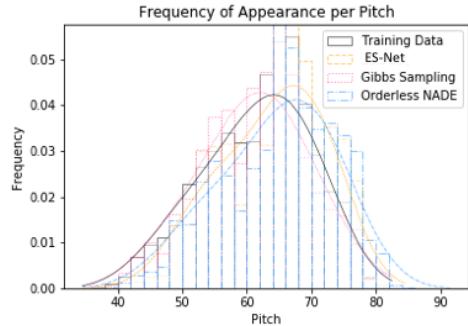
Table 2. Quantitative metrics for each approach. Closer to training data is better. Bold values are best between the three approaches.

We observe that our approach and the Gibbs sampling approach both produce music that have similar characteristics to the dataset, while orderless NADE shows less similarity to the dataset. As seen in Table 2, our approach is the closest for all four metrics, with Gibbs sampling tying for number of unique pitches and pitch classes used.

	Bhattacharyya	Kolmogorov-Smirnov		
		df	D	p
ES-Net	0.028	46	0.17	0.49
Gibbs Sampling	**0.021**	46	0.13	0.83
Orderless NADE	0.049	46	0.17	0.49

Table 3. Various metrics for how far pitch appearance frequency is from the training data. Lower is better and bolded is best for Bhattacharyya distance. The Kolmogorov-Smirnov test is unable to show significant difference between any of the approaches and the training data.

---

[6] The C-Major scale was chosen arbitrarily.
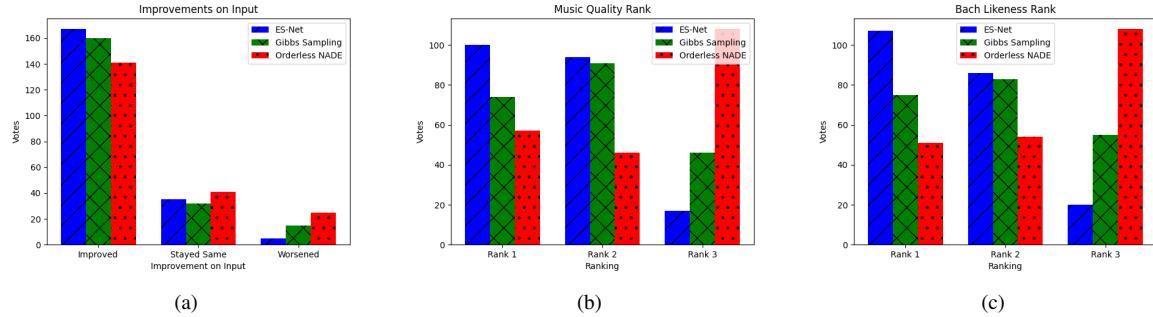
Figure 3. Frequency of occurrence for each pitch bin. Each bin is two pitches (i.e. one bin contains both pitch 31 and 32).

In Figure 3, we plot the frequency of pitch values for each approach and compare with the distribution of pitches in the training data. We observe that the distribution of pitches for all three approaches is very similar to that of the training data. In Table 3, we evaluate the similarity of each approach's pitch appearance frequency to the training data using various metrics. We calculate the Bhattacharyya distance [30] showing Gibbs sampling as the closest to the training data and orderless NADE as the furthest from the training data. We perform Kolmogorov-Smirnov tests and are unable to show significant differences between each approach and the training data.

6.4 Human Evaluation

We conducted a human opinion test in order to compare our approach against orderless NADE and Gibbs sampling. We generated 8 bar samples with a pitch range from 36 to 81. We assume 4/4 time (i.e. 4 beats per bar) and quantize to 16 time steps per bar (i.e. 1/16th note). We assume two notes continuous in time as one note. For orderless NADE, we sample 400 times to generate samples that approximate to 4 pitches per time step. Coconet optimizes the number of iterations it requires. Since our approach allows for the model to both add and remove notes, there is no fixed number of iterations to run the model; instead, the model eventually stabilizes and adds or removes the same set of notes repeatedly. We sample for 10,000 iterations for our approach. When conducting the surveys, we chose a large number of iterations to ensure stabilization; through later experiments, however, we found that the output almost always stabilizes before 2000 iterations.

Each survey contained fifteen randomly chosen sets of comparisons where each set of comparisons contained a random sample from each of the three approaches. Each of the samples in each set were randomly ordered. All three samples in a set were conditioned on the same input track which was also given to the participant. In order to simulate real user input, we created input tracks by taking two bar user inputs from the Bach Doodle Dataset [31]—a dataset of real user inputs to Coconet and its resulting composition—and repeated them four times to form eight

Figure 4. Human Survey Evaluation Ratings: (a) describes whether users thought a sample improved on the input. (b) describes user rankings for music quality. (c) describes user rankings for how similar a sample is to real Bach data. Bars are ordered from left-to-right as ES-Net, Gibbs Sampling, and Orderless NADE.

bars. Bach Doodle ranks their inputs based off of user feedback from the resulting Coconet composition; we chose an equal number of samples randomly from each feedback level. Each survey contained the same fifteen sets of comparisons. These inputs are monophonic (i.e. only one pitch per time step). For each set of comparisons, users were asked (a) if each sample improved on the input, (b) to rank the samples based on music quality, and (c) to rank the samples based on similarity to music composed by Bach.

We receive a total of 207 ratings for question (a), 211 ratings for question (b), and 213 ratings for question (c). [7] For question (a), we see that all approaches are comparable and each approach almost always improved the input as seen in Figure 4(a). For questions (b) and (c), we see in Figures 4(b) and 4(c) that our edit sequence approach is the best approach while the orderless NADE approach is the worst. We perform a Kruskal-Wallis H-test across all ratings for questions (b) and (c). We show that there is a statistically significant difference ($\mathcal{X}^2(2) = 64.47$, $p < 0.001$ for question (a) and $\mathcal{X}^2(2) = 73.07, p < 0.001$ for question (b)) between the three models. We use the Wilcoxon signed-rank test to conduct a pairwise post-hoc analysis. We show that there is a statistically significant difference ($p < 0.001$ for questions (b) and (c)) between our approach and both the Gibbs sampling and orderless NADE approaches.

7. FUTURE WORK

We currently trained on a limited number of datasets, both of which are based on Bach chorales. There is no reason, however, that our approach should be limited to any feature of Bach. By training on other datasets, we will be able to evaluate how well our approach generalizes.

We show that allowing the model to remove notes increases music quality which we believe is due to the model correcting its past mistakes. During our training process, we generate random notes in order to mimic those mistakes. Rather than merely mimicking those mistakes, however, we can generate real mistakes by feeding outputs

from the model back into itself. We believe that this self-adversarial training paradigm will allow the model to capture more realistic sampling mistakes and further improve performance.

Our current data representation does not convey features such as note velocity, repeated notes, or explicit note duration. These features, however, can add to the technical and emotive quality of music. We can map these new features as additional channels and concatenate this information with our existing piano roll. This new data representation will allow our model to learn from these new feature dimensions and produce more expressive and technically challenging music.

An advantage of our algorithm is the ease with which we can extend our approach to other use cases. For instance, currently our model generates fixed length outputs depending on the length of the training samples. In this way, we can extend user melodies up to a fixed length; however, we never explicitly train our model to extend inputs. By augmenting our dataset so that the latter portion of each sample is masked out, we can explicitly train our model to extend melodies. Then, during sampling, we can generate a fixed length output, feed the latter portion of that output back into the model to generate a new output, concatenate those two outputs together, and repeat. This would allow us to extend melody repeatedly rather than up to a fixed length output.

8. CONCLUSION

We show that by modeling removal of notes, we can train a model to produce better music by fixing past mistakes and preventing accumulation of errors. We discuss how our note-by-note approach allows for a finer degree of control and better human and AI collaboration. We demonstrate how to map an edit sequence representation into a piano roll representation and how we can use that to model a distribution of musical pieces. We discuss how we train our model by masking and adding erroneous notes and how we sample from our model during inference. Finally, we show through quantitative metrics and human evaluation that our approach is able to generate musical compositions that are of better quality than orderless NADE and Gibbs sampling.

---

[7] Some users did not answer all three questions per set of samples. Partial or incomplete rankings were discarded.

9. AUTHOR CONTRIBUTION

Authors Wayne Chi and Prachi Kumar contributed equally to this work.

10. REFERENCES

[1] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[2] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.

[3] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," 2018.

[4] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[5] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[6] C. Payne, "Musenet, 2019," *URL https://openai.com/blog/musenet*, 2019.

[7] J. Wu, C. Hu, Y. Wang, X. Hu, and J. Zhu, "A hierarchical recurrent neural network for symbolic melody generation," *IEEE Transactions on Cybernetics*, 2019.

[8] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[9] F. Huszár, "How (not) to train your generative model: Scheduled sampling, likelihood, adversary?" *arXiv preprint arXiv:1511.05101*, 2015.

[10] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Advances In Neural Information Processing Systems*, 2016, pp. 4601–4609.

[11] A. Venkatraman, M. Hebert, and J. A. Bagnell, "Improving multi-step prediction of learned time series models," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[12] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," *arXiv preprint arXiv:1903.07227*, 2019.

[13] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 29–37.

[14] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle, "Neural autoregressive distribution estimation," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 7184–7220, 2016.

[15] B. Uria, I. Murray, and H. Larochelle, "A deep and tractable density estimator," in *International Conference on Machine Learning*, 2014, pp. 467–475.

[16] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation, june 2017," *arXiv preprint arXiv:1612.01010*.

[17] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.

[18] M. Stern, W. Chan, J. Kiros, and J. Uszkoreit, "Insertion transformer: Flexible sequence generation via insertion operations," *arXiv preprint arXiv:1902.03249*, 2019.

[19] W. Chan, N. Kitaev, K. Guu, M. Stern, and J. Uszkoreit, "Kermit: Generative insertion-based modeling for sequences," *arXiv preprint arXiv:1906.01604*, 2019.

[20] J. Gu, C. Wang, and J. Zhao, "Levenshtein transformer," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 181–11 191.

[21] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep learning techniques for music generation–a survey," *arXiv preprint arXiv:1709.01620*, 2017.

[22] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "High-dimensional sequence transduction," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3178–3182.

[23] H. Chu, R. Urtasun, and S. Fidler, "Song from pi: A musically plausible network for pop music generation," *arXiv preprint arXiv:1611.03477*, 2016.

[24] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," *arXiv preprint arXiv:1604.08723*, 2016.

[25] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

[26] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] H.-W. Dong, W.-Y. Hsiao, and Y.-H. Yang, "Pypianoroll: Open source python package for handling multitrack pianoroll," *Proc. ISMIR. Late-breaking paper;[Online] https://github. com/salu133445/pypianoroll*, 2018.

[30] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–109, 1943.

[31] C.-Z. A. Huang, C. Hawthorne, A. Roberts, M. Dinculescu, J. Wexler, L. Hong, and J. Howcroft, "The Bach Doodle: Approachable music composition with machine learning at scale," in *International Society for Music Information Retrieval (ISMIR)*, 2019. [Online]. Available: https://goo.gl/magenta/bach-doodle-paper

MICROTASK CROWDSOURCING FOR MUSIC SCORE TRANSCRIPTIONS: AN EXPERIMENT WITH ERROR DETECTION

Ioannis Petros Samiotis   **Sihang Qiu**   **Andrea Mauri**   **Cynthia C. S. Liem**
Christoph Lofi   **Alessandro Bozzon**
Delft University of Technology, Netherlands
`{i.p.samiotis, s.qiu-1, a.mauri, c.c.s.liem, c.lofi, a.bozzon}@tudelft.nl`

ABSTRACT

Human annotation is still an essential part of modern transcription workflows for digitizing music scores, either as a standalone approach where a single expert annotator transcribes a complete score, or for supporting an automated Optical Music Recognition (OMR) system. Research on human computation has shown the effectiveness of crowdsourcing for scaling out human work by defining a large number of microtasks which can easily be distributed and executed. However, microtask design for music transcription is a research area that remains unaddressed. This paper focuses on the design of a crowdsourcing task to detect errors in a score transcription which can be deployed in either automated or human-driven transcription workflows. We conduct an experiment where we study two design parameters: 1) the size of the score to be annotated and 2) the modality in which it is presented in the user interface. We analyze the performance and reliability of non-specialised crowdworkers on Amazon Mechanical Turk with respect to these design parameters, differentiated by worker experience and types of transcription errors. Results are encouraging, and pave the way for scalable and efficient crowd-assisted music transcription systems.

1. INTRODUCTION

Written musical resources, such as tablature or musical scores, are increasingly being digitized. The physical form of such resources would typically be a book, hence, the most obvious digitized form is a scan of the book, encoded in the form of images. In fact, numerous PDF files of scanned sheet music are commonly found on popular music websites such as the Petrucci Music Library (IMSLP [1]). One of the main disadvantages of scans though, is that the musical content contained in them cannot be computationally accessed. Having easily machine-readable

formats allows for computational analyses, easier enrichment of the digitized musical resources, and, most importantly, to ease the preservation of and the access to our written musical culture.

The majority of transcriptions for professional use involve experts using specialised interfaces, such as Finale [2] and Sibelius [3], to fully transcribe new editions of existing music manuscripts. Optical Music Recognition (OMR) aims at performing the transcription work automatically; state-of-the-art methods show acceptable performance in the case of clean music scores, but their quality quickly degrades in case of hand written notes [1]. In general, they still require substantial human intervention to provide results with consistent quality [1, 2], while interactive systems that could utilize human evaluation in an efficient and scalable way are still an open issue [3].

Microtask crowdsourcing is a popular approach for scaling up digital content annotation tasks. On online microtask crowdsourcing platforms, such as Amazon Mechanical Turk, large groups of individuals - called workers - perform *microtasks* such as image categorization, and audio or text transcription. By splitting a complex and cognitively intensive task into simpler steps, *microtasks* crowdsourcing allows people with little to no expertise, to contribute to knowledge-intensive activities [4].

Explicit control over a crowd's product, is in the heart of microtask crowdsourcing [5, 6]. To that end, microtasks design should allow the measurement of their outcomes in an algorithmic fashion. Few studies addressed the use of microtask crowdsourcing for music scores transcription, and they typically focus on guiding the workers in the transcription of whole scores [7] or by providing support to the experts [8, 9]. However, music scores are complex artefacts that need specific domain knowledge to read and understand, making the task of transcribing a score complex and cognitively demanding. To the best of our knowledge, how to address the task of score transcription through microtask crowdsourcing remains an open research question [10].

This paper contributes towards a better understanding of how music transcription could be supported, and potentially scaled up, through microtask crowdsourcing. We focus on a simple yet fundamental problem: the identification of differences (errors) between two music scores segments. Spotting errors is, in itself, a very useful operation

---

[1] `https://imslp.org/wiki/Main_Page`

[2] `https://www.finalemusic.com/`
[3] `https://www.avid.com/sibelius`

in music transcription workflows, as it could be of assistance for experts transcribing a score, with the creation of labeled data to train automated OMR systems, or with the identification of errors made by such a system. Workers operating in online microtask crowdsourcing platforms are already accustomed to such type of tasks, but the understanding of music scores is not a common skill.

The main research question addressed in this work is: *To what extent are workers from microtask crowdsourcing platforms able to detect errors in transcribed music scores?*. To answer the question, we setup an experiment where two microtask design factors were adjusted respectively, the score transcription's *modality* (spotting errors on visual vs. audio), and the *size* (in terms of measures) to be analysed. We recruited 144 workers from Mechanical Turk, asked them to check 144 segments for several types of errors, and measured their performance in terms of completion time, accuracy, and sustained cognitive load.

Results show that crowd workers were able to achieve maximum precision of 94% and accuracy of 85% using an interface that combines visual and audio modalities, thus showing that microtask crowdsourcing is useful for error detection, and that workers benefit from the audio extract of the transcribed score, both alone and as a support for the visual comparison.

2. RELATED WORK

The topic of microtask crowdsourcing for music transcription is scarcely addressed in literature, with many relevant research questions left unanswered. In Burghardt et al. [7] the *Allegro* system was developed, a tool to allow the transcription of entire scores by a (single) human worker. However, *Allegro* has only been tested on a limited number of users, and it was not deployed on an online microtask crowdsourcing platform. The same limitation holds for the work in [8], one of the first attempts to study human input and how the task design can affect human input. This study focused on analysing segments which are one measure long, which is the smallest unit of analysis in our study as well. We expand this, by studying also how the size of the segment shown to the crowd affect its performance. An important work to mention is OpenScore [11], up to now the largest scale project to incorporate humans in music score transcription. In terms of user participation though, it was mainly carried out by seven community members with extensive musical background. Moreover they report different issues related to the management of data (done manually by the administrators of the platform) and user engagement (without any control they would focus on their preferred music score) admitting in the end that in their project "OMR (involving humans) is not currently a scalable solution".

So far, there is not any literature that has targeted unknown crowds with varying skills for music transcription tasks, thus research questions on [10] about what type of tasks users can perform and how to evaluate them still remain open. In this work we address this research gap by looking into similar crowdsourcing works in other domains. More specifically, in [12] it was found that for knowledge-intensive tasks involving artworks, a crowd with varying and unknown domain-specific knowledge found on online platforms can produce useful annotations when aided by good task design. Research has shown that UI design is an important part of a microtask design [13]. Research so far has experimented with various designs such as showing spectogram visualisations for audio annotation [14] or the use of chat-bots to assist common types of microtasks [15], all of which have yielded positive results on the performance of the crowdworkers. Inspired by this we make the design of the work interface a central point of our study.

3. EXPERIMENTAL DESIGN

The main focus of this work is to study to what extent a general crowd can identify *errors* in a music score transcription. We therefore designed an experiment aimed at testing the ability of crowd workers to spot errors using interfaces having a combination of visual and audio components.

3.1 Task Design

Our aim is to study how different task design factors can influence the crowdworkers performance, focusing on two aspects:

1. The *modality* (*visual* versus *audio*) used to spot errors: as music scores are complex artefacts, and music is primarily an auditory experience. Therefore, we investigate how the score comparison *modality* affect the error detection performance in workers that are potentially not familiar with musical notation. Intuitively, we want to investigate if "hearing" errors is easier that "seeing" errors.

2. The score *size* offered to crowdworkers for annotation. The goal is to assess how the size (in terms of measures) of the score offered to worker affects their performance.

To develop a better understanding on the characteristics of the crowd, we open the tasks with questions about their demographic information (occupational status, level of education and age) and their music sophistication. For the latter, we compiled a list of 6 questions from The Goldsmiths Musical Sophistication Index (Gold-MSI) [16].

Crowd workers' performance with error identification is measured using accuracy, precision and recall and time to execute a microtask. In addition, we measure user confidence with their judgements with a seven-value Likert scale.

Finally, we measure the sustained cognitive load when executing the microtask, measured through the NASA Task Load Index (TLX)[4], which ranges from 0 to 100 (higher the TLX is, the heavier cognitive load the worker perceives).

---

[4] https://humansystems.arc.nasa.gov/groups/TLX/

3.2 Evaluation Dataset

Selecting a suitable music score was our first step preparing our experiments. We use a single classical music score to avoid introducing additional variable in workers' performance. Specifically we use the Urtext of "*32 Variations in C minor*" by Ludwig van Beethoven. It is a piano piece and the music artifacts are all printed typeset forms. This is a slightly easier use case than hand-written scores. The score was retrieved from IMSLP as a PDF [5] .

As a Gold standard transcription of that PDF we used an MEI [6] file that had been transcribed by an expert. This file was accepted as error free, and it allowed us introduce errors in a controlled way for our experiments.

We segmented the music score in varying sizes to investigate how workers cope with shorter or longer tasks. We distinguish 1) *one measure* segments, 2) segments of *two measures* and 3) segments of *three measures*. Both of the two digital versions of the score, the PDF file of the original score and the transcribed MEI file, were segmented using the aforementioned segment sizes. The segmentation of the PDF was performed manually, while for the MEI we used the appropriate identifiers of each measure that was included in the corresponding image segment, to isolate the correct headers in the MEI. Since it's a piano score, each measure contains two staves.
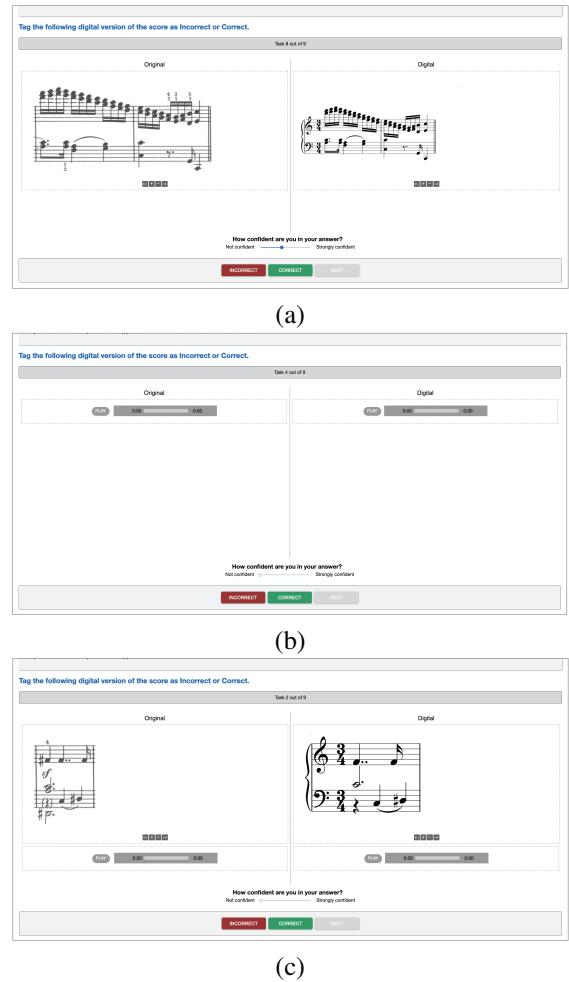
We then introduced errors in the MEI segments derived from common errors that can occur in automatic OMR systems. The type of errors could impact the crowdworkers' ability to spot them and correctly identify them as errors. Some of them can be challenging to notice even to experts of the field. Therefore, we study different types of errors, all focusing on the music notes themselves and their accidentals. Errors on performance annotations, clefs, finger numbers etc, are out of scope in this study. We introduced the following types of error per MEI segment: 1) *Missing notes*; 2) *Wrong vertical position of a note*; 3) *Wrong duration of a note*; 4) *Wrong accidental*.

Each segment that was shown to the user contained only one type of error. The amount of errors per segment was kept constant at 40% of the notes present in the segment. Thus, if a crowdworker is presented with two measures with notes missing, then notes are missing on both measures at a 40% rate of the total notes on both measures combined. No more types of error are present in the segment.

To make the performance comparisons meaningful, we ensured that our dataset is balanced across all error types. In total we used 144 segments derived from the entirety of the selected piano score, with 48 segments per size category, from which 24 were equally distributed to each type of error, while the remaining 24 were kept correct.

3.3 User Interface Design

To test the modalities' effects separately and accurately, we designed three different interfaces: one that would have

Figure 1. Microtask User Interfaces: (a) Visual, (b) Audio and (c) Combination

image to image comparison to test the traditional form of the task, one with only audio to audio comparison, and one with both audio and image comparison. The interfaces are designed to include the following data. 1) **Original Score**: the segment's image from the scanned score. 2) **Correct MEI Render**: a render of the transcribed version of the *Original Score*'s segment; 3) **Incorrect MEI Render**: a render of the MEI transcription containing errors. 4) **Correct MIDI**: the MIDI extract of the correct version of *MEI Transcript*'s segment. 5) **Incorrect MIDI**: the MIDI extract of *MEI Transcript*'s segment containing errors.

We refrained from using audio from a recorded performance against a MIDI extract containing errors to avoid confusing the crowd on what constitutes as "different" or an "error" in the audio comparison task. A recorded performance would introduce performance-related artefacts to the audio, which do not exist in a MIDI extract, thus increasing the chance of false negatives in identifying an audio snippet as "incorrect". Finally, for the combined comparison, we used the same elements as with the individual comparisons. For the renders of the MEI transcripts and MIDI extracts we used Verovio [7] on our interfaces.

From a design perspective, the interfaces needed to be

simple and closely resembling each other to minimize their effect on the workers' judgements. They should also be able to facilitate the different segment sizes without changing the layout. Eventually, we also wanted to accommodate error detection in the same manner for both image and audio comparisons, to avoid differences on the annotation tools being another factor to the crowd's performance. Therefore, we designed the error detection task to ask from the users to annotate a given MEI transcript or MIDI extract as "*Incorrect*" if it exhibit errors and as "*Correct*" if they did not.

In all interfaces, the segments to the left are associated to the the original scanned score and the correct MEI transcription of it, while to the right we always place the segments that need to be annotated. The MEI render or the MIDI extract to the right can be either "*Correct*" or "*Incorrect*" and we calculate the performance of the workers based on identifying this correctly. In addition to the two buttons for the labels, we included a slider to indicate the worker's confidence in their label. Later in the analysis of the results, we used this indicator to study how each interface and segment size affected the confidence of the workers' to their judgements. These design considerations resulted in the following three interface designs:

- **Original Score** against **Correct/Incorrect MEI Render (Visual)**: This user interface, depicted in Figure 1(a), shows the segment of the original scanned score to the left, with the corresponding MEI render to the right. The user needs to compare the two images and spot differences related to the types of errors.
- **Correct MIDI** against **Correct/Incorrect MIDI (Audio)**: In this interface, as shown in Figure 1(b), we let the user listen to the correct MIDI extract on the left and the one generated from the MEI transcription to the right.
- **Original Score and Correct MIDI** against **Correct/Incorrect MEI and Correct/Incorrect MIDI (Combination)**: This final user interface, as shown in Figure 1(c), combines elements of the previous two. The user here has the option to either use the visual comparison, the audio comparison or both to realise if there are errors to the segment to the right. The MEI render and MIDI extraction to the right always originate from the same MEI transcription, therefore both will be correct or both will contain errors.

Each combination of interface with a segment size consists of a microtask. To efficiently and effectively gather performance data, we wanted the same worker to be "exposed" to all nine possible combinations. Therefore, in its final form, a worker would have to execute a task that would begin with a set of demographic and music sophistication questions, followed by the nine microtasks and end with the cognitive load questionnaire. To minimise the impact of issues related to the familiarity of workers with the interface, the task also includes an introductory explana-

tion of the work interface, with examples of errors and expected responses. The results of our experiment are analysed based on the overall, but also on error type, performance of workers.

4. RESULTS

As discussed in previous sections, we published our tasks on Amazon Mechanical Turk (MTurk). The platform offers several configurations for each batch of tasks submitted. We published them as public so they can be accessed by all the users of the platform and we did not require any Mechanical Turk Master (expert workers). Only to avoid malicious workers, we filter them by their previous HIT Approval Rate, and we set it to 95%.

In total, 144 workers executed our tasks on MTurk and we paid them per task execution according to the average US minimal hourly wage [8] . In order to minimize the effect of any biases or learning effect we randomized the order of the presentation of the different task designs (UI-segment size combination). One worker eluded the quality verification on task interface, which results in 143 unique workers.

4.1 Worker Demographics

From a demographic aspect, most of the workers (84.6%) reported that they had a full time occupation. Also, 67.8% of total workers reported their education level was Bachelor's degree, while 14.9% of them had Master's degree. Only 8.3% of the workers were above 45 years old.

Answers to the Gold-MSI questions indicate that the majority of workers seem to be familiar with listening to music, as 56% of them listen to music for at least 1 hour a day and 65% say they can hit the proper notes while listening to a record. Also, the majority of them (75%) state they can properly compare and discuss different performances of the same music piece. On the other hand, 52.4% of the workers reported having up to one year formal training in music theory, where the 26.6% has no prior education on the subject. Also, 41.95% of the workers have trained for maximum one year on a music instrument, while 22.4% of never had. Their answers here suggest that the majority of the crowd has little to no music theory background, and a considerable amount of them also no formal studies on an instrument. The results also suggest that the crowd executing our tasks was mainly composed of workers with little expertise with music theory.

4.2 Accuracy

The results per target segment were aggregated from three different individual workers. Table 1 shows that tasks performed with the *Audio* interface consistently achieved higher accuracy than the *Visual* one. The *Combined* interface achieved good accuracy figures with all segments sizes, but best accuracy with the 3-measure-long segments. The *Visual* interface yielded consistently the lowest recall and accuracy results, for all segment sizes. Interestingly,

---

[8] We estimated an average task completion time of 15'; each crowd-worker was awarded 2.5$

Interface	segment size		
	1	2	3
Visual	**P=66.22** R=59.76 A=60.27	**P=65.28** R=61.84 A=62.24	P=59.72 R=74.14 A=69.23
Audio	P=60.27 **R=81.48 A=72.92**	P=62.50 **R=81.82 A=74.13**	P=64.79 R=80.70 A=74.83
Combined	P=59.70 R=68.97 A=67.63	**P=65.28** R=74.60 A=71.33	<u>**P=68.06 R=83.05 A=76.92**</u>

Table 1. Precision, Recall and Accuracy of *individual* answers, by segment sizes and interfaces.

Interface	segment size		
	1	2	3
Visual	P=60.71 **R=70.83** A=62.50	P=67.86 **R=79.17** A=70.83	P=88.89 R=66.67 A=79.17
Audio	**P=100.0** R=62.50 **A=81.25**	**P=88.24** R=62.50 A=77.08	P=90.00 R=75.00 A=83.33
Combined	P=76.47 R=56.52 A=70.21	P=85.00 R=70.83 **A=79.17**	<u>**P=94.74 R=75.00 A=85.42**</u>

Table 2. Overall Precision, Recall and Accuracy of *aggregated* answers by segment sizes and interfaces. In bold you find the highest precision, recall and accuracy by segment size, while underlined you find the highest overall performance

the precision for this interface on segment size one, was the highest compared to *Audio* and *Combined* for the same size.
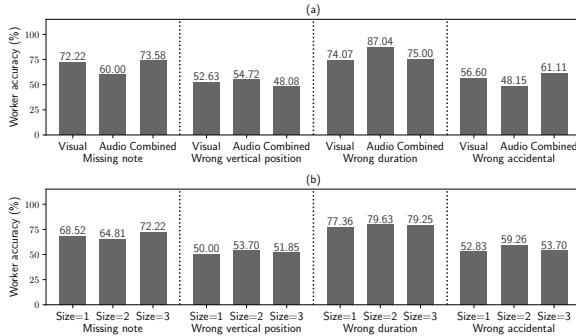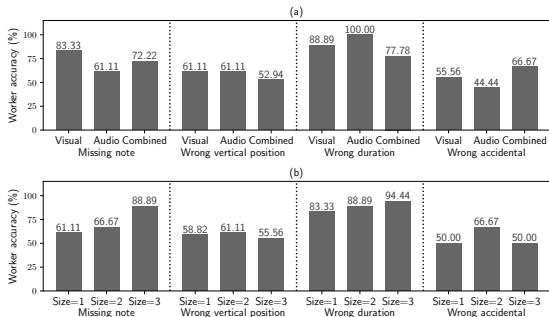
Figure 2. Workers error detection accuracy (unit:%) (a) per user interface and (b) per segment size.

Figure 3. Aggregated error detection accuracy (unit:%) (a) per user interface and (b) per segment size.

Figure 2 shows the accuracy of the workers in detecting specific type of errors. *Wrong duration* error seems to be accurately spotted in any user interface and segment size, with the *Audio* interface resulting in the highest accuracy (87.04%). Workers perform better detecting the *Missing Note* error using the *Combined* interface and the 3-measure

segments. The accuracy obtained with the *Visual* interface though, suggests that workers might rely more on the image of the score rather than the audio for this type of error. The *Wrong Vertical position* error was more difficult to detect in general; the highest accuracy was obtained with the *Audio* interface (54.72%) and with the segment size of 2 measures (53.70%). Finally, the *Wrong accidental* type was the second most demanding error to be detected with the highest accuracy achieved using *Combined* interface (61.11%), with a slight improvement in segments containing 2 measures.

In microtask crowdsourcing it is common to aggregate individual results to improve overall quality. Table 2 shows the performance achieved using a simple *majority voting* aggregation scheme. The *Combined* interface with 3-measure-long segments still achieves best performance with a remarkable 94% in precision, and 85% in accuracy. The *Audio* interface achieves best precision performance for both 1-measure-long and 2-measure-long segments, while the *Visual* interface achieves best recall.
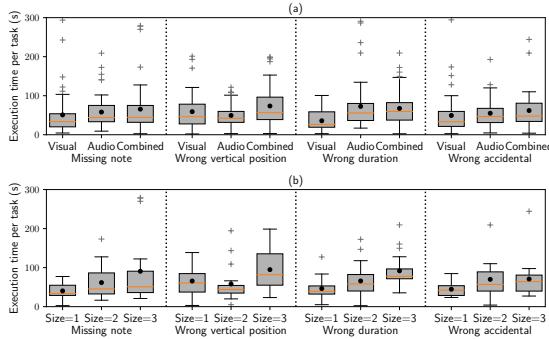
Figure 3 shows the aggregated accuracy in detecting specific type of errors. In terms of *Wrong Duration* error, the accuracy remains the highest after the aggregation. The *Audio* interface and the 3-measure-long segments achieve 100% and 94% in accuracy respectively. *Visual* interface and the 3-measure-long segments obtain the highest accuracy (82% and 88% respectively) in detecting *Missing note* error. The *Wrong Vertical position* error and the *Wrong accidental* error still have relatively low accuracy.

4.3 Execution Time

Figure 4 shows that, as expected, execution time generally increases as the segment size increase. We performed statistical tests (independent t-tests, $\alpha = 0.05$) to find significant differences between interfaces and segment sizes. In the case of *Wrong vertical position* error though, the *Audio* interface allowed the worker to spot the errors significantly faster compared to *Combined* interface ($p = 3.5\text{e-}3$). For the *Wrong duration* error the addition of audio and the

increased segment size can lead to a significantly longer average execution time (for both *Audio* and *Combined* interfaces compared to *Visual*, $p = 1.3e\text{-}4$ and $p = 1.9e\text{-}5$ respectively; and for 3-measure-long segment vs 1-measure-long segment, $p = 2.5e\text{-}3$).

For the *Wrong accidental* case, we see that worker spent less execution time on the *Visual* interface (no significance). However, comparing it with the results, the worker most probably dismissed the segment as "Correct", rather spend more time in case they had missed the error.

Figure 4. Worker execution time (unit: seconds) of each microtask by (a) user interface and (b) segment size.

4.4 Music Sophistication and Cognitive Load

The average score of cognitive task load (NASA-TLX) is 47.7%, a typical value for classification and similar cognitive tasks [17]. To investigate how music sophistication relates to worker performance and cognitive load on spotting music errors, we select and analyze a corresponding question from Gold-MSI questionnaire – "I find it difficult to spot mistakes in a performance of a song even if I know the tune.", where workers need to select an option from `Completely Disagree` to `Completely Agree`, before they execute the music transcription tasks.

Results show that 47% workers agreed with the statement that it was difficult to spot mistakes in a performance of a song; 33% of them disagreed with the statement, and the rest of them (20%) were unsure. We calculated the worker accuracy and the cognitive task load score, and performed significant testings (independent t-test, $\alpha = 0.05$). Workers who reported lower difficulty with spotting music errors (accuracy = $81 \pm 15\%$, TLX score = 44.36 ± 13.05) outperformed workers who had higher difficulty (accuracy = $63 \pm 16\%$, TLX score = 50.86 ± 13.61) in terms of both worker accuracy and perceived cognitive load ($p = 3.3e\text{-}8$ and 0.013 respectively). The workers who reported lower difficulty also had significantly higher accuracy ($p = 0.011$) compared to unsure workers (accuracy = $0.70 \pm 0.20\%$, TLX score = 46.01 ± 14.27). Results suggest that the self-reported music sophistication in some specific aspects strongly relates to actual worker performance in error identification and cognitive load. Nonetheless, workers with lower sophistication still achieved good performance, with a small additional cost in terms of cognitive load.

5. DISCUSSION

As expected, people with some formal knowledge in music, which could be useful to comprehend music scores, are very rare "in the wild". To enable the use of microtask crowdsourcing for music score transcription, good task design is therefore of essence. Results show that error detection is a task that could be successfully performed in a microtask crowdsourcing setting. Offering audio extracts of a target music score can positively affect the performance of the crowdworkers, especially for short segments of one or two measures. With larger segments, even though audio extracts are still yielding better results against to the textual measures of the score, a combination of the two modalities is more preferable. This result gives important indications for task splitting and scheduling purposes, as it suggests that it is possible to evaluate larger portions of scores without incurring accuracy penalties. This has obvious implications in terms of overall transcription costs.

In terms of types of detected errors, results suggest that the *Missing Note* and *Wrong Duration* errors are the easiest to be found, while the crowd had relatively more difficulty detecting *Wrong Accidentals* and *Wrong vertical position* ones. Furthermore, we see the clear effect of user interface and segment sizes in identifying correctly specific errors. Specifically, the *Audio* interface helps in finding *Wrong duration* errors, while the *Combined* one increases the accuracy in finding *Wrong accidental* mistakes. Showing segments longer than two measures seems to slightly hinder the ability of the crowd to detect any errors besides *Missing notes*.

Limitations. Correct MEI render and correct MIDI files of scores to be transcribed are typically not available in the real world. The distribution of errors in the evaluation dataset might not reflect the actual distribution of errors produced, for instance, by OMR systems. Given these limitations, the results of our experiment are probably to be interpreted as an "upper bound" in terms of achievable performance; nonetheless, they clearly indicate that the detection of errors in transcribed music score is an activity that can be successfully performed by crowdworkers.

6. CONCLUSION

Music score transcription is an important activity for written music preservation. Through this work, we show that microtask crowdsourcing can be used to scale up specific transcription activities. Worker interfaces that combine visual and audio modalities allow the evaluation of longer score segments. Focusing on the error detection task, results show that crowd workers can achieve high precision and recall, especially with *Missing Note* and *Wrong Duration errors*. In future work, we plan to expand the evaluation dataset, perform experiments where workers are asked to compare recorded performance, and address a broader set of transcription errors. Finally, we will investigate other types of microtasks, and study to what extent crowd workers could also be employed to *transcribe* scores.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] B. Almeida and S. Spanner, "Allegro: User-centered Design of a Tool for the Crowdsourced Transcription of Handwritten Music Scores," in *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage*, vol. 25, no. 23. New York, New York, USA: ACM Press, 2017, pp. 15–20. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3078081.3078101

[2] P. Bellini, I. Bruno, and P. Nesi, "Assessing Optical Music Recognition Tools," *Computer Music Journal*, vol. 31, no. 1, pp. 68–93, mar 2007. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/comj.2007.31.1.68

[3] J. Calvo-Zaragoza, J. H. Jr., and A. Pacha, "Understanding optical music recognition," *ACM Comput. Surv.*, vol. 53, no. 4, Jul. 2020. [Online]. Available: https://doi.org/10.1145/3397499

[4] J. Oosterman, J. Yang, A. Bozzon, L. Aroyo, and G.-J. Houben, "On the impact of knowledge extraction and aggregation on crowdsourced annotation of visual artworks," *Computer Networks*, vol. 90, pp. 133 – 149, 2015, crowdsourcing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128615002315

[5] E. Law and L. v. Ahn, "Human computation," *Synthesis lectures on artificial intelligence and machine learning*, vol. 5, no. 3, pp. 1–121, 2011.

[6] A. Bozzon, M. Brambilla, S. Ceri, and A. Mauri, "Reactive crowdsourcing," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 153–164.

[7] M. Burghardt and S. Spanner, "Allegro: User-centered design of a tool for the crowdsourced transcription of handwritten music scores," in *Proceedings of the 2Nd International Conference on Digital Access to Textual Cultural Heritage*, ser. DATeCH2017. New York, NY, USA: ACM, 2017, pp. 15–20. [Online]. Available: http://doi.acm.org/10.1145/3078081.3078101

[8] L. Chen and C. Raphael, "Human-Directed Optical Music Recognition," *Electronic Imaging*, vol. 2016, no. 17, pp. 1–9, feb 2017. [Online]. Available: http://www.ingentaconnect.com/content/10.2352/ISSN.2470-1173.2016.17.DRR-053

[9] L. Chen, R. Jin, and C. Raphael, "Human-Guided Recognition of Music Score Images," in *Proceedings of the 4th International Workshop on Digital Libraries for Musicology - DLfM '17*. New York, New York, USA: ACM Press, 2017, pp. 9–12. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3144749.3144752

[10] C. Saitis, A. Hankinson, and I. Fujinaga, "Correcting large-scale omr data with crowdsourcing," in *Proceedings of the 1st International Workshop on Digital Libraries for Musicology*, 2014, pp. 1–3.

[11] M. Gotham, P. Jonas, B. Bower, W. Bosworth, D. Rootham, and L. VanHandel, "Scores of scores: an openscore project to encode and share sheet music," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, 2018, pp. 87–95.

[12] J. Oosterman, A. Bozzon, G.-J. Houben, A. Nottamkandath, C. Dijkshoorn, L. Aroyo, M. H. Leyssen, and M. C. Traub, "Crowd vs. experts: nichesourcing for knowledge intensive tasks in cultural heritage," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 567–568.

[13] U. Gadiraju, A. Checco, N. Gupta, and G. Demartini, "Modus operandi of crowd workers: The invisible role of microtask work environments," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, pp. 1–29, 2017.

[14] M. Cartwright, A. Seals, J. Salamon, A. Williams, S. Mikloska, D. MacConnell, E. Law, J. P. Bello, and O. Nov, "Seeing sound: Investigating the effects of visualizations and complexity on crowdsourced audio annotations," *Proceedings of the ACM on Human-Computer Interaction*, vol. 1, no. CSCW, pp. 1–21, 2017.

[15] P. Mavridis, O. Huang, S. Qiu, U. Gadiraju, and A. Bozzon, "Chatterbox: Conversational interfaces for microtask crowdsourcing," in *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization*, 2019, pp. 243–251.

[16] D. Müllensiefen, B. Gingras, J. Musil, and L. Stewart, "The musicality of non-musicians: an index for assessing musical sophistication in the general population." *PloS one*, 2014.

[17] R. A. Grier, "How high is high? a meta-analysis of nasa-tlx global workload scores," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 59, no. 1, pp. 1727–1731, 2015. [Online]. Available: https://doi.org/10.1177/1541931215591373

SCORE-INFORMED NETWORKS FOR MUSIC PERFORMANCE ASSESSMENT

Jiawen Huang   **Yun-Ning Hung**   **Ashis Pati**
Siddharth Gururani   **Alexander Lerch**
Center for Music Technology, Georgia Institute of Technology, USA
{jhuang448,amyhung,ashis.pati,siddgururani,alexander.lerch}@gatech.edu

ABSTRACT

The assessment of music performances in most cases takes into account the underlying musical score being performed. While there have been several automatic approaches for objective music performance assessment (MPA) based on extracted features from both the performance audio and the score, deep neural network-based methods incorporating score information into MPA models have not yet been investigated. In this paper, we introduce three different models capable of score-informed performance assessment. These are (i) a convolutional neural network that utilizes a simple time-series input comprising of aligned pitch contours and score, (ii) a joint embedding model which learns a joint latent space for pitch contours and scores, and (iii) a distance matrix-based convolutional neural network which utilizes patterns in the distance matrix between pitch contours and musical score to predict assessment ratings. Our results provide insights into the suitability of different architectures and input representations and demonstrate the benefits of score-informed models as compared to score-independent models.

1. INTRODUCTION

A performance is a sonic rendition of a written musical score (in the case of Western classical music). The characteristics of a music performance play a major role in how listeners perceive music, even if performances are based on the same underlying score [14]. To perform a musical piece, the performer must first parse the score, interpret or modify the musical information, and utilize complex motor skills to render the piece on their instrument [21].

From the perspective of the performer, mastery over the art of music performance is often a journey spanning several years of instruction and practice. A major factor in learning and improving one's skill as a performer is to analyze and obtain feedback regarding the performance. Due to the complex nature of music performance, students require regular feedback from trained professionals. Teachers are expected to grade or rate students based on various performance criteria such as note accuracy or musicality. These criteria are often ill-defined and subject to interpretation, thus making objective and consistent music performance assessment (MPA) rather difficult [26, 29]. Regardless, this subjective manner of MPA is still used, e.g., in school systems where ensemble members are selected based on instructors' assessments of student auditions.

Wu et al. discussed the notion of objective descriptors (features) which are potentially useful for automatic MPA [30]. Such features are computed by applying signal processing methods to recorded performances and are used to model teachers' assessments of the performances using machine learning. With the rise of deep learning, neural networks were found to outperform the classical pipeline of feature extraction followed by regression [22]. However, one issue with these approaches is that they ignore the score that the students are meant to play. We will refer to such approaches as *score-independent*. The idea of incorporating score-based features utilizing audio to score alignment was explored, e.g., by Vidwans et al. [27]. Further analysis of hand-crafted features for MPA showed the relative importance of score-based features over score-independent ones [8]. Therefore, the design of deep architectures that incorporate score information is an obvious and overdue extension of previous approaches.

The goal of this paper is to explore different methods to incorporate this score information. Our hypothesis is that including score information will lead to improved performance of deep networks in the objective MPA task. To this end, we present three architectures which combine score and audio features to make a *score-informed* assessment of a music performance. First, we concatenate aligned pitch contours and scores into a 2-dimensional time-series feature representation that is fed to a convolutional neural network (CNN). Second, we propose a joint embedding model for aligned score and pitch contours. The assessment ratings are predicted using the cosine similarity between the score and performance embeddings. Third, we utilize the distance matrix, a mid-level representation combining both the score and pitch contour, as the input to a deep CNN trained to predict the teachers' assessments. Finally, using a fairly large scale dataset of middle school and high school student auditions, we perform an in-depth evaluation comparing these proposed architectures against each other and with a

score-independent baseline approach for MPA .

2. RELATED WORK
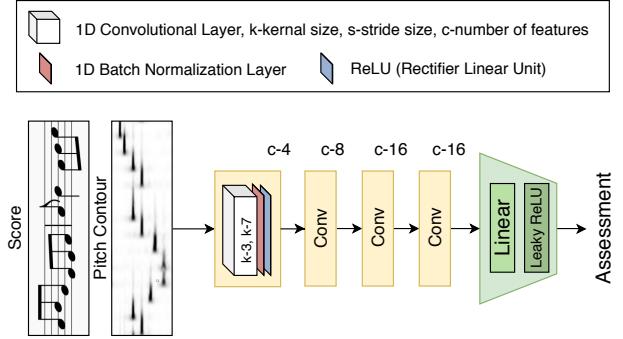
MPA deals with the task of assessing music performances based on audio recordings. Progress in MPA is roughly categorized into feature design-based approaches [8, 13, 20, 24, 30] and feature learning-based approaches [9, 22, 31]. Feature design-based methods rely on signal processing techniques to either extract standard spectral and temporal features [13], or use expert knowledge to extract perceptually motivated features capable of characterizing music performances [20, 24]. The extracted features are then fed into simple machine learning classifiers to train models which predict different performance assessment ratings. Feature learning-based approaches, on the other hand, stem from the argument that important features for modeling performance assessments are not trivial and cannot be easily described. Hence, they rely on using mid-level representations (such as pitch contours or mel-spectrograms) as input to sophisticated machine learning models such as sparse coding [9, 31] and neural networks [22].

Most performances of Western music, however, are based on written musical scores. Hence, performances are also assessed based on their perceived deviations from the underlying score. There has been some prior research on incorporating the score information into the assessment modeling process. Most of the these approaches rely on computing descriptive features using some notion of *distance* between the score representation and the performance representation [3, 6, 11, 17, 19]. The most common approach has been to first use an alignment algorithm, e.g., Dynamic Time Warping (DTW) [25], to temporally align the performance recording with the score and then compute descriptive features which characterize the deviations of the performance from the score [1, 27]. However, to the best of our knowledge, incorporating score information directly into neural network-based models for MPA has not been investigated before.

Score-informed approaches have helped improve results for both related performance analysis tasks and other music information retrieval tasks. Most of these methods have also relied on an alignment between the audio recording and the score as the primary tool for incorporating score information. Aligning audio recordings with scores has been useful for several tasks such as detecting expressive features in music performances [15], identifying missing notes and errors in piano performances [5], and segmenting syllables in vocal performances [23]. Scores have also been used to generate soft labels and/or artificial training data for tasks such as source separation [4, 18].

3. METHODS

We propose and compare three different approaches to incorporate the score information with audio features for MPA. [1] The score information is represented as the MIDI pitch sequence (in ticks) obtained from the sheet music of the score

---

[1] The code is available at: https://github.com/biboamy/FBA-Fall19

Figure 1. Schematic for the SIConvNet. The aligned *score* and *pitch contour* are stacked together and fed into a 4-layer CNN to directly predict the assessment ratings.

to be performed. Henceforth, the MIDI pitch sequence will be referred to as the *score*. The student's *performance* is represented by the pitch contour of the audio. We use pitch contour since it captures both pitch and rhythmic information. Musical dynamics and timbre are ignored in this study; while dynamics are an important expressive tool for the performer [14], the score usually lacks specificity in dynamics instructions and cannot serve as the same absolute reference as for pitch and rhythm.

3.1 Score-Informed Network (SIConvNet)

The first approach that we use is probably the most straightforward way of incorporating score information into the assessment model. A simple CNN is used that relies on both the score and performance as the input and directly predicts the assessment ratings.

3.1.1 Input Representation
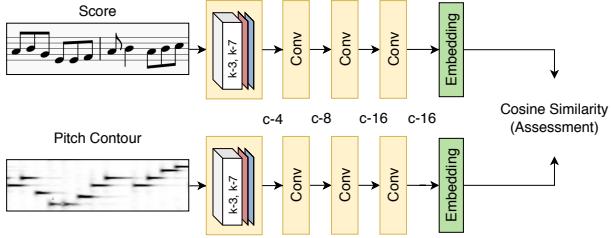
The input representation for this approach is constructed by simply stacking an aligned pitch contour and score pair to create a $N \times 2$ matrix, where N is the sequence length of the pitch contour. The two channels correspond to the pitch contour and score, respectively.

In order to obtain this representation, we first consider a pitch contour snippet of length N (sequence of logarithmic frequencies). Then, we find the corresponding part of the score using a DTW-based alignment process. The obtained score snippet (sequence of MIDI note numbers) is then resampled to have the same length N as the pitch contour.

3.1.2 Model Architecture

A schematic of the model architecture is shown in Figure 1. We use a simple 4-layer CNN based on the architecture proposed by Pati et al. [22] and append a single linear layer which predicts the assessment. Each convolutional stack consists of a 1-D convolution followed by a 1-D batch normalization layer [12] and ReLU non-linearity. The linear layer at the end comprises of a dense layer followed by Leaky ReLU non-linearity.

Figure 2. Schematic of the JointEmbedNet architecture.

3.2 Joint Embedding Network (JointEmbedNet)

The second approach is based on the assumption that performances are rated based on some sort of perceived distance between the performance and the underlying score being performed. Consequently, we use two separate encoder networks to project the score and the pitch contour to a joint latent space and then use the similarity between the embeddings to predict the assessment ratings.

3.2.1 Input Representation

This approach uses the same input representation as SIConvNet (see Section 3.1.1). However, instead of stacking the aligned pitch contour and the score, the individual $N \times 1$ sequences are fed separately to the two encoders.

3.2.2 Model Architecture
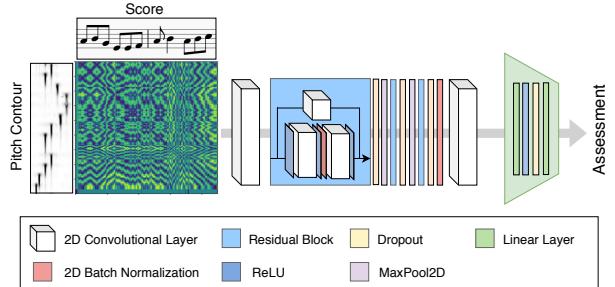
This network (see Figure 2) uses two 1-D convolutional encoders having the same architecture as SIConvNet. Each encoder has 4 convolutional blocks to extract high level feature embeddings. The performance encoder is expected to extract relevant features pertaining to the performance from the pitch contour. On the other hand, the score encoder is expected to extract the important features from the score. Assuming that the assessment rating for the performance is high if these two embeddings are similar, we use the cosine similarity $\cos(E_{\mathrm{score}}, E_{\mathrm{performance}})$ between the two embeddings to obtain the predicted assessment rating. E_{score} and $E_{\mathrm{performance}}$ are the embeddings obtained from the score and performance encoders, respectively. If the two embeddings are similar, the cosine similarity is close to one, and the model will predict a higher rating.

3.3 Distance Matrix Network (DistMatNet)

The final approach uses a distance matrix between the pitch contour and the score as the input to the network. Given the information from both the pitch contour and the score, the task of performance assessment might be interpreted as finding a *performance distance* between them. Thus, the choice of the distance matrix as the input representation allows the model to learn from the pitch differences. A Residual CNN [10] architecture is chosen for the network.

3.3.1 Input Representation

The distance matrix elements are the pair-wise wrapped distances between the pitch contour and the MIDI pitch sequence. The octave-independent wrapped-distance is

Figure 3. Schematic of the DistMatNet architecture.

used to compensate for possible octave errors made by the pitch tracker. To ensure a uniform input size to the network, the matrix is resampled to a square shape of a fixed size. Thus, a performance with constant tempo would result in an aligned path located on the diagonal. Unlike the previous two methods where the input pitch contour and the score are aligned using DTW, the distance matrix input avoids any error propagation caused by alignment errors. The choice of this input representation stems from the success of distance matrices (or self-similarity matrices) in other areas of MIR such as structural segmentation [2, 7] and music generation [28].

3.3.2 Model Architecture

The model architecture is shown in Figure 3. It is composed of 3 residual blocks. Each residual block has 2 convolutional layers. Dropout and max-pooling are added between each residual block. A classifier with two linear layers (128 features) with one ReLU and dropout layer in between is used after the residual network to perform regression prediction. We use (3,3) kernal size and 4 feature maps for all convolutional layers, 0.2 dropout rate, and a (3,3) kernal size for all max-pooling layers.

4. EXPERIMENTS

4.1 Dataset

The dataset we use to evaluate our methods is a subset of a large dataset of middle school and high school student performances. These are recorded for the Florida All State Auditions, which are separated into three bands: (i) middle school band, (ii) concert band, and (iii) symphonic band. The recordings contain auditions spanning 6 years (from 2013 to 2018), and feature several monophonic pitched and percussion instruments. Each student performs rehearsed scores, scales and a sight reading exercise. For the purpose of this study we limit our experiments to the *technical etude* for middle school and symphonic band auditions. We choose *Alto Saxophone*, *Bb Clarinet* and *Flute* performances due to these being the most popular across all pitched instruments. Table 1 shows the distribution of data across different instruments. The average duration of each performance is 30 s for middle school and 50 s for symphonic band students. The dataset also includes the musical scores that the students are supposed to perform for each exercise. The average length (in notes) of the musical

	Middle School	Symphonic Band
Alto Saxophone	696	641
Bb Clarinet	925	1156
Flute	989	1196

Table 1. Number of performances for the different instruments per band.

scores are 136 for middle school and 292 for symphonic band. Note that these scores are the same across all students performing the same instrument in the same year but vary across years and instruments.

The dataset also contains expert assessments for each exercise of a student performance. Each performance is rated by one expert along 4 criteria defined by the Florida Bandmasters' Association (FBA): (i) musicality, (ii) note accuracy, (iii) rhythmic accuracy, and (iv) tone quality. All ratings are on a point-based scale and are normalized to range between 0 to 1 by dividing by the maximum. Since we focus on pitch contours as the primary audio feature, tone quality is excluded from this study.

4.1.1 Data pre-processing

The pitch contours are extracted using the pYIN algorithm [16] with a block size and hop size of 1024 and 256 samples, respectively. The audio sampling rate is 44100 Hz. The extracted frequencies are converted from Hz to MIDI pitch (unlike the MIDI pitches from the musical score, these can be floating point numbers). Both the resulting pitch contour and musical score are normalized by dividing by 127. Finally, for the purpose of model training and evaluation, we divide our dataset into three randomly sampled subsets: training, validation, and testing. We use a ratio of $8 : 1 : 1$ for splitting the dataset.

We use *random-chunking* as a data augmentation tool when training SIConvNet and JointEmbedNet since it has shown to be useful in improving model performance [22]. First, the pitch contour is chunked into snippets of length N by randomly selecting the starting position. The corresponding aligned and length-adjusted score snippet is obtained using the method described in Section 3.1.1. We assume the chunked segment has the same assessment score as the whole recording. We do not perform chunking on our distance matrix since the matrix has already been resampled into a smaller resolution. Instead, we discuss how varying the resampling size could effect the performance in one of the experiments.

4.2 Experimental Setup

We present three experiments to evaluate our proposed methods. First, we compare the overall performance of the proposed architectures against a score-independent baseline system PCConvNet [22] which uses only the randomly-chunked pitch contour as input. This experiment also gives us an indication of the effectiveness of each of the proposed methods. Second, we look at the sensitivity of the SIConvNet and JointEmbedNet to the chunk size N. Finally, we investigate the effect of varying the resolution of the input

SIConvNet	JointEmbedNet	DistMatNet
3,089	6,144	63,417

Table 2. Number of parameters for each model.

distance matrix for the DistMatNet model. The latter two experiments were aimed at understanding the effects of the different hyper-parameters used while constructing the input data for each model. These helped us arrive at the best parameters for each approach.

The number of trainable parameters for each method is shown in Table 2. DistMatNet has a higher number of parameters because it uses a higher-dimensional input with a deeper architecture to capture high level information [10].

For each method, we trained separate models to predict each assessment criterion. Moreover, to measure the variation of each model, we trained each model on 10 different random seeds. We used M_i to represent the model training on different random seed where $i = 0 \dots 9$. A boxplot with median and variation of each M_i is shown to demonstrate the results. All the models are trained based on the mean squared error between estimated and ground truth ratings. All the models are trained with a stochastic gradient descent optimizer with a 0.05 learning rate. We apply early stopping if the validation loss does not improve for 100 epochs. The performance of all models is measured using the coefficient of determination (R^2 score):

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}, \tag{1}$$
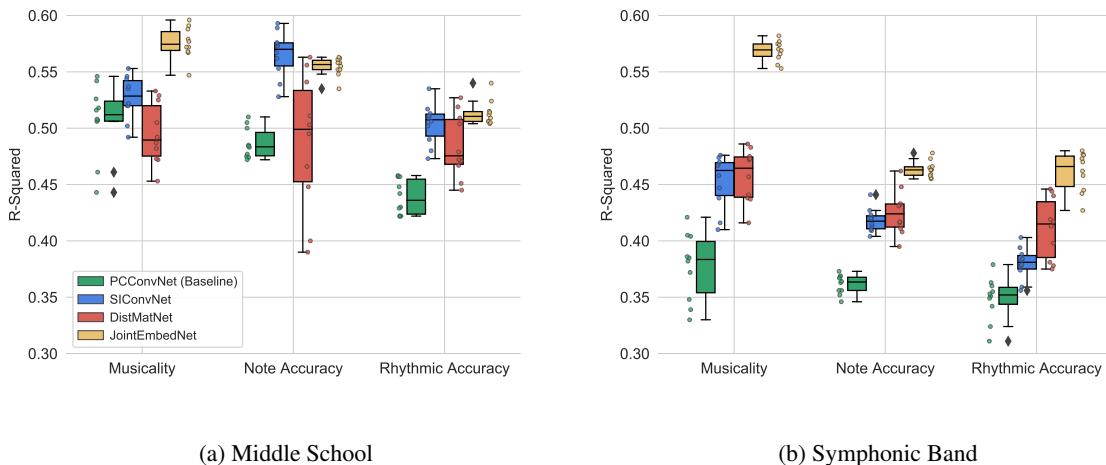
where y_i is the ground truth rating, $\hat{y}_i$ is the estimated rating, and $\bar{y}_i$ is the average of the ground truth rating. R^2 is a common metric to evaluate the fit of a regression prediction to the ground truth value.

5. RESULTS & DISCUSSION

5.1 Overall Performance

Figure 4 shows the comparative performance for all models for middle school and symphonic band. We can make the following observations (with independent t-test results reported):

(i) We compare the performance of various models trained on different band performances. All systems perform better (higher R^2 value) on the middle school recordings than on the symphonic band recordings ($p < 0.01$ except JointEmbedNet for musicality). One possible explanation for this is that symphonic band scores are usually more complicated and longer. For example, symphonic band scores tend to be performed at high tempo with high note density. The chunking into smaller lengths (and the downsampling of the distance matrix) compared to the score length might lead to a less accurate mapping to the assessment rating. An additional factor is that most performers in the symphonic band auditions exhibit greater skill level than middle school performers thus making it

(a) Middle School

(b) Symphonic Band

Figure 4. Box plots showing comparative performance (higher is better) across different models and assessment criteria.

potentially more difficult to model the differences in proficiency levels.

(ii) All score-informed models generally outperform the baseline, implying that score information is indeed helpful for MPA ($p < 0.01$ except SIConvNet for musicality, DistMatNet for musicality and note accuracy on middle school). We notice, however, that the difference between the score-independent baseline and the score-informed models is smaller for the middle school than for symphonic band. Given the significant improvement over the baseline for symphonic band performances (which have complicated scores), we speculate that the score-informed models benefit more from access to score information. In other words, a score reference becomes more impactful with increasing proficiency level while the pitch contour alone contains most relevant information for medium proficiency levels.

(iii) While the two models SIConvNet and JointEmbedNet both use the same input features, JointEmbedNet either outperforms or matches SIConvNet in all experiments. The main difference between these two architectures is that SIConvNet simply performs a regression to estimate the assessments while JointEmbedNet learns a similarity in the embedding space to model the assessments. Therefore, we can assume that JointEmbedNet is able to explicitly model the differences between the input pitch contour and score especially in the case of symphonic band where the scores are more complicated.

(iv) We observe that while DistMatNet and JointEmbedNet both utilize the similarity between the score and pitch contour, albeit at different stages of the network, JointEmbedNet typically performs better across categories and bands, and the gap is larger for musicality than for the other two categories. It is possible that the absolute pitch at the input may be important for the final assessment (octave jumps, for example, would not be properly modeled in the distance matrix). More likely, however, is that the significantly larger input

dimensionality of the matrix (compared to the aligned sequences for JointEmbedNet) negatively impacts performance. Most of the relevant information for MPA centers around the diagonal of the distance matrix with relatively small deviations depending on the students' tempo variation. Most of the distance matrix elements far from the diagonal contain redundant or irrelevant information, thus complicating the task. Another advantage that JointEmbedNet might have over DistMatNet in terms of overall assessment is that the distance is computed on the whole performance while DistMatNet computes a frame-level pitch distance, potentially complicating the task for overall quality measures like musicality.
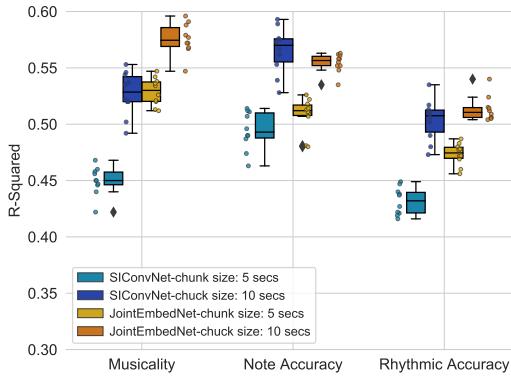
5.2 Chunk Size

In this experiment, we look at the impact of two different chunk sizes for the first two methods. Figure 5 shows the results on middle school (a) and symphonic band (b). For both SIConvNet and JointEmbedNet, a chunk size of 10 s outperforms that of 5 s across all the bands.
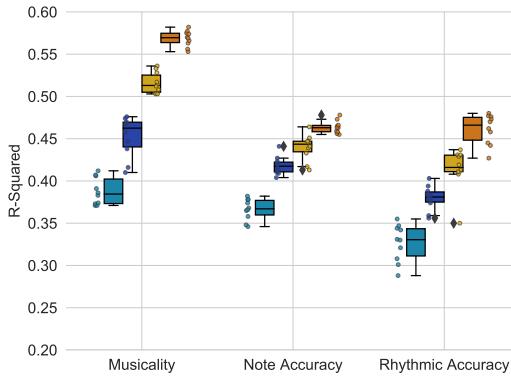
Chunking with random sampling is a form of data augmentation. By using the ground truth rating of the whole performance, the chunks are assumed to reflect the quality of the whole performance. The results show that 5 s chunks might be too short to evaluate the whole performance while 10 s chunks are much better suited regardless of category and score complexity. Chunk lengths greater than 10 s were not tested because we restricted ourselves to the length of the shortest performance in the dataset. Consequently, we used a 10 s chunk size for the experiment in Figure 4.

5.3 Distance Matrix Resolution

In this experiment, we study the impact of the different input matrix resolutions 400×400, 600×600, and 900×900, for the DistMatNet model. The results for both middle school and symphonic band are shown in Figure 6. First, the performance of rhythmic accuracy criterion tends to decrease with increasing distance matrix resolution. It might

(a) Middle School

(a) Middle School
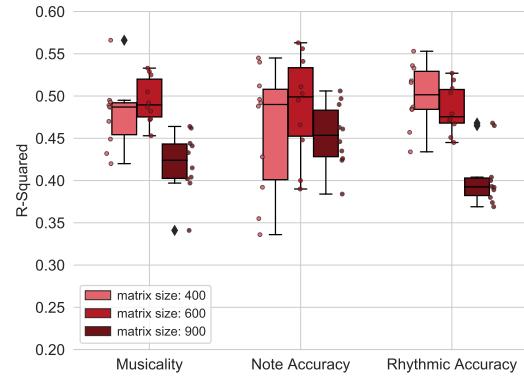
(b) Symphonic Band

(b) Symphonic Band

Figure 5. Box plots showing comparative performance (higher is better) across different chunk sizes for SIConvNet and JointEmbedNet.

Figure 6. Box plots showing comparative performance (higher is better) across different matrix sizes for the distance matrix network (DistMatNet).
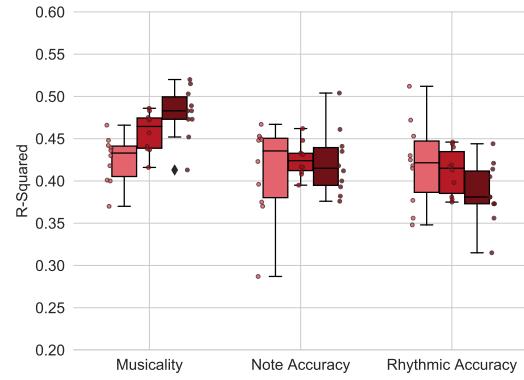
be more difficult for the same model structure to capture the complexity inside a larger matrix. This can also explain the result for middle school: although increasing the input resolution from 400×400 to 600×600 will capture more details, the performance decreases when the matrix resolution is further increased. Second, an input matrix size of 600×600 leads to a slightly higher average score (0.46) on both symphonic and middle school than the other two resolutions (0.45 for 400×400 and 0.43 for 900×900). We ended up using the 600×600 resolution for the experiment in Figure 4.

6. CONCLUSION

This paper presents three novel neural network-based methods that combine score information with a pitch representation of an audio recording to assess a music performance. The methods include: (i) a CNN with aligned pitch contour and score as the input, (ii) a joint embedding model that learns the assessment as the cosine similarity of the embeddings of both the aligned pitch contour and the score, and (iii) a distance-matrix based CNN, using a differential repre-

sentation of pitch contour and score at the input. The results show that all the methods outperform the score-independent baseline model. The joint embedding model achieves the highest average performance.

Beyond the obvious applications in software-based music tutoring systems, score-informed performance assessment models (and objective MPA in general) can benefit the broader area of music performance analysis. Models capable of rating performances along different criteria could serve as useful tools for objective evaluation of generative systems of music performance. In addition, such models could also be explored for objective analysis of inter-annotator differences in rating music performances.

In the future, we plan to incorporate timbre and dynamics information into the models as it has been shown to improve accuracy [22]. This will also enable the model to assess performances in terms of tone quality, the criterion ignored in this study. We also plan to investigate other instruments and to examine cross-instrument relationships by training instrument-specific models. Furthermore, the musical score reference could be replaced with other representations such as the pitch contour of a highly-rated performance.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] B. Bozkurt, O. Baysal, and D. Yuret. A dataset and baseline system for singing voice assessment. In *Proc. of International Symposium on Computer Music Multidisciplinary Research (CMMR)*, pages 430–438, Matosinhos, Porto, 2017.

[2] Alice Cohen-Hadria and Geoffroy Peeters. Music structure boundaries estimation using multiple self-similarity matrices as input depth of convolutional neural networks. In *Proc. of Audio Engineeing Society (AES) International Conference on Semantic Audio*, Erlangen, Germany, 2017.

[3] Johanna Devaney, Michael I Mandel, and Ichiro Fujinaga. A Study of Intonation in Three-Part Singing using the Automatic Music Performance Analysis and Comparison Toolkit (AMPACT). In *Proc. of 13th International Society of Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 2012.

[4] Sebastian Ewert and Mark B Sandler. Structured dropout for weak label and multi-instance learning and its application to score-informed source separation. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2277–2281, New Orleans, USA, 2017.

[5] Sebastian Ewert, Siying Wang, Meinard Müller, and Mark Sandler. Score-informed identification of missing and extra notes in piano recordings. In *Proc. of 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016.

[6] Felipe Falcao, Baris Bozkurt, Xavier Serra, Nazareno Andrade, and Ozan Baysal. A Dataset of Rhythmic Pattern Reproductions and Baseline Automatic Assessment System. In *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[7] Thomas Grill and Jan Schluter. Music boundary detection using neural networks on spectrograms and self-similarity lag matrices. In *Proc. of 23rd European Signal Processing Conference (EUSIPCO)*, pages 1296–1300, Nice, France, 2015.

[8] Siddharth Gururani, Ashis Pati, and Alexander Lerch. Analysis of objective descriptors for music performance assessment. In *Proc. of International Conference on Music Perception and Cognition (ICMPC)*, Montréal, Canada, 2018.

[9] Yoonchang Han and Kyogu Lee. Hierarchical approach to detect common mistakes of beginner flute players. In *Proc. of 15th International Society of Music Information Retrieval Conference (ISMIR)*, pages 77–82, Taipei, Taiwan, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, USA, 2016.

[11] Jiawen Huang and Alexander Lerch. Automatic assessment of sight-reading exercises. In *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of 32nd International Conference on Machine pyin (ICML)*, pages 448–456, Lille, France, 2015.

[13] Trevor Knight, Finn Upham, and Ichiro Fujinaga. The potential for automatic assessment of trumpet tone quality. In *Proc. of 12th International Society of Music Information Retrieval Conference (ISMIR)*, pages 573–578, Miami, USA, 2011.

[14] Alexander Lerch, Claire Arthur, Ashis Pati, and Siddharth Gururani. Music performance analysis: A survey. In *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[15] Pei-Ching Li, Li Su, Yi-Hsuan Yang, Alvin WY Su, et al. Analysis of expressive musical terms in violin using score-informed and expression-based audio features. In *Proc. of 16th International Society of Music Information Retrieval Conference (ISMIR)*, pages 809–815, Málaga, Spain, 2015.

[16] Matthias Mauch and Simon Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663, Florence, Italy, 2014.

[17] Oscar Mayor, Jordi Bonada, and Alex Loscos. Performance analysis and scoring of the singing voice. In *Proc. of Audio Engineering Society (AES) Convention*, pages 1–7, London, UK, 2009.

[18] Marius Miron, Jordi Janer Mestres, and Emilia Gómez Gutiérrez. Monaural score-informed source separation for classical music using convolutional neural networks. In *Proc. of 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.

[19] Emilio Molina, Isabel Barbancho, Emilia Gómez, Ana Maria Barbancho, and Lorenzo J Tardón. Fundamental frequency alignment vs. note-based melodic similarity for singing voice assessment. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 744–748, Vancouver, Canada, 2013.

[20] Tomoyasu Nakano, Masataka Goto, and Yuzuru Hiraga. An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features. In *Proc. of International Conference on Spoken Language Processing (INTERSPEECH)*, pages 1706–1709, Pittsburg, USA, 2006.

[21] Caroline Palmer. Music performance. *Annual review of psychology*, 48(1):115–138, 1997.

[22] Ashis Pati, Siddharth Gururani, and Alexander Lerch. Assessment of student music performances using deep neural networks. *Applied Sciences*, 8(4):507, 2018.

[23] Jordi Pons Puig, Rong Gong, and Xavier Serra. Score-informed syllable segmentation for a cappella singing voice with convolutional neural networks. In *Proc. of 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.

[24] Oriol Romani Picas, Hector Parra Rodriguez, Dara Dabiri, Hiroshi Tokuda, Wataru Hariya, Koji Oishi, and Xavier Serra. A real-time system for measuring sound goodness in instrumental sounds. In *Proc. of Audio Engineering Society Convention*, Philadelphia, USA, 2015.

[25] Hiroaki Sakoe and Seibi Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[26] Sam Thompson and Aaron Williamon. Evaluating evaluation: Musical performance assessment as a research tool. *Music Perception: An Interdisciplinary Journal*, 21(1):21–41, 2003.

[27] Amruta Vidwans, Siddharth Gururani, Chih-Wei Wu, Vinod Subramanian, Rupak Vignesh Swaminathan, and Alexander Lerch. Objective descriptors for the assessment of student music performances. In *Proc. of Audio Engineeing Society (AES) International Conference on Semantic Audio*, Erlangen, Germany, 2017.

[28] I-Chieh Wei, Chih-Wei Wu, and Li Su. Generating strcutured drum pattern using variational autoencoder and self-similarity matrix. In *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[29] Brian C Wesolowski, Stefanie A Wind, and George Engelhard. Examining rater precision in music performance assessment: An analysis of rating scale structure using the multifaceted rasch partial credit model. *Music Perception: An Interdisciplinary Journal*, 33(5):662–678, 2016.

[30] Chih-Wei Wu, Siddharth Gururani, Christopher Laguna, Ashis Pati, Amruta Vidwans, and Alexander Lerch. Towards the objective assessment of music performances. In *Proc. of International Conference on Music Perception and Cognition (ICMPC)*, pages 99–103, San Francisco, USA, 2016.

[31] Chih-Wei Wu and Alexander Lerch. Learned features for the assessment of percussive music performances. In *Proc. of International Conference on Semantic Computing (ICSC)*, Laguna Hills, California, USA, 2018.

HIERARCHICAL TIMBRE-PAINTING AND ARTICULATION GENERATION

Michael Michelashvili
Tel Aviv University
`mosheman5@gmail.com`

Lior Wolf
Tel Aviv University
`wolf@cs.tau.ac.il`

ABSTRACT

We present a fast and high-fidelity method for music generation, based on specified f0 and loudness, such that the synthesized audio mimics the timbre and articulation of a target instrument. The generation process consists of learned source-filtering networks, which reconstruct the signal at increasing resolutions. The model optimizes a multi-resolution spectral loss as the reconstruction loss, an adversarial loss to make the audio sound more realistic, and a perceptual f0 loss to align the output to the desired input pitch contour. The proposed architecture enables high-quality fitting of an instrument, given a sample that can be as short as a few minutes, and the method demonstrates state-of-the-art timbre transfer capabilities. Code and audio samples are shared at https://github.com/mosheman5/timbre_painting.

1. INTRODUCTION

The melody, as depicted by a sequence of notes, or alternatively by a sequence of frequencies, is one generic aspect of the musical experience. The dynamic loudness signal is another prominent aspect that is also almost instrument-invariant. Due to the invariance property of these two aspects, it is natural to employ them as specifications to the instrument-independent essence of a musical piece.

A prominent aspect that does depend on the instrument is the timbre. The music-AI task of timbre-transfer considers methods that receive, as input, an audio segment and a target instrument, and output the analog (melody preserving) audio in the target domain, by replacing the timbre of the original audio clip with that of the specified instrument.

Another aspect that defines a musical instrument is articulation, or the joining-up of notes. Timbre transfer methods address this implicitly with varying degrees of success. The physical properties of the instrument lead to constraints and subsequently different characteristic ways to move from one note to the next in a smooth manner. This aspect, therefore, varies considerably, e.g., between violin, guitar, and trumpet.

While this interpolation process is second nature for trained musicians, it can be sophisticated and involves the introduction of new frequencies that are not part of the original notes. See Fig. 1.

In this work, we build a hierarchical music generator network. Given a fundamental frequency (f0) and loudness inputs, the network generates audio in four different scales. While the different scales share the same architecture, they have different roles. The first (lower) scale introduce the articulation, while the top scales introduce much of the timbre and the final audio-spectrum quality, which we call timbre-painting. See Fig. 2.

The model is trained on a relatively short sample from the target instrument, typically consisting of few minutes. The network is trained to minimize multiple losses: an adversarial loss encourages the output to be indistinguishable from audio in the output domain, multi-scale reconstruction losses in the frequency domain are used to ensure that the network can recreate the training sample, and the f0 of the output is compared to the specifications.
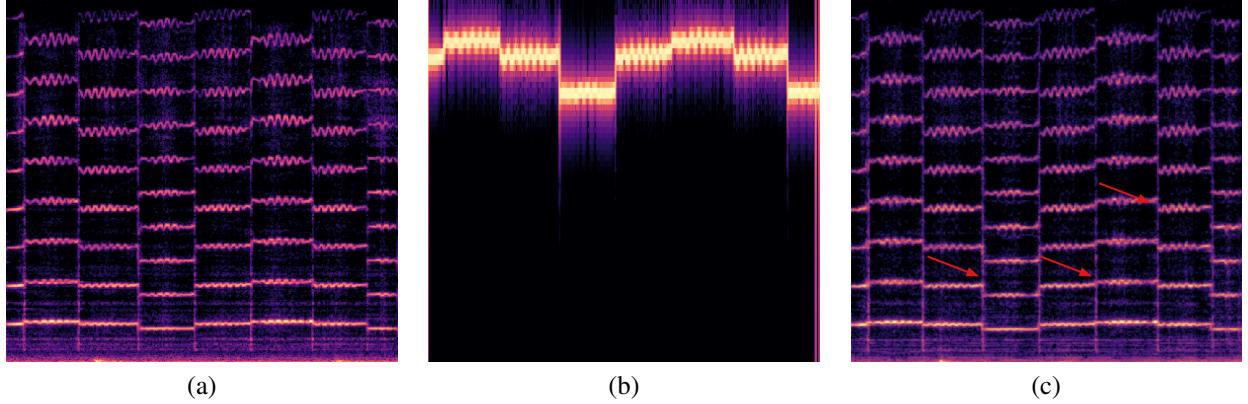
One possible application of the network is for the task of music domain transfer, similar to the application of other timbre-transfer methods. In this case, the f0 and loudness inputs are extracted from an existing audio clip and the network generates the analog music in the target domain. Our experiments show that our method generates audio that sounds more realistic and is perceived to be of a better fit to the original melody than the recent state-of-the-art method DDSP [1].

2. RELATED WORK

The task of timbre-transfer was tackled by [2]. An image-to-image pipeline that uses cycle consistency losses [3] is applied to the audio domain by representing audio signals as 2D images with the Constant-Q-Transform (CQT). To move back from the CQT representation, a WaveNet [4] synthesizer that is conditioned on CQT representation was used. Another prominent work [5] suggested to learn the audio melody by using a WaveNet Autoencoder architecture [6]. One "universal" encoder is used to represent melody from raw data, and multiple domain-specific decoders are used for audio generation. By presenting domain-adversarial loss on the encoding, this method represents only the domain-invariant data needed for generation, which is predominantly the melody. Even though this method presents impressive results on timbre transfer and audio translation, it has few major disadvantages: the reliance on large amounts of data, and the heavy computation resources required (tens of GPUs).

Figure 1. An illustration of our method's articulation capabilities. (a) The spectrogram of a violin audio. (b) The extracted fundamental frequency (f0) of the violin. (c) The results of our method. Both the timbre and the articulations were manipulated. See arrows for a few specific locations where the violin's articulation is demonstrated.

The differentiable digital signal processing (DDSP) method [1], which was proposed recently, is much more efficient with regards to both data and computational needs. The method presents a DSP hybrid model in which a synthesizer with learned parameters is used. Like our method, DDSP conditions the signal generation on f0 and the loudness signal. Therefore, it can apply timbre-transfer to any audio for which a pitch tracker, e.g., CREPE [7], can successfully extract the f0 signal.

DDSP and other methods [8] follow the high fidelity speech synthesizer of [9] in employing convolutional neural networks as shape-shifting filters to a sine-wave input. While many speech generation techniques condition the network on the f0 signal, this line of methods employ the corresponding sine-wave.

Hierarchical generation was shown to be effective for image generation tasks. The progressive GAN method [10] breaks down the generation scheme into cascading generators and discriminators, improving the image generation quality and stabilizing the training process. The SinGAN method [11] performs convincing image-retargeting and image generation, using multi-scales learning from a single input image.

3. METHOD

Our method is hierarchical and consists of generators in four different scales. All generators have the same architecture of a non-autoregressive WaveNet applied on (scale-dependent) input and conditioned on extracted audio features on each scale. The learning process is optimized to: (i) decrease the distance between the spectral representations of the generated and the target audio, (ii) minimize pitch perceptual loss in order to improve pitch coherence, and (iii) create realistically sounding examples by the usage of an adversarial loss.

3.1 Input Features

An audio sample is denoted by $\boldsymbol{x^n} = (x_1^n, \ldots, x_T^n)$, where T is the length of the signal and n is the finest scale we consider. The scaled version of it are denoted by $\boldsymbol{x^{n-1}}$,

$\boldsymbol{x^{n-2}}$, up to $\boldsymbol{x^0}$, which is the coarsest scale. The scaling is carried out by down-sampling,

$$x^{n-1}[t] = \sum_{k=0}^{K-1} x^n[tM - k]h[k] \qquad (1)$$
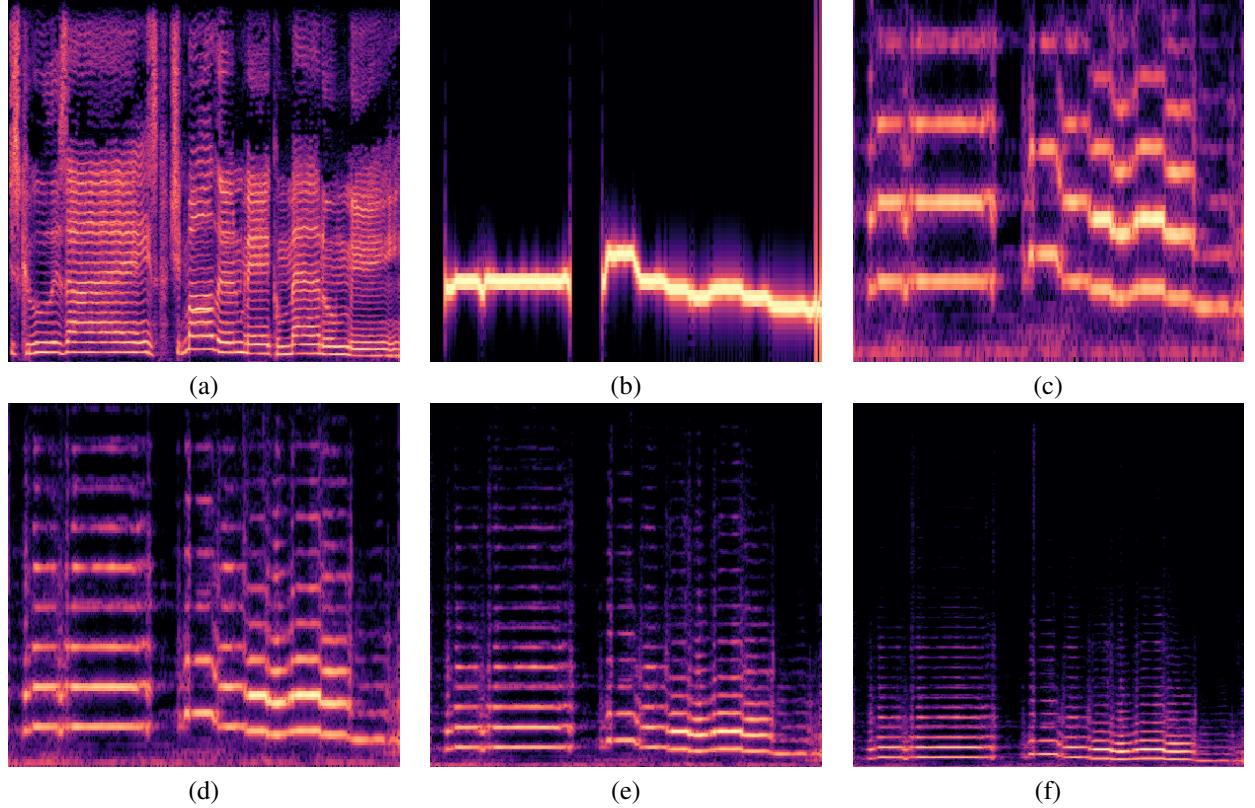
Where M is the reduction factor, h a FIR anti-aliasing filter and K the length of the filter.

In our experiments, we use four scales $j = 0..3$. The finest generates audio in 16 kHz, while the coarsest generates audio in 2 kHz. We chose the coarsest scale to be as small as possible on the articulation generation phase, yet to include the f0 signal of our target instruments (max of 1kHz as given by Nyquist rule)

In our method, audio is generated based on the specifications of the loudness of the output audio and its pitch. The other characteristics (timbre, articulation, and spectral quality) are being added by the model, based on the training sample. The loudness is given, following [12], by the A-Weighting scheme, which is a weighted sum of the log of the power spectrum. We denote the loudness extraction computation by $\text{loud}(\boldsymbol{x^j})$, which is a 1D signal of a length that is 32 times shorter than the length of the input $\boldsymbol{x^j}$, $j = 0..3$, due to the power spectrum extraction.

The fundamental frequency f0, which is also a 1D signal, is extracted using the CREPE pitch tracking network [7], as is done in [1, 13]. We denote the extracted signal by $f0(\boldsymbol{x^n})$ and compute it only at the finest-resolution scale. The CREPE network has a resolution of 250Hz, which differs from the sampling rate of our network. However, this conditioning is provided as a sine-wave at the resolution of the coarsest layer (2kHz).

Specifically, following previous work in speech [9] and music synthesis [1, 8], we apply what is known as "neural source-filtering". In this technique, instead of conditioning the generated sample directly on the extracted f0 signal, the generator is conditioned on a raw waveform that is synthesized via a single sinusoid sine-excitation, calculated from upsampled $f0(\boldsymbol{x^n})$. The $f0$ is downsampled by 32 from the input signal $\boldsymbol{x^n}$ and the coarsest scales $j = 0$, which is generated first, has a frequency that is one eighth of the

Figure 2. An illustration of the hierarchical generation process. (a) A spectrogram depicting the original melody as sang by a male singer. (b) The extracted fundamental frequency (f0) of the melody. (c-f) The generated audio for a saxophone from the coarsest scale x^0 to the finest x^3, respectively. While articulation-based manipulations are already seen in x^0, the full effect of the timbre and spectral-quality is only observed at the final output x^3.

original audio. Scaling is, therefore, by a factor of 4. We denote the generated waveform by $\eta(f0(x^n))$.

$$\eta(f0(x^n)) = sin(\sum_{k=0}^{T} 2\pi\uparrow f0(x^n)_k/f_s), \qquad (2)$$

where f_s is the sample rate of the audio and $\uparrow$ denotes an upsampling operator.

3.2 Hierarchical Generation

The generated waveform $\eta(f0(x^n))$ serves as the input to the lowest scale generator in the hierarchy, which is denoted by G^0. Similarly to our other generators and unlike conventional GAN generators, the generator does not receive random noise as input.

In our method, we propose a conceptual relaxation to the audio generation task, and divide the generation into two distinct phases: timbre painting and articulation on the lowest scale, followed by upsampling networks which learn to generate higher resolution audio based on the previous scale. By doing so, we separate what we consider the most difficult part in the generation, namely converting a sine wave into well-articulated music, from the aspects of timbre painting and spectral quality adjustment. Therefore, fewer errors are introduced during the generation process and the method produces more coherent audio samples.

Denote by $z^j = loud(x^j)$. A set of input encoding networks E^j transforms the raw input signal z^j into a sequence of vectors, which G is conditioned upon.

The lowest scale generator operates as follows:

$$\hat{x}_0 = G^0(\eta(f0(x^n)), E^0(z^0)), \qquad (3)$$

where the second input is the conditioning signal.

The following generation steps receive as input the output of the previous scale generator:
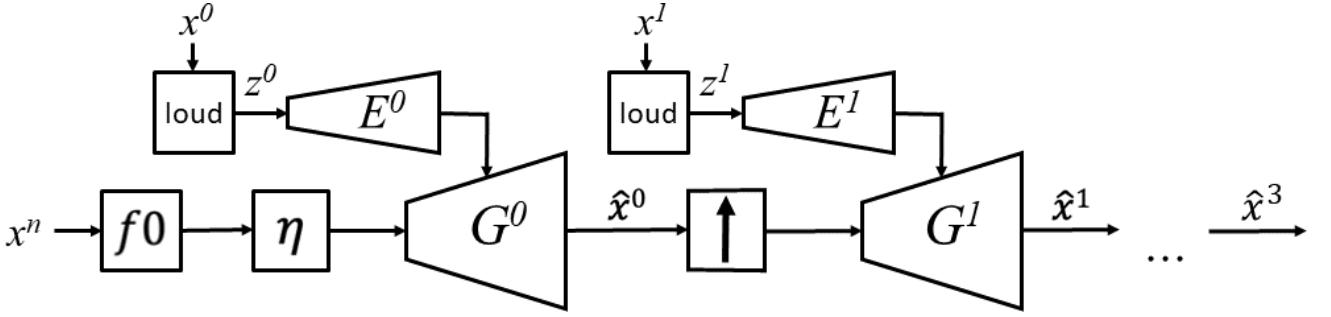
$$\hat{x}^j = G^n(\uparrow(\hat{x}^{j-1}), z^j), \qquad (4)$$

where $\uparrow(\hat{x}_{n-1})$ is an upsampled signal that matches the next scale. An illustration of the generation process is given in Fig. 3.

3.2.1 Architecture

The architecture of the generators and discriminators is similar to that of [14]. Each generator is composed of 30 layers stacked into three stacks. The kernel size is 3, using 64 residual channels and 64 skip channels. The dilation is exponentially growing in each stack, providing a receptive field of 3072 samples, which translates to a window size of 1.5sec on the lowest scale and 192ms on the finest.

The input encoder E^j is composed of instance normalization, followed by 1D-convolution with kernel size of 1 that is applied on the condition input z. The number of output channels is 80. The output of E^j is provided after upsampling via convolutional layers and nearest neighbor interpolation to the temporal dimension of the input signal.

Figure 3. An illustration of the generation process. The generator of the coarsest scale receives as input a sine-wave that is based on the fundamental frequency of the input sample. All generators are conditioned on the loudness signal of the appropriate scale. The output of the first generator G^0 serves as the input for the subsequent generator G^1 and so on.

Training involves a set of discriminators D^j, one per scale. Each discriminator is composed of 10 layers of 1D-convolution, followed by leakyReLU with negative slope of 0.2. The kernel size is 3, and 64 channels are used per layer. The dilation is growing linearly. Weight normalization is applied both on the generator and the discriminator.

3.3 Training

The learning setup and objective functions are the same for all the scales, with respect to the target audio signal. Conveniently, each generator G^j is trained separately, after the previous generator G^{j-1} is completely trained. We found that using the weights of the previous scale generator G^{j-1} to initialize the weights of G^j leads to faster convergence than random initialization on every scale. Similarly, the discriminator D^j that provides the adversarial training signal to the generator G^j is initialized based on D^{j-1}.

At each scale j, we obtain a training set S^j by dividing the training sample, after it has been downsampled to scale j to audio clips x^j of length 2sec.

3.3.1 Objective function

A time-frequency reconstruction loss is used to align to the generated audio sample with the target audio. Specifically, the spectral amplitude distance loss [15, 16], in multiple FFT resolutions [1, 9, 14] is used. For a given FFT size m, the spectral amplitude distance loss is defined as follows:

$$\mathcal{L}_{\text{recon}}^{(m,j)} = \sum_{\boldsymbol{x}^j \in S^j} \left(\frac{\||\text{STFT}(\boldsymbol{x}^j)| - |\text{STFT}(\hat{\boldsymbol{x}}^j)|\|_F}{\|\text{STFT}(\boldsymbol{x}^j)\|_F} + \frac{\|\log|\text{STFT}(\boldsymbol{x}^j)| - \log|\text{STFT}(\hat{\boldsymbol{x}}^j)|\|_1}{N} \right)$$
(5)

where $\hat{\boldsymbol{x}}^j$ is given by Eq. 3 and Eq 4, $\|\cdot\|_F$ and $\|\cdot\|_1$ denotes the Frobenius and the L_1 norms, respectively. The first element in the sum penalizes dominant bins in the magnitude while the second penalizes the silent parts. STFT denotes the magnitude of a Short-time Fourier transform with N elements in the spectrogram.

The multi-resolution loss is defined as the mean of the above loss for multiple scales:

$$\mathcal{L}_{recon}^j = \frac{1}{N_M} \sum_{m \in M} \mathcal{L}_{recon}^{(m,j)}$$
(6)

where $M = [2048, 1024, 512, 256, 128, 64]$ and $N_M = 6$ is the number of FFT scales. Using the multi-resolution loss, we implicitly constrain the phase of the output signal to be correct and prevent artifact noises.

To make the generated quality of the audio signals sound realistic, we introduce an adversarial loss. On each scale, we apply a different discriminator D^j to account for different statistics between scales. We follow the least-squares GAN [17], where the discriminator minimizes the loss

$$\mathcal{L}_D^j = \sum_{\boldsymbol{x} \in S^j} [||1 - D^j(\boldsymbol{x}^j)||_2^2 + ||D^j(\hat{\boldsymbol{x}}^j)||_2^2]$$
(7)

Each trained generator G^j minimizes the adversarial loss (recall that $\hat{\boldsymbol{x}}^j$ is computed with G^j):

$$\mathcal{L}_{adv}^j = \sum_{\boldsymbol{x}^j \in S^j} ||1 - D^j(\hat{\boldsymbol{x}}^j)||_2^2$$
(8)

To further improve the generation quality, we add a perceptual loss [18] on the generator output, using the CREPE network [7]. Denoting the mapping between the input signal x and the intermediate activations the CREPE network as $h(\uparrow x)$, which requires an upsampling to 16kHz, this loss takes the form:

$$\mathcal{L}_{percep}^j = \sum_{\boldsymbol{x}^j \in S^j} \|h(\uparrow \boldsymbol{x}^j) - h(\uparrow \hat{\boldsymbol{x}}^j)\|_1 .$$
(9)

The optimization with this loss requires the upsampling operator to be differentiable.

In order to support a more direct comparison of the methods, following DDSP [1], the fifth max-pool layer of the small CREPE model is employed.

Overall, the optimization loss for a generator G^j, is defined as:

$$\mathcal{L}_G^j = \mathcal{L}_{recon}^j + \alpha \mathcal{L}_{adv}^j + \beta \mathcal{L}_{percep}^j$$
(10)

where α, β are weight factors that balance the contribution of each loss term.

4. EXPERIMENTS

We conduct timbre-transfer experiments for multiple instruments, and compare the results to the state-of-the-art timbre transfer method DDSP [1].

Instrument/Method	Target Similarity		Melody Similarity	
	DDSP	Our	DDSP	Our
Cello	4.11 ± 0.16	4.24 ± 0.16	4.00 ± 0.32	4.01 ± 0.49
Saxophone	3.09 ± 0.53	3.47 ± 0.54	3.87 ± 0.41	3.91 ± 0.53
Trumpet	3.29 ± 0.45	4.01 ± 0.33	3.99 ± 0.29	4.11 ± 0.51
Violin	4.02 ± 0.35	4.13 ± 0.27	4.13 ± 0.39	4.22 ± 0.39
All samples	3.63 ± 0.60	3.96 ± 0.46	4.00 ± 0.36	4.06 ± 0.50

Table 1. MOS evaluation for the timbre transfer task for multiple target instruments.

4.1 Datasets

We used the University of Rochester Music Performance (URMP) dataset [19], a multi-modal audio-visual dataset containing classical music pieces. The music is assembled from separately recorded audio stems of various monophonic instruments. For our experiments, we used only the separated audio stems for each instrument. f0 extraction was carried out by CREPE [7], although the URMP dataset provides ground truth melody signals, since we wanted to apply similar methods during train and test.

We trained both the baseline DDSP [1] method and our model on generating four different instruments from the URMP dataset: cello, saxophone, trumpet and violin. As a prerocessing step the audio files were resampled to 16kHz. To improve the ability of learning meaningful f0 representation we removed in each dataset samples which achieved less than 0.85 mean confidence on CREPE extractor. Each dataset was separated into a training and evaluation set by 0.85/0.15 split. After the preprocessing, we ended up with small dataset sizes: 6.5 minutes of cello, 6 minutes of saxophone, 17 minutes of trumpet and 39 minutes of violin.

4.2 Experiment Setup

Our models were trained with α=1 and β=1. We used the Adam optimizer [20] with a learning rate of 0.0005 for the generators and 0.0001 for the discriminators. Each scale was trained for 120K iterations, with batch sizes of 32, 16, 8 and 4, from coarsest to finest. The learning rates were halved after 60K iterations. The discriminators were introduced to the training process on iteration 30K. To improve the robustness of our method we added a random Gaussian noise with a standard deviation of 0.003 to the $\eta(f0(\boldsymbol{x^n}))$ signal, inspired by [9].

For the baseline evaluation of the DDSP method, the open source GitHub implementation [1] provided by the authors of [1] was used. The experiments were carried out for 100K iterations with a batch size of 16. The hyperparameters used are the ones provided by the recipe available in that repository.

4.3 User Study

To inspect the results of the timbre transfer experiments we carried out a mean opinion scores (MOS) evaluation. We sampled six audio clips varying from 5-10s, long enough

for good evaluation. The origin instruments are: clarinet, saxophone, female singer, male singer, trumpet and violin. For each audio sample, we conducted timbre transfer using the four models of the target instruments, resulting in a matrix of 24 inspection files for our method and 24 for the baseline. The timbre-transfer was done by extracting the loudness and pitch features from the source audio, aligning pitch key to the target (if needed) and generation procedure. The evaluations samples are available in the supplementary material. Twenty raters were asked to rate the generated outputs by two criteria: (i) target similarity to the transferred instrument, and (ii) the melody similarity to the original tune. Scores vary on a scale of one to five.

4.4 Results
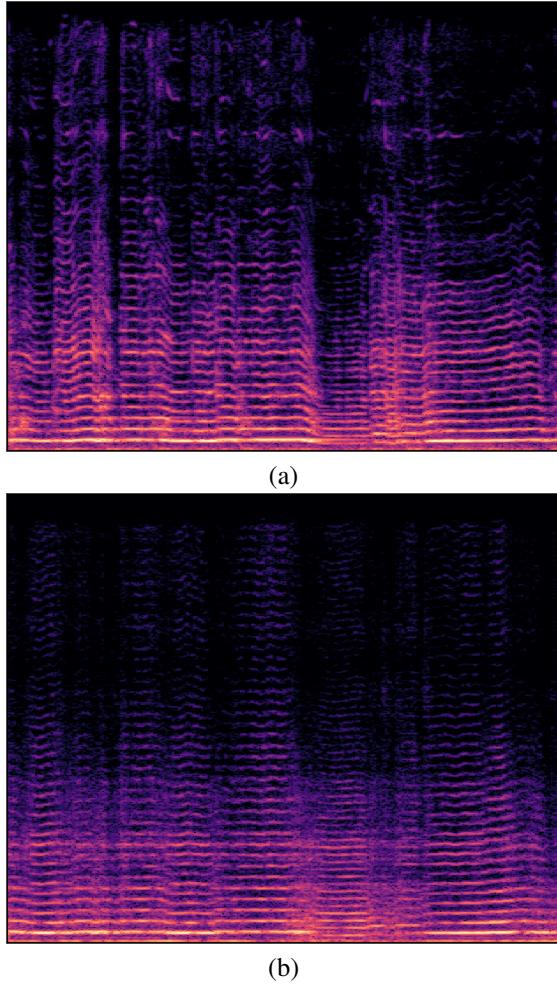
As can be seen in Tab. 1, our method outperforms DDSP both by the melody similarity and target similarity. While the baseline method gets a relatively close score on melody similarity, it is inferior in sound quality and its ability to mimic the target instrument. For example, in some cases DDSP fails to imitate the target domain timbre, and produces a sine-sounding signal in the correct pitch. An example of a challenging conversion is depicted in Fig. 4.

The high melody preserving results of both methods reflect the fact that both utilize a meaningful f0 sine-wave signal, which aligns the output melody well with the input melody. However, the target similarity results can be explained through the crux of the DDSP mechanism: the learnable function on this network optimizes control parameters of a deterministic noise-additive synthesizer, thus it is upper bounded by the quality of the best-setup synthesizer. Our method, on the other hand, enjoys the expressiveness of a fully capable neural generator, thus can deviate considerably from the source, if needed, in order to generate realistic sounds.

4.5 Data efficiency

Another advantage of our method is the need for a minimal amount of training data to generate high quality samples. Successful timbre-transfer results are produced from datasets of few minutes long. For comparison we have trained a state-of-the-art WaveNet based model for music translation [5] on two different datasets: URMP, as presented above, and a 30min subset of MusicNet [21], as discussed in Sec. 5. In both cases, the music translation method [5] failed due to the limited amount of data.

(a)

(b)

Figure 4. A challenging conversion from a female voice to a cello. (a) The results of DDSP. (b) Our results. While DDSP introduces synthetic noise in order to bridge the different characteristics of the two domains, our method successfully manages to overcome and adapt the input signal to cello's articulation and timbre .

5. DISCUSSION

Recent music AI models vary in the number of parameters and in the required size of the training data. The very recently introduced jukebox model [22] was trained on over a million songs using hundreds of GPUs, and include 7 billions parameters. The autoencoder-based music-domain translator [5] was trained on hours of audio, using tens of GPUs and includes 42M parameters. In comparison, models such as ours are trained on a single GPU, require minutes of audio, and have orders of magnitude less parameters, 1.4M on each scale in our case. The total number of parameters is even smaller than the lean DDSP model, which is of 6M parameters. Taking into account the fact that each scale is trained separately, our model is much more accessible to universities and other small-scale research labs than the other models in the literature.

The method generates sound by shape-shifting a sinewave, which serves as the skeleton of the rich-timbre painted output. Using scales reflects the inherent structure of the musical audio signal, which is composed of

harmonies on different pitch resolutions. The utilization of this strong prior allows us to achieve state-of-the-art results much more efficiently.

The hierarchical structure, which is natural for music generation, also exists in other methods, but in a different way. In the jukebox model, the hierarchy is used separately in the encoders and in the decoders, i.e., all encoder scales are applied, followed by the decoder scales. In our model, there is an interleaving structure in which generation is completed at the lowest scale (including both input encoding and the WaveNet decoder), moving to the processes of the next scale and so on.

Limitations   The economic nature of the model is not without limitations. Unlike the jukebox model, our model does not produce a discrete encoding that can be used (together with an sizable transformer model) for composing new music. In order to add a similar capability, we would need to quantize the input encoding modules (E^j) using techniques such as VQ-VAE [23] and to train an autoregressive model for each level of the hierarchy. Alternatively, any composition method can be used to generate the bare-bones input signal of the network, which would then add the articulation and the timbre to create a richer musical experience.

In the current form, unlike both jukebox and the autoencoder music translator, our model does not share information between different domains, and needs to be retrained on each domain. It is not difficult, however, to modify it to be conditioned on multiple target domains, a path that has been followed many times in the past for other WaveNet-based generators.

The current method relies on the f0 signal as extracted by a pretrained network that has been trained on monophonic instruments. Since the pitch tracker we employ was trained on monophonic instruments [7], the results on polyphonic instrument are mostly reasonable but not always. When successful, our method is successful in transforming the melody to the learned monophonic target domains. However, training polyphonic target instruments remains a challenge since it relies on such a success across the training samples. In the supplementary we present results obtained for polyphonic instruments (keyboard and piano samples from MusicNet), for both our method and DDSP. Both methods succeed to some degree with our method presenting what we consider to be a slight advantage (see supplementary samples). As future work, we note that our method can be readily extended to employ encoders, such as the ones of [5, 22], which were trained on large collections of polyphonic music.

6. CONCLUSIONS

We present a novel method of music generation which relies on neural source filtering and hierarchical generation. The method achieves high quality audio generation despite training on small training datasets. The generated input is conditioned on loudness and pitch signals, which are almost source-agnostic, and the characteristic articulation and timbre of the target instrument are introduced through a series of generators.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *ICLR*, 2020.

[2] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer," in *ICLR*, 2019.

[3] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[4] A. v. d. Oord, S. Dieleman *et al.*, "WaveNet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[5] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A universal music translation network," in *ICLR*, 2019.

[6] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *ICML*, 2017.

[7] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *ICASSP*, 2018.

[8] Y. Zhao, X. Wang, L. Juvela, and J. Yamagishi, "Transferring neural speech waveform synthesizers to musical instrument sounds generation," in *ICASSP*, 2020.

[9] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter-based waveform model for statistical parametric speech synthesis," in *ICASSP*, 2019.

[10] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *ICLR*, 2018.

[11] T. R. Shaham, T. Dekel, and T. Michaeli, "Singan: Learning a generative model from a single natural image," in *ICCV*, 2019.

[12] B. C. Moore, B. R. Glasberg, and T. Baer, "A model for the prediction of thresholds, loudness, and partial loudness," *Journal of the Audio Engineering Society*, vol. 45, no. 4, pp. 224–240, 1997.

[13] L. Hantrakul, J. Engel, A. Roberts, and C. Gu, "Fast and flexible neural audio synthesis," in *ISMIR*, 2019.

[14] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *ICASSP*, 2020.

[15] A. v. d. Oord *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," *ICML*, 2018.

[16] S. Ö. Arık, H. Jun, and G. Diamos, "Fast spectrogram inversion using multi-head convolutional neural networks," in *IEEE Signal Processing Letters*, 2018.

[17] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *ICCV*, 2017.

[18] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.

[19] B. Li, L. Xinzhao, D. Karthik, D. Zhiyao, and S. Gaurav, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *IEEE Transactions on Multimedia 21.2 (2018)*, pp. 522–535, 2018.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2016.

[21] J. Thickstun, Z. Harchaoui, and S. Kakade, "Learning Features of Music From Scratch," in *ICLR*, 2017.

[22] P. Dhariwal, H. Jun, C. Payne, J. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv:2005.00341*, 2020.

[23] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural Discrete Representation Learning," in *NIPS*, 2017.

FROM MUSIC ONTOLOGY TOWARDS ETHNO-MUSIC-ONTOLOGY

Polina Proutskova[1]   **Anja Volk**[2]   **Peyman Heidarian**[3]
György Fazekas[1]
[1] Center for Digital Music, Queen Mary University of London, UK
[2] Department of Information and Computing Sciences, Utrecht University, Netherlands
[3] Department of Computer Science The University of Waikato, NZ

proutskova@googlemail.com, g.fazekas@qmul.ac.uk

ABSTRACT

This paper presents exploratory work investigating the suitability of the Music Ontology [33] - the most widely used formal specification of the music domain - for modelling non-Western musical traditions. Four contrasting case studies from a variety of musical cultures are analysed: Dutch folk song research, reconstructive performance of rural Russian traditions, contemporary performance and composition of Persian classical music, and recreational use of a personal world music collection. We propose semantic models describing the respective domains and examine the applications of the Music Ontology for these case studies: which concepts can be successfully reused, where they need adjustments, and which parts of the reality in these case studies are not covered by the Music Ontology. The variety of traditions, contexts and modelling goals covered by our case studies sheds light on the generality of the Music Ontology and on the limits of generalisation "for all musics" that could be aspired for on the Semantic Web.

1. INTRODUCTION

Non-Western musical traditions are of interest to MIR research for several reasons: firstly, alongside Western classical and popular music, which have been studied extensively in MIR, the musics of other cultures are analysed in their own right [6, 13, 16, 25]; secondly, other musical cultures often present difficult, non-standard datasets and examples, showing the limits of existing MIR approaches [22, 30, 39]; finally, including non-Western musical traditions allows for a broader view of music and leads to new, more generally applicable technical solutions [24, 28, 40]. In this paper, we explore the latter avenue with the view of generalising existing standards of semantic modelling in music to include non-Western musical traditions.

Ontology in computer science is commonly defined as an "explicit formal specification of a shared conceptualisation of a domain " [15]. An ontology represents consen-

sual knowledge about the entities and their relationships in an area, preferably expressed using a machine-processable formal language that supports some form of inference [12]. This could be logic based, while more recently, ontologies found their use in machine learning as a mechanism to help structuring training data, formalise constraints, or become an integral part of the inference process [42].

The Music Ontology (MO) [32, 33] is among the most comprehensive ontologies for the music domain, with broad ranging applications [9, 38] from recommendation systems [49] to live performance [51], and numerous extensions covering music production [10, 11] and audio effects [53, 54], audio features [1], music theoretical concepts [34, 43, 46], smart instruments and more generic or other "Musical Things" [48]. The ontology is based upon several broadly accepted domain models (see Section 2) adopted to the music domain. Moreover, it has been applied successfully to model jazz [31], a tradition distinct from Western classical and pop music; and it was found to be beneficial for modelling Chinese musical tradition due to its flexibility and layered structure [45]. This makes it a primary candidate for our analyses.

The Music Ontology makes general claims about representing discographic information, music creation, performance, production and consumption. Yet it has so far mainly been applied to Western music. MIR researchers with expertise in ethnomusicology [29, 50] suggest that computational approaches to non-Western music should all be culture and use case specific. We therefore aim to answer the following questions: Is the Music Ontology capable of representing the domains of non-Western musical traditions? What are the gaps that the Music Ontology fails to model? Can or should the Music Ontology be generalised to encompass many (or all) musical traditions? What are the limits of such generalisation?

While political and geographic borders, language and religion play an important role in forming musical traditions, modelling the domain of such a tradition goes far beyond adding a geo location. For instance, Kurdish music in northern Iraq is different from Kurdish music in Iran and Turkey; Persian musics in Iran, Afghanistan and Tajikistan are also different, even though people speak in the same language (Persian); likewise Azerbaijan and Armenia have different religions and languages, but their musics are very close. Also, music of a diaspora sometimes adheres closely to the original traditions and sometimes fuses with the mu-

sic of the host community, importing new elements and establishing new trends.

In addition to differences in musical systems and repertoires, how people create, perform and listen to music varies between cultures. To account for this diversity, we chose four case studies from a variety of musical traditions. The use cases are representative of the chosen traditions, based on authors' expertise as practitioners, researchers and consumers of those musical cultures. While by no means exhaustive, this investigation employs a qualitative approach to test the usefulness and representativeness of the Music Ontology in a large variety of contexts outside of Western classical and popular music.

First, we look at a Dutch state institution collecting folk songs of a tradition now largely extinct and how research on this collection is conducted (Sec. 3). Secondly, contemporary performance and composition in Iranian music are explored, encompassing Persian classical art music, folk music of many Iranian population groups and Western influences (Sec. 4). Thirdly, we take the genre of world music into consideration, where recording and consumption are broadly in line with Western popular music (Sec. 5). Additionally, we turn to Russian village music and how it is being actively revived through field research and performance (see Supplement [1]).
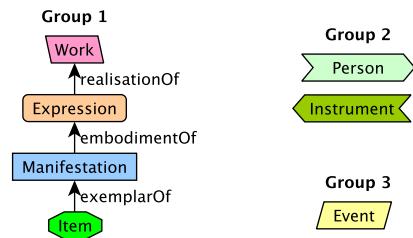
Dutch folk songs are a representative of folk music traditions of Western Europe and North America in our study; Russian village music is a polyphonic vocal tradition, which are common throughout Eastern Europe, and are found in other parts of the world. Iranian music is a maqamic tradition, strongly connected to the modal traditions encompassing the Near East, North Africa, and South Asia. The world music genre does not represent any particular culture and can include all kinds of musical content from around the world.

To illustrate how the Music Ontology (MO) classes and properties can or cannot be used to model our case studies, we introduced a consistent form- and colour coding throughout this paper: MO classes and their subclasses have solid line borders while other classes have dashed line borders; MO properties are thick blue arrows with straight heads; properties not present in the Music Ontology are thin red.

2. PREVIOUS WORK

The main standard for semantic modelling in cultural heritage is FRBR (Functional Requirements for Bibliographic Records). It is a conceptual model for describing entities and relationships in libraries, museums, and archives [47]. FRBR was developed by the International Federation of Library Associations and Institutions (IFLA) and is widely used by cultural institutions around the world, in particular for electronic cataloguing of physical and digital objects. It provides the basis for interoperability between holdings, collections, and datasets [3].

FRBR Group 1 defines four main entities to represent the products of intellectual or artistic endeavour: "Work

Figure 1. FRBR conceptual model. The shape and colour coding exemplified here is used throughout this paper to indicate classes implementing FRBR concepts

(a distinct intellectual or artistic creation) and expression (the intellectual or artistic realisation of a work) reflect intellectual or artistic content. Manifestation (the physical embodiment of an expression) and item (a single exemplar of a manifestation) reflect physical form." Group 2 includes persons and corporate bodies responsible for the custodianship of Group 1 intellectual or artistic endeavours (e. g., creators, consumers). Group 3 includes events and places [17] (Fig. 1).

The Music Ontology [32] provides a vocabulary for publishing and linking a wide range of music-related data on the Web [2]. It builds on four main ontologies: FRBR Ontology [3] (Fig. 1), the Timeline Ontology [4], the Event Ontology [5] and FOAF [6]. Fig. 2 illustrates how the Music Ontology classes implement FRBR. It has been extended to describe a variety of musical domains, such as audio content (Audio Features Ontology [7] [1]), recording sessions (Studio Ontology [8] [10]) and exploration, transformation and redistribution of audio content (AudioCommons Ontology [9] [4]). The Jazz Ontology [31] is a semantic model successfully developed on the basis of MO. It illustrates how the Music Ontology requires "tweaking" with shortcuts, new or qualified properties and some additional concepts to describe a musical tradition other than Western popular or classical music.

MusicBrainz [10] is the largest crowd-sourced collection of music metadata online. It has its own semantic model [18], focused on discographic information about published CDs, therefore less suitable to musical traditions which are not centred around published products.

Tian et al. [45] presented a detailed analysis of metadata standards in existence in 2013, including the Music Ontology, and their ability to model the domain of Chinese traditional music. They identified several aspects which were not covered by existing standards: function (purpose of creation, occasion of performance), performance practice (vocal style, stage performance, cosmetics and props, per-

---

[1] Supplementary material: https://osf.io/5qxdb/

[2] http://musicontology.com
[3] http://vocab.org/frbr/core.html
[4] http://purl.org/NET/c4dm/timeline.owl
[5] http://purl.org/NET/c4dm/event.owl
[6] Friend of a Friend ontology, describing relationships between persons:http://xmlns.com/foaf/spec/
[7] https://w3id.org/afo/onto/
[8] http://isophonics.net/content/studio-ontology
[9] https://w3id.org/ac-ontology/aco
[10] https://musicbrainz.org
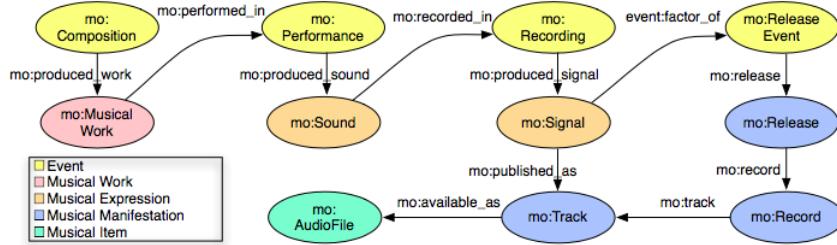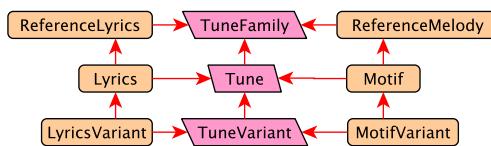
Figure 2. A fragment of the Music Ontology: key concepts and selected properties describing the music production workflow, showing the FRBR layers [12]

forming skills), musical characteristics (intonation, temperament), historical context, ethnic group, etc.

Further to cultural heritage sector, Coladangelo [5] provides a comprehensive description of contemporary (2020) semantic frameworks representing cultural heritage, including those related to music. Goienetxea et al. [14] describe an ontology representing Basque folk songs based on CIDOC Conceptual Reference Model (CIDOC CRM) - a complimentary standard for cultural heritage used primarily in the context of architecture and museum collections. FRBRoo is an object oriented model harmonising FRBR and CIDOC CRM [36]. Strle and Marolt [41] modelled Slovenian folk songs and chimes music based on FRBRoo. DOing REusable MUSical (DoReMus) Project [23] developed a model, also based on FRBRoo, describing varied collections from three French cultural institutions.

In this paper we conduct four diverse case studies originating from different musical traditions, analysing the ability of the Music Ontology to model their domains. The following sections describe the case studies: the cultural context, the musical content to be modelled, specific domain characteristics, providing diagrams of semantics models. We wrap up with a discussion of commonalities and differences displayed by the case studies and their application of the Music Ontology, the advantages and the limits of a generalised Ethno-Music-Ontology.
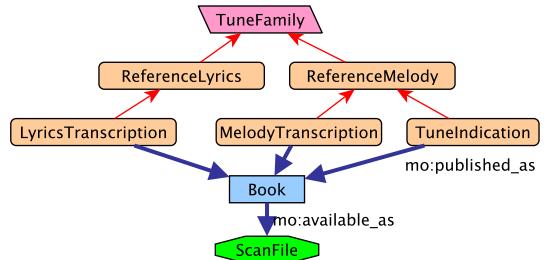
3. CASE STUDY: FOLK SONG RESEARCH ON THE DUTCH FOLK SONG ARCHIVE

Figure 3. The oral transmission (songs learnt and passed on through listening and participation) introduced continuous changes, giving rise to the coexistence of *TuneVariants*
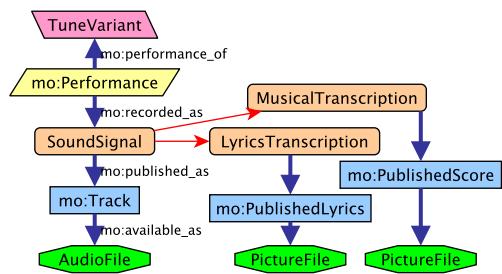
The history of Western European folk song collection and research stretches back centuries: before the emergence of audio recording, folklorists wrote down folk songs performed by their informants or encountered in the field, which were then released in printed collections.

Songbooks would often only contain the lyrics; later, more research oriented editions would include a notated melody transcription. The idea of *TuneFamilies* (Fig. 3) – clusters of tunes descending from a common "ancestor" - was in line with other disciplines such as linguistics [2,7]. This line of inquiry was strengthened by the requirements of the medium – the book – used to publish the songs: usually only one representative of a tune family would be included in a print collection to avoid repetition (Fig. 4).
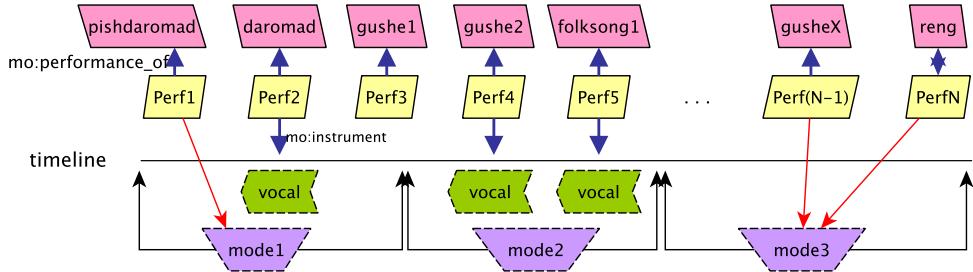
Figure 4. A *TuneFamily* was represented in a print book collection by only one of its member tunes.

The assignment of a tune to a family is performed manually by experts. An annotation experiment [52] has shown that the most salient feature for experts to assign songs to the same tune family was the presence of common melodic *Motifs*. It also confirmed the emergence of a prototypic *ReferenceMelody* representing a tune family (Fig. 3).

Figure 5. Audio field recordings capture real-life performances of folk songs and are stored in digital files.

When audio recordings of Dutch folk songs and digital processing were introduced, there was no need to limit the publication to just one representative of a family. A

Figure 6. Traditional Iranian performance: The daràmad (opening section) comes at or near the beginning of the performance, and the following gushés are organised according to a gradually ascending pitch scheme, until a forud (cadence) leads a return to the original mode. During the modulations, the modal tonic gradually moves upwards. Usually, metered gushés are played between non-metered gushés. [55]

contemporary database of the Dutch folk songs (the Dutch Song Database [21]) contains metadata on 173K song occurrences from song books, manuscripts and field recordings from the 12th century to the present day. Songs from the earlier printed collections are linked to audio *Recordings* of their *Performances*, the *Lyrics*, the *Scans* of the book pages and *Transcriptions* in digital notation (Fig. 5). The songs are linked to other songs with the same lyrics, or the same *ReferenceMelody*, or the same *MelodicIncipit*. A general melodic similarity search on the whole database is a new tool that facilitates song relationship discovery.

The diagrams in Figs. 3, 4 and 5 show that all the instances can be represented by MO classes or their subclasses (solid borders). Often MO properties (thick blue arrows) can be used. Yet connections between *Tunes* and *TuneVariants* from a *TuneFamily* (Fig. 3), which are paramount in folk song research, are not represented in the Music Ontology (red arrows). The relationships between primary and derivative kinds of *MusicalExpressions*, e.g. a melody and its transcription (Figs. 4 and 5), could be modeled via an event of transcription, analogously to *Sound -> RecordingEvent -> Signal* connections in the Music Ontology (Fig. 2). In contrast, the relationship between a *MusicalWork* and its *Expression*, which is similarly represented via a composition event in the Music Ontology (Fig. 2), cannot be used in the context of folk songs or traditional music more generally: there is usually no composer and no single event in which a song is created.

4. CASE STUDY: PERSIAN MUSIC - CONTEMPORARY COMPOSITION AND PERFORMANCE

Music in Iran is categorised, according to a scheme devised by Farhat, into urban, ethnic and pop [8]. Urban music, prevalently heard in the larger cities, includes both classical art music and pop music. Classical Persian music consists of free-rhythmic pieces (àvàz) and rhythmic pieces, typically in 2/4, 4/4, or 6/8. Ethnic music, which is in an Iranian form of maqàm, is that of the various ethnic groups living in towns, villages, deserts, and in the mountains. In addition to pieces in free and simple rhythms, irregular rhythms such as 5/8 and 7/8 are more often encountered in ethnic music. Classical Persian music uses

more ornaments, complex melodies and free rhythms than ethnic music. Iranian pop music, which has dominated the music scene in Iran since the mid-twentieth century [8], draws on either or both of the classical and ethnic traditions; it tends to simplify them and to reflect influences from other cultures, notably Western pop music.
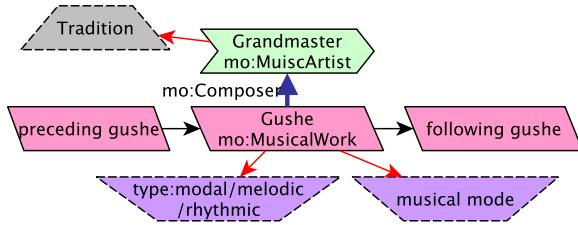
The process of creative performance, called bedàhe navàzi (improvisation), which is at the heart of Persian music, is different from improvisation in Western music, as it involves both composition and new ways of rendering classical pieces (gushés); thus, there is no distinction between the role of the performer and the composer [27]. A performance is usually centred on a set of important gushés, whose order is conventionally accepted (Fig. 6). The texture of Persian ensemble music is heterophonic, meaning that the members of the ensemble play the melodic scheme simultaneously in different ways, characterised by a high degree of improvisation and ornamentation.

4.1 Persian modes and repertoire
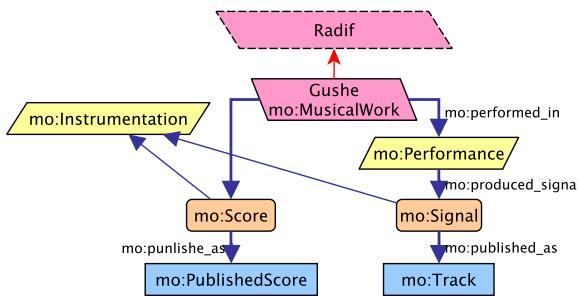
Persian music is based on a modal system of seven main modes and their five derivatives that are collectively called the twelve dastgàhs [8, 16]. In a maqàm performance, different pieces are played in a single mode, while the performance in a dastgàh comprises a certain sequence of modulations from an opening section in the main mode of a dastgàh (daràmad), to derivative modes (àvàz) and finally a return to the starting mode. (Fig. 6).

A student of Persian music studies a Radif - a body of classical repertoire created by a Grandmaster (Fig. 7) - to form the basis of their performance and composition. In the past the transmission took place orally in a teacher-student relationship that would last for many years; nowadays musicians refer to scores and recordings of classical pieces performed by outstanding masters (Fig. 8).

In Fig. 6 a traditional Iranian performance is represented on a timeline (black line) using the Timeline Ontology (black arrows). The modes are specific to Iranian music and are not part of the Music Ontology, which only provides a concept of a Western-centric major/minor *Key*, though these could potentially be added and the concept generalised to represent modes. Alongside mode the gushe type also has to be documented (Fig. 7, for which there is

Figure 7. Radif - the core repertoire of Iranian classical music, a body of compositions by a *Grandmaster* from a particular regional tradition.

Figure 8. Documenting Radif

no correspondence in the Music Ontology. A Grandmaster is part of a Tradition, which provides the musical and cultural context (grey trapezoid) to the performance - see discussion to Fig. 1 of the Russian case study in the Supplementary Material. In Fig. 8 all relationships can be represented including the creation of a Score, with one exception: the belonging of a gushe to a particular radif. This is similar to the within-repertoire relationships discussed in Section 3 in relation to Dutch folk songs.

5. CASE STUDY: PERSONAL WORLD MUSIC COLLECTION

Personal world music [44] collections are ubiquitous and the exact specifics of their usage will differ between users. They are put together for enjoyment, to create playlists, share music with others, to have an overview of a variety of genres and traditions. The difference to other case studies is that the owner of the collection is not an expert in the majority of the styles represented in the collection. Moreover, a world music collection would be heterogeneous and include a large variety of cultures, genres, languages, instruments, contexts, etc. The authority of the sources and the authenticity/expertise of the performers are not always clearly documented or known; all kinds of cultural and stylistic mixtures can occur: for instance, a piece can originate from one culture but be performed in a different style; the musicians might have their roots in more than one culture, including diasporas; music can be performed using instruments not present in the culture of its origin; a mixture of styles can be deliberate or accidental.

Because the consumer is not an expert, the artwork, liner notes and other textual information play an impor-

tant role (Fig. 9). Tracks are commonly compiled into playlists which can be devoted to a particular theme (love songs), reflect or create a certain mood (chill out) or serve a function (music for exercise) (Fig. 10).

Fig. 9 shows that, apart from the cultural context, the Music Ontology is perfectly suitable to represent discographic information about world music collections. Yet we observe in Fig. 10 that factors determining the content of a playlist - aspects of cultural context or musical characteristics - are beyond the domain of the Music Ontology.

6. DISCUSSION
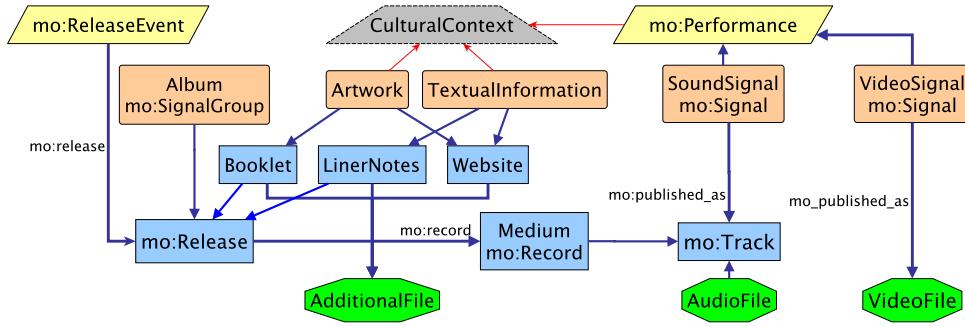
The Music Ontology captures FRBR group 1 concepts in all case studies, modelling the process of performance documentation from *Expressions* over *Manifestations* to *Items* (Figs. 4, 5, 8, 9). It is also well suited, in combination with the Timeline and the Event Ontologies, to document musical events (Figs. 6, 10). We identified three areas where the Music Ontology lacks descriptions: cultural contexts, musical characteristics and relationships within or between repertoires.

Our case studies demonstrate how varied cultural contexts (grey downward trapezoid in the diagrams) can be: function, social group and performance practice (Fig. 1 in Supplement [11]) in Russian traditional music; regional tradition in Persian Music (Fig. 7); culture, function, mood and theme in world music (Fig. 10). This is a very complex area, which is often described and discussed differently depending on the language, organisation or school of thought. It is not practical to construct a single taxonomy to describe pagan rituals and music for exercise; wedding songs alongside remembering sunrise; an Easter Vesper as well as indecent humorous couplets. Therefore, Broad categories could be offered like *SocialFunction*, *PerformancePractice*, *Mood*, whereas more detailed modelling should be culture- and use case specific.

Musical characteristics (magenta upwards trapezoid in the diagrams) are specific for each culture: traditions and repertoires can differ greatly in the complexity and variation in modality, rhythm, harmony, ornamentation. Case studies vary in which musical characteristics are important: mode and rhythm type are crucial in Persian music (Figs.6, 7) but are less important in other case studies. Therefore, it seems most viable to model musical characteristics separately for each musical tradition, choosing a subset of the model relevant for the use case. MusicOWL [19] and the Music Theory Ontology [34] offer a model for Western music. Modelling for other traditions should be conducted in collaboration with ethnomusicologists and tradition experts. It is important to keep in mind the gap between theory and practice [16]. Related musical cultures, such as maqamic traditions or Eastern European polyphonic vocal styles, could possibly benefit from a systemic view and a more generic modelling, which would facilitate crosscultural interoperability of the models.

Relationships within repertoires are crucial in some cultures and contexts: the order and modulations of gushes in

---

[11] Supplementary material: https://osf.io/5qxdb/

Figure 9. A world music collection: *TextualInformation* captures the cultural affiliations and the social context of the song/work performed as well as of the performance itself

Persian music (Figs. 6, 7) or the relationship between tunes in a folk music tune family (Fig. 3); they are less important in other cultures, like Russian village music. Cross-repertoire relationships, such as between tune families or Radifs, are of interest for comparative research. Such relationships will differ between cultures, yet there might be scope for generalisation within larger musical regions.

We also noticed that modelling musical instruments (Fig. 1 of supplement, Fig. 10) will represent a challenge, due to their variety and linguistic barriers. While a general instrument classification has long been established [37] and the Music Ontology refers to an instrument taxonomy [20], constructing a cross-cultural taxonomy of musical instruments is a long-term task. Alongside instruments, an addition of approximate dates (e.g. Figs. 1, 2 of supplement) as it was done in the Jazz Ontology [31] would be beneficial, since references to periods of the past and absence of exact dates are a common phenomenon in many traditional musics.
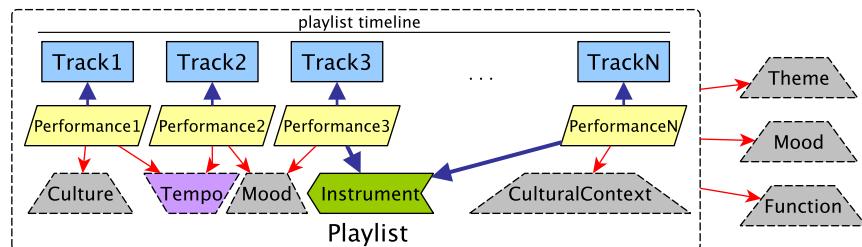
Caution must be exercised when using the FRBR/MO concept of *MusicalWork*. As Riley [35] noted, it is less well suited to describe traditional and folk music. It is often difficult to delineate works: are highly similar tunes one work or two? If a song has changed through oral transmission, or has been transformed through improvisation, is it still the same work? It is related to the problem of labelling works, when titles, lyrics and incipits vary between localities or through improvisation, such as instrumental tunes in our Russian example [26].

We conclude that the Music Ontology is a very useful

standard to implement for the domains of musical cultures other than Western classical and popular music. However, its further generalisation seems to offer few advantages, since cultural contexts, musical characteristics, intra- and inter-repertoire relationships are mostly culture specific: small domain specific extensions would be more useful than trying to build one big generic ontology.

In future work we suggest to investigate CIDOC Concept Reference Model as a way to provide generalised categories for cultural context, to interface with the Music Ontology. One option would be to adjust the Music Ontology to implement FRBRoo, the object-oriented model harmonising FRBR and CIDOC-CRM. This might allow to blend the advantages of FRBR for modelling music creation and consumption with the modelling of cultural contexts to some extent, though the simplicity and transparency of the Music Ontology's current version would suffer. Similarly, the usefulness of Hornbostel-Sachs categories to generalise musical instruments should be explored critically through case studies.

This generalisation approach could be taken further in relation to cross-cultural comparative research. We suggest to concentrate on two or three loosely related repertoires from a broad cultural area, for instance a Persian Radif, a Turkish Makam and an Indian Raga. Modelling similar use cases for such repertoires would allow to evaluate generalisation opportunities and advantages (or the lack thereof) in cultural context, musical characteristics, relationships within and between repertoires and musical instruments.

Figure 10. World music playlists are often compiled for variety, each track from a new culture, with different instrumentation and texture. Tempo and mood may be kept constant or raised gradually, depending on the aim of the playlist.

7. REFERENCES

[1] A. Allik, G. Fazekas, and M.B. Sandler. An ontology for audio features. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) 7-11 August, New York, USA*, pages 73–79, 2016.

[2] S Bayard. Prolegomena to a study of the principal melodic families of british-american folk song. *Journal of American Folklore*, 63(247):1–44, 1950.

[3] A. Carlyle. Understanding frbr as a conceptual model. *Library Resources Technical Services*, 50(4):264-73, 2006.

[4] M. Ceriani and G. Fazekas. Audio commons ontology: a data model for an audio content ecosystem. In *Proc. of the 17th International Semantic Web Conference (ISWC'18)*, pages 20–35, Monterey, CA, USA, 8-12 Oct 2018. Springer.

[5] LP Coladangelo. *Ontology and Domain Knowledge Base Construction for Contra Dance as an Intangible Cultural Heritage: A Case Study in Knowledge Organization of American Folk Dance*. PhD thesis, Kent State University, 2020.

[6] Darrell Conklin and Christina Anagnostopoulou. Comparative pattern analysis of cretan folk songs. *Journal of New Music Research*, 40(2):119–125, 2011.

[7] J. R. Cowdery. A fresh look at the concept of tune family. *Ethnomusicology*, 28(3):495–504, 1984.

[8] H. Farhat. *The Dastgàh Concept in Persian Music*. Cambridge University Press, Cambridge, 1990.

[9] G. Fazekas, Y. Raimond, K. Jakobson, and M. Sandler. An overview of Semantic Web activities in the OMRAS2 Project. *Journal of New Music Research special issue on Music Informatics and the OMRAS2 Project*, 39(4):295–311, 2011.

[10] G. Fazekas and M. Sandler. The Studio Ontology Framework. In *Proceedings of the International Society for Music Information Retrieval conference*, pages 24–28, 2011.

[11] G. Fazekas and M. Sandler. Describing audio production workflows on the Semantic Web. In *Proc. of the 14th IEEE International Workshop on Image and Audio Analysis for Multimedia Interactive Services (WIAMIS) 3–5 July, Paris, France*, 2013.

[12] György Fazekas and Mark B Sandler. Knowledge representation issues in audio-related metadata model design. In *Audio Engineering Society Convention 133*. Audio Engineering Society, 2012.

[13] Ali C. Gedik and Barış Bozkurt. Pitch-frequency histogram-based music information retrieval for turkish music. *Signal Processing*, 90(4):1049 – 1063, 2010. Special Section: Ethnic Music Audio Documents: From the Preservation to the Fruition.

[14] Izaro Goienetxea Urkizu, Iñaki Arrieta Urtizberea, Jon Bagüés, Arantza Cuesta, Pello Leiñena, and Darrell Conklin. Ontologies for representation of folk song metadata. Technical report, University Of The Basque Country, Department of Computer Science and Artificial Intelligence, 2012.

[15] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995.

[16] P. Heydarian. *Automatic Recognition of Persian musical modes in audio musical signals*. PhD thesis, London Metropolitan University, 2016.

[17] IFLA. *Functional Requirements for Bibliographic Records*. IFLA FRBR Study Group, Munich, 1998.

[18] Kurt Jacobson, Simon Dixon, and Mark Sandler. Linkedbrainz: Providing the musicbrainz next generation schema as linked data. In *Late-Breaking Demo Session at the 11th International Society for Music Information Retrieval Conference*, 2010.

[19] Jim Jones, Diego de Siqueira Braga, Kleber Tertuliano, and Tomi Kauppinen. Musicowl: The music score ontology. In *Proceedings of the International Conference on Web Intelligence*, pages 1222–1229, 2017.

[20] Sefki Kolozali, György Fazekas, Mathieu Barthet, and Mark B Sandler. Knowledge representation issues in musical instrument ontology design. In *Proc. of the 12th International Society for Music Information Retrieval (ISMIR'11) conference*, pages 465–470, Miami, Florida, USA, 24-28 Oct 2011.

[21] P. Van Kranenburg, M. De Bruin, and A. Volk. Documenting a song culture: the dutch song database as a resource for musicological research. *International Journal on Digital Libraries*, 20(1):13–23, 2019.

[22] Thomas Lidy, Carlos N. Silla Jr., Olmo Cornelis, Fabien Gouyon, Andreas Rauber, Celso A.A. Kaestner, and Alessandro L. Koerich. On the suitability of state-of-the-art music information retrieval methods for analyzing, categorizing and accessing non-western and ethnic music collections. *Signal Processing*, 90(4):1032 – 1048, 2010. Special Section: Ethnic Music Audio Documents: From the Preservation to the Fruition.

[23] Pasquale Lisena, Konstantin Todorov, Cecile Cecconi, Francoise Leresche, and Isabelle Canno et al. Controlled vocabularies for music metadata. *19th International Society for Music Information Retrieval Conference Proc. (ISMIR2018)*, 2018.

[24] Michela Magas and Polina Proutskova. A location-tracking interface for ethnomusicological collections. *Journal of New Music Research*, 42(2), 2013.

[25] Joaquín Mora, Francisco Gómez, Emilia Gómez, Francisco-Borrego Escobar, and José Miguel-Báñez Díaz. Characterization and melodic similarity of a cappella flamenco cantes. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[26] Ullrich Morgenstern. *Die Musik der Skobari. Studien zu lokalen Traditionen instrumentaler Volksmusik im Gebiet Pskov (Nordwestrußland)*. Cuvillier, Göttingen, 2007.

[27] L. Nooshin. Improvisation as 'other': creativity, knowledge and power – the case of iranian classical music. *Journal of the Royal Musical Association*, 128, 2003.

[28] Alastair Porter, Mohamed Sordo, and Xavier Serra. Dunya: A system for browsing audio music collections exploiting cultural context. In *Proc. 14th International Society for Music Information Retrieval Conference (ISMIR) 4-8 Nov. 4-8, Curitiba, Brazil*, 2013.

[29] Polina Proutskova. Musical memory of the world - data infrastructure in ethnomusicological archives. *Proceedings of the International Symposium on Music Information Retrieval*, 2007.

[30] Polina Proutskova and Michael Casey. You call that singing? ensemble classification for multi-cultural collections of music recordings. *Proceedings of the International Symposium on Music Information Retrieval*, 2009.

[31] Polina Proutskova, Daniel Wolff, György Fazekas, Kalus Frieler, Frank Höger, Olga Velichkina, Gabriel Solis, Tillman Weyde, Martin Pfleiderer, Hélène Camille Crayencour, and Simon Dixon. The jazz ontology: A semantic model and large-scale rdf repositories for jazz. *Journal of Web Semantics*, 2020 submitted.

[32] Y. Raimond, F. Giasson, K. Jacobson, G. Fazekas, T. Gangler, and S. Reinhardt. The music ontology specification. In *Online Specification Document: http://musicontology.com/*, 2010.

[33] Yves Raimond, Samer A Abdallah, Mark B Sandler, and Frederick Giasson. The music ontology. In *Proc. 8th International Conference on Music Information Retrieval, ISMIR'07, 23-27 Sept., Vienna, Austria*, 2007.

[34] Sabbir M Rashid, David De Roure, and Deborah L McGuinness. A music theory ontology. In *Proceedings of the 1st International Workshop on Semantic Applications for Audio and Music*, pages 6–14, 2018.

[35] Jenn Riley. Application of the functional requirements for bibliographic records (frbr) to music. *Proceedings of the International Symposium on Music Information Retrieval*, 2008.

[36] Pat Riva, Martin Doerr, and Maja Zumer. Frbroo: enabling a common view of information from memory institutions. In *World Library and Information Congress: 74th IFLA General Confrence and Council*, 2008.

[37] Curt Sachs and Erich Moritz von Hornbostel. Systematik der musikinstrumente. *Berliner Gesellschaft für Anthropologie, Ethnologie und Urgeschichte*, 46(4-5), 1914.

[38] Mark Sandler, David De Roure, Steven Benford, and Kevin Page. Semantic web technology for new experiences throughout the music production-consumption chain. In *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 49–55. IEEE, 2019.

[39] Xavier Serra. A multicultural approach in music information research. In Klapuri A. and Leider C., editors, *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference; 2011 October 24-28; Miami, Florida (USA). Miami: University of Miami; 2011.* International Society for Music Information Retrieval (ISMIR), 2011.

[40] Joren Six and Olmo Cornelis. Tarsos - a platform to explore pitch scales in non-western and western music. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011.* International Society for Music Information Retrieval, 2011.

[41] Gregor Strle and Matija Marolt. The ethnomuse digital library: conceptual representation and annotation of ethnomusicological materials. *International Journal on Digital Libraries*, 12(2-3):105–119, 2012.

[42] Yiwei Sun and Shabnam Ghaffarzadegan. An ontology-aware framework for audio event classification. *IEEE International Conference on Acoustics, Speech, and Signal Processing, 4-8 May, Barcelona, Spain*, 2020.

[43] Christopher Sutton, Yves Raimond, and Matthias Mauch. The Chord Ontology Specification. In *Online Specification Document: http://purl.org/ontology/chord/*, 2007.

[44] Timothy Dean Taylor and John D Taylor. *Global pop: World music, world markets*. Psychology Press, 1997.

[45] Mi Tian, György Fazekas, Dawn Black, and Mark Sandler. Towards the representation of chinese traditional music: A state of the art review of music metadata standards. In *International Conference on Dublin Core and Metadata Applications*, pages 71–81, 2013.

[46] Dan Tidhar, György Fazekas, Matthias Mauch, and Simon Dixon. TempEst: Harpsichord temperament estimation in a Semantic Web environment. *Journal of New Music Research*, 39(4), 2010.

[47] B. Tillett. Frbr: A conceptual model for the bibliographic universe. Technical report, Library of Congress Cataloging Distribution Service, 2004.

[48] L. Turchet, F. Antoniazzi, F. Viola, F. Giunchiglia, and G. Fazekas. The internet of musical things ontology. *Journal of Web Semantics*, Vol. 60, 2020.

[49] L. Turchet, J. Pauwels, C. Fischione, and G. Fazekas. Cloud-smart musical instrument interactions: Querying a large music collection with a smart guitar. *ACM Transactions on the Internet of Things*, 1(3), 2020.

[50] George Tzanetakis, Ajay Kapur, W. Andrew Schloss, and Matthew Wright. Computational ethnomusicology. *journal of interdisciplinary music studies*, 1(2):1–24, 2007.

[51] Fabio Viola, Ariane Stolfi, Alessia Milo, Miguel Ceriani, Mathieu Barthet, and György Fazekas. Playsound.space: enhancing a live performance tool with semantic recommendations. In *Proceedings of the 1st International Workshop on Semantic Applications for Audio and Music (SAAM'18) held in conjunction with the Semantic Web conference, 9 October, Monterey, CA, USA*. ACM, 2018.

[52] Anja Volk and Peter van Kranenburg. Melodic similarity among folk songs: An annotation study on similarity-based categorization in music. *Musicae Scientiae*, 16(3):317–339, 2012.

[53] T. Wilmering, G. Fazekas, and M. Sandler. The audio effects ontology. In *Proc. of the 14th International Society for Music Information Retrieval Conference, ISMIR'13, November 4-8, Curitiba, Brazil*, 2013.

[54] T. Wilmering, G. Fazekas, and M. Sandler. Aufx-o: Novel methods for the representation of audio processing workflows. In *15th International Semantic Web Conference (ISWC)*, volume 9982 of *Lecture Notes in Computer Science,*, pages 229–237. Springer, Cham, 2016.

[55] O Wright. *Touraj Kiaras and Persian music: An analytical perspective*. SOAS Musicology Series. Ashgate, Farnham, 2009.

Author Index

Author Index