Sponsoring Committee: Professor Juan Pablo Bello, Chairperson
Professor Robert Rowe
Doctor Eric J. Humphrey

AUTOMATIC MUSIC TRANSCRIPTION IN THE DEEP LEARNING ERA:

PERSPECTIVES ON GENERATIVE NEURAL NETWORKS

Jong Wook Kim

Program in Music Technology
Department of Music and Performing Arts Professions

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in the
Steinhardt School of Culture, Education, and Human Development
New York University
2019

ABSTRACT

The problem of automatic music transcription (AMT) is considered by many researchers as the holy grail of the field, because of the notorious complexity and difficulty of the problem. Meanwhile, the current decade has seen an unprecedented surge of deep learning where neural network methods have achieved tremendous success in many machine learning tasks including AMT. The success of deep learning is largely enabled by the ever-increasing amount of available data and the innovation of GPU hardware, allowing a deep learning model to enjoy the increased capacity to process such scale of data. While having more data and higher capacity translates better performance in general, there still remains the question of how to design an AMT model that can effectively incorporate the inductive bias for the task and best utilize the increased capacity.

This thesis hypothesizes that an effective way to address this question is through the use of generative neural networks. Starting with a simplified setup of monophonic transcription, we learn the effectiveness of convolutional representation and the roles of dataset choices in data-driven models for music analysis. In the subsequent chapters, we examine the applications of deep generative models in music analysis and synthesis tasks, by introducing a WaveNet-based music synthesis model that learns a multi-dimensional timbre representation and a music language model applied in an adversarial manner to improve a piano transcription model. Finally, we combine the analysis and synthesis methods to develop a multi-instrument polyphonic music transcription system. From these observations, we conclude that deep generative models can be used to improve AMT in many ways, and they will be a crucial component for further advancing AMT.

To Nayoung.

ACKNOWLEDGEMENTS

I am truly grateful to many wonderful people who believed in me along this long journey. This dissertation would not have been able to come into existence without their support and guidance.

First of all, I would like to express my sincere gratitude to my supervisor, Prof. Juan Pablo Bello. I am incredibly lucky to have you as my doctoral advisor; thank you for being a dependable teacher, an incredible researcher, and a welcoming friend to me. To my committee members and readers — Prof. Robert Rowe, Dr. Eric Humphrey, Prof. Johanna Devaney, and Prof. Brian McFee — I deeply appreciate taking your time to read through my awkward sentences and giving insights to make them better. And to all professors with whom I have had the pleasure of working with: thank you for your classes, chats, and smiles.

I have been fortunate enough to become friends with almost all of MARL's PhD students and postdocs, and I am thankful to every one of them. To the MARL-doctors — Taemin, Areti, Jon, Aron, Braxton, Finn, Eric, Uri, Rachel, and Finn — thanks for showing the ways that I can follow, including but not limited to the coffee shops and bars. To the MARL-doctor-to-be's — Andrea, Andrew, Marta, Peter, Yu, Ho-Hsiang, Willie, Dirk, Tom, Jason, and Chris — it was so much fun sharing office and hanging out with you, and I will miss the occasional beer sessions. To the MARL postdocs — Brian, Justin, Mark, Charlie, Ron, Vincent, Claire, Magdalena, Hitomi — thank you for the insights during the lab meetings and collaborations, and also for sometimes bearing with my unscholarliness.

And to my Korean friends: thanks for being on KakaoTalk whenever I had silly memes to share with, but more importantly for being always welcoming and cheering for me every time I visited Korea, especially that time when you guys gladly spent a whole day at my wedding.

I am honored to have been a recipient of Samsung Scholarship, which is another factor that made all this possible; thank you for your support, networking, and gifts from Leeum. I would also like to thank everyone I worked with during my brief industry experiences — NCSOFT, Kakao, Pandora, and Spotify — for allowing me to learn and achieve what I could not do in schools, and of course for all the free foods and swags.

To my parents who have always believed me and prayed for me, I can't express enough gratitude for your limitless love and unwavering support. Your passion in education made me grow from an aspiring teenager to a respectable scientist. Now that there are no more degrees for you to worry about, please take it easy and enjoy your 60s!

Finally, to my wife Nayoung, you are the foremost reason why I am writing these words now. I cannot believe how lucky I am to have met you and plowed through this journey mixed with joys and tears with you. Thank you for being my best friend, a fellow researcher, a delightful travel partner, a world-class cook, a witty comedian, and a lovely cheerleader who makes me a better person every day.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

“What I cannot create, I do not understand.”

-Richard Phillips Feynman

CHAPTER I

INTRODUCTION

As listening is a core constituent of human perception, an essential component of artificial intelligence is *machine listening*. The purpose of machine listening research is to enable computers to process and understand sounds as humans do. In recent years, there have been an unprecedented amount of successes in the field of *machine learning*, a near-synonym to artificial intelligence with a connotation of statistical and/or probabilistic methodologies, which redefined what a computer vision or natural language processing systems can do and made previously unimaginable applications such as semi-autonomous driving[1] and a superhuman StarCraft-playing AI (Vinyals et al., 2019) into reality.

In this context, this thesis focuses on improving the machine understanding of music in order to automatically transcribe music, which largely remains an unsolved problem despite decades of research. The recent rapid development in *deep learning* research, however, hints at many new possibilities for improving the performance or even achieving human-level accuracy in music transcription.

1 Statement of Problem

Automatic music transcription (AMT) refers to an automated process that can identify musical events in the input audio and convert them into musical notations.

[1] Fully autonomous driving is under active development; e.g. Tesla Autopilot and Waymo.

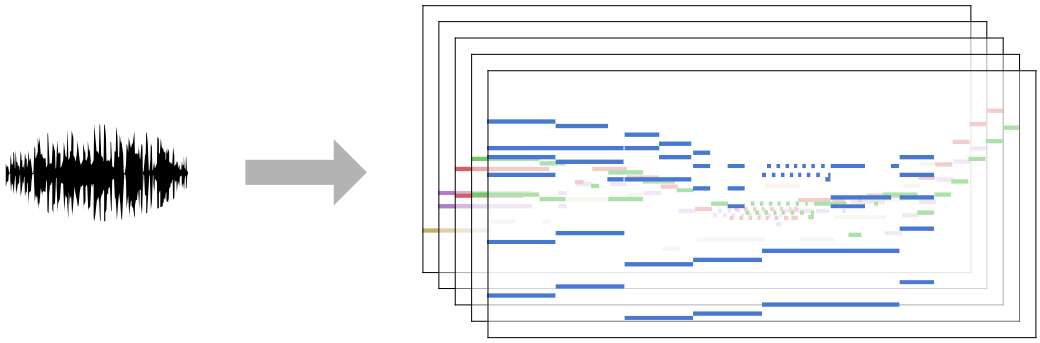


Figure 1: The automatic music transcription setup to be used in this thesis. Using per-instrument piano-roll representations is easier for machines to process, and avoids variability and subjectivity that may arise from symbolic and textual notations.

Historically, the definition of automatic music transcription varied by author, usually in terms of the form of the output representation. In earlier works (Moorer, 1977; Piszczalski & Galler, 1977), the final output of the transcription system was to be the common music notation, i.e. a score, while later literature generalizes the problem by defining it as "the analysis of an acoustic musical signal so as to write down the pitch, onset time, duration, and source of each sound that occurs in it" (Klapuri & Davy, 2006) or "the process of converting an acoustic musical signal into some form of musical notation" (Benetos et al., 2013). This thesis adopts per-instrument piano-rolls as the resulting representation of automatic music transcription, as shown in Figure 1, and defers the "piano-roll to score" conversion as an out-of-scope task, which involves higher-level nontrivial tasks such as tempo and meter tracking, key signature detection, and music structure identification. This can be justified since it allows the transcription model to focus on source separation and multi-pitch tracking, which are already highly challenging problems (Cemgil et al., 2006).

To perform automatic music transcription, various properties of musical events, such as pitch, timbre, harmony, beats, etc., need to be defined and

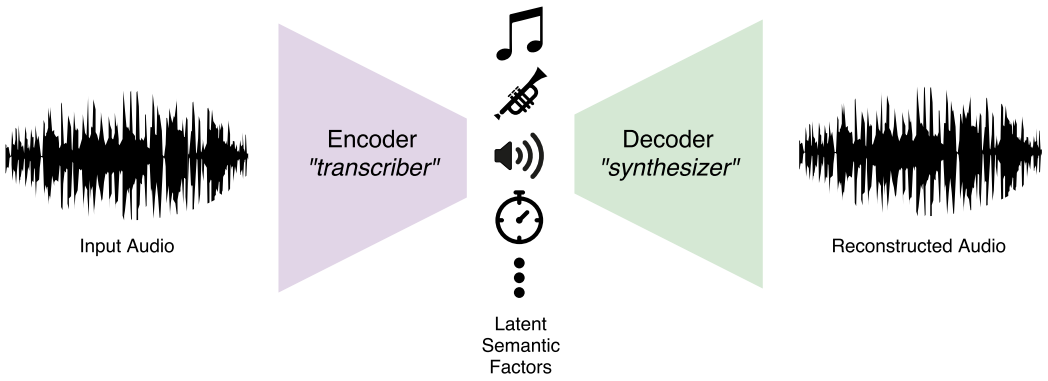


Figure 2: A generative model has to know all of necessary information required to reconstruct the audio data, including pitch, timbre, loudness, and duration. Generative models can be jointly trained with an encoder that finds those semantic information, giving a transcriber-synthesizer pair.

extracted from the audio. In this sense, the setup of AMT is *discriminative* in nature, meaning that it aims to identify different attributes from given audio, as opposed to *generative* models concerning how to construct audio signals according to given conditions about those attributes. Meanwhile, when a generative model is jointly trained with an encoder, it can learn to generate data samples from a small number of latent factors, while the encoder learns to extract those factors from the audio in a compact representation, as depicted in Figure 2. Recently, with the increased capacity of machine learning models and hardware, many *deep generative models* have been proposed and shown to be capable of processing high-dimensional multimedia data. Furthermore, significant research efforts have been made towards learning disentangled representations of data, meaning that the latent factors contain meaningful information that can be easily separated and isolated. To this end, the goal of this thesis is to study representation learning methods powered by deep generative models, to obtain disentangled information from audio signals that can achieve better performance in music transcription.

2 Research Questions

To achieve the goal of improving music transcription with generative models, several research questions needs to be addressed. This thesis considers the following questions regarding the proposed approach towards automatic music transcription.

1. What kinds of deep models and representations can be used for effectively extracting pitch from audio?
2. How does the choice of datasets affect the accuracy and the generalizability of a trained model?
3. How can we encode the concept of timbre in a way that is useful for music synthesis and transcription?
4. How can a transcription model make informed predictions incorporating the knowledge of a music language model?
5. Can a music synthesizer component based on a deep generative model be used to improve music transcription?

We aim to address each of these questions in the technical chapters of the thesis. Through extensive experimental analysis in each chapter, we draw the conclusion on the effectiveness of deep generative models in the context of automatic music transcription.

3 Limitations

Because of the sophisticated and open-ended nature of automatic music transcription, it is necessary to define the scope of the tasks and data that this thesis will be concerned with. The purpose of this section is to define those limitations in terms of the scope of music that the proposed AMT system can process, the required capability of symbolic music processing, and the need for perceptual studies regarding the validity of AMT systems.

3.1 Scope of Music

Music signals typically contain both harmonic and percussive sources. From a signal processing point of view, harmonic sounds are quasi-periodic and contain energy only at certain frequencies, roughly at the multiples of the fundamental frequency, whereas percussive or highly inharmonic sounds have aperiodic frequency spectra in which it is not possible to define a fundamental frequency. Consequently, transcription models for harmonic sounds and percussive sounds require different techniques according to their nature.

This thesis will limit the focus on the transcription of harmonic sounds and therefore use the per-instrument piano roll notation (Figure 1) as the output representation. This is a realistic trade-off to make, because of a number of reasons. First, learning to simultaneously model harmonic and percussive sounds is a harder problem both conceptually and computationally. Secondly, it is possible to plug a harmonic-only model into a pipeline consisting of harmonic-percussive source separation (HPSS) and a percussion transcription model as an alternative to a comprehensive approach. Lastly, polyphonic transcription is considered to be the most difficult problem in the domain of

automatic transcription, and it is sensible to tackle this as a standalone problem in the simplest possible setup. Excluding percussive sounds will disallow using most of pop music tracks as-is, but multi-track datasets can still be utilized since they contain each track separately.

Additional limitations should be considered on the types of the instruments and their sound variations. Depending on the instrument, the same score could be performed using a variety of expressive techniques, such as vibrato, tremolo, pizzicato, and the usage of mutes or harmonics, among others. In order to accurately produce a piano roll transcription that is invariant to such techniques, the model has to be trained to classify them as nonessential information, requiring the availability of datasets with the annotations for those techniques. While an ideal model should learn those concepts as humans do, too much timbral or temporal variation for an instrument will prevent the model from learning a consistent representation corresponding to the instrument. Therefore, for the immediate purpose of this thesis, distributions of music signals that do not contain too much of said variations will be employed.

3.2 Symbolic Processing of Notes

As mentioned and justified in the problem statement (Section 1), by choosing per-instrument piano rolls as the output of transcription, concerns of symbolic music processing such as beat quantization and score typesetting are excluded from the scope of this study. The difference between a piano roll output and a full human-readable score output becomes apparent when we compare the piano rolls in Figure 1 with Figure 3, which is the original score from which the piano rolls are plotted. There are many aspects in producing the score output that are

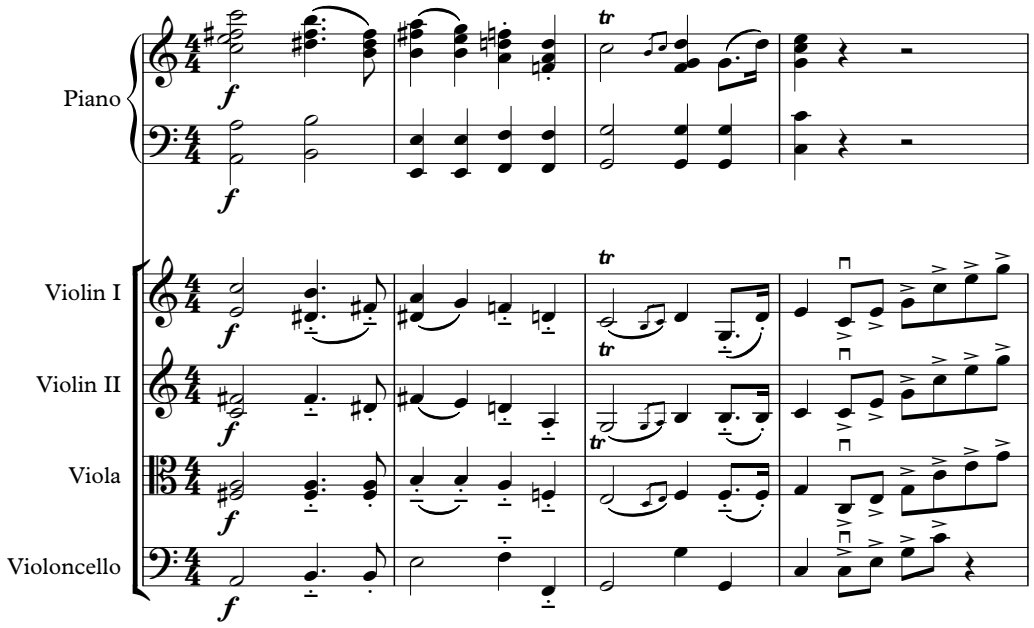


Figure 3: The full score notation of the music used to build the piano rolls in Figure 1. To fully recover this level of notations from the audio, the transcriber has to make many additional decisions than for the piano rolls, such as determining the key signature, time signature, clefs, dynamics, trills, bowing instructions, etc.

highly subjective and difficult to derive a consistent evaluation metric from, such as the interpretation of legatos or staccatos and the aesthetic choices for typesetting, providing an additional justification for using the piano roll notation.

The MIDI file format is suitable for encoding data equivalent to per-instrument piano rolls, consisting of multiple tracks of `note_on` and `note_off` events with the corresponding timestamps. MIDI will therefore be a supported output format of the proposed AMT system in addition to the piano roll representations, which can also be conveniently played back by media player software. Music typesetting software such as Sibelius or Finale can render a MIDI file into score notation using the metadata in the file as well as some heuristics for beat quantization. However, the readability of the score rendered from a

transcribed MIDI file is limited, due to imperfect transcription and the absence of metadata such as the time and key signature.

3.3 On the Need for Perceptual Studies

This study of automatic music transcription is entirely quantitative and does not involve subjective tests on human participants. We only aim to discover the systematic relations between audio signals and corresponding note sequences present in the datasets we use for training. However, music is essentially a perceived notion, and thus are the core qualities of sound — pitch, timbre, and loudness — which are the output of automatic transcription. Limited human performance and subjectivity make transcription of polyphonic music an error-prone process, where any two annotators may produce drastically different annotations. Although this problem of inaccuracy and subjective difference is often overcome by using a ground-truth dataset synthesized from known frequency information, the gap still persists between what a model can learn from synthesized audio and how it will respond to real-world sounds. This thesis uses both kinds of datasets, from synthesized and recorded audio signals, and thus the real-world applicability of each of the proposed models should be carefully examined.

The goal of automatic transcription is at a lower level than for tasks such as chord recognition and melody tracking, which may incur even more subjective disagreements caused by the imprecise definitions of chords and melodies. Being directly related to the physical concept of fundamental frequency, pitch is relatively precisely defined in this sense, and formulating automatic music transcription as an audio-to-piano-roll conversion mostly eliminates the ambiguity

that exists in chord recognition and melody tracking. There exist some cases where the mathematical definition of fundamental frequency still cannot be applied for all pitched sounds, such as the Shepard tone (Shepard, 1964) where the pitch of a harmonic sound fails to be consistently mapped to a fundamental frequency. However, disregarding these few edge cases, this study assumes that the piano roll notation can convey an objective transcription for many practical purposes, allowing us to formulate AMT as a mathematical problem which does not require experiments on human subjects.

4 Need for Study

The nature of music transcription is multifold; to create a complete transcription, one has to identify all instruments, onsets, dynamics, and the pitch traces for every instrument present in the music, and it would still be far from achieving the human-level accuracy. The need for this study arises naturally, not only because this is an intriguing problem in the intersection of music and technology that has remained unsolved for decades, but also because the solution to this problem can provide practical benefits to many applications.

In order to bolster the need for this study, a few of such applications are introduced in this section, followed by discussions on the advantages of employing generative models as a means of better capturing musical semantics. A brief perspective on AMT is presented in the context of wider AI research, followed by the organization of the chapters.

4.1 Applications of Automatic Music Transcription

Many applications of the techniques in the realm of automatic music transcription is on interactive music systems. Vercoe (1984) proposed a quest for a *synthetic performer*, which can listen, perform, and learn in the context of live performance. Relevant sub-fields include automatic *real-time accompaniment* (Dannenberg, 1985) based on dynamic programming was one of the first successful demonstrations of AMT techniques, and *score following*, a general term referring to the synchronization of a computer with a performer playing a known score (Orio et al., 2003). An offline music-to-score matching algorithm can also be applied to intelligent audio editors (Dannenberg & Hu, 2003).

Music recommender systems can combine many kinds of information for improved music retrieval and personalization (Celma, 2010). Content-based music recommender systems can utilize not only the metadata but also the audio content, and methods using timbral (Magno & Sable, 2008), temporal (Q. Li et al., 2007), and tonal features (Lu & Tseng, 2009) have been introduced. These music recommender systems can be further improved when the complete annotations are made available through AMT.

AMT system can help create databases for query-by-humming (Ghias et al., 1995) by automatically estimating melody annotations, where users can retrieve music by humming an excerpt of the song. Such databases can also facilitate large-scale musicological analyses (Abdallah et al., 2015), as well as the development of computer-aided music composition (A. Agostini & Ghisi, 2013) that incorporates musicological knowledge.

4.2 Generative Modeling for Fully Capturing Semantics

Being "generative" means that a model is capable of generating new samples in the domain of the original data. Generation in the symbolic domain creates new musical scores, and a generative model in the audio domain creates audio waveforms. These two kinds of generative systems are familiar to computer music artists and are referred to as algorithmic composition (Fernández & Vico, 2013) and sound synthesis (Cook, 2002) models. This thesis defines the term "generative model" more specifically, as a model that can learn the distribution of provided data and can sample new samples in the original distribution. This differs from the term "generative" used in computer music in a sense that it aims to accurately model the probability distribution and learn to regenerate the real-world audio to be used in music transcription, rather than focusing on the artistic aspects of generating new kinds of sounds and music.

By learning to generate data using fewer parameters than the scale of the dataset, a model has to discover the underlying natural features from the distribution of data. In music transcription, these features correspond to the musical concepts such as pitch, timbre, and rhythm. Philosophically, this idea follows what Richard Feynman once wrote on his blackboard, *"What I cannot create, I do not understand"* and *"Know how to solve every problem that has been solved"*. He meant that the marker for truly understanding something is the ability to construct it completely from scratch. Studies on generative models have an advantage of not requiring explicitly labeled data, and they hence tend to take self-supervised or unsupervised learning approaches. LeCun (2016) described unsupervised learning as a cake, with supervised learning as icing and reinforcement learning as the cherry on the top, by which he meant that generative

Figure 4: Increasingly realistic qualities of the generated faces using generative adversarial networks as shown in (Brundage et al., 2018); images are taken from (Goodfellow et al., 2014), (Radford et al., 2015), (Liu & Tuzel, 2016), (Karras et al., 2018), and (Karras et al., 2019).

models need to predict at a much larger scale of information and should be able to learn the underlying "common sense".

Inherently, unsupervised learning is less well-defined than supervised learning, and this is the reason why unsupervised learning is sometimes synonymous with clustering, because finding clusters is usually as much an unsupervised learning system can do. However, the recent success of deep learning introduced a new breed of generative models, enabling the end-to-end generation of complex data such as photos and audio signals. *Generative adversarial networks* (GAN) (Goodfellow et al., 2014) are the most notable among them, and their performance in generating realistic images has been improving at an extraordinary pace, as shown in Figure 4. Combined with the various techniques for manipulating the semantic information in GANs as will be introduced in Section III.4, this facilitates completely new kinds of generative methodologies for audio processing.

In this context, this thesis aims to design and develop improved methods for automatic music transcription powered by deep generative models. The idea specifically hypothesizes that by training a generative model, it is possible to learn disentangled representations, from which the information necessary for

transcription can be easily extracted, as depicted in Figure 2. By doing so, the ultimate objective is to build an end-to-end model that connects the piano roll representation to audio signals, in order to perform automatic music transcription — obtaining the most likely piano roll representation for given a audio waveform.

4.3 On the Broader Context of Machine Listening in AI Research

Using generated audio data and generative models is partly motivated by the fact that synthesized music is more prevalent and perceptually more familiar to people than synthesized texts or pictures. Many commercial music tracks are often produced entirely using software instruments, except for the vocal parts. This suggests that synthesized and generated audio may more accurately model the distribution of the real audio data to be transcribed. This generative approach also aligns well with how actual musicians transcribe music, where they match given audio with their knowledge of how the instruments sound when played in a certain combination of rhythms and melodies. Therefore it is reasonable to claim that machines should also be able to perform in a similar way, provided that a proper representation of knowledge about the music and instruments is available.

The task of automatic music transcription shares many common values with other machine learning tasks, such as image segmentation, machine translation, and speech recognition, in the sense that the core task is to build an intelligent system that can extract and process useful information conveyed in complex signals. This is an essence of artificial intelligence (AI) — a system that perceives its environment and takes actions that maximizes the utility (Russell & Norvig, 2009) — where an intelligent system has to understand the semantics of complex data coming from the environment in order to perform well in its tasks. To this

end, the problem of automatic music transcription is not just an intriguing task in music technology but will also be a key component of the AI-enabled future society, constituting a musical component of the artificial general intelligence (AGI), in the form of advanced machine musicianship (Rowe, 2003).

4.4 Organization of The Thesis

This thesis examines the possibility of using deep generative models to learn relevant musical concepts and inform a music transcription model to incorporate them. There exists a rich history of research aiming at the understanding of musical sounds and automatic music transcription; to validate this claim as a feasible research direction and place this thesis in the context of this continuum of research, Chapter II provides a review of the standard methods and the current state of the art in automatic music transcription research. Deep learning techniques are employed as a building block throughout this thesis, and a general introduction to deep learning and deep generative models is provided in Chapter III, with a focus on generative adversarial networks.

In Chapter IV, we first consider a subproblem of music transcription, i.e. monophonic pitch estimation, and learn that convolutional neural networks predicting two-dimensional time-frequency representations can constitute an effective strategy, which we build and extend in the subsequent chapters. Chapter V examines the applications of deep generative models in music analysis and synthesis tasks, by introducing a WaveNet-based music synthesis model that learns a multi-dimensional timbre representation. In Chapter VI, generative adversarial networks are used to apply a music language model that can help improve a piano transcription model. In Chapter VII, we combine the analysis and synthesis

methods developed in the preceding chapters and present a multi-instrument polyphonic music transcription system. Finally, concluding remarks are provided in Chapter VIII.

CHAPTER II

MUSIC INFORMATION RETRIEVAL FOR TRANSCRIPTION

Being able to accurately identify all musical events from audio and transcribe them into musical notations is an essential skill for musicians as well as a paramount goal of music machine learning research. Enabling an automatic conversion between musical audio and symbolic notations, automatic music transcription opens up many new possibilities.

Due to the complexity and difficulty of creating a completely end-to-end music transcription system, many existing approaches focus on a specific subtask of the problem (Casey et al., 2008), e.g. extracting onsets and beats, recognizing timbre and instruments, tracking monophonic and polyphonic pitches, or separating audio sources from a mixture. Each of these subtasks poses interesting goals and applications even without the lofty goal of end-to-end music transcription, and they are classified as subproblems of *music information retrieval* (MIR). Although this term has existed since 1960s (Kassler, 1966), it was only after the late 1990s when active research on this area has spun off from computer music and computational musicology literature. During the last two decades, numerous sophisticated and novel approaches for each of these subproblems have been introduced, that have continuously improved the performance in terms of the accuracy in predicting the correct annotations. This chapter starts by introducing the common concepts and techniques employed in many AMT models, followed by reviews of the state-of-the-art techniques in each area of music transcription.

The purpose of this chapter is to provide a survey over the history of MIR research related to automatic music transcription, as well as to show a clear common pattern over the areas of MIR where the machine learning models have been evolving from simple heuristics based on hand-crafted features to sophisticated deep learning models with millions of parameters. Many methods employing deep neural networks are referenced in this chapter, and the concepts and the formulation of those models such as convolutional newral networks (CNN) and recurrent neural networks (RNN) are described in detail in Chapter III.

1 Introduction

Audio data is huge in volume; a typical audio track contains 44,100 real-numbered samples per second, and sometimes even more. Therefore, computational methods for extracting musical information from audio usually contains a pipeline of feature extraction stages to reduce the dimensionality and increase the interpretability of input data, as shown in Figure 5. The pipeline includes a few techniques widely used in speech processing, as well as many feature extraction stages created for music-specific purposes.

While there are many MIR tasks that operate on the track level, such as music recommendation, tagging, and genre classification, most subtasks of music transcription involve the prediction of labels that are dependent on time, operating either in the sample-level or frame-level. Frames are created by taking a series of overlapping short-time audio segments, where the length of a segment typically ranges from 10 to 50 milliseconds, and optionally multiplying them by a window function. Taking discrete Fourier transforms on the frames produces a *short-time Fourier transform* (STFT), and the squared magnitude of an STFT gives a

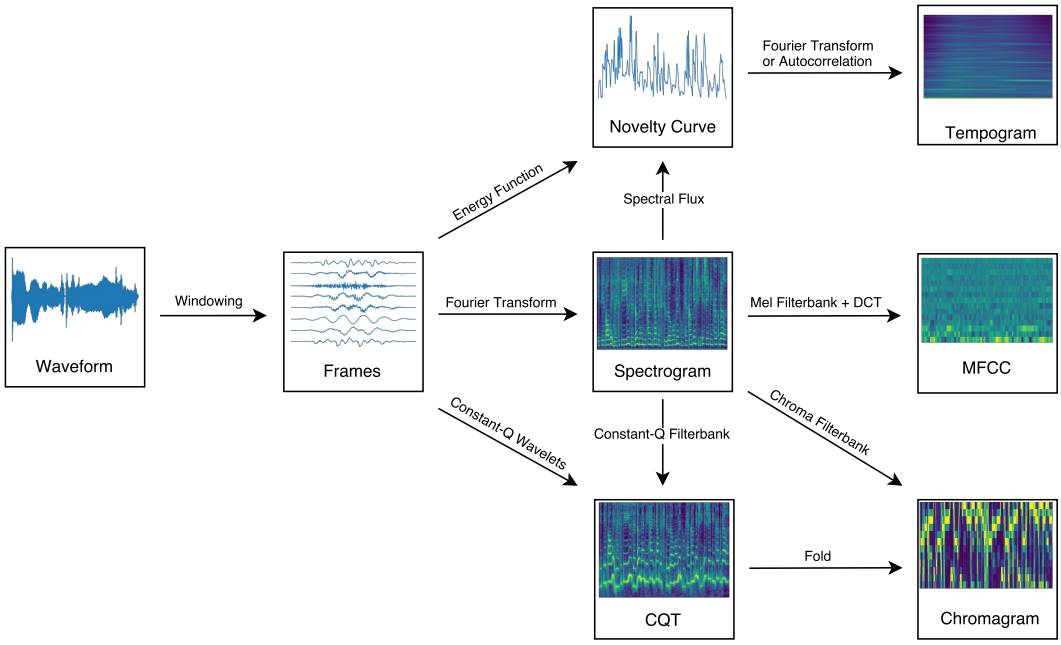


Figure 5: The standard pipeline for music feature extraction. An appropriate set of feature extraction methods needs to be heuristically selected depending on the task.

spectrogram; i.e. for a signal $x[n]$ and a window function $w[n]$:

$$\textbf{STFT}\{x[n]\}(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[m-n]e^{-j\omega n}, \tag{1}$$

$$\textbf{Spectrogram}\{x[n]\}(m,\omega) = \left|\textbf{STFT}\{x[n]\}(m,\omega)\right|^2. \tag{2}$$

Spectrograms give very rich information about the audio; for example, the contour of melodies and the dynamics of music are usually identifiable from the spectrogram image. Spectrograms are expressive enough to be used as an output of sound synthesis or a source separation algorithm, and the corresponding audio signals can be reconstructed without incurring significant perceptual inconsistencies (Griffin & Lim, 1984; Le Roux et al., 2010). However, the dimensionality of a spectrogram is still quite high, making it computationally prohibitive to run many algorithms directly on an STFT or a spectrogram.

This necessitated further transformations by the means of filterbanks, such as *Mel-Frequency Cepstral Coefficients* (MFCC) (Logan, 2000) by applying the Mel filterbank inspired by the human auditory perception and taking the first few DCT components that contain independent factors describing the spectral shape. *Constant-Q transform* (CQT) (Schörkhuber & Klapuri, 2010) uses a filterbank where the center frequencies of filters have a constant Q factor, which is the ratio between the center frequency and the 3 dB bandwidth of a filter. By configuring CQT to produce 12 filters per octave, it is possible to obtain the coefficients corresponding to each musical tone, and to fold the representation to produce a *chromagram* (Harte & Sandler, 2005). Tonnetz (Harte et al., 2006) is a 6-dimensional feature space signifying harmonic relationships, and more sophisticated feature extractors include the summary ACF (Tolonen & Karjalainen, 2000), Specmurt (Saito et al., 2008), and bispectrums (Argenti et al., 2011).

To extract the beat and tempo information, a heuristic function, such as the first-order difference of the time-domain log energy function or the *spectral flux* that measures the total energy increase over the STFT frequency bins, is applied to formulate a novelty curve. This curve can then be used to measure energy bursts that are typically present in the onsets of notes (Bello et al., 2005). The onset information can be further processed to obtain tempo information via *tempogram* (Cemgil et al., 2000) or cyclic tempogram (Grosche et al., 2010).

Meanwhile, many recent approaches have successfully eliminated some or all feature transformation stages in the standard MIR pipeline by training a deep model directly on spectrograms or audio waveforms. Applications of deep learning arose in virtually all types of MIR tasks, including melody extraction (Bittner et al., 2017), beat tracking (Vogl et al., 2017), and genre classification (Oramas et al.,

2017), and outperformed previous feature-based approaches. Apart from a small number of end-to-end approaches, most deep learning models for music still rely on predefined feature transforms such as STFT or CQT, because those features leverage the prior knowledge that music signals are often harmonically sparse and make it easier for a model to learn meaningful concepts without overfitting, using a smaller number of parameters and hence being achievable in a limited hardware capacity. However, these feature extraction stages typically induce a loss of information, and the best-performing model would benefit most from the raw audio data, given enough amount of training data and hardware (Pons et al., 2018).

As is the case for feature extraction, musical prior knowledge is often applied to the algorithmic design of models as well, such as the assumption that sudden changes in music is rare and most changes happen gradually. In the time domain, median filtering (Oudre et al., 2009) is a simple heuristic that can suppress spurious changes, and *Hidden Markov models (HMM)* are widely employed for modeling sequence data such as chord progressions (T. Cho et al., 2010) as well as to smooth sequence outputs as a post-processing step (Khadkevich & Omologo, 2009). Prior knowledge about musical notes can be incorporated more specifically. These approaches include detecting onsets and offsets for note transcription (Benetos & Dixon, 2011), modeling note attacks and decays (Cheng et al., 2016), and more generally the temporal evolution of notes (Cogliati & Duan, 2015). In the frequency domain, the *spectral smoothness principle* states that the spectral envelopes of real sounds tend to be slowly varying as a function of frequency (Klapuri, 2003). The principle has been implemented in a number of ways, such as a moving-average filter for iterative source estimation and separation (Klapuri, 2003), a score function

for F0 candidate (Yeh et al., 2010), and a low-order autoregressive overtone modeling (Emiya et al., 2010).

We have reviewed the common feature extraction stages and identified how the musicological or mathematical prior knowledge on music data guides the algorithmic design of MIR models. In the following sections, more in-depth literature reviews on the different subtasks relevant to automatic music transcription are provided, starting from the simplest problem of monophonic picth tracking.

2 Monophonic Pitch Estimation

Monophonic pitch estimation, or pitch tracking, refers to the task of extracting the fundamental frequency (F0) values from monophonic audio signals. Formally, pitch is defined as a subjective quality of perceived sounds and does not precisely correspond to the physical property of the fundamental frequency (Hartmann, 1997). However, apart from a few rare exceptions, pitch can be quantified using fundamental frequency, and thus they are often used interchangeably in the MIR literature outside psychoacoustical studies. Since differentiating the physical and the perceptual aspects of pitch is a non-goal, the two terms are interchangeably throughout this thesis as well.

Computational methods for monophonic pitch estimation have been studied for more than a half-century (Noll, 1967), and many reliable methods have been proposed since. Earlier methods commonly employ a certain candidate-generating function, accompanied by pre- and post-processing stages to produce the pitch curve. Those functions include the cepstrum (Noll, 1967), the autocorrelation function (ACF) (Dubnowski et al., 1976), the average magnitude

difference function (AMDF) (Ross et al., 1974), the normalized cross-correlation function (NCCF) as proposed by RAPT (Talkin et al., 1995) and PRAAT (Boersma, 1993), and the cumulative mean normalized difference function as proposed by YIN (de Cheveigné & Kawahara, 2002). More recent approaches include SWIPE (Camacho & Harris, 2008), which performs template matching with the spectrum of a sawtooth waveform, and pYIN (Mauch & Dixon, 2014), a probabilistic variant of YIN that uses a Hidden Markov Model (HMM) to decode the most probable sequence of pitch values. According to a few comparative studies, the state of the art is achieved by YIN-based methods (von dem Knesebeck & Zölzer, 2010; Babacan et al., 2013), with pYIN being the best performing method to date.

Since the methods for monophonic pitch tracking are usually built based on the assumption that at most one pitch is present at a time, they cannot be directly applied to polyphonic music where multiple concurrent notes and sound sources are present. Different approaches are therefore needed to accurately estimate multiple concurrent pitches, and such methods are reviewed in the next section.

3 Multiple Fundamental Frequency Estimation

Among the subtasks of automatic music transcription, estimating and tracking all pitches from a polyphonic recording poses the most difficult challenges, as apparent from the recent stream of results from MIREX challenges (Downie et al., 2014). The task is commonly referred to as *multiple fundamental frequency estimation* (Multi-F0 estimation, or MFFE) or *multi-pitch estimation* (MPE). This task is in some sense a superset of other MIR tasks like onset detection, beat tracking, chord recognition, and melody extraction, since the frequency tracking task has to

indicate the presence of every pitch, and tracking chords and melodies becomes much easier when the correct annotations for all pitch values are available.

Early approaches for polyphonic pitch tracking are often based on rather strict assumptions on the types of timbre and the number of polyphony in the audio to be transcribed (Moorer, 1977; Piszczalski & Galler, 1977). *Blackboard systems* (Martin, 1996; Dixon, 2000) are one of the first methods that enabled polyphonic transcription to work under milder assumptions, by integrating both signal processing and musicological knowledge into hierarchical problem abstraction. Limitations of the blackboard approach include the overall complexity and the exhaustive searches required by the system. Accordingly, later approaches often use statistical models or factorization-based methods to more efficiently capture pitch information.

Non-negative matrix factorization (NMF) (Lee & Seung, 1999, 2001) refers to an algorithm that finds two matrices that factorize a given matrix where the elements of all three matrices are non-negative. Starting with Smaragdis and Brown (2003), many successful methods for music transcription have been built based on NMF (Benetos et al., 2019). These methods commonly take the approach of factorizing a time-frequency representation $\mathbf{X} \in \mathbb{R}_{+}^{F \times T}$ as a product of a dictionary matrix $\mathbf{D} \in \mathbb{R}_{+}^{F \times K}$ and an activation matrix $\mathbf{A} \in \mathbb{R}_{+}^{K \times T}$, where K is the number of pitch labels to be transcribed, e.g. 88 keys for piano transcription. This allows for an intuitive interpretation of each matrix, where each column of $\mathbf{D}$ contains a spectral template for a pitch label, and each row of $\mathbf{A}$ contains the activation of the corresponding pitch over time. Various extensions of factorization-based methods have been proposed to leverage sparsity (Abdallah & Plumbley, 2004; Cont, 2006; Costantini et al., 2013), non-negative matrix division

(Niedermayer, 2008), β-divergence (Dessein et al., 2010), adaptive estimation of harmonic spectra (E. Vincent et al., 2010; Fuentes et al., 2013), a Bayesian framework for encouraging harmonicity and temporal smoothness (Bertin et al., 2009, 2010; Peeling et al., 2010), and modeling of attack and decay sounds (Benetos & Dixon, 2013; Ewert & Sandler, 2016). With carefully designed regularizers and the *alternating direction method of multipliers (ADMM)* for training NMF, Ewert and Sandler (2016, 2017) achieved close-to-perfect accuracy for piano transcription in a studio setting, i.e. when the exact spectral profile of the piano notes to be transcribed is known in advance.

A probabilistic approach closely related to NMF is *probabilistic latent semantic analysis (PLSA)* (Hofmann, 1999), a simple probabilistic graphical model that factorizes the joint probability distribution of time and frequency into conditional distributions involving a latent semantic factor. PLSA is is equivalent to NMF when the KL divergence is minimized under L_1 normalization (Gaussier & Goutte, 2005; Ding et al., 2008), while the probabilistic framework enables statistical learning and introducing transformation invariances (Smaragdis et al., 2006). In (Smaragdis, 2009; Benetos & Dixon, 2012), a shift-invariant extension to PLSA has been applied to multi-instrument MFFE by using a convolution over constant-Q spectrograms. Grindlay and Ellis (2010) proposed an extension to PLSA based on the *subspace NMF* algorithm (Grindlay & Ellis, 2009) for multi-instrument MFFE, incorporating additional parameters for instrument sources and possible pitch values.

Many probabilistic models for music transcription employ a Bayesian framework, in which the audio signal is modeled using a probability distribution conditional on unobserved variables such as the note frequencies and timing. The transcription is then performed through Bayesian inference on those parameters,

using either *Markov-chain Monte Carlo (MCMC)* or *variational inference*. Bayesian models for music transcription are usually designed with prior and conditional distributions incorporating musicological and acoustical knowledge on the music and the instruments to be transcribed, such as the de-tuning of partials (Davy & Godsill, 2003). Existing approaches in directions include a state-space model for musical harmonics (Cemgil et al., 2003), a decomposition of audio signals into Gabor atoms (Davy et al., 2006), a hierarchical graphical model (Pesek et al., 2017), and an overtone corpus encoding the harmonic structure (Sakaue et al., 2013). A nonparametric Bayesian model proposed in (Yoshii & Goto, 2012) employs infinite Gaussian mixtures and noninformative hyperprior distributions, automating the model selection process. Bayesian models provide a powerful tool connecting the whole generative process of music, but usually suffer from the high computational cost and the complexity of the algorithm.

Unlike the aforementioned approaches which incorporate as much prior knowledge on the music as possible, discriminative models employ a simpler approach of learning a direct mapping between the audio features and the labels from large training data, making minimal assumptions on the acoustical or musical structure. This data-driven approach is made possible by the availability of large datasets and increased computational capabilities. *Support vector machines (SVM)* were commonly used in earlier discriminative approaches, by constructing one-versus-all SVM classifiers on STFT magnitudes (Poliner & Ellis, 2006), on learned features using *deep belief networks (DBN)* (Nam et al., 2011), or on the result of NMF (Weninger et al., 2013). While being flexible and straightforward to be applied on any features of choice, SVMs have high time and space complexities which limit the size of training dataset and hence the capability of the model.

Deep learning (LeCun et al., 2015) methods for music transcription are increasingly popular (Benetos et al., 2019), as larger labeled datasets and more powerful hardware become accessible. These approaches commonly employ *neural networks (NN)* to produce music transcriptions from the input audio representation, and these are relatively recent phenomena that started in the last decade, with a notable exception of Marolt (1999, 2004). Nam et al. (2011) used deep belief networks (G. Hinton et al., 2006) to extract audio features which are subsequently fed to pitch-wise SVM-HMM pairs to predict the target piano rolls. More recent approaches are based on *convolutional neural networks (CNN)* and/or *recurrent neural networks (RNN)*. Transcription models using CNNs are relatively simpler to train and deploy since they can be easily applied in a frame-wise manner in parallel, and they are shown effective especially for the cases where the notes mostly contain sustained sounds, such as bowed strings, wind instruments, and vocals (Kelz et al., 2016; Bittner et al., 2017). The sequential nature of RNNs is suitable for modeling the temporal variations in music, and many neural transcription models include recurrent connections in their architecture. RNN architectures proposed for AMT include bidirectional RNNs (Böck & Schedl, 2012), recurrent temporal restricted Boltzmann machines (Boulanger-Lewandowski et al., 2012), an acoustic model combined with an RNN-based music language model (Sigtia et al., 2015, 2016; Q. Wang et al., 2018), a sequence-to-sequence model (Ullrich & Van Der Wel, 2017), a dual-objective loss for onsets and frames (Hawthorne et al., 2018), and a *convolutional-recurrent neural networks (CRNN)* (Thomé & Ahlbäck, 2017), which scored the top accuracy in the MFFE subtask of the MIREX 2017 competition.

A few recent studies tried to identify the limitations of simply improving

the conventional metrics and argued the need for more systematic analysis and assessment of the music transcription problem. These include a study of invariances under data augmentation (Thickstun et al., 2018), a study of the entanglement of note representations that may prevent accurate predictions for unseen combinations of notes (Kelz & Widmer, 2017), and a musically inspired evaluation metric that also takes account of voice separation and harmonic analysis (Mcleod & Steedman, 2018).

4 Source Separation and Music Translation

Source separation refers to the task of separating sound sources from a mixture signal, and is closely related to automatic music transcription, because it provides a means to separate each instrument sources from multi-instrument music. This problem is also called as a *cocktail party problem*, based on humans' ability to focus on a single voice at a noisy cocktail party. Sound source separation has been a popular research topic since the seminal work on *auditory scene analysis* by Bregman (1990), from which stemmed computational auditory scene analaysis (CASA) (Brown & Cooke, 1994), a problem of using computational models to analyze an auditory scene, identifying the sources and location of all nearby sounds. In this sense, automatic music transcription and sound source separation are particular aspects of auditory scene analysis (Plumbley et al., 2002), and source separation enables similar kinds of applications to AMT, such as music editing, 3D sound rendering, and information retrieval systems.

Blind source separation refers to the situation where no information about the sources or the mixing process is known (Bell & Sejnowski, 1995), whereas *informed source separation* (E. Vincent et al., 2014) concerns the case where some level of

side information is available, e.g. the presence of a score (Ewert et al., 2014). Blind sound source separation has to resort to using purely statistical approaches such as indepenent component analysis (Saruwatari et al., 2006) or robust PCA (P. S. Huang et al., 2012), whereas informed source separation can leverage the knowledge on the musical structure (Rafii & Pardo, 2013; Liutkus et al., 2012) or the timbral differences of the sources (Y. Li & Wang, 2007; Ono et al., 2010) for better separation. Probabilistic models for source separation (Ozerov et al., 2007; Leglaive et al., 2016) have also been developed, and source-filter modeling (Heittola et al., 2009; Durrieu et al., 2011) is a generative approach which separately models a source that creates a sound and a filter that shapes the timbre. The proposed AMT model is similar to source-filter models in a sense that it describes the generative process of each sound, but also relates to timbre-informed source separation models (Miron, 2018), because it needs to learn the concept of timbre to produce per-instrument piano rolls.

A closely related problem to source separation is audio translation, which concerns mapping input audio to a corresponding output with some desired properties, such as speech with reduced noise, singing voice separated from music, or the same speech content in the voice of a different speaker. Barry and Kim (2018) applied the style transfer algorithm (Gatys et al., 2015) to an ensemble of STFT, CQT, and Mel spectrograms, to transfer musical styles capturing harmonic, rhythmic, and timbral elements. The *U-Net* architecture (Ronneberger et al., 2015) uses an encoder-decoder framework with skip connections between the hidden layers at the same level of abstraction to perform image translation, and a singing voice separation model can be trained using this architecture (Jansson et al., 2017). The encoder-decoder architecture with skip connections can also be trained with

GAN objectives, and a few audio translation models working on spectrograms have been developed; examples include singing voice separation (Fan et al., 2017; Stoller et al., 2017), source separation (Subakan & Smaragdis, 2017), and speech enhancement (Pascual et al., 2017; Donahue et al., 2017).

5 Machine Learning Models for Music Synthesis

The recent deep generative models have been very successful in synthesizing breathtakingly high-quality audio signals. We would want the synthesized music and audio signals to capture the long-term dependencies such as beats, measures, and chord progressions that ranges up to a few seconds, while the raw audio signals typically have the order of 10 thousand sameples per second. This made end-to-end synthesis models more difficult to train than image synthesis and translation models which it usually suffices to capture dependencies ranging a few hundred pixels. SampleRNN (Mehri et al., 2017), to be discussed in Chapter III in the context of deep autoregressive models, is one of the first successful deep generative models for audio and formed a basis for the techniques used by Lyrebird, an AI startup founded by University of Montréal students that provides API for synthesized voice of a specific person, e.g. Barack Obama. WaveNet (van den Oord, Dieleman, et al., 2016), developed by Google DeepMind, uses a causal architecture using dilated convolutions to generate time-domain audio samples, and is able to produce realistic human voices and piano sounds. WaveNet learns acoustically meaningful representations including pitch and spectral features (Hua, 2018). There also exist faster approaches using recurrent neural networks to produce vocal and musical audio, as found in (Nayebi & Vitelli, 2015) and (Kalingeri & Grandhe, 2016), albeit with lower quality when compared to WaveNet. Tacotron (Y. Wang

et al., 2017; Shen et al., 2018) is a fully end-to-end speech synthesizer that works directly on a sequence of characters, which can learn the pronunciation of unseen complex words and different ways of reading the same word according to the phrase semantics and punctuations. A newer RNN-based model called WaveRNN (Kalchbrenner et al., 2018) is capable of generating audio that matches WaveNet in quality, yet with an enough efficiency to be able to run real-time on GPUs or even on mobile phones. WaveGlow (Prenger et al., 2019) uses a flow-based approach to synthesize waveform samples and is also shown to produce WaveNet-quality audio while being able to run efficiently in parallel. A singing synthesis model (Blaauw & Bonada, 2017) based on the WaveNet architecture is also capable of synthesizing voice parametrically, separating the influence of pitch and timbre in the model.

Generative Adversarial Networks (GANs), also to be reviewed extensively in Chapter III, have also been used as generative models for audio. A GAN architecture using one-dimensional convolutions called *WaveGAN* was introduced by Donahue, McAuley, and Puckette (2018) and is capable of generating 1-second audio segments from the latent representations. In a newer approach called *GANSynth* (Engel et al., 2019), the GAN architecture was used for generating magnitude spectrograms together with the corresponding two-dimensional representation of instantaneous frequencies, which produced significantly more stable output compared to WaveGAN. Training GANs for arbitrary-length audio sequences and extending it as a tool for disentanglement of latent semantic information or a conditional audio synthesis framework remains a challenge.

6 Music Language Models for Symbolic Music Generation

Symbolic music processing refers to the techniques for processing music at a symbolic level, such as in the form of sheet music, MIDI signals, or piano roll representations. Problems in this domain include optical music recognition (Rebelo et al., 2012), algorithmic composition (Fernández & Vico, 2013), and computational music theory (Hamanaka et al., 2013), while the subject most relevant to music transcription research would be *music language models*. A music language model is a statistical model, often a generative model, that encodes music theoretic knowledge to describe the structural composition and arrangement of musical elements (Patel, 2008), similarly to how computational linguists build language models to describe the structure of natural languages. A well-designed music language model can be an important component for a generative model for music, because it can serve as a prior for latent representations and can be combined with conditional synthesis models or software instruments to produce audio.

The first systematic approach of applying a linguistic theory to music was the *generative theory of tonal music* (Lerdahl & Jackendoff, 1983), which was inspired by Noam Chomsky's generative grammar (Chomsky, 1966) and was influential in music theory, music psychology, and cognitive musicology. Music language models are loosely connected to this idea of generating music according to its grammar, and its implementations typically use statistical models that are also used in natural language processing. These include many kinds of approaches for symbolic music generation, such as hidden Markov models (Farbood & Schoner, 2001), generative grammars (Chemilier, 2001), cellular automata (Burraston et al., 2004), and genetic algorithms (Miranda et al., 2007). More recently, deep

learning models such as recurrent neural networks have been used to build music language models (Sigtia et al., 2014). The latest approaches to generate realistic music sequences in the symbolic domain include an application of variational autoencoder (Teng et al., 2017; Tikhonov & Yamshchikov, 2017), a generative adversarial network (Yang et al., 2017), and a transformer (C.-Z. A. Huang et al., 2019).

7 Summary

In this chapter, a broad range of MIR techniques related to automatic music transcription have been discussed, in the fields of monophonic and polyphonic pitch tracking, source separation and music translation, music synthesis, and music language models. A clear observation in each subtask of AMT is that many recent methods employ deep learning models, and this is because deep models have more flexibility and capacity to learn complex statistical relations of interest. In order to build the solid foundation of the various deep learning techniques used throughout this thesis, Chapter III will provide an extensive review on the various techniques and models that are collectively classified as deep learning.

CHAPTER III

DEEP LEARNING

Recently, a family of machine learning research under the term *deep learning* has incurred many groundbreaking changes to the world of artificial intelligence, making the long-waited dream of the *artificial general intelligence* (AGI)[1] look not so distant in the future[2]. The impact of deep learning has been so dramatic that many successful applications of deep learning like DeepMind's AlphaGo outplaying the human Go champion and Google's neural machine translation have became familiar to the general public. The core idea of using artificial neural networks to process complex information traces back to the earliest days of computing (Kleene, 1951) but has long been considered less effective than alternative methods, such as support vector machines or probabilistic graphical models. Since around 2012, it has been increasingly shown that neural networks can substantially outperform those other approaches and have much more capability for further improvements, and that the lower performance of neural networks in the past was mostly due to insufficient data, the lack of computational power, and some numerical tricks that had not been employed before. This finding has opened the era of deep learning — a term coined after the fact that neural networks often employ multiple layers of

[1] a loosely defined term referring to human-level intelligence, i.e. an AI system that can solve complex problems in varied and possibly previously-unseen domains with self-understanding and autonomous self-control. (Goertzel & Pennachin, 2007)

[2] Machine learning researchers, according to a large survey by Grace, Salvatier, Dafoe, Zhang, and Evans (2018), believe that there is 50% chance that AI will outperform human in all tasks in the next 45 years.

learned feature transformations — and is continuing to innovate virtually all fields of science and engineering, including, of course, music technology.

This chapter reviews the essential concepts and terminologies of deep learning, from the basic architectures and techniques to the most recent advances in deep generative models. The purpose of this chapter is to present a solid foundation as well as a historical perspective for the deep learning techniques used throughout the subsequent chapters such as autoregressive waveform synthesis models and generative adversarial networks, starting from the simplest concepts. At the same time, this chapter aims to provide a concise and timely overview of the field of deep learning as a whole, rather than explaining only the specific deep learning techniques used in the later chapters.

1 Neural Network Architectures

The key idea of an artificial neural network in the simplest setting is to find an appropriate matrix W to model the relationship between variables $\boldsymbol{x}$ and $\boldsymbol{y}$, represented as real-valued vectors, so that

$$\boldsymbol{y} = \sigma(W\boldsymbol{x}) \tag{3}$$

is a good approximation, where σ is a nonlinear function like the sigmoid or the hyperbolic tangent. This model in Equation 3 is also known as a *perceptron* (Rosenblatt, 1957), one of the first artificial neural networks in history. This computation — a matrix multiplication followed by a nonlinear activation — can be applied multiple times, like

$$\boldsymbol{y} = \sigma(W_3\sigma(W_2\sigma(W_1\boldsymbol{x}))), \tag{4}$$

which gives the model more expressive power, meaning that it can learn more complex relationship in the data that the previous model could not discern, e.g. the XOR problem (Riedmiller, 1994). The model in Equation 4 is called a *multilayer perceptron (MLP)* in a sense that it is a concatenation of perceptrons, and the fact that it contains multiple layers is why these neural networks are called "deep".

A multilayer perceptron is a special case of feedforward neural networks, which refer to any computational graph that does not contain a cycle. A popular model under this category is *convolutional neural networks (CNN)* (LeCun et al., 1995), which uses a convolution (a cross-correlation, to be precise) with fixed-size kernels instead of matrix multiplications. A 2-D convolutional layer takes input arrays $X_c \in \mathbb{R}^{H \times W}$, $c \in \{1, \cdots, C\}$, and produces output arrays $Y_d \in \mathbb{R}^{H \times W}$, $d \in \{1, \cdots, D\}$. The kernels $K_{cd} \in \mathbb{R}^{K_1 \times K_2}$, $c \in \{1, \cdots, C\}$, $d \in \{1, \cdots, D\}$, and the biases $\boldsymbol{b} \in \mathbb{R}^D$ are the parameters to be optimized, and the output is calculated as:

$$Y_d[i,\ j] = \sum_{m=1}^{K_1} \sum_{n=1}^{K_2} \sum_{c=1}^{C} K_{cd}[m,\ n] X_c[i+m,\ j+n] + b_d \tag{5}$$

There are various options to this operation including whether to pad the input or trim the output of the convolution to according to the kernel size, and how much to stride the kernels while moving along the input arrays. The 2-D convolution is suitable for image data, where the initial C can be the 3 RGB channels of color images; this allows the convolutional layer to extract features that are spatially local but across all channels. For similar reasons, 1-D and 3-D convolutions are often used with time-series and video data respectively.

Using convolutional layers results in a fewer number of parameters to learn in each layer than the equivalent multilayer perceptron, allowing deeper models for the same total number of parameters. LeNet (LeCun et al., 1995) for digit

classification is what pioneered the technique of using convolutional layers in neural networks, which has become an essential building block of the majority of deep learning methods. Such models include those that surpassed the human-level accuracy in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Szegedy et al., 2015; He et al., 2016). A standard practice of building a CNN is to stack a multiple convolutional layers along with pooling layers to obtain a compact and hierarchical feature repesentation that is also translation-invariant, which is then fed to a multi-layer perceptron as in Equation 4 to produce output. The layers of MLP are called fully connected or dense layers, because unlike the convolutional layers, the weight matrix used in the matrix multiplication associates every pair of the input and output features. *Fully convolutional networks*, which omit the fully connected layers that are typically placed at the last stages of neural networks, do not require a fixed input and output size and are known to perform well for image segmentation (Shelhamer et al., 2017). Using the ability of deep convolutional layers to extract complex semantic information from images, many artistic applications have been developed, such as the transfer of artistic style from one image to another (Gatys et al., 2015), and a captivating transformation of images using neural network weights known as *Deep Dream* (Mahendran & Vedaldi, 2016).

A network with cyclic connections is called a *recurrent neural network* (RNN; Rumelhart, Hinton, & Williams, 1986), and has been successfully applied to modeling sequential data. Because it is hard for a recurrent neural network to propagate long-range dependencies through a chain of recurrent connections, specific recurrent units called *long short-term memory* (LSTM; Hochreiter & Schmidhuber, 1997) and *gated recurrent unit* (GRU; K. Cho et al., 2014) are devised

to resolve the problem and are considered essential for recurrent neural networks. A formulation of recurrent neural network called the sequence-to-sequence model (K. Cho et al., 2014; Sutskever et al., 2014), which can model a mapping from variable-length input to variable-length output, is well known to be very effective for machine translation, and is deployed in production in Google's translation services (Y. Wu et al., 2016). An important technique for building recurrent neural networks is *attention* (Bahdanau et al., 2015), which allows the network to focus on specific parts of a sequence for generating outputs. The attention mechanism is shown to be effective in tasks including not only machine translation as in the original paper, but also in image description generation (Karpathy & Fei-Fei, 2017), speech recognition (Chorowski et al., 2015), and question answering (Sukhbaatar et al., 2015).

Reinforcement learning is a formulation of machine learning where a software agent takes actions in an environment to maximize the reward given according to the actions (Sutton & Barto, 2018). This formulation is inspired by behaviorist psychology and is well-suited for partially-observed environments that involve delayed rewards and require explorations by the agent, such as robotics and games. *Deep Q-Network* (DQN; Mnih et al., 2015) is a neural network model designed for reinforcement learning, which has been successfully applied to automatically playing Atari games (Mnih et al., 2013) and the agent playing the game of Go that surpassed the human level of skill (Silver et al., 2016).

2 Performance Optimization Techniques

The success of deep learning was possible not only because of the architectural design of deeper models and the hardware capable of supporting such models,

but also due to the numerous elaborate techniques and clever tricks that enabled previously impossible performances.

Training a neural network involves optimization of its parameters, e.g. the weights W in Equation 3-4 and the kernels K_{cd} in Equation 5, to minimize a task-specific loss function, which is a mapping from the parameters to a real number that we desire to be as low as possible. Such optimization processes usually require the gradient of the loss function, i.e. the partial derivatives with respect to all of the model's parameters. It is feasible to manually derive the gradient for simple models, but for deep neural networks it is often too complex and error-prone to calculate the derivative by hand. For this reason, a method called backpropagation (Werbos, 1982; Rumelhart, Hinton, & Williams., 1986) was introduced based on the ideas of automatic differentiation (Linnainmaa, 1970) and revived neural network research that had been largely abandoned. The popularization of *backpropagation* in the 1980s partly contributed to the ending of the first AI winter, leading to the first commercially successful application of neural network in optical digit recognition and speech recognition. Backpropagation is still a fundamental element of deep learning, and many deep learning frameworks are capable of automatically calculating gradients using backpropagation when a compute graph is given. This enables the developer to write only the forward calculation and run the backpropagation automatically, greatly improving the productivity.

Once a gradient is known, the standard way of optimizing a neural network is to use a variant of *stochastic gradient descent* (SGD; Robbins & Monro, 1951), where the direction of the gradient descent is determined only based on a mini-batch of training data. Although using only a tiny subset of training data makes the gradient

noisy, in practice, stochastic gradient descent converges faster than batch gradient descent using the same amount of training samples. Adding momentum (Polyak, 1964; Sutskever et al., 2013) in the gradient descent optimizer has shown to be effective for finding the convergence even faster, and many schemes for applying the momentum have been introduced, such as Adagrad (Duchi et al., 2011), RMSprop (G. Hinton, 2012), Adadelta (Zeiler, 2012), and Adam (Kingma & Ba, 2015). While Adam is by far the most popular choice of optimizer, a few modification to Adam's algorithm have been proposed, including Eve (Koushik & Hayashi, 2016) using feedback from the objective function, Nadam (Dozat, 2016) incorporating Nesterov's accelerated gradient descent (Nesterov, 1983), and AMSgrad (Reddi et al., 2018) fixing a failure case of Adam where it does not converge to the optimum even in a simple convex optimization problem.

Historically, the sigmoid and the hyperbolic tangent function have been popular choices for the nonlinearity, but it is surprisingly shown (Nair & Hinton, 2010) that the *rectified linear units* (ReLU),

$$f(x) = \max\{x, 0\}, \tag{6}$$

generally improves the accuracy of deep learning models. It is also known that neural networks with ReLU activations converge faster, and are more robust to the vanishing gradient problem. A number of ReLU variants, including leaky ReLU (Xu et al., 2015), parametric ReLU (PReLU; He, Zhang, Ren, & Sun, 2015), SReLU (Jin et al., 2015), have been devised and shown to be effective in some cases.

As with any other machine learning methods, overfitting is a problem to overcome for deep learning models as well. While directly adding a L1 or L2 regularization term of weights is possible, a few clever tricks for preventing

overfitting have been devised and widely employed, and they are treated as regularization methods in a wider sense. *Dropout* (Srivastava et al., 2014) is a simple yet powerful regularization method that turns off a random subset of activations during the training process. Because the network has to learn how to make accurate predictions using only a random subset of its components, the training becomes more robust and less susceptible to overfitting. *Batch normalization* (Ioffe & Szegedy, 2015) is a method to reduce the covariance shift of activations, by performing normalization for each training mini-batch so that the activations of each layer have zero mean and unit variance. It has also been empirically shown to improve the generalizability of the trained model. Despite being relatively new, dropout and batch normalization are drop-in methods that can be added to most deep architectures with almost no changes to code and yet significantly improve the performance and are thus included almost by default in the majority of newer deep models. *Scaled exponential linear units* (SELU; Klambauer, Unterthiner, Mayr, & Hochreiter, 2017) use a special activation function that induces a self-normalizing property over layers, making the activations have zero mean and unit variance without using batch normalization explicitly.

Additionally, because typical neural networks contain thousands to millions of parameters to train, a proper initialization of the weights prior to training is important. In early days of deep learning, unsupervised pre-training of weights (Bengio et al., 2007; Erhan et al., 2010) was considered necessary, but recently it is shown that a simple random initialization of weights is sufficient with the current computational power of the hardware. A widely practiced way of initializing the weights without unsupervised pre-training is to sample from a Gaussian or uniform

distribution, scaled according to the number of input and output nodes (Glorot & Bengio, 2010; He et al., 2015).

3 Toward Deep Generative Models

Statistical models that describe how data is generated are called *generative models* and provide means of generating samples of data, either by directly modeling the data distribution or through a sampling procedure specified by the model which implicitly defines the probability distribution. This is in contrast with *discriminative models*, which can only predict the labels corresponding to the given data samples.

3.1 Traditional Generative Models

Classic examples of generative models include *naïve Bayes classifiers* (Maron, 1961) which model a conditional distribution of each feature assuming they are conditionally independent given the label and use Bayes' theorem to predict the labels. *Gaussian mixture models* (GMM; Everitt & Hand, 1981) approximates the data distribution with a mixture of multivariate gaussian distributions.

While these simple models work effectively to a certain degree with a well-crafted set of features, it is desirable to have generative models that can capture more intricate geometry of the data distribution. *Probabilistic graphical models* (PGM; Pearl, 1988) specify the structural dependencies between random variables using graphs, with nodes representing random variables and connections between them representing their dependencies. The graphs can have directed or undirected connections to formulate the joint probability distributions of the variables. *Hidden Markov models* (HMM; Rabiner, 1989) and *latent Dirichlet*

allocation (LDA; Blei, Ng, & Jordan, 2003) are special cases of directed probabilistic graphical models, also called *Bayesian networks*, and are widely used for sequence modeling and topic modeling, respectively. Undirected graphical models, also called *Markov random fields* (MRF; Kindermann & Snell, 1980), have many applications in image processing, typically by having connections between nodes corresponding to adjacent pixels. Undirected graphical models where every pair of nodes has a connection are called *Boltzmann machines* (G. E. Hinton et al., 1984) and are capable of learning internal representations of data.

3.2 Early Deep Generative Models and Autoregressive Models

Restricted Boltzmann machines (RBM; Smolensky, 1986) are simplified variants of Boltzmann machines consisting of two layers of nodes with only interlayer connections, and unlike Boltzmann machines, there exists a relatively efficient algorithm (Carreira-Perpiñán & Hinton, 2005) for training restricted Boltzmann machines. *Deep belief networks* (DBN; G. Hinton et al., 2006) are composed of multiple layers of restricted Boltzmann machines, which can be trained using greedy layer-wise optimization, and are capable of classifying hand-written digits as well as conditionally generating them. Although deep belief networks are one of the first successful deep learning methods and gave many architectural and algorithmic insights to the development of deep learning in the subsequent years, the algorithms for training DBNs are not as scalable as those for other discriminative models like stochastic gradient descent, and eventually faded away in favor of the discriminative models that runs more effectively with a larger scale of data.

An alternative method to generate data samples using a neural network is

to produce one element (i.e. one audio sample or one pixel) at a time by feeding the previous elements to the network. This approach is called an autoregressive model, and many architectures based on this idea including NADE (Larochelle & Murray, 2011), DARN (Gregor et al., 2014), RIDE (Theis & Bethge, 2015), DRAW (Gregor et al., 2015), PixelCNN/PixelRNN (van den Oord, Kalchbrenner, & Kavukcuoglu, 2016), SampleRNN (Mehri et al., 2017), WaveNet (van den Oord, Dieleman, et al., 2016), and WaveRNN (Kalchbrenner et al., 2018) are proposed and shown to be capable of generating image and audio samples. However, in addition to being inevitably slow having to repetitively run the model for every element, a drawback of autoregressive models is the difficulty of interpreting the representation, because the autoregressive model only encodes the local dependency of one sample on the adjacent elements and does not provide a compact latent representation corresponding to the global structure.

3.3 Variational Autoencoders

A straightforward method to obtain a compact latent representation from unlabeled data is to build an encoder that transforms the input data into a smaller latent dimension, followed by a decoder that maps it back to the original data. This architecture is called an *autoencoder* (Bengio, 2009), and being a deep extension to principal component analysis (PCA), it is capable of learning a nonlinear mapping for dimensionality reduction. The autoencoder architecture are shown to be effective at encoding the latent representation of data, through a few successful variants including sparse autoencoder (Ng, 2011) which produces a sparse representation of the input data, denoising autoencoder (P. Vincent et al., 2008) which is capable of reducing noise or recovering a redacted portion of an

image, and contractive autoencoder (Rifai et al., 2011) which adds a regularization term to make the model robust to slight variations of input values.

Autoencoders are not generative models in a strict sense, because, while its decoder part can produce data samples from their latent representations, it lacks the ability to randomly sample the points in the latent dimensions that corresponds to the data distribution. *Variational autoencoders* (VAE; Kingma & Welling, 2014) address this problem by restricting the posterior latent distribution to be Gaussian. This is achieved by variational inference, reformulating the evidence lower bound (ELBO) of the data log-likelihood as:

$$\log p(\boldsymbol{x}) \geq \mathcal{L}(p_\theta, q_\phi) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})), \tag{7}$$

where $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ models the encoder and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ models the decoder, and both are parameterized using neural networks which provides a flexible and differentiable family of functions. Note that maximizing $\mathcal{L}$ will maximize the first term of RHS, the log likelihood of the reconstructed data, and minimize the second term, the KL divergence between the encoded data distribution and the Gaussian prior, serving as a regularizer that induces the posterior to be Gaussian. The Gaussian prior gives the KL divergence a closed-form solution making it straightfoward to derive the derivative according to ϕ, whereas the first term contains an expectation over a distribution depending on ϕ, which disallows moving the gradient operator into the expectation and makes the stochastic gradient descent and backpropagation impossible. A reparameterization trick is used to address this issue, by setting $\boldsymbol{z} = g(\epsilon, \boldsymbol{x}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \epsilon \cdot \boldsymbol{\sigma}_\phi(\boldsymbol{x})$ where $\epsilon \sim \mathcal{N}(0, 1)$, which gives:

$$\nabla_\phi \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] = \nabla_\phi \mathbb{E}_\epsilon[\log p_\theta(\boldsymbol{x}|g(\epsilon, \boldsymbol{x}))] = \mathbb{E}_\epsilon[\nabla_\phi \log p_\theta(\boldsymbol{x}|g(\epsilon, \boldsymbol{x}))], \tag{8}$$

making the stochastic gradient ascent and backpropagation on the ELBO possible.

A drawback of variational autoencoders, however, is the blurriness in reconstructed images, that may come from the inexactness of the Gaussian assumption and the variational lower bound used by the model (Doersch, 2016). There have been many attempts to overcome this by allowing more flexible prior distributions (Rezende & Mohamed, 2015) as well as better latent representations (Kingma et al., 2016). VQ-VAE (van den Oord et al., 2017) uses discrete prior and posterior distributions, and is able to generate less blurry images and perform speaker conversion using raw audio. Variational autoencoders are powerful deep generative models with the advantages of having a single objective function to be optimized and thus having a stable training scheme. Despite being one of the most successful types of deep generative models to date, it remains to be seen if variational autoencoders can be extended to become a building block of an automatic music transcription system.

4 Generative Adversarial Networks

Generative adversarial networks (GAN; Goodfellow et al., 2014) are a family of deep generative models that have become extremely popular. Unlike other deep neural network models that use optimization to find the weights minimizing the loss function, GANs try to find a Nash equilibrium between its two components, the generator and discriminator. Given the training data $\boldsymbol{x} \sim p_{\text{data}}$ and the prior of latent vectors $\boldsymbol{z} \sim p_{\boldsymbol{z}}$ which typically is a multivariate Gaussian distribution, GAN performs the following minimax game:

$$\min_G \max_D \Big[\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \log D(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim p_z} \log \left(1 - D(G(\boldsymbol{z}))\right) \Big], \tag{9}$$

where the generator G learns to transform a noise vector $\boldsymbol{z}$ into a data point that can fool the discriminator as if it is a real data sample, while the discriminator D tries to correctly distinguish the output of generator $G(\boldsymbol{z})$ from the real data $\boldsymbol{x}$. Because the second expectation has a near-zero gradient where $D(G(\boldsymbol{z})) \approx 0$, i.e. the discriminator classifies the generated samples as fake, the authors suggests using a non-saturating loss for training the generator:

$$\max_{G} \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} \log D(G(\boldsymbol{z})), \tag{10}$$

which maximizes the log-likelihood of the discriminator classifying the generated samples as real.

4.1 Evolution of the GAN Architecture

The original formulation of GAN uses neural networks, which can only be applied to simple datasets of up to 32×32 images such as MNIST, CIFAR-10, and Toronto Face Dataset. LAPGAN (Denton et al., 2015) is the first GAN formulation to generate 64×64 images, which builds upon a Laplacian pyramid of convolutional layers that conditionally generates an image that is twice larger, gradually building 64×64 images from 4×4 samples. DCGAN (Radford et al., 2015) provides a simpler method of training convolutional GANs by following a list of architectural choices, making adversarial training of 64×64 images possible only using one generator and one discriminator. Together with Improved GAN (Salimans et al., 2016) which proposed now-standard tricks such as feature matching, minibatch discrimination, historical averaging, and one-sided label smoothing to generate 128×128 images, the DCGAN architecture is employed by virtually all subsequent GAN applications. For photo-realistic images synthesis in a higher resolution,

StackGAN (H. Zhang et al., 2017a) uses a multi-stage GAN architecture to synthesize 256×256 images, and progressive growing of GANs (Karras et al., 2018) uses a training scheme that gradually switches to larger GANs to generated photo-realistic images of size 1024×1024. The GAN architecture is also applicable to 3-D (J. Wu et al., 2016) and 1-D (Donahue et al., 2018) synthesis, suitable for generation of video and audio data, respectively.

4.2 The GAN Zoo

GANs are notoriously difficult to train (Arjovsky & Bottou, 2017); its convergence is unstable because of the minimax nature of its formulation, and the generator network may simply memorize and output just a few samples of training data, an undesirable phenomena called mode collapsing. A plethora of variations of GAN have been proposed to mitigate this problem, aiming to stabilize the training and/or improve the perceptual quality of generated samples. A common approach among them is to devise a different loss function or to add a regularization term to the loss function that penalizes mode collapsing.

f-GAN (Nowozin & Cseke, 2016) is a generalization of the original GAN using a family of f-divergences in addition to the original GAN's formulation which uses Jensen-Shannon divergence. Wasserstein GAN (WGAN) (Arjovsky et al., 2017) minimizes the Wasserstein distance between the model and real distribution, and was later extended to WGAN-GP (Gulrajani et al., 2017) which uses a gradient penalty that does not require weight clipping as in Wasserstein GAN. Based on WGAN's observation that Lipschitz continuity of GAN is beneficial, spectral normalization (Miyato et al., 2018) is another technique to impose Lipschitz

continuity that is more stable and provides higher diversity in generated images than gradient penalty.

Least-Square GAN (LSGAN) (Mao et al., 2017) uses least-square losses instead, and also can be trained with gradient penalty (Mao et al., 2018) which achieves a training stability similar to that of WGAN-GP. Energy-Based GAN (EBGAN) (Zhao et al., 2017) views the discriminator as an energy function that puts low energy near the data manifold, and uses an autoencoder to model the discriminator. Boundary Equilibrium GAN (BEGAN) (Berthelot et al., 2017) extends the EBGAN architecture using Wasserstein distance to balance the generator and discriminator during training. While there are many more GAN formulations claiming to be superior than others, a large-scale empirical study (Lucic et al., 2017) suggested that the none of the popular variants actually outperforms the original GAN, provided that the hyperparameters are sufficiently optimized.

4.3 Conditional Generation and Other Applications

It is usually desirable to generate samples according to certain conditions, e.g. generating images for a specific digit. Conditional GAN (cGAN) (Mirza & Osindero, 2014) is an architecture where the generator can use the class label as well as the noise input to produce samples. Auxiliary Classifier GAN (AC-GAN) (Odena et al., 2016) is an extension to cGAN in which the discriminator can also serve as a classifier for categories of data. InfoGAN (Chen et al., 2016) can perform conditional generation in a completely unsupervised manner, i.e. without using class labels during training, by maximizing the mutual information between a subset of the latent variables and the observations.

All of the above architectures use conditional latent components

concatenated to the other feature components, which makes training a conditional GAN with a large number of classes difficult. (Miyato & Koyama, 2018) overcomes this by performing projections on the feature space of the discriminator, successfully demonstrating conditional image synthesis on the 1,000 classes of images of the ILSVRC dataset (Russakovsky et al., 2015).

Image-to-image translation models are also considered as a conditional generation problem, where the input image is the condition. Many such models have been developed using GANs with artistically interesting results. `pix2pix` (Isola et al., 2017) is trained on pairs of cross-domain images and learns to translate images between the domains, e.g. satellite images to corresponding maps, day photos to night photos, and edges of images to the original. DiscoGAN and CycleGAN (T. Kim et al., 2017; Zhu et al., 2017) are capable of performing similar tasks, but does not require paired training samples, greatly expanding the ranges of datasets that can be used for training. StarGAN (Y. Choi et al., 2017) learns to translate between more than two domains.

GAN has been successfully applied to many other tasks, including image super-resolution (AffGAN) (Sønderby et al., 2017), text-to-image synthesis (StackGAN) (H. Zhang et al., 2017b, 2017a), text generation (Boundary-seeking GAN, BGAN) (Hjelm et al., 2018), and speech enhancement (SEGAN) (Pascual et al., 2017).

4.4 Evaluation of Generated Samples

Unlike discriminative models that can be evaluated using well-defined metrics, it is not as straightforward to evaluate the performance of generative models (Theis et al., 2016). Structural similarity (SSIM) (Z. Wang et al., 2004) and its multi-scale

extension MS-SSIM (Z. Wang et al., 2003) can measure the similarity between two images using luminance, contrast, and structure information.

Another popular method for evaluating the perceptual quality of generated images is the Inception score (Salimans et al., 2016), based on the image classification model under the same name (Szegedy et al., 2016) which is a 1000-class image classifier trained on the ILSVRC dataset (Russakovsky et al., 2015). Assuming that meaningful images would have a low-entropy conditional label distribution, and a diverse set of generated images would have a high-entropy marginal label distribution, the Inception score can be obtained by feeding generated images to the Inception classifier and calculating:

$$\exp\left(\mathbb{E}_{\boldsymbol{x}}\ \mathrm{KL}\left(p(y|\boldsymbol{x})||p(y)\right)\right]\right). \tag{11}$$

While Inception scores correlate well with human perception, a drawback of this is that the distribution of real data is not considered in the calculation, which is problematic for a metric measuring how realistic the generated samples are.

Fréchet Inception distance (FID) (Heusel et al., 2017) addresses this problem and provides a distance metric between the data distribution and model distribution. FID is defined using the activations of a coding layer of the Inception-v3 model, `pool_3` to be specific, as the Fréchet distance between multivariate Gaussian approximations of the two distributions:

$$d^2 = \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2 + \mathrm{Tr}\left(C_1 + C_2 - 2(C_1C_2)^{1/2}\right), \tag{12}$$

where $\boldsymbol{\mu}_i$ and C_i are the mean vector and the covariance matrix of the coding layer for each distribution.

4.5 Theories on GAN Convergence

The formulation of GAN training and ways to overcome its instability have been studied from various angles. Mohamed and Lakshminarayanan (2017) generalized the GAN objective function to a wider range of implicit generative models, and Fedus et al. (2018) studied the relations of the divergence function and the equilibrium. Arjovsky and Bottou (2017) and Kodali, Abernethy, Hays, and Kira (2017) studied the training dynamics of GANs around the local equilibria and its relation to the mode collapse problem, and Daskalakis, Ilyas, Syrgkanis, and Zeng (2018) used optimistic mirror descent to reduce the cycling behavior in GAN training. Mescheder, Nowozin, and Geiger (2017) observed that the training is locally convergent when all eigenvalues of the Jacobian have negative real-part. Nagarajan and Kolter (2017) showed that with an absolute continuity assumption on the data and generator distributions ensure this case, but Sønderby et al. (2017) confirmed that this assumption does not necessarily hold in general.

Despite these advances in theoretical understanding, many of these studies are inconclusive, and there is not yet a out-of-the-box method that can make a GAN always converge. Unfortunately, the best practice at the moment is to carefully monitor the divergences and the Jacobian to determine if the training is leading to a convergence.

5 Summary

This chapter provided a comprehensive introduction to deep learning and deep generative models, with an emphasis on generative adversarial networks. The fast progress of the research on deep generative models, together with the clear trend toward data-driven MIR research as discussed in Chapter II, further motivates the

utilization of deep learning and especially deep generative models in automatic music transcription research.

CHAPTER IV

CREPE: DEEP MONOPHONIC PITCH ESTIMATION

As outlined in Chapter I, we start with a simpler formulation of music transcription that is later extended to ultimately support multi-instrument polyphonic music transcription. This chapter concerns the first step where we assume that the input is monophonic, i.e. the audio contains at most one pitched sound at each instant. Through this simplified formulation, we aim to gain insights on how a supervised learning framework for music transcription tasks can be designed. Furthermore, we seek to identify important aspects of the proposed approach that can be applied to or extended to polyphonic and multi-instrument transcription tasks. The content of this chapter is largely based on the work presented at IEEE ICASSP 2018 (J. W. Kim et al., 2018).

1 Introduction

Estimating the fundamental frequency (f0) of a monophonic audio signal, also known as pitch tracking or pitch estimation, is a long-standing topic of research in audio signal processing. Pitch estimation plays an important role in music signal processing, where monophonic pitch tracking is used as a method to generate pitch annotations for multi-track datasets (Bittner et al., 2014) or as a core component of melody extraction systems (Bosch & Gómez, 2014; Mauch et al., 2015). Pitch

estimation is also important for speech analysis, where prosodic aspects such as intonations may reflect various features of speech (Zubizarreta, 1998).

We have reviewed a few approaches for monophonic pitch tracking in Section II.2, including YIN (de Cheveigné & Kawahara, 2002) and pYIN (Mauch & Dixon, 2014). A notable trend in those methods is that the derivation of a better pitch detection system solely depends on cleverly devising a robust candidate-generating function and/or sophisticated post-processing steps, i.e. heuristics, and none of them are directly learned from data, except for manual hyperparameter tuning. This contrasts with many other problems in music information retrieval like chord ID (Humphrey & Bello, 2012) and beat detection (Böck & Schedl, 2011), where data-driven methods have been shown to consistently outperform heuristic approaches. One possible explanation for this is that since fundamental frequency is a low-level physical attribute of an audio signal which is directly related to its periodicity, in many cases heuristics for estimating this periodicity perform extremely well with accuracies (measured in raw pitch accuracy, defined later on) close to 100%, leading some to consider the task a solved problem. This, however, is not always the case, and even top performing algorithms like pYIN can still produce noisy results for challenging audio recordings such as a sound of uncommon instruments or a pitch curve that fluctuates very fast. This is particularly problematic for tasks that require a flawless f0 estimation, such as using the output of a pitch tracker to generate reference annotations for melody and multi-f0 estimation (Salamon et al., 2017; Bittner et al., 2017).

In the following sections, a novel, data-driven method for monophonic pitch tracking based on a deep convolutional neural network operating on

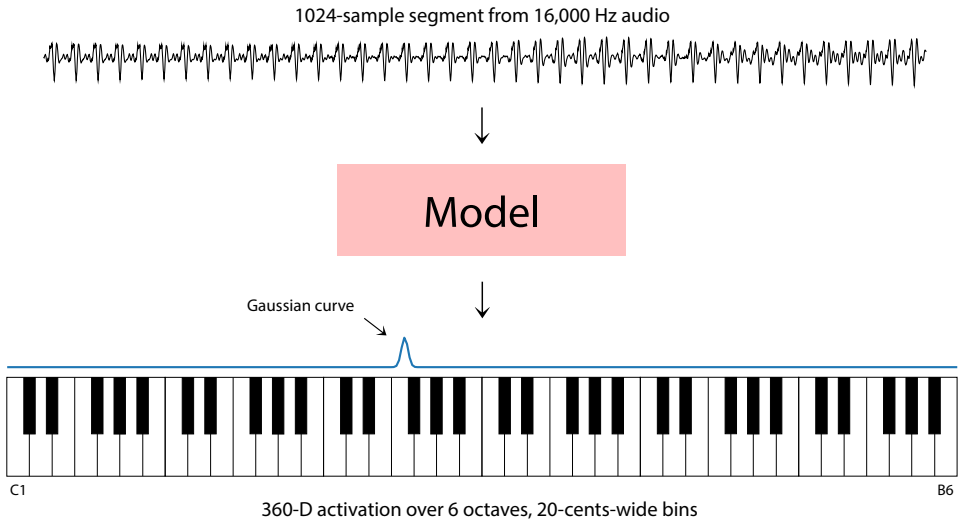


Figure 6: The input and output representations of the CREPE model. The input is 1024 time-domain samples, and the output is a 360-dimensional vector that contains a Gaussian curve centered at the ground-truth frequency (see Equation 16). For details on the convolutional architecture used in the model, see Figure 7.

the time-domain signal is presented. This approach, CREPE (Convolutional Representation for Pitch Estimation), obtains state-of-the-art results, outperforming heuristic approaches such as pYIN and SWIPE while being more robust to noise too. It is further shown that CREPE is highly precise, maintaining over 90% raw pitch accuracy even for a strict evaluation threshold of just 10 cents.

2 Architecture

CREPE consists of a deep convolutional neural network which operates directly on the time-domain audio signal to produce a pitch estimate. The input and output representations used in the model are illustrated in Figure 6. The input is a 1024-sample excerpt from the time-domain audio signal, using a 16 kHz sampling rate, and the output is a 360-dimensional vector $\hat{\mathbf{y}}$, from which the resulting pitch estimate is calculated deterministically.

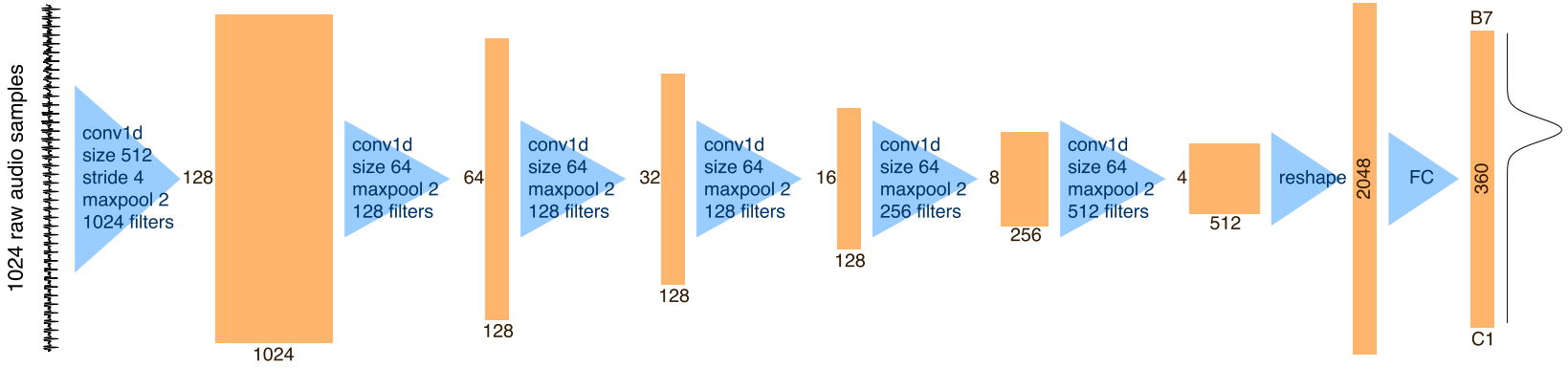


Figure 7: The architecture of the CREPE pitch tracker. The six convolutional layers predicts the Gaussian curve centered at the ground-truth frequency. The output is then used to extract the exact pitch estimate as in Equation 14-15.

Each of the 360 dimensions in the output corresponds to a specific pitch value, defined in cents. Cent is a unit representing musical intervals relative to a reference pitch f_{ref} in Hz, defined as a function of frequency f in Hz:

$$c(f) = 1200 \cdot \log_2 \frac{f}{f_{\text{ref}}}, \tag{13}$$

where $f_{\text{ref}} = 10$ Hz throughout the experiments. This unit provides a logarithmic pitch scale where 100 cents equal one semitone. The 360 pitch values are denoted as $c_1, c_2, \cdots, c_{360}$ and are selected so that they cover six octaves with 20-cent intervals between C1 and B6, corresponding to 32.70 Hz and 1975.5 Hz. The resulting pitch estimate $\hat{c}$ is the weighted average of the associated pitches c_i according to the output $\hat{\mathbf{y}}$, which gives the frequency estimate in Hz:

$$\hat{c} = \frac{\sum_{i=1}^{360} \hat{y}_i c_i}{\sum_{i=1}^{360} \hat{y}_i}, \tag{14}$$

$$\hat{f} = f_{\text{ref}} \cdot 2^{\hat{c}/1200}. \tag{15}$$

The target outputs we use to train the model are 360-dimensional vectors, where each dimension represents a frequency bin covering 20 cents (the same as the model's output). The bin corresponding to the ground truth fundamental frequency is given a magnitude of one. As in Bittner et al. (2017), in order to soften the penalty for near-correct predictions, the target is Gaussian-blurred in frequency such that the energy surrounding a ground truth frequency decays with a standard deviation of 25 cents:

$$y_i = \exp\left(-\frac{(c_i - c_{\text{true}})^2}{2 \cdot 25^2}\right), \tag{16}$$

This way, high activations in the last layer indicate that the input signal is likely to have a pitch that is close to the associated pitches of the nodes with high activations.

A detailed block diagram of the proposed architecture is provided in Figure 7. There are six convolutional layers that result in a 2048-dimensional latent representation, which is then connected densely to the output layer with sigmoid activations to produce the 360-dimensional target vector. The network is trained to minimize the binary cross entropy between the target vector $\mathbf{y}$ and the predicted vector $\hat{\mathbf{y}}$:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{360} \left(-y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)\right), \tag{17}$$

where both y_i and $\hat{y}_i$ are real numbers between 0 and 1. This loss function is optimized using the ADAM optimizer (Kingma & Ba, 2015), with the learning rate 0.0002. The best performing model is selected after training until the validation accuracy no longer improves for 32 epochs, where one epoch consists of 500 batches of 32 examples randomly selected from the training set. Each convolutional layer is preceded with batch normalization (Ioffe & Szegedy, 2015) and followed by a dropout layer (Srivastava et al., 2014) with the dropout probability 0.25. This architecture and the training procedures are implemented using Keras (Chollet, 2015).

3 Experiments

3.1 Datasets

In order to objectively evaluate CREPE and compare its performance to alternative algorithms, we require audio data with perfect ground truth annotations. This is especially important since the performance of the compared algorithms is already

very high. In light of this, we cannot use a dataset such as MedleyDB (Bittner et al., 2014), since its annotation process includes manual corrections which do not guarantee a 100% perfect match between the annotation and the audio, and it can be affected, to a degree, by human subjectivity. To guarantee a perfectly objective evaluation, we must use datasets of synthesized audio in which we have perfect control over the f0 of the resulting signal. We use two such datasets: the first, RWC-synth, contains 6.16 hours of audio synthesized from the RWC Music Database (Goto et al., 2002) and is used to evaluate pYIN in (Mauch & Dixon, 2014). It is important to note that the signals in this dataset were synthesized using a fixed sum of a small number of sinusoids, meaning that the dataset is highly homogenous in timbre and represents an over-simplified scenario. To evaluate the algorithms under more realistic (but still controlled) conditions, the second dataset we use is a collection of 230 monophonic stems taken from MedleyDB and re-synthesized using the methodology presented in (Salamon et al., 2017), which uses an analysis/synthesis approach to generate a synthesized track with a perfect f0 annotation that maintains the timbre and dynamics of the original track. This dataset consists of 230 tracks with 25 instruments, totaling 15.56 hours of audio, and henceforth referred to as MDB-stem-synth.

3.2 Methodology

We train the model using 5-fold cross-validation, using a 60/20/20 train, validation, and test split. For MDB-stem-synth, we use artist-conditional folds, in order to avoid training and testing on the same artist which can result in artificially high performance due to artist or album effects (Sturm, 2013). The evaluation of an algorithm's pitch estimation is measured in raw pitch accuracy (RPA) and raw

chroma accuracy (RCA) with 50 cent thresholds (Salamon et al., 2014). These metrics measure the proportion of frames in the output for which the output of the algorithm is within 50 cents (a quarter-tone) of the ground truth. We use the reference implementation provided in `mir_eval` (Raffel et al., 2014) to compute the evaluation metrics.

We compare CREPE against the current state of the art in monophonic pitch tracking, represented by the pYIN (Mauch & Dixon, 2014) and SWIPE (Camacho & Harris, 2008) algorithms. To examine the noise robustness of each algorithm, we also evaluate their pitch tracking performance on degraded versions of MDB-stem-synth, using the Audio Degradation Toolbox (ADT) (Mauch & Ewert, 2013). We use four different noise sources provided by the ADT: pub, white, pink, and brown. The pub noise is an actual recording of the sound in a crowded pub, and the white noise is a random signal with a constant power spectral density over all frequencies. The pink and brown noise have the highest power spectral density in low frequencies, and the densities fall off at 10 dB and 20 dB per decade respectively. Seven different signal-to-noise ratio (SNR) values are used: ∞, 40, 30, 20, 10, 5, and 0 dB.

3.3 Results

3.3.1 Pitch Accuracy

Table 1 shows the pitch estimation performance tested on the two datasets. On the RWC-synth dataset, CREPE yields a close-to-perfect performance where the error rate is lower than the baselines by more than an order of magnitude. While these high accuracy numbers are encouraging, those are achievable thanks to the highly homogeneous timbre of the dataset. In order to test the generalizability of

Dataset	Metric	CREPE	pYIN	SWIPE
RWC-synth	RPA	**0.999±0.002**	0.990±0.006	0.963±0.023
	RCA	**0.999±0.002**	0.990±0.006	0.966±0.020
MDB-stem-synth	RPA	**0.967±0.091**	0.919±0.129	0.925±0.116
	RCA	**0.970±0.084**	0.936±0.092	0.936±0.100

Table 1: Average raw pitch/chroma accuracies and their standard deviations, tested with the 50 cents threshold.

Dataset	Threshold	CREPE	pYIN	SWIPE
RWC-synth	50 cents	**0.999±0.002**	0.990±0.006	0.963±0.023
	25 cents	**0.999±0.003**	0.972±0.012	0.949±0.026
	10 cents	**0.995±0.004**	0.908±0.032	0.833±0.055
MDB-stem-synth	50 cents	**0.967±0.091**	0.919±0.129	0.925±0.116
	25 cents	**0.953±0.103**	0.890±0.134	0.897±0.127
	10 cents	**0.909±0.126**	0.826±0.150	0.816±0.165

Table 2: Average raw pitch accuracies and their standard deviations, with different evaluation thresholds.

the algorithms on a more timbrally diverse dataset, the performance is evaluated on the MDB-stem-synth dataset as well. It is notable that the degradation of performance from RWC-synth is more significant for the baseline algorithms, implying that CREPE is more robust to complex timbres compared to pYIN and SWIPE.

Finally, to see how the algorithms compare under scenarios where any deviation in the estimated pitch from the true value could be detrimental, Table 2 reports the RPA at lower evaluation tolerance thresholds of 10 and 25 cents as well as the RPA at the standard 50 cents threshold for reference. The table indicates that as the threshold is decreased, the difference in performance becomes more

accentuated, with CREPE outperforming by over 8 percentage points when the evaluation tolerance is lowered to 10 cents. The high accuracy in 10 cents despite the 20-cent resolution of the model output indicates that the weighted average solution in Equation 14 is effective at predicting the precise frequencies even between the adjacent frequency bins. This suggests that CREPE is especially preferable when even minor deviations from the true pitch should be avoided as best as possible. Obtaining highly precise pitch annotations is perceptually meaningful for transcription and analysis/resynthesis applications.

3.3.2 Noise Robustness

Noise robustness is key to many applications like speech analysis for mobile phones or smart speakers, or for live music performance. Figure 8 shows how the pitch estimation performance is affected when an additive noise is present in the input signal. CREPE maintains the highest accuracy for all SNR levels for pub noise and white noise, and for all SNR levels except for the highest level of pink noise. Brown noise is the exception where pYIN's performance is almost unaffected by the noise. This can be attributed to the fact that brown noise has most of its energy at low frequencies, to which the YIN algorithm (on which pYIN is based) is particularly robust.

To summarize, CREPE performs better in all cases where the SNR is below 10 dB while the performance varies depending on the spectral properties of the noise when the noise level is higher, which indicates that this approach can be reliable under a reasonable amount of additive noise. CREPE is also more stable, exhibiting consistently lower variance in performance compared to the baseline algorithms.

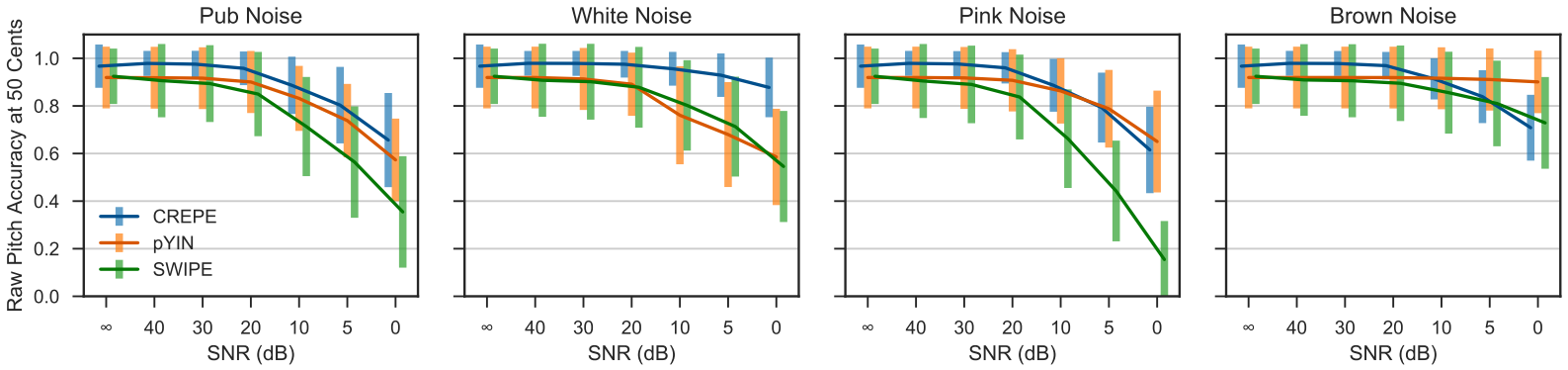


Figure 8: Pitch tracking performance when additive noise signals are present. The error bars are centered at the average raw pitch accuracies and span the first standard deviations. With brown noise being a notable exception, CREPE shows the highest noise robustness in general.

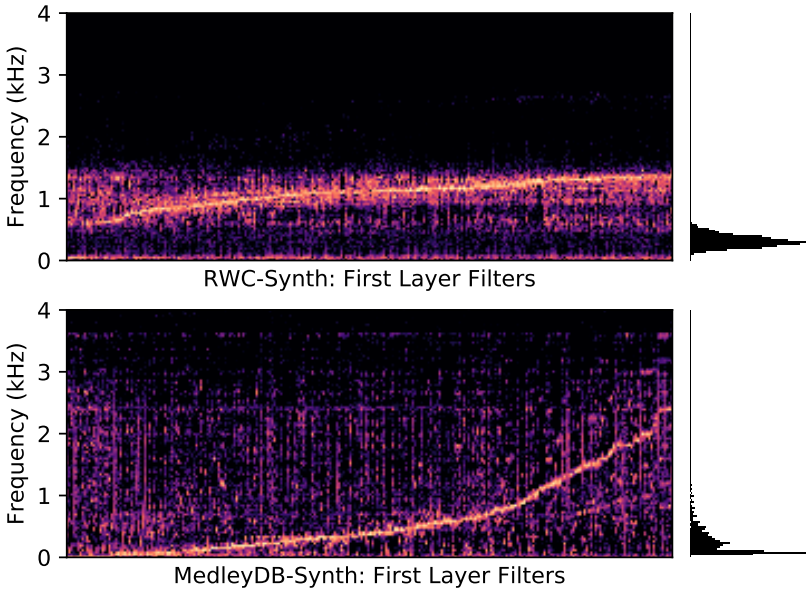


Figure 9: Fourier spectra of the first-layer filters sorted by the frequency of the peak magnitude. Histograms on the right show the distribution of ground-truth frequencies in the corresponding dataset.

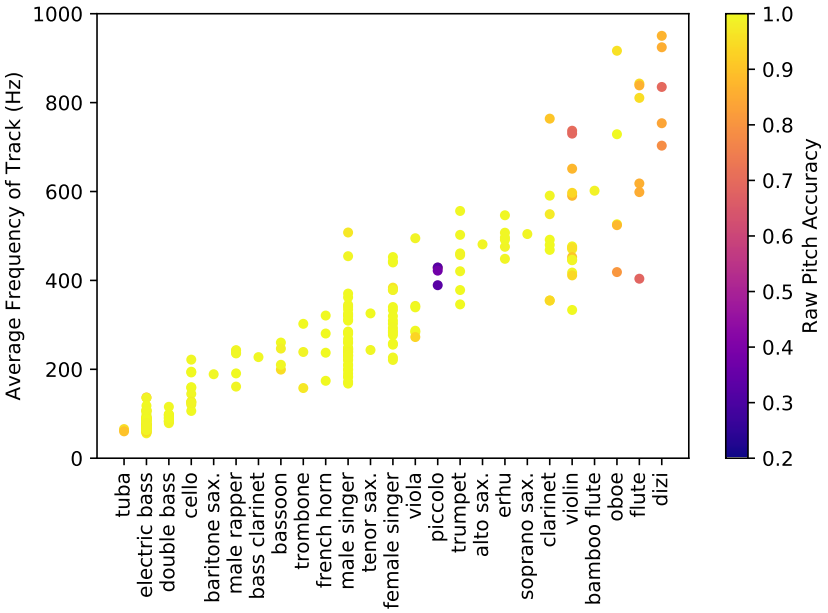


Figure 10: The raw pitch accuracy (RPA) of CREPE's predictions on each of the 230 tracks in MDB-stem-synth with respect to the instrument, sorted by the average frequency.

3.3.3 Model Analysis

To gain some insight into the CREPE model, we visualize in Figure 9 the spectra of the 1024 convolutional filters in the first layer of the neural network, with histograms of the ground-truth frequencies to the right of each plot. It is noticeable that the filters learned from the RWC-synth dataset have the spectral density concentrated between 600 Hz and 1500 Hz, while the ground-truth frequencies are mostly between 100 Hz and 600 Hz. This indicates that the first convolutional layer in the model learns to distinguish the frequencies of the overtones rather than the fundamental frequency. These filters focusing on overtones are also visible for MDB-stem-synth, where peak frequencies of the filters range well above the f0 distribution of the dataset, but in this case, the majority of the filters overlap with the ground-truth distribution, unlike RWC-synth. A possible explanation for this is that since the timbre in RWC-synth is fixed and identical for all tracks, the model is able to obtain a highly accurate estimate of the f0 by modeling its harmonics. Conversely, when the timbre is heterogeneous and more complex, as is the case for MDB-stem-synth, the model cannot rely solely on the harmonic structure and requires filters that capture the f0 periodicity directly in addition to the harmonics. In both cases, this suggests that the neural network can adapt to the distribution of timbre and frequency in the dataset of interest, which in turn contributes to the higher performance of CREPE compared to the baseline algorithms.

3.3.4 Performance by Instrument

The MDB-stem-synth dataset contains 230 tracks from 25 different instruments, where electric bass (58 tracks) and male singer (41 tracks) are the most common while there are instruments that occur in only one or two tracks. Figure 10

shows the performance of CREPE on each of the 230 tracks, with respect to the instrument of each track. It is notable that the model performs worse for the instruments with higher average frequencies, but the performance is also dependent on the timbre. CREPE performs particularly worse on the tracks with the dizi, a Chinese transverse flute, because the tracks came from the same artist, and they are all placed in the same split. This means that for the fold in which the dizi tracks are in the test set, the training and validation sets do not contain a single dizi track, and the model fails to generalize to this previously unseen timbre. There are 5 instruments (bass clarinet, bamboo flute, and the family of saxophones) that occur only once in the dataset, but their performance is decent, because their timbres do not deviate too far from other instruments in the dataset. For the flute and the violin, although there are many tracks with the same instrument in the training set, the performance is low when the sound in the tested tracks is too low (flute) or too high (violin) compared to other tracks of the same instruments. The low performance on the piccolo tracks is due to an error in the dataset where the annotation is inconsistent with the correct pitch range of the instrument. Unsurprisingly, the model performs well on test tracks whose timbre and frequency range are well-represented in the training set.

4 Open-Sourcing CREPE

Releasing an implementation of research work as an open-source software helps ensure better quality, reproducibility, and longevity of the research code (McFee et al., 2019). To facilitate easier adoption of CREPE's pitch tracking functionality for wider audience, we open-sourced[1] the implementation of CREPE under the

[1] `https://github.com/marl/crepe`

MIT License. The main purpose of the open-source release is to make the usage of the CREPE software as easy and effective as possible, so we aimed to ensure that the pitch estimation can work accurately for the broad range of the real-world monophonic signals. As shown in Figure 9, a model trained on a specific dataset specializes to the characteristics of the sounds in the dataset and will fail to work well across the previously unknown, real-world inputs. To address this problem, we used six different datasets for the pre-trained model included in the open-source release of CREPE: MIR-1K (Hsu & Jang, 2010), Bach10 (Duan et al., 2010), RWC-Synth (Mauch & Dixon, 2014), MedleyDB (Bittner et al., 2014), MDB-STEM-Synth (Salamon et al., 2017), and NSynth (Engel et al., 2017). These datasets contain various instrumental and vocal sounds, and the pre-trained model is expected to work well on these types of sounds. A visualization of the first-layer filters for the model trained with these datasets are shown in Figure 11. Compared to Figure 9, the peak-frequency curve better covers the full range of frequencies and in a smoother manner. An interesting feature of this visualization is that the curve is somewhat linear until 1 kHz and rapidly grows for the higher frequencies. This roughly coincides with the Mel scale, where the mapping from the perceived frequency to the actual frequency is linear below 1 kHz and logarithmic above (Logan, 2000).

4.1 Python Package and Command-Line Interface

CREPE is distributed as a Python package and is hosted on the Python Package Index (PyPI). It can be installed simply by running the following command:

```
$ pip install crepe
```

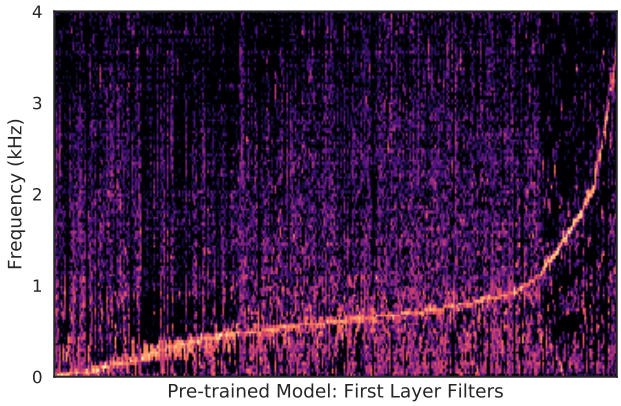


Figure 11: The first layer filters of the CREPE model trained with six different datasets, visualized using the same method as in Figure 9. Compared to the previous cases where the filters are specialized to one dataset, the peak-frequency curve is smoother and covers the full frequency range.

in a Python environment. This command will install the Python package `crepe` usable in Python scripts as well as a command-line interface that can be called with one or more audio file names:

```
$ crepe audio_file.wav
```

The estimated pitch values are saved using the CSV format, like:

```
time,frequency,confidence
0.00,185.616,0.907112
```

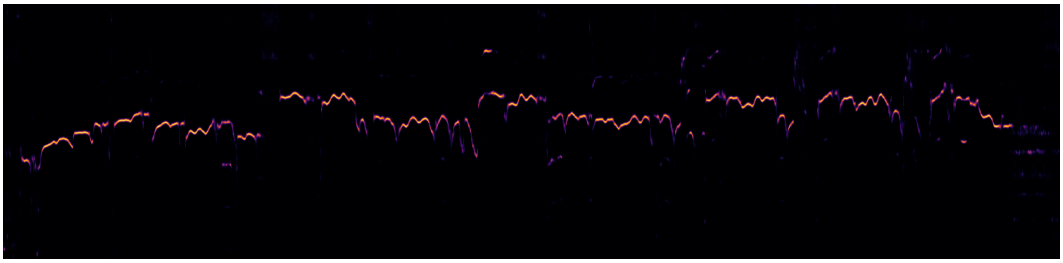


Figure 12: Visualization of the target activation, which can be saved as in image file by using the `--save-activation` option in the CREPE command-line interface. The audio clip used contains an excerpt of male singing voice.

```
0.01,186.764,0.844488
0.02,188.356,0.798015
0.03,190.610,0.746729
0.04,192.952,0.771268
0.05,195.191,0.859440
0.06,196.541,0.864447
0.07,197.809,0.827441
0.08,199.678,0.775208
...
```

where the third column indicates the confidence level of the presence of a pitch, which is obtained by taking the maximum activation value at each frame.

The command-line interface supports various options, such as to use Viterbi smoothing (Viterbi, 1967) of the pitch curves as in pYIN (Mauch & Dixon, 2014) and to save the activation in an image file as shown in Figure 12. The package contains five pre-trained models with different capacities, where the number of convolutional channels are scaled to 12.5% (tiny), 25% (small), 50% (medium), 75% (large), or 100% (full) relative to the model size used in the experiments (see Figure 7), and the specific model capacity to use can be specified using the `--model-capacity` option. Smaller-capacity models run significantly faster than the full-capacity model, suitable for resource-constrained use cases where faster computation is preferred at the cost of slightly lower pitch estimation accuracy.

4.2 Real-Time Web Demo

In addition to the Python package, we have developed a real-time web demo[1] where the CREPE model runs on the web browser using the audio recorded from the system microphone and displays the target activation as well as the predicted pitch value in real time. To enable this, the demo used the W3C Web Audio API for real-time audio input and TensorFlow.js (Smilkov et al., 2019) for running inference on the deep model. Under the hood, TensorFlow.js uses WebGL which allows GPU acceleration for faster computation of the convolutional layers. The web demo uses the "tiny" capacity model in order to run faster in browsers, which uses less than 3% of the parameters compared to the full model. The web demo is also open-source and distributed in the same Github repository as the Python package.

4.3 Argmax-Local Weighted Averaging

In the process of training the model with diverse datasets, we have observed a side-effect where the model produces more octave errors, i.e. the predicted pitch is an octave apart from the ground truth. These errors resulted in multiple peaks in the target output vector around the harmonics of the ground-truth frequency, as depicted in Figure 13. When the weighted average over the all 360 bins is used, this ends up predicting a completely different pitch. To alleviate this, we have revised Equation 14 in the open-source version of CREPE to calculate the weighted average

[1] `https://marl.github.io/crepe`

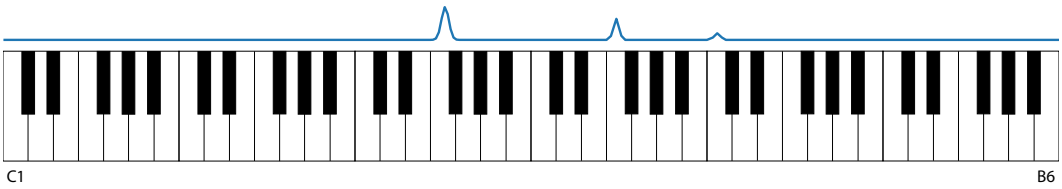


Figure 13: An example of an inaccurate target output where there exist multiple peaks around the harmonics of the ground-truth frequency. Equation 18 ensures that the predicted pitch is calculated using only the frequency bins around the highest peak.

using only the frequency bins around the highest peak in the output:

$$\hat{c} = \frac{\sum_{i=m-4}^{m+4} \hat{y}_i c_i}{\sum_{i=m-4}^{m+4} \hat{y}_i}, \qquad m = \arg\max_i \hat{y}_i. \tag{18}$$

Using Equation 18 improved the transcription accuracies of real-world signals significantly. However, a quantitative evaluation of the robustness in the real-world situations is inherently difficult and is out of the scope of this chapter, because it is virtually impossible to obtain a representative and objective dataset encompassing all such situations.

4.4 Data-Driven Models and Real-World Applications

The series of modifications on the CREPE model are taken in order to release the open-source software that are effective not only for evaluation in an academic setting but also in various real-world scenarios of pitch estimation. These give an important lesson that should be considered when developing data-driven models for music analysis tasks. Especially with the rise of deep learning models that typically have a large number of parameters, it is very easy to overfit a model to work well with a specific dataset while not being able to generalize. This problem can persist even if the dataset is properly splitted for training and evaluation,

because it is common for such dataset to have shared characteristics such as recording and mixing conditions that are not particularly representative of the broad range of the real-world data for which the model should supposedly work well. Therefore, a great amount of care should be taken when selecting the datasets to be used for evaluation of a music analysis model, and qualitative analyses on out-of-distribution real-world data need to be accompanied.

5 Conclusions

In this chapter, we presented a novel data-driven method for monophonic pitch tracking based on a deep convolutional neural network operating on time-domain input, CREPE. We showed that CREPE obtains state-of-the-art results, outperforming pYIN and SWIPE on two datasets with homogeneous and heterogeneous timbre respectively. Furthermore, we showed that CREPE remains highly accurate even at a very strict evaluation threshold of just 10 cents. We also showed that in most cases CREPE is more robust to added noise. In addition, we discussed the challenges that arose during the open-source release of the model, which included how to make the model more generalizable to the real-world sounds and the revised pitch extraction strategy to better deal with increased octave errors.

Ideally, we want the model to be invariant to all transformations that do not affect pitch, such as changes due to distortion and reverberation. Some invariance can be induced by the architectural design of the model, such as the translation invariance induced by pooling layers in our model as well as in deep image classification models. However, it is not as straightforward to design the model architecture to specifically ignore other pitch-preserving transformations.

While it is still an intriguing problem to build an architecture to achieve this, we could use data augmentation to generate transformed and degraded inputs that can effectively make the model learn the invariance. The robustness of the model could also be improved by applying pitch-shifts as data augmentation (McFee et al., 2015) to cover a wider pitch range for every instrument. In addition to data augmentation, various sources of audio timbre can be obtained from software instruments; NSynth (Engel et al., 2017) is an example where the training dataset is generated from the sound of software instruments.

Pitch values tend to be continuous over time, but CREPE estimates the pitch of every frame independently without using any temporal tracking, unlike pYIN which exploits this by using an HMM to enforce temporal smoothness. We can potentially improve the performance of CREPE further by statistically modeling how the pitch value changes over time. Employing recurrent neural networks (RNNs) is a natural choice for this, and we will explore in the following chapters how RNNs can model the temporal characteristics of music, focusing on two specific tasks: music synthesis (Chapter V) and piano transcription (Chapter VI).

CHAPTER V

LEARNING TIMBRE SPACE FOR MUSIC SYNTHESIS

In this chapter, we consider the problem of music synthesis. Although seemingly unconnected to music transcription, studying music synthesis models can provide an interesting perspective in understanding the music transcription problem: viewing synthesis as an inverse problem of analysis. One important outcome of this approach is that it allows us to computationally model the variations of musical timbre, a perceptual quality of sound that is less rigorously defined compared to other notions such as pitch and loudness. In light of this, we develop in this chapter a music synthesis model that can be used not only for synthesizing music, but also for better understanding the relation between audio signals and the qualities of musical sounds such as timbre and pitch.

The recent success of raw audio waveform synthesis models like WaveNet motivates a new approach for music synthesis, in which the entire process — creating audio samples from a score and instrument information — is modeled using generative neural networks. In this chapter, we develop a neural music synthesis model, which consists of a recurrent neural network conditioned on a learned instrument embedding followed by a WaveNet vocoder. The synthesis model is capable of learning a timbre embedding space that successfully captures the diverse variations in timbres within a large dataset. The model can therefore seamlessly connect the note sequence input and the timbre information to the audio signal, serving as the synthesizer component in Figure 2. The content of

this chapter is largely based on the work presented at IEEE ICASSP 2019 (J. W. Kim et al., 2019).

1 Introduction

Musical synthesis, most commonly, is the process of generating musical audio with given control parameters such as instrument type and note sequences over time. The primary difference between synthesis engines is the way in which *timbre* is modeled and controlled. In general, it is difficult to design a synthesizer that both has dynamic and intuitive timbre control and is able to span a wide range of timbres; most synthesizers change timbres by having presets for different instrument classes or have a very limited space of timbre transformations available for a single instrument type.

In this chapter, we present a flexible music synthesizer named Mel2Mel, which uses a learned, non-linear instrument embedding as timbre control parameters in conjunction with a learned synthesis engine based on WaveNet. Because the model has to learn the timbre – any information not specified in the note sequence – to successfully reconstruct the audio, the embedding space spans over the various aspects of timbre such as spectral and temporal envelopes of notes. This learned synthesis engine allows for flexible timbre control, and in particular, timbre morphing between instruments, as demonstrated in our interactive web demo.

2 Background

2.1 Timbre Control in Musical Synthesis

Methods for music synthesis are based on a variety of techniques such as FM synthesis, subtractive synthesis, physical modeling, sample-based synthesis, and granular synthesis (Pejrolo & Metcalfe, 2017). The method of controlling timbre and the level of flexibility depends on the parameters of the exact method used, but in general, there is a trade-off between flexible timbre control over synthetic sounds (e.g. FM or subtractive synthesis) and a limited timbre control in more "realistic" sounds (e.g. sample-based, physical model-based, or granular synthesis). Our work is aimed at achieving the best of both worlds: flexibly controlling a variety of realistic-sounding timbres.

2.2 Timbre Morphing

'Morphing' of a sound can be generally described as making a perceptually gradual transition between two or more sounds (Caetano & Rodet, 2010). A common approach is to use a synthesis model and define sound morphing as a numerical interpolation of the model parameters. Sinusoidal models can directly interpolate between the energy proportions of the partials (Osaka, 1995; Boccardi & Drioli, 2001). Other models use parameters characterizing the spectral envelope (Slaney et al., 1996; Ezzat et al., 2005) or psychoacoustic features for perceptually linear transition (Caetano & Rodet, 2013). A limitation of these approaches is that morphing can only be applied among the range of timbres covered by a certain synthesis model, whose expressiveness or parameter set may be limited. To

overcome this, we employ a data-driven approach for music synthesis that is generalizable to all timbres in the dataset.

2.3 Timbre Spaces and Embeddings

Timbre is often modeled using a timbre space (Peeters et al., 2011), in which similar timbres lie closer than dissimilar timbres. In early work in psychoacoustics, multidimensional scaling (MDS) was used to obtain a timbre space which preserves the timbral dissimilarities measured in perceptual experiments (Grey, 1977; Wessel, 1979). Meanwhile, in music content analysis, timbre similarity is measured using computed features such as the Mel-frequency cepstral coefficients (MFCCs) (Logan, 2000), descriptors of the spectral envelope (A. Agostini & Ghisi, 2013), or hidden-layer weights of a neural network trained to distinguish different timbres (Humphrey et al., 2011). A recent method (Esling et al., 2018) used a variational autoencoder (Kingma & Welling, 2014) to obtain a timbre space, and unlike the above embeddings, the method is able to generate monophonic audio for a particular timbre embedding but does not consider the temporal evolution of notes such as attacks and decays. In our work, we generate a timbre embedding as a byproduct of polyphonic synthesis, which can utilize both spectral and temporal aspects of timbre.

2.4 Neural Audio Synthesis using WaveNet

WaveNet (van den Oord, Dieleman, et al., 2016) is a generative audio synthesis model that is able to produce realistic human speech. WaveNet achieves this by learning an autoregressive distribution which predicts the next audio sample from the previous samples in its receptive field using a series of dilated convolutions.

Tacotron (Shen et al., 2018) and Deep Voice (Ping et al., 2018) are WaveNet-based text-to-speech models which first predict a Mel spectrogram from text and use it to condition a WaveNet vocoder.

There are also a few notable applications of WaveNet in music, including NSynth (Engel et al., 2017), an autoencoder architecture which separately encodes monophonic pitch with learned timbral features, and the universal music translation network (Mor et al., 2019) which uses a denoising autoencoder architecture that can extract and translate between musical styles while preserving the melody. In Hawthorne et al. (2019), a WaveNet is used for music synthesis conditioned directly on note sequences but only supports piano sounds. Our model is similarly built around WaveNet for its synthesis capability but uses a learned embedding space to flexibly control the timbre of polyphonic music.

3 Method

The neural network shown in Figure 14, dubbed Mel2Mel, concerns the task of synthesizing music corresponding to given note sequences and timbre. The note sequences are supplied as a MIDI file and converted to a piano roll representation, which contains the note timings and the corresponding note velocities for each of the 88 piano keys. We use a fixed step size in time, and the piano roll representation is encoded as a matrix by quantizing the note timings to the nearest time step. The input to the neural network is a concatenation of two 88-dimensional piano roll representations, one for onsets and one for frames, comprising 176 dimensions in total:

$$
\begin{aligned}
\mathbf{X} &= [\,\mathbf{X}^{\text{onset}}\,;\,\mathbf{X}^{\text{frame}}\,] \\
\mathbf{X}^{\text{onset}}_{p,t} &= v_{\text{the active note}} \cdot \mathbb{1}_{\text{a note at pitch } p \text{ is } \textit{first} \text{ active at time } t} \\
\mathbf{X}^{\text{frame}}_{p,t} &= v_{\text{the active note}} \cdot \mathbb{1}_{\text{a note at pitch } p \text{ is active at time } t}
\end{aligned}
$$

where $\mathbb{1}$ is the indicator function, and v denotes the MIDI velocity scaled to $[0, 1]$. This input representation is inspired by (Hawthorne et al., 2018) which showed that jointly training on onsets and frames performs better than using frame information only; similarly, we want the network to maximally utilize the onsets which have the most relevant information on the attack sounds, while still receiving the frame information. Another reason for using both onsets and frames is that, because of the time quantization, repeated notes become indistinguishable only using $\mathbf{X}^{\text{frame}}$ when an offset is too close to the subsequent onset.

The input goes through a linear 1x1 convolution layer, which is essentially a time-distributed fully connected layer, followed by a FiLM layer, to be described in the following subsection, which takes the timbre embedding vector and transforms the features accordingly. After a bidirectional LSTM layer and another FiLM layer for timbre conditioning, another linear 1x1 convolution layer produces the Mel spectrogram prediction. The resulting Mel spectrogram is then fed to a separately trained WaveNet vocoder to produce the music; Mel spectrograms compactly convey sufficient information for audio synthesis and have been successfully used for conditioning WaveNet (Shen et al., 2018; Ping et al., 2018). The use of bidirectional LSTM is appropriate because Mel spectrograms are constructed using a larger window than the step size, making it non-causal. The only nonlinearities

in the network are in the LSTM, and there are no time-domain convolutions except in WaveNet.

3.1 Timbre Conditioning using FiLM Layers

A FiLM layer (Perez et al., 2017) is a neural network that can take side information; it first learns functions f and h, which are linear layers mapping the timbre embedding $\mathbf{t}$ to $\gamma = f(\mathbf{t})$ and $\beta = h(\mathbf{t})$. The affine transformation, or FiLM-ing, of intermediate-layer features $\mathbf{F}$ is then applied using feature-wise operations:

$$\text{FiLM}(\mathbf{F} \mid \gamma, \beta) = \gamma\mathbf{F} + \beta = f(\mathbf{t})\mathbf{F} + h(\mathbf{t}). \tag{19}$$

We can think of the model architecture as a multi-step process that shapes a piano roll into its Mel spectrogram, by applying appropriate timbre given as side information. A FiLM layer is a suitable choice for this task, because it can represent such action of shaping using an affine transformation of intermediate-layer features. For each instrument to model, its timbre is represented in an embedding vector $\mathbf{t}$, implemented as a learned matrix multiplication on one-hot encoded instrument labels. The values of instrument embedding vectors do not vary in time and are learned jointly with the rest of the model.

At a higher level, the affine transformations learned by the FiLM layers are nonlinearly transformed by the recurrent and convolutional layers to respectively form temporal and spectral envelopes, which are two important aspects that characterize instrumental timbre. Using the first FiLM layer is essential because the recurrent layer needs to take timbre-dependent input to apply the temporal dynamics according to the timbre, and the second FiLM layer can apply additional spectral envelope on the recurrent layer's output.

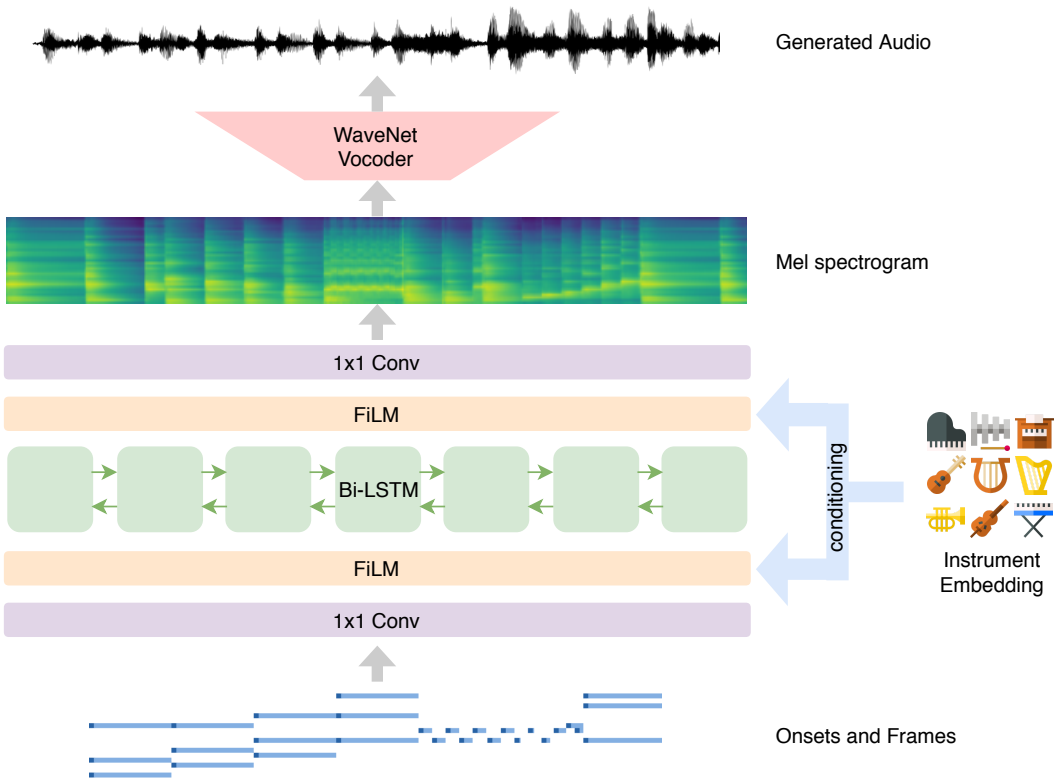


Figure 14: The overall architecture of the proposed Mel2Mel model. The note sequences and instrument embeddings are combined to predict the Mel spectrogram, which is then fed to the WaveNet vocoder.

3.2 Model Details

The sampling rate of 16 kHz and μ-law encoding with 256 quantization levels are used for all audio as in the original WaveNet paper (van den Oord, Dieleman, et al., 2016). The predicted Mel spectrograms are defined using 80 area-normalized triangular filters distributed evenly between zero and the Nyquist frequency in the Mel scale. The STFT window length of 1,024 samples and the step size of 128 samples are used, which translate to 64 milliseconds and 8 milliseconds, respectively. Unless specified otherwise, we use 256 channels in all hidden layers and a two-dimensional embedding space for timbre conditioning.

For Mel spectrogram prediction, an Adam optimizer with the initial learning

rate of 0.002 is used, and the learning rate is halved every 40,000 iterations. The model is trained for 100,000 iterations, where each iteration takes a mini-batch of 128 sequences of length 65,536, or 4.096 seconds. Three different loss functions are used and compared; for linear-scale Mel spectrograms S_{true} and S_{pred}:

$$\textbf{abs MSE} = \mathbb{E}\left[\left(S_{\text{true}} - S_{\text{pred}}\right)^2\right] \tag{20}$$

$$\textbf{log-abs MSE} = \mathbb{E}\left[\left(\log S_{\text{true}} - \log S_{\text{pred}}\right)^2\right] \tag{21}$$

$$\textbf{tanh-log-abs MSE} = \mathbb{E}\left[\left(\tanh \tfrac{1}{4} \log S_{\text{true}} - \tanh \tfrac{1}{4} \log S_{\text{pred}}\right)^2\right] \tag{22}$$

All logarithms above are natural, and the spectrogram magnitudes are clipped at -100 dB. Prepending tanh gives a soft-thresholding effect where the errors in the low-energy ranges are penalized less than the errors close to 0 dB.

For the WaveNet vocoder, we used nv-wavenet[1], a real-time open-source implementation of autoregressive WaveNet by NVIDIA. This implementation limits the recurrent channel size at 64 and the skip channels at 256, because of the GPU memory capacity. A 20-layer WaveNet model was trained with the maximum dilation of 512, and the Mel spectrogram input is upsampled using two transposed convolution layers of window sizes 16 and 32 with strides of 8 and 16, respectively. An Adam optimizer with the initial learning rate of 0.001 is used, and the learning rate is halved every 100,000 iterations, for one million iterations in total. Each iteration takes a mini-batch of 4 sequences of length 16,384, i.e. 1.024 seconds.

[1] `https://github.com/NVIDIA/nv-wavenet`

4 Experiments

4.1 Datasets

While it is ideal to use recorded audio of real instruments as the training dataset, the largest multi-instrument polyphonic datasets available such as MusicNet (Thickstun et al., 2017) is highly skewed, contains a limited variety of solo instrument recordings, and is expected to have a certain degree of labeling errors as the authors suggest. Instead, we used synthesized audio for training and collected MIDI files from `www.piano-midi.de`, which are also used in the MAPS Database (Emiya et al., 2010); these MIDI files are recorded from actual performances and contain expressive timing and velocity information. We have selected 10 General MIDI instruments shown in Figure 16 covering a wide variety of timbres, and 334 piano tracks are synthesized for each instrument using FluidSynth with the default SoundFont from MuseScore 3. The 334 tracks are randomly split into 320 for training and 14 for validation. The total size of the synthesized dataset is 3,340 tracks and 221 hours.

For later experiments, we also generate a similar dataset using 100 manually selected instrument classes using a high-quality collection of SoundFonts, which contains a wide variety of timbres.

4.2 Ablation Study on Model Design

In this series of experiments, we examine how slight variations in the model architecture affect the performance and show that the proposed model achieves the best performance in accurately predicting Mel spectrograms. The first two variations use either the frame data or the onset data only as the input. The

Variations	Train loss ($\times 10^3$)	Validation loss ($\times 10^3$)
Proposed	4.09 ± 0.30	$\mathbf{4.75 \pm 0.05}$
Frame input only	5.58 ± 0.32	5.92 ± 0.08
Onset input only	5.88 ± 0.37	6.97 ± 0.06
First FiLM only	4.55 ± 0.32	4.99 ± 0.06
Second FiLM only	7.65 ± 0.34	8.76 ± 0.08
Forward LSTM only	5.70 ± 0.43	5.56 ± 0.09
ReLU activation	3.97 ± 0.35	5.04 ± 0.06
3x1 convolutions	3.66 ± 0.28	5.12 ± 0.08
5x1 convolutions	3.49 ± 0.30	5.06 ± 0.08
2-layer LSTM	2.98 ± 0.20	4.96 ± 0.12

Table 3: Train and validation losses as defined in Equation 22 with respect to different variations on the architecture, either limiting or increasing the model capacity. The proposed architecture has the lowest validation loss, indicating that it predicts the most accurate Mel spectrograms while not being prone to overfitting.

next three omit an architectural component: one of the two FiLM layers or the backward LSTM. The last four increase the network's capacity by adding the ReLU nonlinearity after the first convolution, using kernel sizes of 3 or 5 time steps in convolutions, or adding another LSTM layer. The train and validation losses as defined in Equation 22 are shown[1] in the table above for each variation. Using both onsets and frames is indeed more effective than using only one of them in the input. The first FiLM layer plays a more crucial role than the second, because only the first can help learn a timbre-dependent recurrent layer. As expected, removing the backward LSTM also hurt the performance.

On the other hand, any variations increasing the model capacity make the model overfit and fail to generalize to validation data. This implies that the proposed model has the optimal architecture among the tested variations, and

[1] The means and standard deviations over the model checkpoints in 90k-100k iterations are reported, to minimize the variability due to SGD.

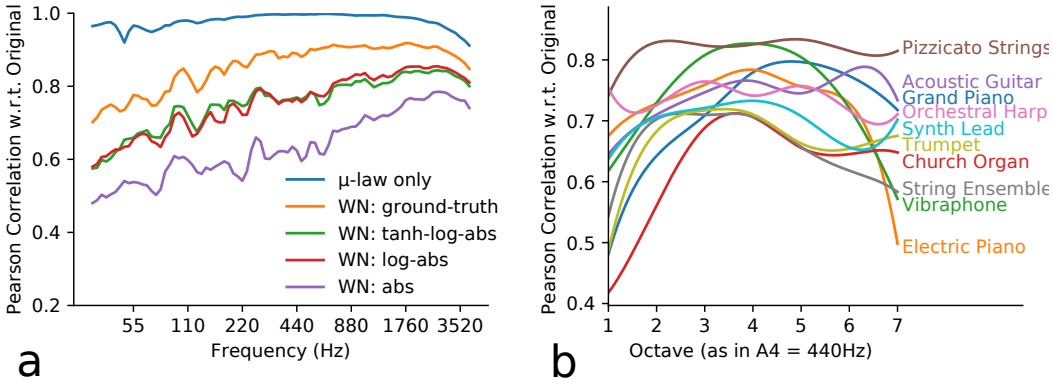


Figure 15: Pearson correlations between the reconstructed and original audio. (a) each stage of degradation. (b) per-instrument breakdown of the green curve on the left. The curves are smoothed and drawn with respect to each octave for readability.

more specifically, having the nonlinearity only in the single recurrent layer helps the model better generalize in predicting Mel spectrograms from unseen note sequences. A possible interpretation is that the increased capacity is being used for memorizing the note sequences in the training dataset, as opposed to learning to model the timbral features independent of specific notes.

4.3 Synthesis Quality

4.3.1 Numerical Analysis of Audio Degradation

The model goes through several stages of prediction, and each stage incurs a degradation of audio quality. There necessarily exists some degradation caused by the μ-law quantization, and WaveNet adds additional degradation due to its limited model capacity. The generated audio is further degraded when imperfect Mel spectrogram predictions are given. As an objective measure of audio quality degradation at each stage and for each instrument, we plot the Pearson correlations between the synthesized and original audio in Figure 15. To calculate and visualize the correlations with respect to evenly spaced octaves, we use 84-bin log-magnitude

constant-Q transforms with 12 bins per octave starting from C1 (≈ 32.70Hz) and 512-sample steps. For ideal synthesis, the Pearson correlation should be close to 1, and lower correlations indicate larger degradation from the original.

Figure 15a shows the correlations for each stage of degradation and for different loss functions used for training the model. The degradations are more severe in low frequencies in general, where the WaveNet model sees fewer periods of a note within its fixed receptive field length. The orange curve showing the correlations for WaveNet synthesis using ground-truth Mel spectrograms already exhibits a significant drop from the top curve; this defines an upper bound of Mel2Mel's synthesis quality. The lower three curves correspond to the loss functions in Equations 20-22, among which the abs MSE loss clearly performs the worse than the other two which have almost identical Pearson correlation curve, indicating that the MSE loss is more effective in the log-magnitude scale.

Figure 15b shows the breakdown of the curve corresponding to Equation 22 into each of the 10 instruments. There are rather drastic differences among instruments, and most instruments have low Pearson correlations in low pitches except pizzicato strings. The implications of these trends are discussed in (J. W. Kim et al., 2019), in comparison with the subjective audio quality test.

4.4 The Timbre Embedding Space

To make sense of how the learned embedding space conveys timbre information, we construct a 320-by-320 grid that encloses all instrument embeddings and predict the Mel spectrogram conditioned on every pixel in the grid. The spectral centroid and the mean energy corresponding to each pixel are plotted in Figure 16, which are indicative of the two main aspects of instrumental timbres: the spectral and

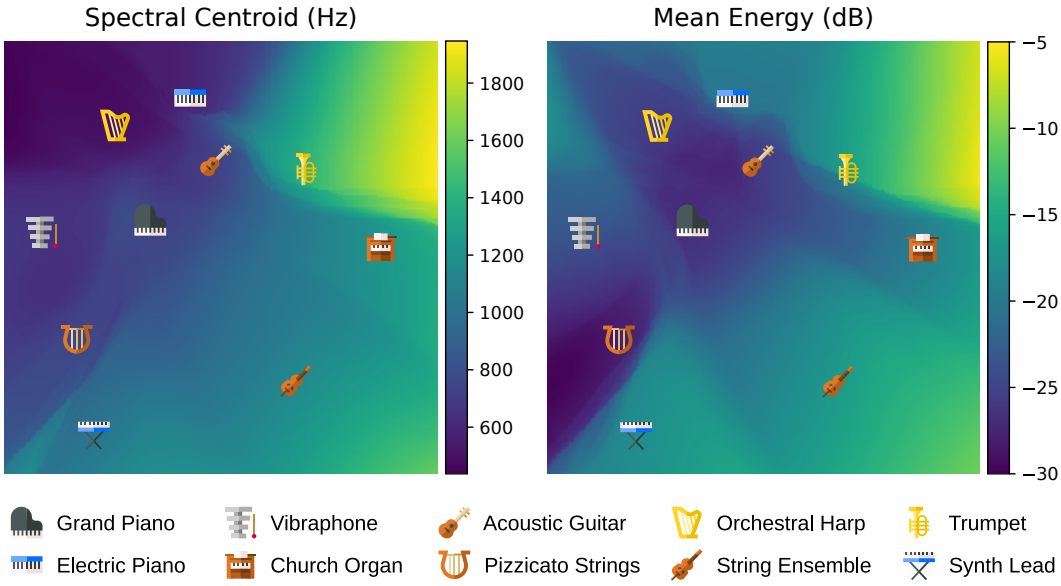


Figure 16: Visualization of the embedding space trained in 2 dimensions, using spectral centroids and mean energy. The continuous colors are obtained for each pixel in the 320-by-320 grid and are related to the spectral and temporal envelopes of the timbres.

temporal envelopes. For a fixed f0, a higher spectral centroid signifies stronger harmonic partials in high frequency, while a lower spectral centroid indicates that it is closer to a pure sine tone. Similarly, higher mean energy implies a more sustained tone, and low mean energy means that the note is transient and decays rather quickly. The points corresponding to the 10 instruments are annotated with instrument icons.[1] These plots show that the learned embedding space forms a continuous span over the timbres expressed by all instruments in the training data. This allows us to use the timbre embedding as a flexible control space for the synthesizer, and timbre morphing is possible by interpolating along curves within the embedding space.

To illustrate how the model scales with more diverse timbre, we train the Mel2Mel model with 100 instruments using a 10-dimensional embedding, and we

[1] The icons are made by Freepik and licensed by CC 3.0 BY.

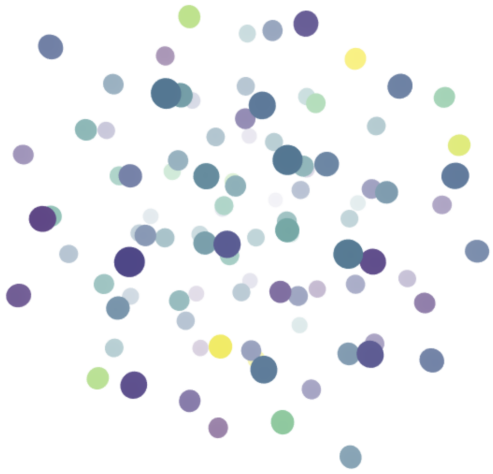


Figure 17: A t-SNE visualization of the 10-dimensional timbre embedding space learned using 100 instruments, color-coded according to the spectral centroid. In the web demo, users can rotate the dots to navigate and click on the dots to play the corresponding audio segments.

refer the readers to the web demo[1] for an interactive t-SNE visualization (van der Maaten & Hinton, 2008) of the embedding space. Figure 17 shows a screenshot of the web demo. The 10-dimensional embedding space also contains a locally continuous timbre distribution of instruments, as in Figure 16, implying that the Mel2Mel model is capable of scaling to hundreds of instruments and to a higher-dimensional embedding space.

In addition to the audio samples used in the experiments, the interactive web demo showcases the capability of flexible timbre control, where the Mel2Mel model runs on browser to convert preloaded or user-provided MIDI files into Mel spectrograms using a user-selected point in the embedding space.

[1] `https://neural-music-synthesis.github.io`

5 Conclusions and Future Directions

We showed that it is possible to build a music synthesis model by combining a recurrent neural network and FiLM conditioning layers, followed by a WaveNet vocoder. It successfully learns to synthesize musical notes according to the given note sequence and timbre embedding in a continuous timbre space, providing the ability of flexible timbre control for music synthesizers.

The capacity of the WaveNet, such as the number of residual channels and the number of layers, is limited due to the memory requirements of the nv-wavenet implementation, and the degradation from μ-law quantization is also apparent in the experiments. These limitations can be overcome by Parallel WaveNet (van den Oord et al., 2018) or WaveGlow (Prenger et al., 2019), which does not require a specialized CUDA kernel for fast synthesis and uses a continuous probability distribution for generation, thereby avoiding the quantization noise. Our earlier experiments on continuous emission failed to stably perform autoregressive sampling due to teacher forcing, and the future work includes investigating this phenomenon comparing with (Hawthorne et al., 2019), which used a mixture of logistics distributions to produce high-quality piano sounds.

A notable observation is that the WaveNet vocoder is able to synthesize plausible polyphonic music from Mel spectrograms containing only 80 frequency bins, which are not even aligned to the tuning of the audio files. This implies that Mel spectrograms contain a close-to-sufficient amount of information for what we perceive from polyphonic music. While more information available from the increased bins should help synthesize more accurate audio, predicting the higher-dimensional representation becomes more compute-intensive and inaccurate, making 80 bins a sweet spot for use with WaveNet. Introducing

an adversarial loss function for predicting high-resolution images (Ledig et al., 2017) can be a viable direction for predicting more accurate and realistic Mel spectrograms for conditioning WaveNet.

To summarize, we have demonstrated in this chapter that a MIDI-to-audio synthesizer can be learned directly from audio, and that this learning allows for flexible timbre control through interpolation in the learned timbre embedding space. While it is certainly a possible future research direction to improve the synthesis quality through the usage of a better vocoder component and a larger and more realistic dataset, the main takeaway of having the synthesis model in the context of this thesis is that it is an end-to-end model that can convert semantic information of music into audio signals. This end-to-end approach not only allows for a simpler and streamlined process of synthesizing music, but also enables using the synthesis model as a differentiable component in a larger neural network. We will examine this possibility in Chapter VII later in this thesis, by combining a music synthesis model with a multi-pitch transcription model introduced in Chapter VI.

CHAPTER VI

ADVERSARIAL LEARNING FOR PIANO TRANSCRIPTION

In the previous chapter, we have introduced a WaveNet-based music synthesis model that is conditioned on predicted Mel spectrograms. A Mel spectrogram is a representation of audio whose dimension is in the order of hundreds at each frame. Accurate prediction of such high-dimensional representations requires sophisticated modeling of their statistical properties as well as extensive computational power to realize it. Piano roll representations of music, which are formulated in Chapter I as the output representation of polyphonic music transcription throughout this thesis, are another example of high-dimensional representations of audio which contain hundreds of data points at each instant. Therefore, unlike the single-valued objective in monophonic pitch estimation tasks, prediction of piano roll representations is inherently a multi-label problem and thus warrants a mathematical model that is right for high-dimensional representation.

In this chapter, as hinted in the conclusions in the last chapter, we discuss a method to include adversarial loss to allow the model to predict the piano roll target more accurately. We first address a limitation in the conventional, element-wise definition of loss functions in which the inter-label probabilistic dependencies are not accurately modeled. Based on this observation, we show that appending an adversarial discriminator to a discriminative piano transcription model can help producing more confident predictions which in turn improve the transcription

accuracy. The content of this chapter is largely based on the work presented at ISMIR 2019 (J. W. Kim & Bello, 2019).

1 Introduction

Automatic music transcription (AMT) is a multifaceted problem and comprises a number of subtasks, including multi-pitch estimation (MPE), note tracking, instrument recognition, rhythm analysis, score typesetting, etc. MPE predicts a set of concurrent pitches that are present at each instant, and it is closely related to the task of note tracking, which predicts the onset and offset timings of every note in audio. In this chapter, we address an issue in the recent approaches for MPE and note tracking, where the probabilistic dependencies between the labels are often overlooked.

A common approach for MPE and note tracking is through the prediction of a two-dimensional representation that is defined along the time and frequency axes and contains the pitch tracks of notes over time. Piano rolls are the most common example of such representations, and deep salience (Bittner et al., 2017) is another example that can contain more granular information on pitch contours. Once such representation is obtained, pitches and notes can be decoded by thresholding (Kelz et al., 2016) or other heuristic methods (J. W. Kim et al., 2018; Hawthorne et al., 2018).

To train a model that predicts a two-dimensional target representation $\hat{\mathbf{Y}} \in \mathbb{R}^{P \times T}$ from an input audio representation $\mathbf{X}$, where P is the number of pitch labels and T is the number of time frames, a common approach is to minimize the

element-wise sum of a loss function $\mathcal{L}$:

$$\text{minimize } \mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y}) = \sum_{p=1}^{P} \sum_{t=1}^{T} \mathcal{L}(\hat{\mathbf{Y}}_{pt}, \mathbf{Y}_{pt}), \tag{23}$$

where $\mathbf{Y} \in \mathbb{R}^{P \times T}$ is the ground truth. In a probabilistic perspective, we can interpret $\mathcal{L}$ as the negative log-likelihood of the model parameters ϑ of a discriminative model $p_\vartheta(\mathbf{Y}|\mathbf{X})$:

$$p_\vartheta(\mathbf{Y}|\mathbf{X}) = e^{-\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})} = \prod_{p=1}^{P} \prod_{t=1}^{T} e^{-\mathcal{L}(\hat{\mathbf{Y}}_{pt}, \mathbf{Y}_{pt})} = \prod_{p=1}^{P} \prod_{t=1}^{T} p_\vartheta(\mathbf{Y}_{pt}|\mathbf{X}) \tag{24}$$

which indicates that each element of the label $\mathbf{Y}$ is conditionally independent with each other given the input $\mathbf{X}$. This encourages the model to predict the average of the ground-truth distribution, making blurry predictions when the target is multimodal, e.g. natural images (Dosovitskiy & Brox, 2016).

Music data is highly contextual and multimodal, and the conditional independence assumption does not hold in general. This is why many computational music analysis models employ a separate post-processing stage after sequence prediction. One approach is to factorize the joint probability using the chain rule and assume the Markov property:

$$p_\vartheta(\mathbf{Y}|\mathbf{X}) \approx \prod_{p=1}^{P} \prod_{t=1}^{T} p_\vartheta(\mathbf{Y}_{pt}|\mathbf{Y}_{\cdot(t-1)}, \mathbf{X}). \tag{25}$$

This corresponds to appending hidden Markov models (HMMs) (Poliner & Ellis, 2006) or recurrent neural networks (RNNs) (Sigtia et al., 2016; Hawthorne et al., 2018) to the transcription model. The Markov assumption is effective for one-dimensional sequence prediction tasks, such as chord estimation (Ni

et al., 2012) and monophonic pitch tracking (Mauch & Dixon, 2014), but when predicting a two-dimensional representation, it still does not address the inter-label dependencies along the frequency axis.

There exist a number of models in the computer vision literature that can express inter-label dependencies in two-dimensional predictions, such as the neural autoregressive distribution estimator (NADE) (Larochelle & Murray, 2011), PixelRNN (van den Oord, Kalchbrenner, & Kavukcuoglu, 2016), and PixelCNN (van den Oord, Kalchbrenner, Vinyals, et al., 2016). However, apart from a notable exception using a hybrid RNN-NADE approach (Boulanger-Lewandowski et al., 2012), the effect of learning the joint multi-label distribution for polyphonic music transcription has not been well studied.

To this end, we propose a new approach for effectively leveraging inter-label dependencies in polyphonic music transcription. We pose the problem as an image translation task and apply an adversarial loss incurred by a discriminator network attached to the baseline model. We show that our approach can consistently and significantly reduce the transcription errors in *Onsets and Frames* (Hawthorne et al., 2018), a state-of-the-art music transcription model.

2 Background

2.1 Automatic Transcription of Polyphonic Music

Automatic transcription models for polyphonic music can be classified into frame- or note-level approaches. Frame-level transcription is synonymous with multi-pitch estimation (MPE) and operates on tiny temporal slices of audio, or frames, to predict all pitch values present in each frame. Note-level transcription, or note tracking, operates at a higher level, predicting a sequence of note events

that contains the pitch, the onset time, and optionally the offset time of each note. Note tracking is typically implemented as a post-processing stage on the output of MPE (Benetos et al., 2019), by connecting and grouping the pitch estimates over time to produce discrete note events. In this sense, we can say that MPE is at the core of polyphonic music transcription.

Among the approaches for MPE reviewed in Section II.3, two categories have been most successful in recent years: matrix factorization (Lee & Seung, 2001) and deep learning (LeCun et al., 2015). Commonly in both of these approaches, an iterative gradient-descent algorithm is used to minimize an element-wise loss function that is defined on two-dimensional representation. NMF-based methods are designed to minimize a divergence function between the matrix factorization and the target matrix, and similarly in deep learning models, neural networks are optimized to predict a two-dimensional representation that makes an element-wise loss function as small as possible.

In this chapter, we use Onsets and Frames (Hawthorne et al., 2018), a state-of-the-art piano transcription model based on deep learning, as our baseline. It uses multiple columns of convolutional and recurrent neural network layers to predict onsets, offsets, velocities, and frame labels from the Mel spectrogram input, as shown in Figure 18. Predicted onset and frame activations are then used for decoding the note sequences, where a threshold value is used to create binary onset and frame activations, and frame activations without the corresponding onsets are disregarded. Onsets and Frames also uses an element-wise optimization objective which does not consider the inter-label dependencies. This motivates the adversarial training scheme that is outlined in the following subsection.

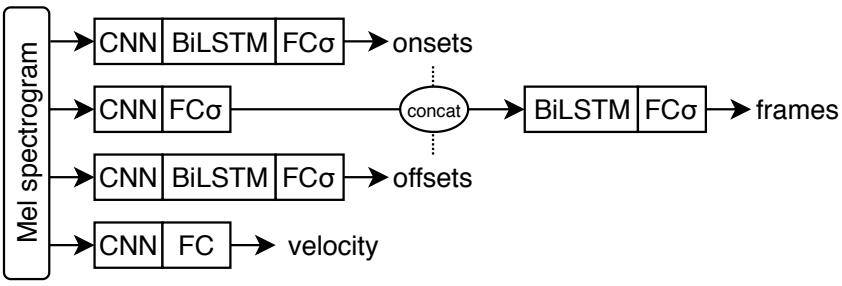


Figure 18: The Onset and Frames model. CNN denotes the convolutional acoustic model taken from (Kelz et al., 2016), FC denotes a fully connected layer, and σ denotes sigmoid activation. Dotted lines mean stop-gradient, i.e. no backpropagation.

2.2 Generative Adversarial Networks and `pix2pix`

As extensively reviewed in Section III.4, generative adversarial networks (GANs) (Goodfellow et al., 2014) refer to a family of deep generative models which consist of two components, namely the generator G and the discriminator D. Given a data distribution $\mathbf{x} \sim p(\mathbf{x})$ and latent codes $\mathbf{z} \sim p(\mathbf{z})$, GAN performs the following minimax game:

$$\min_G \max_D \underbrace{\mathbb{E}_{\mathbf{x}} \log D(\mathbf{x}) + \mathbb{E}_{\mathbf{z}} \log(1 - D(G(\mathbf{z})))}_{\mathcal{L}_{\text{GAN}}(G,D)} . \tag{26}$$

G and D are implemented as neural networks trained in an adversarial manner, where the discriminator learns to distinguish the generated samples from the real data, while the generator learns to produce realistic samples to fool the discriminator. GANs are most renowned for their ability to produce photorealistic images (Karras et al., 2019) and have shown promising results on music generation as well (Engel et al., 2019; Dong et al., 2017; Yang et al., 2017). We refer the readers to (Goodfellow, 2016; Creswell et al., 2017) for a comprehensive review of the techniques, variants, and applications of GANs.

The second term in Equation 26 has near-zero gradients when $D(G(\mathbf{z})) \approx 0$,

which is usually the case in early training. To avoid this, a *non-saturating* variant of GAN is suggested in (Goodfellow et al., 2014) where the generator is trained with the following optimization objective instead:

$$\max_G \ \mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z})). \tag{27}$$

The non-saturating GAN loss is used more often than the minimax loss in Equation 26 and is implemented by flipping the labels of fake data while using the same loss function. *Least-squares GAN* (Mao et al., 2017) is an alternative method to address the vanishing gradient problem, which replaces the cross entropy loss in Equations 26-27 with squared errors:

$$\begin{aligned} &\min_D \quad \mathbb{E}_{\mathbf{x}}(D(\mathbf{x}) - 1)^2 + \mathbb{E}_{\mathbf{z}} D(G(\mathbf{z}))^2 \ , \\ &\min_G \quad \mathbb{E}_{\mathbf{z}}(D(G(\mathbf{z})) - 1)^2. \end{aligned} \tag{28}$$

The non-saturating GAN (NSGAN) and least-squares GAN (LSGAN) losses are examples of GAN losses that define the objective used during the minimax optimization, and we will use these two losses in conjunction with the conditional GAN setup described below.

While the default formulation of GAN concerns unconditional generation of samples from $p(\mathbf{x})$, conditional GANs (cGAN) (Mirza & Osindero, 2014) produce samples from a conditional distribution $p(\mathbf{y}|\mathbf{x})$. To do this, the generator and the discriminator are defined in terms of the condition variable $\mathbf{x}$ as well:

$$\min_G \max_D \ \underbrace{\mathbb{E}_{\mathbf{x},\mathbf{y}} \log D(\mathbf{x},\ \mathbf{y}) + \mathbb{E}_{\mathbf{x},\mathbf{z}} \log(1 - D(\mathbf{x},\ G(\mathbf{x},\ \mathbf{z}))}_{\mathcal{L}_{\text{cGAN}}(G,D)} \ . \tag{29}$$

`pix2pix` (Isola et al., 2017) is an image translation model that learns a mapping between two distinct domains of images, such as aerial photos and maps. A `pix2pix` model takes paired images $(\mathbf{x}, \mathbf{y})$ as its training data and minimizes the conditional GAN loss along with an additional L1 loss:

$$\mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{z}} \left\| \mathbf{y} - G(\mathbf{x}, \mathbf{z}) \right\|_1, \tag{30}$$

which encourages the conditional generator to learn the forward mapping from $\mathbf{x}$ to $\mathbf{y}$. It can be thought that the GAN loss in Equation 29 is fine-tuning the mapping learned by the L1 loss in Equation 30, resulting in a predictive mapping that better respects the probabilistic dependencies within the labels $\mathbf{y}$.

We adapt this approach to music transcription tasks and show that we can indeed improve the performance by introducing an adversarial loss to an existing music transcription model.

3 Method

We describe a general method for improving an NN-based transcription model G that performs prediction of a two-dimensional target $\mathbf{Y}$ from an input audio representation $\mathbf{X}$. Say the original model G is trained by minimizing the loss $\mathcal{L}_{\text{task}}(G(\mathbf{X}), \mathbf{Y})$ between the predicted target $\hat{\mathbf{Y}} = G(\mathbf{X})$ and the ground-truth $\mathbf{Y}$. The main idea of our method is to adapt `pix2pix` (Isola et al., 2017) to this setup, by introducing an adversarial discriminator D during the training process. The adversarial training objective includes the conditional GAN loss $\mathcal{L}_{\text{cGAN}}$ (Equation 29):

$$\min_G \max_D \mathbb{E}_{\mathbf{X},\mathbf{Y}} \ \nu \mathcal{L}_{\text{task}}(G(\mathbf{X}), \mathbf{Y}) + \mathcal{L}_{\text{cGAN}}(G, D) \ , \tag{31}$$

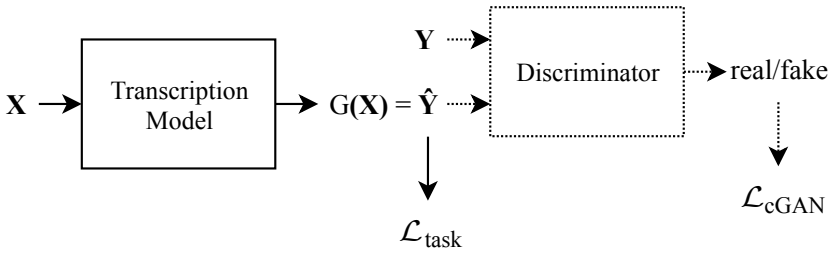


Figure 19: A computation graph showing how a discriminator is appended to the original model. The appended parts are shown as dotted components.

where ν is a hyperparameter that controls how much the conditional GAN loss contributes to the gradient steps relative to the discriminative loss $\mathcal{L}_{\text{task}}$. Figure 19 illustrates how the two components are connected in the computation graph and how the loss terms are calculated.

Adversarial training with $\mathcal{L}_{\text{cGAN}}$ allows the model to learn the inter-label dependencies as desired, even when $\mathcal{L}_{\text{task}}$ is defined only in terms of element-wise operations between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$, as in Equation 23. In the next subsection, we describe a neural network architecture for the cGAN discriminator that leverages prior knowledge on music.

3.1 Musically Inspired Adversarial Discriminator

Following `pix2pix`, we use a fully convolutional architecture (Long et al., 2018) for the discriminator. By being fully convolutional, the discriminator has translation invariance not only along the time axis (as in HMMs and RNNs) but also along the frequency axis. Since the discriminator determines how realistic a polyphonic note sequence is, the translation invariance enforces that the decision does not depend on the musical key, but only on the relative pitch and time intervals between the notes. This effectively implements a music language model

(MLM) (Boulanger-Lewandowski et al., 2012; Sigtia et al., 2016) and biases the transcription toward more realistic note sequences.

Unlike the image-to-image translation problem, the input representations (e.g. Mel spectrograms) and the output representations (e.g. piano rolls) of a music transcription model can have different dimensions. This makes combining $\mathbf{X}$ and $\mathbf{Y}$ in a fully convolutional manner difficult. For this reason, we make the discriminator a function of $\mathbf{Y}$ only, simplifying the objective in Equation 29 to:

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{\mathbf{Y}} \log D(\mathbf{Y}) + \mathbb{E}_{\mathbf{X}} \log(1 - D(G(\mathbf{X}))). \tag{32}$$

Note that $\mathbf{z}$ is also omitted in Equation 32, as we follow Isola et al. (2017) and implement the stochasticity of $\mathbf{z}$ only in terms of dropout layers (Srivastava et al., 2014), without explicitly feeding random noises into the generator. This causes a mode collapse problem where the learned $p(\mathbf{Y}|\mathbf{X})$ is not diverse enough, but it does not harm our purpose of producing more realistic target representations.

3.2 TTUR and *mixup* to Stabilize GAN Training

Although an ideal GAN generator can fully reconstruct the data distribution at the global optimum (Goodfellow et al., 2014), training of GANs in practice is notoriously difficult, especially for high-dimensional data (Goodfellow, 2016). This led to the inventions of a plethora of techniques for stabilizing GAN training, among which we employ the two-timescale update rule (TTUR) (Heusel et al., 2017) and *mixup* (H. Zhang et al., 2018). TTUR means simply setting the generator's learning rate a few times larger than that of the discriminator, which has been empirically shown to stabilize GAN training significantly.

The other technique, *mixup*, is an extension to empirical risk minimization where training data samples are drawn from convex interpolations between pairs of empirical data samples. For a pair of feature-target tuples $(\mathbf{X}_i, \mathbf{Y}_i)$ and $(\mathbf{X}_j, \mathbf{Y}_j)$ sampled randomly from the empirical distribution, their convex interpolation is given by:

$$\begin{aligned} \tilde{\mathbf{X}} &= \lambda \mathbf{X}_i + (1 - \lambda)\mathbf{X}_j \\ \tilde{\mathbf{Y}} &= \lambda \mathbf{Y}_i + (1 - \lambda)\mathbf{Y}_j \end{aligned} \tag{33}$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$, and α is the *mixup* hyperparameter which controls the strength of interpolation. When $\alpha = 0$, the Beta distribution becomes Bernoulli(0.5) which recovers the usual GAN training without *mixup*.

Input: Generator $G_\vartheta(\mathbf{X})$ with initial parameters ϑ, learning rate η, and loss function $\mathcal{L}_{\text{task}}(\hat{\mathbf{Y}}, \mathbf{Y})$, discriminator $D_\varphi(\mathbf{Y})$ with initial parameters φ, learning rate β, and loss function $\ell \in \{\text{BCE}, \text{MSE}\}$, batch size m, training data distribution $p(\mathbf{X}, \mathbf{Y})$, `pix2pix` weight ν, *mixup* strength α.
Output: Trained conditional generator $G_\vartheta(\mathbf{X})$.

---

while φ *and* ϑ *have not converged* **do**
  $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1,\cdots,m} \leftarrow m$ samples from $p(\mathbf{X}, \mathbf{Y})$
  for $i = 1, \cdots, m$ **do**
    $\hat{\mathbf{Y}}_i \leftarrow G_\vartheta(\mathbf{X}_i)$
    $\lambda_i \leftarrow$ sample from Beta(α, α)
    $\tilde{\mathbf{Y}}_i \leftarrow \lambda_i \mathbf{Y}_i + (1 - \lambda_i)\hat{\mathbf{Y}}_i$
  end
  $\mathcal{L}^D_{\text{cGAN}} \leftarrow \sum_{i=1}^{M} \ell(D_\varphi(\tilde{\mathbf{Y}}_i), \lambda_i)$
  $\varphi \leftarrow \varphi - \beta \cdot \nabla_\varphi \mathcal{L}^D_{\text{cGAN}}$
  $\mathcal{L}^G_{\text{cGAN}} \leftarrow \sum_{i=1}^{M} \ell(D_\varphi(\tilde{\mathbf{Y}}_i), 1 - \lambda_i)$
  $\vartheta \leftarrow \vartheta - \eta \cdot \nabla_\vartheta \sum_{i=1}^{m} \nu \mathcal{L}_{\text{task}}(\hat{\mathbf{Y}}_i, Y_i) - \mathcal{L}^G_{\text{cGAN}}$
end

Algorithm 1: Training of a *mixup* Conditional GAN.

mixup is readily applicable to the binary classification task of GAN discriminators. In our conditional GAN setup, we have an additional advantage of having paired samples of a real label $\mathbf{Y}$ and a fake label $\hat{\mathbf{Y}} = G(\mathbf{X})$, which allow us to replace Equation 32 with:

$$\min_{G} \max_{D} \mathbb{E}_{\mathbf{X},\mathbf{Y},\lambda} \; -\ell(D(\lambda\mathbf{Y} + (1-\lambda)G(\mathbf{X})),\ \lambda) \ . \tag{34}$$

where $\ell(p, y) = -y \log p - (1-y)\log(1-p)$ is the binary cross entropy (BCE) function. With this *mixup* setup, the discriminator now has to operate on the convex interpolation between the predicted representation and the corresponding ground truth. This makes the discriminator's task even more difficult when the prediction gets close to the ground truth, which is desirable because the discriminator should be inconclusive (i.e. $D = \frac{1}{2}$ everywhere) at the global optimum (Goodfellow et al., 2014).

Algorithm 1 details the procedure of training the conditional GAN using *mixup*, based on Equations 32 and 34. Note that for training the generator network, we perform label flipping in $\mathcal{L}^{G}_{\text{cGAN}}$ similarly as in Equation 27. Also, to train a least-squares GAN (Equation 28) instead, we can simply replace ℓ with a mean squared error (MSE) loss.

4 Experimental Setup

To verify the effectiveness of our approach, we compare Onsets and Frames (Hawthorne et al., 2018), a state-of-the-art piano transcription model, with variants of the same model that are trained with the adversarial loss. We also aim to evaluate the choices of the GAN loss and the *mixup* strength α.

4.1 Model Architecture

We use the extended Onsets and Frames model (Hawthorne et al., 2019) which increased the CNN channels to 48/48/96, the LSTM units to 256, and the FC units to 768. The extended model has a total of 26.5 million parameters. We do not use the frame loss weights described in (Hawthorne et al., 2018) in favor of the offset stack introduced in the extended version (see Figure 18). During inference, we first calculate the activations corresponding to overlapping chunks of audio, with the same length as the training sequences, and perform overlap-add (OLA) using Hamming windows to obtain the full-length activation matrix. We perform OLA instead of applying the recurrent calculations for the full length, because the effects of adversarial learning are best achieved within the training sequence length.

The input to the discriminator has two channels for the onset and frame predictions. The discriminator has 5 convolutional layers: `c32k3s2p1`, `c64k3s2p1`, `c128k3s2p1`, `c256k3s2p1`, `c1k5s1p2`, where the numbers indicate the number of output channels, the kernel size, the stride amount, and the padding size. At each non-final layer, dropout of probability 0.5 and leaky ReLU activation with negative slope 0.2 are used. The mean of the final layer output along the time and frequency axes is taken as the discriminator output.

4.2 Hyperparameters

Table 4 summarizes the hyperparameters used during the experiments, which are mostly taken directly from Hawthorne et al. (2018) and Isola et al. (2017). Also following Hawthorne et al. (2018), we use Adam (Kingma & Ba, 2015) and apply learning rate decay of factor 0.98 in every 10,000 iterations, for both the generator

Hyperparameter	Values
Generator learning rate η	0.0006
Discriminator learning rate β	0.0001
Discriminator loss function ℓ	{BCE, MSE}
Batch size m	8
`pix2pix` weight ν	100
mixup strength α	{0, 0.2, 0.3, 0.4}
Activation threshold τ	0.5
Training sequence length	327,680

Table 4: Hyperparameters used during the experiments.

and the discriminator. We examine two types of GAN losses, the non-saturating GAN ($\ell = \text{BCE}$) and the least-squares GAN ($\ell = \text{MSE}$). For each GAN loss, multiple values of *mixup* strengths are compared with $\alpha = 0$, i.e. no *mixup*. Training runs for one million iterations, and the iteration that best performs on the validation set are used for evaluation on the test set.

4.3 Dataset

We use the MAESTRO dataset (Hawthorne et al., 2019), which contains Disklavier recordings of 1,184 classical piano performances. The dataset consists of 172.3 hours of audio, which are provided with 140.1, 15.3, and 16.9 hours of train/validation/test splits such that recordings of one composition only appear in the same split. We resample the audio to 16 kHz and down-mix into a single channel. Following Hawthorne et al. (2018), an STFT window of 2,048 samples is used for producing 229-bin Mel spectrograms, and a hop length of 32 ms is used. Training sequences sliced at random positions are used, unlike the official implementation which slices training sequences at silence or zero crossings.

4.4 Evaluation Metrics

The Onsets and Frames model perform both frame-level and note-level predictions, and their performance can be evaluated with the standard precision, recall, and F1 metrics. For multi-pitch estimation, we also report the error rate metrics defined by Poliner and Ellis (2006), which include total error, substitution error, miss error, and false alarm error. We use the `mir_eval` (Raffel et al., 2014) library for all metric calculations. For the note-level metrics, we use the default settings of the library, which use 50 ms for the onset tolerance, 50 ms or 20% of the note length (whichever is longer) for the offset tolerance, and 0.1 for the velocity tolerance.

5 Results

5.1 Comparison with the Baseline Metrics

Table 5 and 6 summarize the transcription performance, clearly showing a consistent improvement in the conditional GAN models over the Onsets and Frames baseline. Table 5 shows that both non-saturating GAN and least-squares GAN achieve the highest frame and note F1 scores when the *mixup* strength

			mixup strength α			
	Baseline	GAN type	0	0.2	0.3	0.4
Frame F1	0.899	Non-Saturating	0.664	0.912	**0.914**	0.907
		Least-Squares	0.904	0.903	0.906	0.898
Note F1	0.942	Non-Saturating	0.717	0.953	**0.956**	0.951
		Least-Squares	0.944	0.947	0.950	0.943

Table 5: Frame and note F1 scores are the highest when the non-saturating GAN loss and $\alpha = 0.3$ are used.

			Non-Saturating GAN			Least-Squares GAN		
		Baseline	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$
Frame Metrics	F1	0.899	0.912	**0.914**	0.907	0.901	0.906	0.898
	Precision	0.946	0.937	0.931	0.939	0.940	0.942	0.946
	Recall	0.857	0.889	0.898	0.879	0.865	0.875	0.855
	E_{total}	0.179	0.157	0.156	0.166	0.176	0.167	0.181
	E_{subs}	0.013	0.013	0.012	0.013	0.014	0.013	0.013
	E_{miss}	0.130	0.097	0.089	0.108	0.121	0.113	0.132
	E_{fa}	0.036	0.047	0.054	0.045	0.042	0.042	0.036
Note	F1	0.942	0.953	**0.956**	0.951	0.944	0.950	0.941
	Precision	0.990	0.974	0.981	0.973	0.986	0.988	0.989
	Recall	0.899	0.933	0.932	0.930	0.905	0.916	0.898
Note with Offsets	F1	0.802	0.811	**0.813**	0.799	0.799	0.810	0.798
	Precision	0.842	0.828	0.835	0.817	0.835	0.841	0.838
	Recall	0.765	0.794	0.793	0.782	0.767	0.781	0.762
Note with Offsets and Velocity	F1	0.790	0.799	**0.802**	0.787	0.788	0.799	0.787
	Precision	0.830	0.816	0.823	0.805	0.823	0.830	0.827
	Recall	0.755	0.783	0.782	0.770	0.757	0.771	0.752

Table 6: Summary of transcription performance. The non-saturating GAN loss achieves the best performance across all F1 metrics. The average metrics across the tracks in the MAESTRO test dataset are reported, and the model checkpoint where the average of frame F1 and note F1 is the highest on the validation dataset is used.

$\alpha = 0.3$ is used, and they both outperform the baseline. The binary piano rolls are easy to distinguish from the non-binary predictions, which may cause imbalanced adversarial training. *mixup* allows non-binary piano rolls to be fed to the discriminator, making its task more challenging and leading to higher performance.

Table 6 shows an important trend of the cGAN results compared to the baseline that cGAN trades off a bit of precision for a significant improvement in recall; this is a side effect of the cGAN producing more confident predictions, as will be discussed in the following subsections.

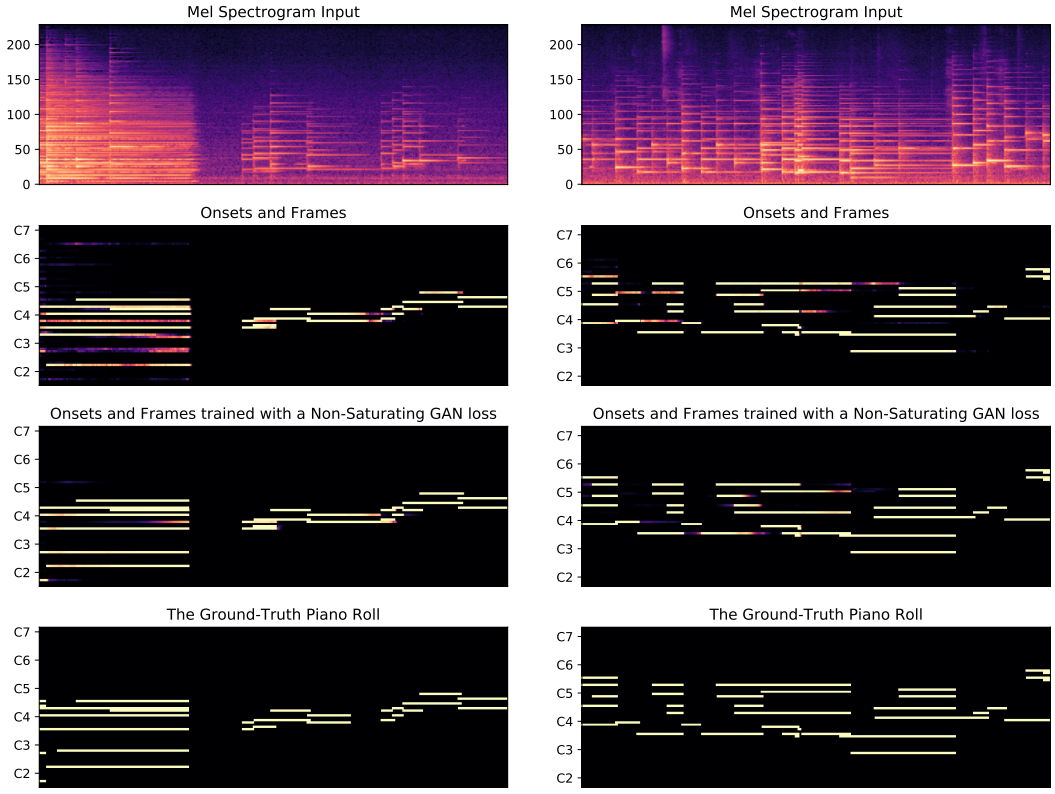


Figure 20: Comparisons of the frame activations predicted by the baseline and our model ($\ell = \text{BCE}$, $\alpha = 0.3$), on three example segments. The input Mel spectrograms and the target piano rolls are shown together. The GAN version produces more confident predictions compared to the noisy baselines, leading to more accurate predictions.

While the percentage differences are moderate, our method achieves statistically significant improvements in F1 metrics on the MAESTRO test dataset ($p < 10^{-14}$ for all 4 metrics, two-tailed paired t-test). The distribution of per-track improvement in each F1 metric is shown in Figure 21, which indicates that the improvements are evenly distributed across the majority of the tracks. These improvements are especially promising, considering that Onsets and Frames is already a very strong baseline.

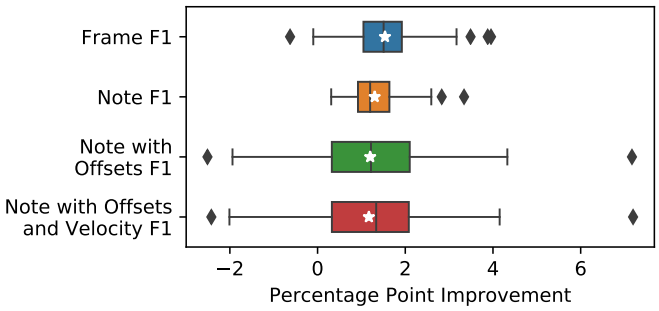


Figure 21: Distribution of the F1 score improvements over the baseline, tested on the MAESTRO test tracks.

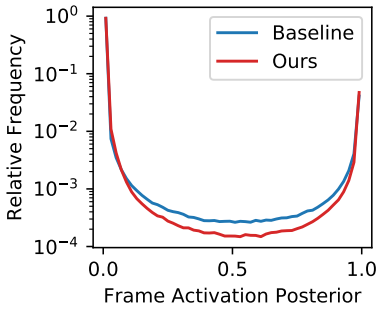


Figure 22: Distribution of frame activation values. Our model outputs more confident predictions, as indicated by the lower relative frequency in (0.1, 0.9).

5.2 Visualization of Frame Activations

To better understand the inner workings of the conditional GAN framework, we visualize the frame activations created by the baseline and the best performing conditional GAN model in Figure 20. In contrast to the baseline which have many blurry segments, the activations generated by our method mostly contain segments with solid colors, meaning that the model is more confident in its prediction. Figure 22 shows that the proportion of frame activation values in $(0.1, 0.9)$ is noticeably higher in the baseline, thus making the output less sensitive to the threshold choice. This is because indecisive predictions are penalized by the discriminator, since they are easy to distinguish from the ground-truth which

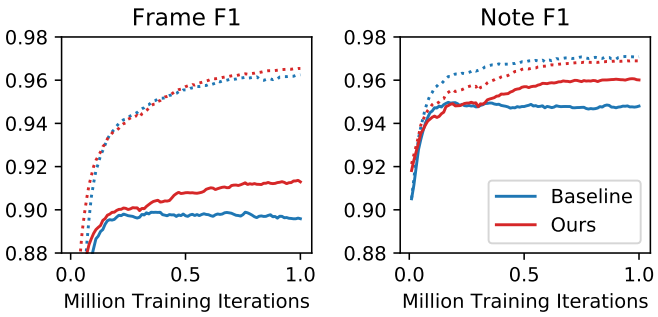


Figure 23: Learning curves showing the generalization gaps; training curves are drawn as dotted lines, and test curves are drawn as solid lines.

contains only binary labels. The generator is therefore encouraged to output the most probable note sequences even when it is unsure, rather than producing blurry activations that might hamper the decoding process. This allows for an interpretation in which the GAN loss provides a prior for valid onset and frame activations, and the model learns to perform MAP estimation based on this prior.

5.3 Training Dynamics and The Generalization Gap

Figure 23 shows the learning curves for the frame F1 and note F1 scores, where the scores on the training dataset are plotted in dotted lines. It is noticeable in the figure that the validation F1 scores for the baseline stagnate after 300k iterations, while the F1 scores of our model steadily grow until the end of 1 million iterations. Thanks to this, the generalization gap — the difference between the training and validation F1 scores — is significantly smaller for the conditional GAN model. This means that the GAN loss works as an effective regularizer that encourages the trained model to generalize better to unseen data, rather than memorizing the note sequences in the training dataset as LSTMs are known to be capable of (Zaremba et al., 2014).

6 Conclusions

We have presented an adversarial training method that can consistently outperform the baseline Onsets and Frames model, using the standard frame-level and note-level transcription metrics and visualizations that show how the improved model predicts more confident output. To achieve this, a discriminator network is trained competitively with the transcription model, i.e. a conditional generator, so that the discriminator serves as a learned regularizer that provides a prior for realistic note sequences. After training, the discriminator can be disregarded for inference, incurring no additional computational cost during transcription.

Our results show that modeling the inter-label dependencies in the target distribution is important and brings measurable performance improvements. Our method is generic, and any model that involves predicting two-dimensional representation should be able to benefit from including an adversarial loss. These approaches are common not only in transcription models but also in speech or music synthesis models that predict spectrograms as an intermediate representation (Shen et al., 2018; J. W. Kim et al., 2019). This implies that the findings in this chapter is applicable to a broader set of problems not only in multi-pitch estimation but also in the field of music information retrieval in general.

Our results do not include the effects of using data augmentation (Hawthorne et al., 2019), which is orthogonal to our approach and should bring additional performance improvements when applied. As discussed, the discriminator imposes the prior on the target domain whereas data augmentation enriches the input audio distribution. This implies that our method would be less effective

when the majority of errors are due to the discrepancy in the audio distribution between the training and test datasets. How to apply adversarial learning for better generalization on the input distribution is a potential future research direction.

We have argued that the adversarial discriminator serves as a regularizer providing the prior knowledge of how the note sequences in the dataset should look like, which complements the conditionally independent output of the Onsets and Frames model. The fully-connected output layer also does not take into account a very important characteristic of pitch, that each pitch corresponds to quasi-periodic signals in the input with a known frequency that are geometrically spaced — rather, the predictions are instead made as if each pitch is completely independent pieces of information, not utilizing the knowledge on pitch at all. Another important concept that is not considered in this chapter is the timbre. Although the MAESTRO dataset used in this chapter contains various recording conditions and therefore somewhat different timbres among the tracks, they are all recordings of piano performances which have limited timbral diversity. These aspects motivate building an improved model that can leverage the regularity of pitch as well as the knowledge of instrumental timbres. In the next chapter, we discuss how to extend our music transcription model to incorporate such aspects, specifically by using a synthesizer model similar to the one used in Chapter V.

CHAPTER VII

SYNTHESIZER-AIDED MULTI-INSTRUMENT TRANSCRIPTION

In Chapter I, we introduced the idea of extending a music transcription model with an additional synthesizer component that can convert the transcribed information back to the input audio, forming an analysis/synthesis framework as depicted in the encoder-decoder architecture in Figure 2. By doing so, we anticipated that the two components — the transcriber and the synthesizer — can work together to better translate information between the two representations, i.e. audio waveform and the transcribed semantic information of the music. One motivation for employing such analysis/synthesis framework is that the advent of many successful deep learning techniques and the improved hardware capability may allow us to break apart an audio signal completely into the constituent pieces of musical information that we want to transcribe, that are entangled in an extremely sophisticated way in the audio signal. In the preceding chapters, we have considered deep learning solutions to various problems in music analysis and synthesis, such as monophonic pitch tracking (Chapter IV), music synthesis with controllable timbre (Chapter V), and polyphonic music transcription (Chapter VI). These problems correspond to a certain subset of the full encoder-decoder architecture, and in each chapter we discussed how the design choices in the model architecture and dataset construction affect the experimental results and the real-world applicability.

Having learned the lessons from these, in this chapter we aim to construct an

automatic music transcription system that encompasses the full analysis/synthesis cycle, by considering the general problem of multi-instrument polyphonic music transcription. We propose an autoencoder architecture that appends a synthesis model on top of a transcription model, and we verify the effects of the appended synthesizer in the experiments. As in Chapter V, the synthesizer component is capable of capturing the different aspects of various timbres in the dataset using a learned timbre space representation. We show that the synthesizer component can serve as a regularizer to the transcriber, providing a form of prior knowledge on the spectral and temporal characteristics of the timbres and thereby improving the transcription performance as well.

1 Introduction

Multi-instrument music transcription aims to extract not only the occurrences of multiple simultaneous pitches and notes but also the types of one or more instruments corresponding to those notes. Since it poses additional challenges to the already difficult problem of polyphonic music transcription, studies on the complete multi-instrument polyphonic transcription problem are relatively rare. Instead, most studies on analyzing multi-instrument audio either choose to develop a discriminative model for polyphonic audio where timbre information is disregarded (Bittner et al., 2017; W. Zhang et al., 2018) or separately work on instrument recognition, which does not aim to identify the individual notes that belongs to the recognized instruments (G. Agostini et al., 2003; Lostanlen & Cella, 2016). To take both polyphonic notes and multiple instruments into account, a model has to carefully incorporate the prior knowledge of how pitch and timbre are reflected in the audio signal, e.g. using a probabilistic latent

component model (Benetos & Weyde, 2015) or a variant of non-negative matrix factorization (Grindlay & Ellis, 2009). To produce outputs in multiple domains, these models have to include an intricate set of design choices to implement a mathematical representation of the prior knowledge.

Meanwhile, the main premise of deep learning is quite the opposite of how those models are designed. Rather than writing down the set of rules manually, layers of simple nonlinear calculations can serve as a very effective function approximator, that can produce just about anything when an enough amount of data is fed to the black box. In these models, the prior knowledge baked into the model architecture is minimal and more abstract. Take the Onsets and Frames model (Hawthorne et al., 2018) for example; the model output has 88 dimensions corresponding to each key of the piano, but the 88 keys are treated as separate multi-label targets, without utilizing the crucial knowledge that each key should represent a certain frequency that the quasi-periodic signal in the input follows. Because of this lack of "inductive bias", deep learning models typically have to be trained with a very large amount of data; in other words, they are less sample efficient.

Generative models for transcription, such as a Bayesian network representation of spectrogram (Berg-Kirkpatrick et al., 2014), takes the opposite approach. These models define the process of generating sound entirely from the parameterized sources that typically convey a physical interpretation, such as the modeling of attack-decay-sustain-release (ADSR) curves or the energy distribution along harmonics. Compared to data-driven models with a lot of learnable parameters, however, those interpretable transcription models often lack the flexibility to be applicable for a wider variety of inputs.

In this chapter, we explore the possibility of finding a compromise between the two extremes. Specifically, we examine how to induce the transcription model to make predictions such that, when synthesized back into audio, resemble the original audio input. We do this by appending a neural music synthesis model to the transcription model. The synthesis model is specifically designed to describe the sound generation process with a small number of interpretable parameters, which is restrictive enough to guide the transcription model to more accurate predictions yet flexible enough to be learnable in a data-driven manner. We evaluate this approach with a multi-instrument transcription task and show that a synthesis model can serve as an effective regularizer, through a qualitative analysis of the synthesis model and a quantitative analysis of the transcription preformance.

2 Related Work

2.1 Multi-Instrument Music Transcription

Transcribing polyphonic music with multiple instruments is a particularly difficult problem, and there has been relatively few successful attempts, as it requires solutions to instrument classification in addition to multiple pitch estimation. While some approaches perform source separation and instrument recognition as a separate pre-processing step for per-instrument polyphonic transcription, (Heittola et al., 2009; Itoyama et al., 2011), a number of approaches attempt to jointly solve the two problems, by modeling the possible instruments as a constraint to non-negative matrix factorization (NMF) (Grindlay & Ellis, 2009), applying spectral templates to a probabilistic latent component model (Benetos & Weyde, 2015), and using a U-Net architecture to separate a

multi-channel representation of audio into per-instrument transcriptions (Y.-T. Wu et al., 2019). Joint estimation of instrument and multiple pitches allows for flexible modeling of diverse sounds and enables an end-to-end approach for multi-instrument polyphonic transcription, while employing two separate stages of source separation followed by per-instrument transcription allows focusing on each stage separately. In this chapter, we use the former approach and develop a model that jointly performs instrument classification and polyphonic transcription.

2.2 Deep Clustering

Deep clustering (Hershey et al., 2016) is a technique for audio source separation which learns a mapping from each time-frequency bin of the input audio into an embedding space. The original formulation uses the class assignment matrix $Y = \{y_{i,c}\}$ indicating whether element i belongs to cluster c and learns the embedding matrix $V = \{\mathbf{v}_i\}$ as a function of the input audio that minimizes the Frobenius distance between the assignment matrix:

$$\mathcal{L} = \|VV^\top - YY^\top\|_\mathrm{F}^2 = \sum_{i,j} \left(\langle \mathbf{v}_i, \mathbf{v}_j \rangle - \langle \mathbf{y}_i, \mathbf{y}_j \rangle\right)^2 . \tag{35}$$

Since $\langle \mathbf{y}_i, \mathbf{y}_j \rangle$ equals 1 if the elements i and j belongs to the same cluster and 0 otherwise, the learned embedding vectors are optimized to be as close as possible for the vectors which belong to the same cluster and as orthogonal as possible otherwise. Alternative objective functions to Equation 35 employing deep neural networks have been suggested by (Z. Q. Wang et al., 2018). The deep clustering technique has been successfully applied to the multi-speaker separation (Isik et al., 2016) and the singing voice separation problem (Luo et al., 2017).

This chapter's work borrows the idea of deep clustering and builds an embedding space that captures the timbres of different instruments, while relying on a synthesizer component rather than inducing every pair of embedding vectors to be orthogonal to each other. By allowing different embedding vectors to be similar but not necessarily orthogonal, the synthesizer can learn to map similar-sounding instruments to vectors that are close to each other. Informed by this embedding space, the multi-instrument transcription model can more easily learn to classify the similar instruments than when the instruments are treated independently.

2.3 Generative Modeling for Transcription

Bayesian approaches for music transcription consider the audio generation process as a probabilistic model that can describe the generated audio in terms of its sound events. In these approaches, an audio representation such as a spectrogram is considered as an observation, and the model parameters that maximizes the observation is found using Bayesian inference techniques such as particle filtering (Dubois & Davy, 2005), message propagation (Cemgil et al., 2006), and block-coordinate ascent (Berg-Kirkpatrick et al., 2014). These models typically work well for small test datasets with a homogeneous timbral profile, but because they rely on a small, manually designed set of parameters for sound generation, they often struggle to generalize to diverse inputs.

In the spirit of utilizing the synthesizer component for transcription, S. Li (2017) explored the idea of using on-the-fly synthesized training dataset for piano transcription, using a simple fully-connected neural network operating on the CQT representation. K. Choi and Cho (2019) proposed appending a drum synthesizer

component to enable fully unsupervised training of drum transcription. The work in this chapter draws several ideas from these previous approaches of incorporating a synthesizer component into the loop, while also leveraging the flexibility and the generalizability of data-driven deep learning.

3 Method

We first describe the synthesizer model, followed by the modifications to the Onsets and Frames (Hawthorne et al., 2018) transcription model made to allow the concatenation of the two models.

3.1 Synthesizer Model

The synthesizer model takes a per-instrument piano roll representation of symbolic music (see Figure 1) and uses a learned instrument embedding to predict the Mel spectrogram of the corresponding audio. We are assuming here that predicting Mel spectrograms is equivalent to synthesizing the audio for our purposes, since we have seen in Chapter V that Mel spectrograms contain most of the perceivable information present in the audio content, and the original audio can readily be reconstructed using sample-based synthesis models like WaveNet (van den Oord, Dieleman, et al., 2016).

The synthesizer consists of two subcomponents, the envelope estimator and the waveshaper, respectively implementing the temporal and spectral characteristics of the sound being synthesized. The computation in each subcomponent is performed independently for each pitch-instrument combinations, and their results are combined to obtain the resulting Mel spectrogram, as shown in Figure 24. Both subcomponents take the instrument

embedding in order for the synthesized audio to exhibit different temporal and spectral behaviors depending on the instrument, and together they ensure that the synthesis model only creates harmonic sounds with the frequency corresponding to the pitch of input notes and the spectral envelope consistent among each instrument.

The envelope estimator component predicts the temporal envelope of each note, by mapping the note activities into a latent space using a one-dimensional convolution layer, followed by a FiLM layer (Perez et al., 2017) to condition on the instrument. A GRU layer (K. Cho et al., 2014) is then applied in order to model the temporal behavior, such as the attack-decay-sustain-release (ADSR) curves. Meanwhile, the waveshaper component performs a subtractive synthesis to obtain the spectrum corresponding to each pitch and instrument. We use two types of source waves, sawtooth and triangular, to make it easily adaptable to the timbres that primarily consists of odd harmonics, such as the clarinet. The instrument embedding is used to predict the transfer function in the Mel frequency domain to be applied to each source spectrum, which is modeled as a polynomial function mapping Mel frequencies to log magnitudes. The Mel spectrogram corresponding to each note can then be calculated in the complex domain, by assigning a random initial phase to each pitch-instrument pair and multiplying the temporal envelope predicted by the envelope estimator. All of the per-note Mel spectrograms are then added to obtain the final Mel spectrogram output.

3.2 Training Transcriber with Appended Synthesizer

To append the synthesizer component to the output of the transcription model while keeping the full model differentiable, we expand the fully-connected layer

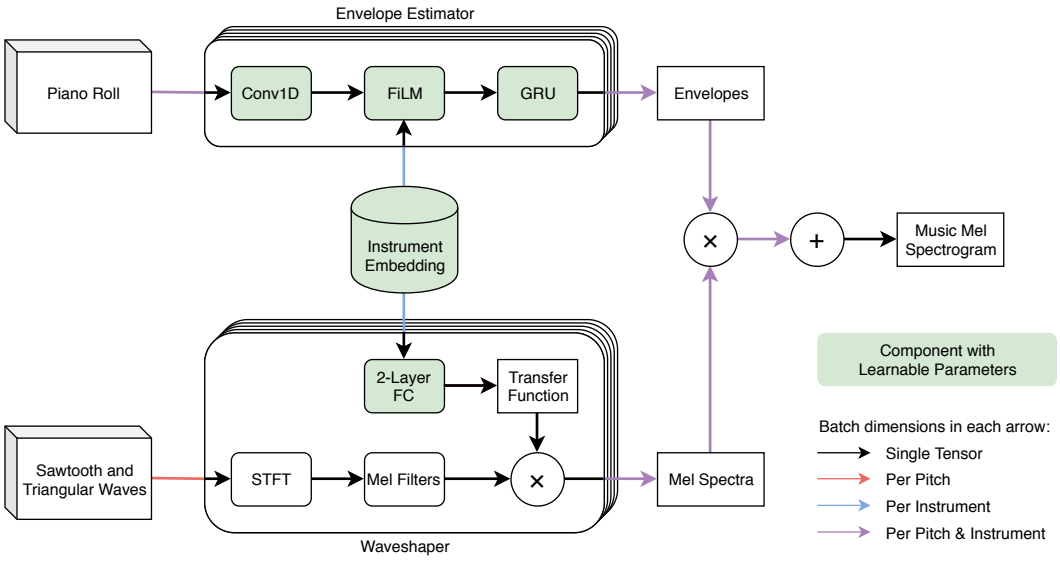


Figure 24: The synthesizer architecture. The temporal and spectral characteristics of each notes are respectively modeled by the envelope estimator and the waveshaper. This structure enforces each instrument to have consistent spectral and temporal envelopes across the pitch.

predicting the frame activation to additionally predict the instrument embedding corresponding to each time-frequency bin. Using the expanded outputs, the frame activations and the predicted embedding can be fed to the synthesizer component. The frame activations are binarized before feeding to the synthesizer, and we use a gradient-stop for this connection.

The synthesizer model is separately trained, and the transcriber model is then optimized to minimize the sum of three losses:

$$\mathcal{L}_{\text{Overall}} = \mathcal{L}_{\text{Onsets \& Frames}} + \lambda \ \mathcal{L}_{\text{Embedding}} + \mathcal{L}_{\text{Synthesizer}} \tag{36}$$

The first term, $\mathcal{L}_{\text{Onsets \& Frames}}$, is the loss of the Onsets and Frames model which is the sum of the binary cross entropy of the onsets, offsets, and frame predictions. λ is a hyperparameter that scales the ratio between the losses from the transcription and the synthesis models. We define $\mathcal{L}_{\text{Embedding}}$ as the mean squared error (MSE)

between the predicted instrument embedding and the ground truth, and $\mathcal{L}_{\text{Synthesizer}}$ is defined as the mean cosine distance between the predicted and the ground-truth Mel spectra:

$$\mathcal{L}_{\text{Synthesizer}} = \mathbb{E}_{\hat{\mathbf{s}},\mathbf{s}}\left[1 - \frac{\hat{\mathbf{s}} \cdot \mathbf{s}}{\|\hat{\mathbf{s}}\|\|\mathbf{s}\|}\right] \tag{37}$$

where $\mathbf{s}$ follows the distribution of Mel spectra in the input audio, and $\hat{\mathbf{s}}$ is the Mel spectra predicted by the synthesizer based on the transcription predicted from $\mathbf{s}$.

4 Experimental Setup

We use the MusicNet dataset (Thickstun et al., 2017) to train both the synthesizer and the transcriber. The dataset contains 2,048 minutes of classical chamber music audio and labels, which comprises 330 recordings of various ensembles performed by musicians. The dataset contains annotations of 11 General MIDI instruments: acoustic grand piano, harpsichord, violin, viola, cello, contrabass, french horn, oboe, bassoon, clarinet, and flute. Different recordings in the dataset have drastically different tuning, ranging more than a semitone in some recordings which would result in a wrong transcription even by a perfect transcriber. To alleviate this, we preprocess the dataset by running a tuning estimator implemented in librosa (Mcfee et al., 2015) on each track and pitch-shifted any recordings that are more than 20 cents apart from the A440 tuning.

We use the Onsets and Frames model (Hawthorne et al., 2018) with the hidden size of 512, which contains 10 million learnable parameters. We use a 2-dimensional instrument embedding space to represent the distribution of the 11 instruments, and the polynomial transfer functions used in the subtractive synthesis are modeled by cubic Bézier curves. We additionally train and compare

a multi-label prediction model, which replaces the last layer of the Onsets and Frames model with a multi-label binary classifier that predicts the activation of each of the 11 instruments.

The audio is resampled to 16kHz, and a training batch is composed of eight 20-second audio segments and labels. We use Adam optimizer (Kingma & Ba, 2015) with learning rate 0.0006, and the learning rate decay of factor 0.98 is applied every 1000 steps. We ran the optimization for 180,000 steps and report the transcription accuracy evaluated on the 10 test recordings provided in the MusicNet dataset. We use $\lambda = 10^{-4}$, which was obtained from a hyperparameter search.

5 Results

In this section, we summarize the experimental results, first by qualitatively analyzing the behavior of the synthesizer model, followed by quantitative comparisons of transcription accuracies of the baseline and proposed transcription models. We also introduce a web interface that allows for easier exploration of the contents of the MusicNet dataset.

5.1 Synthesizer Output

Figure 25 shows example input, target, and output representations of the synthesizer component, and Figure 26 shows the learned timbre embedding space containing the 11 instruments in the MusicNet dataset. The input to the synthesizer contains the frame information in a piano roll format, along with the corresponding timbre embeddings (not shown in Figure 25). The synthesizer is trained to predict the corresponding audio's Mel spectrogram by minimizing the cosine distance. Figure 26 indicates that the synthesizer model is able to learn a

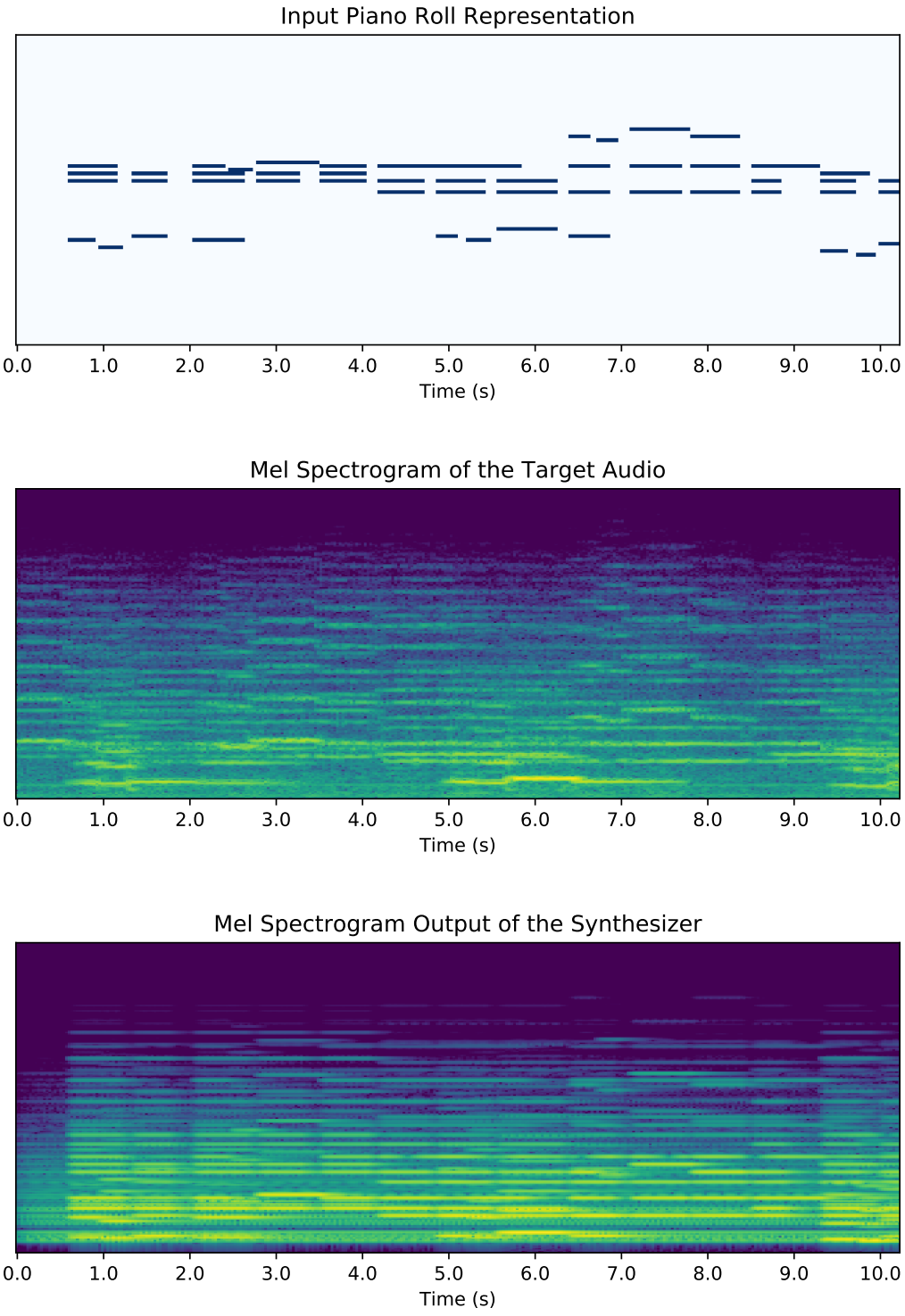


Figure 25: An example of the input, target, and output representations of the synthesizer. The output resembles the target Mel spectrogram but shows a regular pattern as constrained by the synthesizer model.

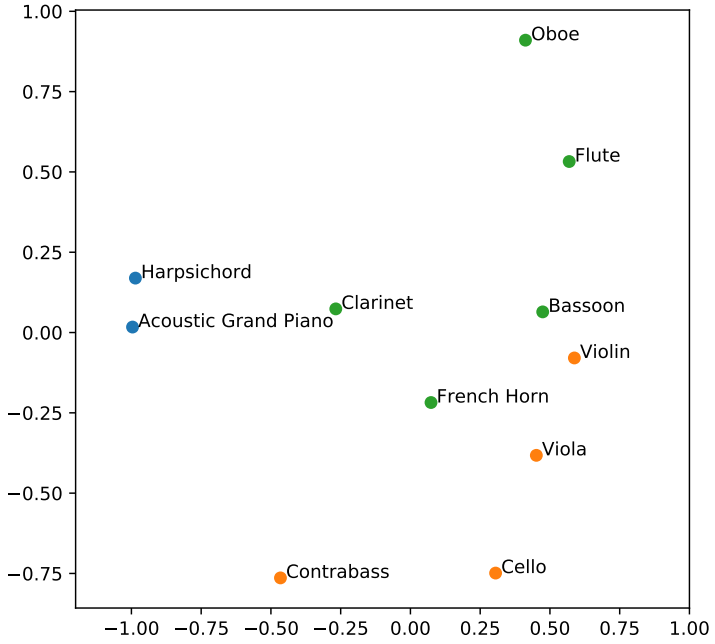


Figure 26: The learned timbre embedding space mapping the 11 instruments in the MusicNet dataset into the corresponding points in the space. Types of instruments (keyboards, winds, and strings) are color-coded.

timbre embedding space that maps similar kinds of instruments into neighboring regions in the space.

Because the synthesizer is strictly limited by its architecture (Figure 24) to produce harmonic sound only, the output Mel spectrogram exhibits highly regular patterns compared to the target Mel spectrogram. While this output Mel spectrogram may not represent an audio signal that perfectly matches the target, it motivates us to find out whether this synthesizer can give informative signals to a transcription model.

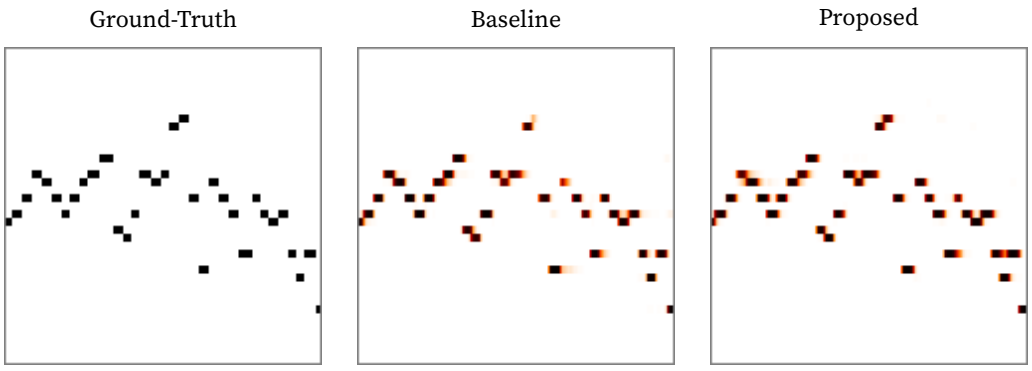
	Baseline (single label)	Baseline (multi-label)	Proposed (synthesizer-aided)
F1 Score	0.720	0.650	**0.729**
Precision	**0.736**	0.714	0.735
Recall	0.711	0.600	**0.726**

Table 7: A comparison of instrument-agnostic frame transcription accuracies for the baseline models and the proposed synthesizer-aided transcription model. The proposed model achieves better recall while staying at the same precision.

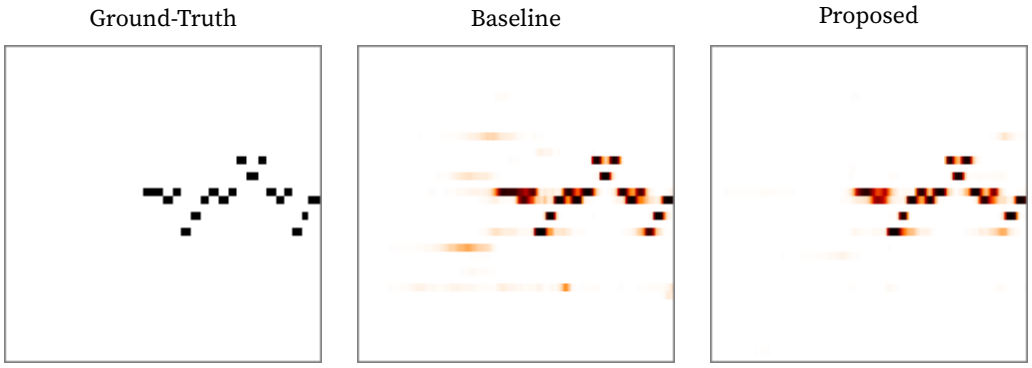
5.2 Transcription Accuracy

We compare the accuracy of the transcription model trained using the additional loss terms given by the appended synthesizer model, as in Equation 36, with the accuracy of the equivalent transcription model trained without a synthesizer component. Table 7 summarizes the frame transcription metrics of the two configuration, in addition to a multi-label prediction model that performs a multi-label classification for each instrument. The multi-label baseline exhibits significantly lower performance in these instrument-agnostic metrics, and the proposed model slightly outperforms the single-label baseline.

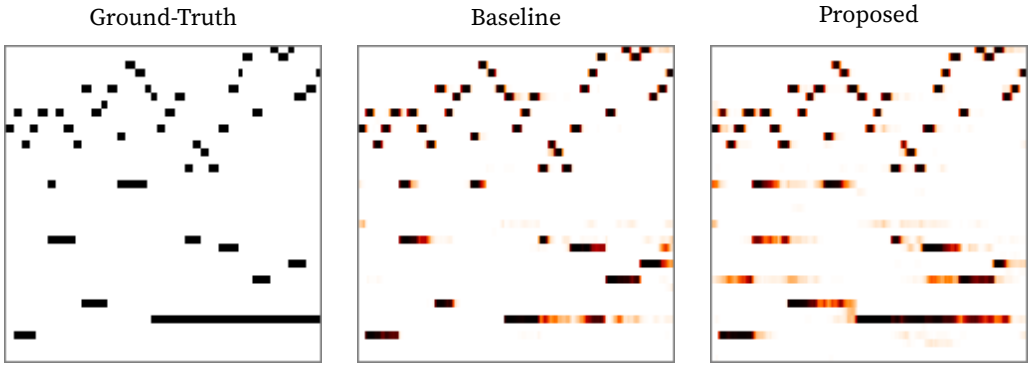
Specifically, the proposed model achieves better recall than the single-label baseline while maintaining the same level of precision. To find out how the different behaviors of the two models contribute the differences in these metrics, we visualize in Figure 27 a few interesting excerpts in the test dataset that explains the better recall score of the proposed model. Figure 27a is an excerpt of a solo violin track (MusicNet ID 2191) and shows that the proposed model is able to identify the highest note in the input that is not detected by the baseline model. In Figure 27b, which is the beginning of a solo cello piece (MusicNet ID 2298), the baseline model produces frame activations of a few notes before the first note, falsely predicted from the background noise in the input audio. Although



(a) The proposed model correctly finds the highest note that the baseline model did not detect.



(b) The proposed model makes less false positives in the silent part in the beginning of the excerpt.



(c) The proposed model better predicts the low sustained note in the latter half of the excerpt, while being more prone to false positive predictions than the baseline.

Figure 27: Examples of the frame predictions by the baseline and the proposed model compared to the ground-truth, showing their qualitative differences in performance.

the transcription metrics are not affected in this case since the activation values are below the threshold, it shows that the synthesizer model can suppress false predictions by the transcription model to a certain degree. Finally, in a solo piano excerpt shown in Figure 27c (MusicNet ID 2303), the low sustained note in the latter half of the input is better predicted by the proposed model than the baseline. Transcription of a long sustained piano note is difficult because of the volume decay of a note, and this example implies that the appended synthesizer is able to encourage an accurate prediction of long notes. In all of these cases, the synthesizer helps in more accurately detecting frames, which in turn contributes to the better transcription metrics.

5.3 Multi-Instrument Transcription

In this subsection, we compare the multi-instrument transcription performance of the proposed model with the multi-label baseline model. The instrument labels are assigned by finding the nearest instrument embedding to the embedding

	Baseline (multi-label)			Proposed (synthesizer-aided)		
	F1	Precision	Recall	F1	Precision	Recall
Acoustic Grand Piano	0.667	0.694	0.650	0.546	0.739	0.435
Violin	0.467	0.433	0.590	0.213	0.610	0.136
Viola	0.195	0.128	0.412	0.233	0.290	0.209
Cello	0.402	0.364	0.575	0.331	0.717	0.252
French Horn	0.255	0.164	0.602	0.246	0.218	0.299
Bassoon	0.290	0.213	0.473	0.065	0.152	0.043
Clarinet	0.386	0.305	0.526	0.376	0.332	0.440

Table 8: Per-instrument accuracy of the multi-instrument baseline and the proposed model, showing only the instruments in the test tracks. The multi-instrument baseline generally outperforms the proposed model, which although shows higher precision across all instruments.

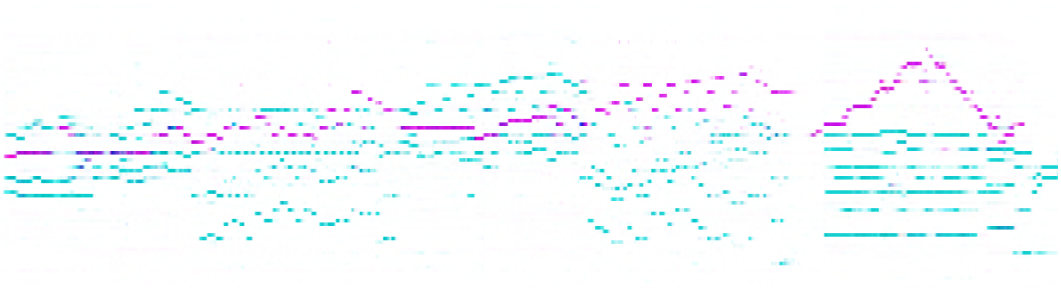


Figure 28: An example of multi-instrument transcription of a violin piece accompanied by piano (MusicNet track 2628), where the predictions of keyboard and string instruments are color-coded in cyan and magenta, respectively.

vector predicted by the transcription model. Table 8 summarizes the transcription metrics for each instrument and indicates that the proposed model actually performs worse than the multi-label baseline, despite the advantages in the instrument-agnostic evaluation in the previous subsection. One contributing factor for the lower performance is that the nearest-neighbor assignment of the instrument labels cannot produce the same pitch activated by multiple instruments, i.e. playing in unison. While the improvement in the individual multi-instrument performance remains a future work, the visualization of Figure 28 shows that the model is capable of classifying the different classes of instruments, such as the piano and the strings.

5.4 MusicNet Inspector

Finally, we introduce a web interface[1] developed for easy inspection of the audio tracks in the MusicNet dataset, which helped identify the characteristics and imperfections of the music and the annotations in the dataset. The UI is shown in Figure 29; the users can select one of the 330 tracks in the list showing the details

[1] `https://musicnet-instpector.github.io`

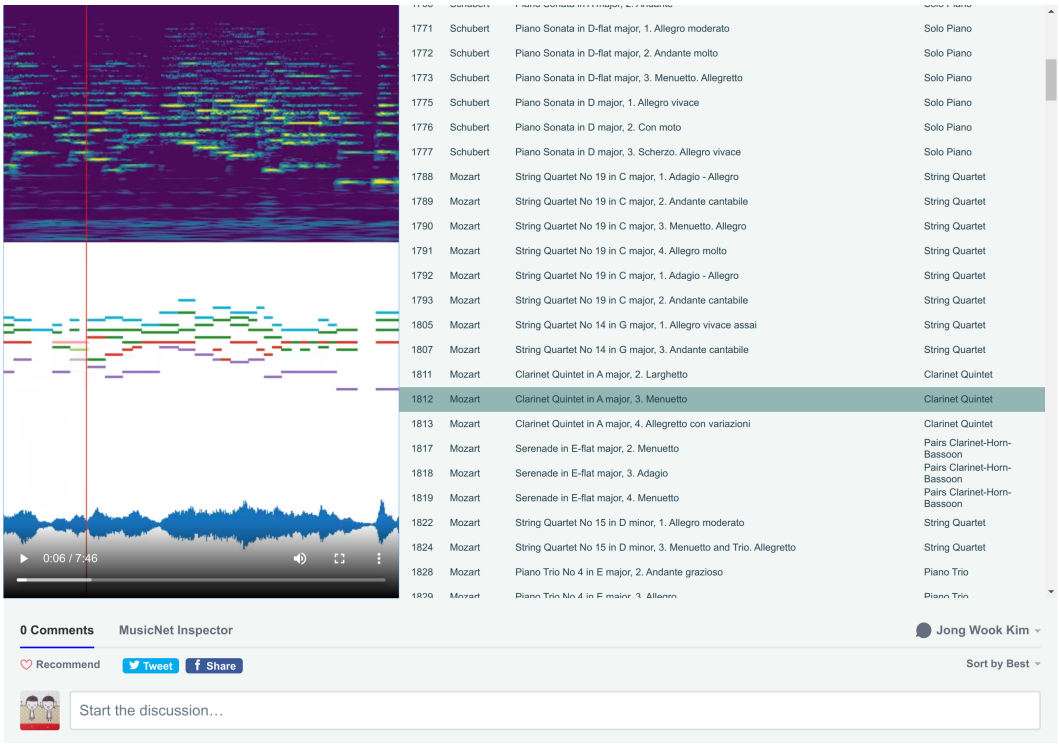


Figure 29: The MusicNet Inspector web interface. Users can select among the 330 tracks in the MusicNet dataset, and the selected track is played together with the CQT, piano roll, and amplitude visualizations.

of each track, and a video containing a CQT representation, color-coded piano-roll representation, and an audio amplitude visualization are played together with the audio. Additionally, the users can participate in the discussion about each track, e.g. any errors in the annotations, in the comment section. This interface can serve as a useful tool for researchers using the MusicNet dataset, providing an easy way to inspect the content of the dataset.

6 Future Work

The experimental results presented above has shown that a marginal improvement by the appended synthesizer component is possible. However, significant further work is required to prove the practical viability of the idea. First of all,

although MusicNet is the largest freely available dataset for multiple-instrument music transcription, it contains a non-negligible amount of inaccuracy in the annotations. This is largely because the annotations were obtained by matching each audio track with a MIDI file containing the same piece using dynamic time warping (Thickstun et al., 2017). This caused many inaccurate time mappings near the beginning and the end of each track, and there exist a few cases where inaccurate annotations continue over extended periods, because the music content of the audio and the MIDI file sometimes do not exactly match due to different repetitions or revisions of the music. Furthermore, because of the inhomogeneous sources of the audio tracks, the recording conditions and the tuning vary significantly, which made the model training even more difficult, given the relatively small number of tracks in the dataset. While the availability of a significantly larger and more accurate dataset is desirable, some of the drawbacks can be alleviated by performing data augmentation (McFee et al., 2015) such as pitch shifting. Since the audio input is not volume-normalized during training, improved robustness to different dynamic ranges of audio can be achievable by augmenting the training data with the variations in the audio volume.

We have used the cosine distance loss (Equation 37) which is invariant to the scaling of either the predicted or the target Mel spectrograms, because the annotations does not contain the velocity information. If a dataset containing the velocity information can be obtained, alternative loss functions such as spectral convergence (Sturm, 2013) can be used for more accurate prediction of Mel spectrograms by the synthesizer.

7 Conclusions

In this chapter, we have explored the overarching idea of this thesis: using a music synthesis model to achieve better music transcription performance. To do so, we have proposed a novel synthesizer architecture which is expressive enough to learn the distribution of different instruments, while being restricted to produce only harmonic sounds matching the input and also having a low number of parameters to be able to effectively serve as a regularizer to a transcription model. We observed improved performance when the transcription model is appended by a pre-trained music synthesis model, which can help the transcriber achieve better recall while suppressing incorrect predictions. Additionally, we have tested the multi-instrument transcription performance of the same setup and introduced a web interface for easier navigation of the MusicNet dataset used in the experiments.

This chapter concludes the technical work of this thesis, which started with proposing a framework for deep learning-based music analysis tasks (Chapter IV), followed by introducing a WaveNet-based music synthesis model that can also learn the distribution of instrumental timbre (Chapter V) and an adversarial technique that can impose a music language model on an existing transcription model (Chatper VI). We continue our discussion of our findings so far in the context of the series of research questions raised in Chapter I, as well as about the implications and the prospects of this line of research, in the next and final chapter of this thesis.

CHAPTER VIII

CONCLUSIONS AND FINAL REMARKS

This thesis has presented a number of novel approaches demonstrating the effectiveness of deep generative models in automatic music transcription research. In this chapter, we summarize our findings and discuss the future prospects of this line of research.

1 Summary and Takeaways

Through the experiments and analyses conducted in this thesis, we have answered the research questions raised in Section I.2. We summarize our findings and discuss the takeaways from the thesis in the following paragraphs, by addressing each of the research questions raised in Section I.2.

Question 1: What kinds of deep models and representations can be used for effectively extracting pitch from audio? In Chapter IV where a data-driven method for monophonic pitch estimation is proposed, we have found that a frame-wise convolutional computation predicting a time-frequency representation is an effective way to extract pitch information from audio. This approach is succesfully employed in the subsequent chapters of this thesis, where we predicted Mel spectrograms for music synthesis (Chapter V), piano-roll representation for music transcription (Chapter VI), and both (Chapter VII).

Question 2: How does the choice of datasets affect the accuracy and the generalizability of a trained model? We have learned that the choice of dataset can cause a drastic effect in a transcription model's performance, and caution regarding the generalizability is needed when releasing a model as open-source. For an open-source release of a pre-traind model, the best practice is to train the model using many datasets from various sources that are as diverse as possible, because audios in a single dataset are likely to be highly correlated, compared to image datasets (Thickstun et al., 2018). In general, the performance of a model is a function of the quality and the quantity of the dataset, very much more so than it is a function of the novelty in the model's design, and it is a motivating factor to develop automated data augmentation methods to be discussed in the next section.

Question 3: How can we encode the concept of timbre in a way that is useful for music synthesis and transcription? In Chapter V, we have presented a WaveNet-based music synthesis method that can learn a continuous timbre embedding space, on which one can interpolate between different timbres. This is achieved by conditioning a recurrent neural network with the timbre embedding to produce instrument-specific Mel spectrograms. The timbre embedding space encodes continuous variations in musical timbre in which it is possible to interpolate timbres between points, and it can contain the timbre of virtually unlimited number of instruments. How to reduce discontinuities and irregularities to make smoother timbre transitions remains a future work.

Question 4: How can a transcription model make informed predictions incorporating the knowledge of a music language model? In Chapter VI, we have used a conditional generative adversarial network model to impose a prior on music transcription, borrowing ideas from image translation

literature. By appending an adversarial discriminator that encourages the resulting transcription to be realistic, this approach implements a music language model that works in conjunction with the Onsets and Frames transcription model.

Question 5: Can a music synthesizer component based on a deep generative model be used to improve music transcription? Lastly, in Chapter VII, we combined the ideas from the preceding chapters to create a music transcription model that can benefit from the musical knowledge contained in a music synthesizer model. A synthesizer-aided transcription model achieves better recall than the equivalent model without a synthesizer, and it implies that the appended synthesizer provides useful information to the transcription model as a form of regularization.

The main takeaway from the approaches above is that it is crucial to find the best way to implement a good prior into a music transcription system, be it an adversarial discriminator or an appended music synthesis model. Those priors can effectively serve as a regularizer to the problem and enable the model to be more efficiently trained with given amount of computing power and data. This rule of thumb should be applied in future music transcription research; with the computing power and data availability becoming exponentially more abundant in the future, we can expect increased benefits of supplying the prior knowledge to task-specific models.

2 Future Research Directions

We discuss in this section how the automatic music transcription approaches proposed in this thesis can be extended and further improved in future research. An immediate future research direction is to combine the findings of Chapter VI

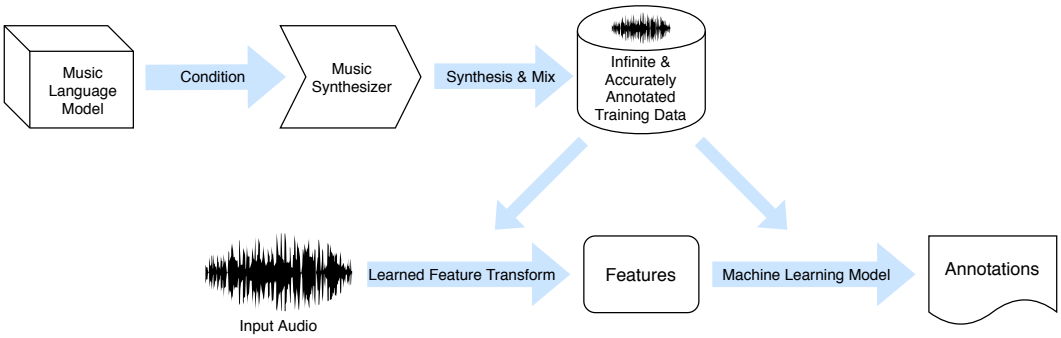


Figure 30: A combination of music language model and music synthesizer can be used as an infinite source of accurately annotated training data for a transcription model.

and Chapter VII to create a music transcription system with both an adversarial discriminator and an appended synthesizer component. Although left as out-of-scope in this thesis because of the computational constraints, this will in theory provide better regularization for the overall system and be able to achieve better performance in music transcription.

Data augmentation is a technique that can provide additional improvement in most music analysis tasks (McFee et al., 2015). While the conventional methods such as time stretch and pitch shift can be used for the purpose of music data augmentation, expressive music language models and music synthesis models can provide more powerful means of enriching the distribution of music data for training a transcription model. This idea is illustrated in Figure 30; an unlimited source of training data can be created using a combination of a powerful music language model and a good music synthesizer, which can then be used to train music analysis models such as a music transcription model. When this pipeline can be successfully implemented, it is possible to scale the downstream model capacity and therefore improve the state-of-the-art performance in its analysis task, since the dataset size has been the main limitation in the recent data-driven music analysis models. This idea is particularly promising

considering the incredible performance of the recent transformer-based language models (Vaswani et al., 2017) and generative pre-training approaches based on them (Radford et al., 2018; Devlin et al., 2018), which have been shown to be also effective in producing coherent symbolic music (C.-Z. A. Huang et al., 2019).

More powerful computational hardware will eventually enable a longer context length to be used in larger transformer models, which suggests another intriguing future research direction: to learn a music language model in the audio waveform domain. This will help create a more expressive synthesizer component, which can be used to augmenting a music transcription model. A compression method such as VQ-VAE (van den Oord et al., 2017) would be helpful in allowing even longer context length to be used in such language model.

Finally, in the discussion regarding the future of AMT, we need to consider the fundamental limitation on any music transcription task, either automatic or manual, and when it can be said that AMT is solved. Polyphonic music contains a

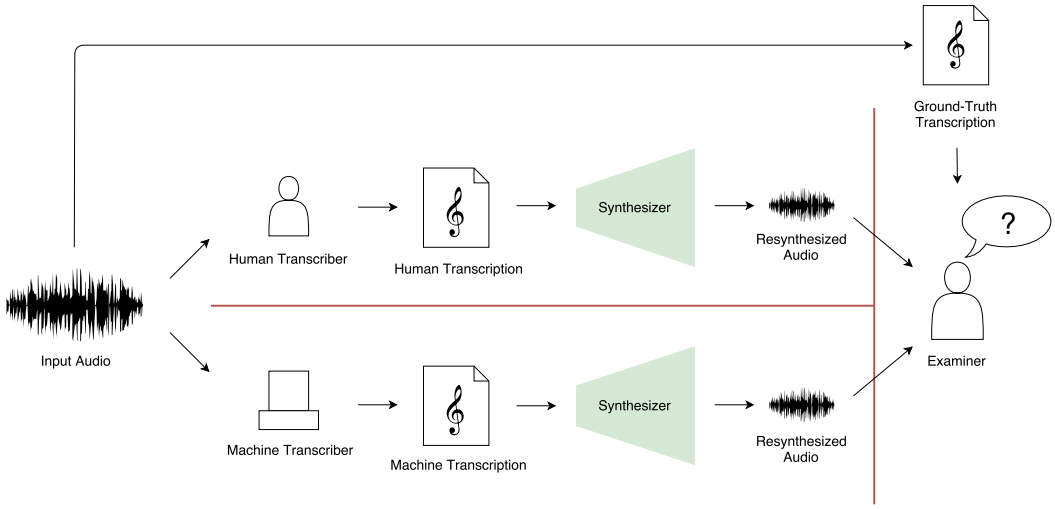


Figure 31: The *transcriptional Turing test*, to test whether an automatic music transcription algorithm has reached human-level. When an automatically generated transcription is indistinguishable by an expert human listener from one produced by a skilled musician, it can be said that AMT is solved.

mixture of sounds with an indefinite number of notes being played simultaneously; even the most experienced musicians may not be able to identify every note. It would be unreasonable to expect anyone to perfectly transcribe all notes in the score of an orchestral piece from an audio file, but it would be sensible for a trained musician to produce a version of the score that, when played by the same orchestra, sounds indistinguishable to the original recording. Considering these limitations, passing this "transcriptional Turing test" as shown in Figure 31, rather than achieving 100% accuracy on a certain dataset, should be the ultimate goal of automatic music transcription, at which point it could be said to mimic human expert-level knowledge on this task.

Considering the astounding progress of deep generative models in the domains such as image generation (Karras et al., 2019) and natural language generation (Radford et al., 2018), it is reasonable to predict that generative modeling will play an important role in the coming decade for enabling a deeper computational understanding of music, to a degree that it becomes difficult to tell apart a machine-generated music transcription from one created by an expert musician.

BIBLIOGRAPHY

Abdallah, S., Alencar-Brayner, A., Benetos, E., Cottrell, S., Dykes, J., Gold, N., ... Wolff, D. (2015). Automatic Transcription and Pitch Analysis of the British Library World and Traditional Music Collection. In *Proceedings of the International Workshop on Folk Music Analysis (FMA)* (pp. 10–12). 10.

Abdallah, S., & Plumbley, M. (2004). Polyphonic Music Transcription by Non-Negative Sparse Coding of Power Spectra. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 23.

Agostini, A., & Ghisi, D. (2013). Real-Time Computer-Aided Composition with bach. *Contemporary Music Review*, *32*(1), 41–48. 10 and 77.

Agostini, G., Longari, M., & Pollastri, E. (2003). Musical Instrument Timbres Classification With Spectral Features. *EURASIP Journal on Advances in Signal Processing*, 97–102. 113.

Argenti, F., Nesi, P., & Pantaleo, G. (2011). Automatic Transcription of Polyphonic Music Based on the Constant-Q Bispectral Analysis. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(6), 1610–1630. 19.

Arjovsky, M., & Bottou, L. (2017). Towards Principled Methods for Training Generative Adversarial Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)* (pp. 1–17). 47 and 51.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. *arXiv preprint arXiv:1701.07875*. 47.

Babacan, O., Drugman, T., D'Alessandro, N., Henrich, N., & Dutoit, T. (2013). A Comparative Study of Pitch Extraction Algorithms on a Large Variety of Singing Sounds. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 7815–7819). 22.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 37.

Barry, S., & Kim, Y. (2018). Style Transfer for Musical Audio Using Multiple Time-Frequency Representations. *OpenReview:BybQ7zWCb*. 28.

Bell, A. J., & Sejnowski, T. J. (1995). An Information-Maximization Approach to Blind Separation and Blind Deconvolution. *Neural Computation, 7*(6), 1129–1159. 27.

Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., & Sandler, M. B. (2005). A Tutorial on Onset Detection in Music Signals. *IEEE Transactions on Speech and Audio Processing, 13*(5), 1035–1046. 19.

Benetos, E., & Dixon, S. (2011). Polyphonic Music Transcription Using Note Onset and Offset Detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 20.

Benetos, E., & Dixon, S. (2012). A Shift-Invariant Latent Variable Model for Automatic Music Transcription. *Computer Music Journal, 36*(4), 81–94. 24.

Benetos, E., & Dixon, S. (2013). Multiple-Instrument Polyphonic Music Ttranscription Using a Temporally Constrained Shift-Invariant Model. *The Journal of the Acoustical Society of America, 133*(3), 1727–1741. 24.

Benetos, E., Dixon, S., Duan, Z., & Ewert, S. (2019). Automatic Music Transcription: An Overview. *IEEE Signal Processing Magazine, 36*(1), 20–30. 23, 26, and 95.

Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., & Klapuri, A. P. (2013). Automatic Music Transcription: Challenges and Future Directions. *Journal of Intelligent Information Systems, 41*(3), 407–434. 2.

Benetos, E., & Weyde, T. (2015). An Efficient Temporally-Constrained Probabilistic Model for Multiple-Instrument Music Transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 701–707). 114 and 115.

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127. 43.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., & Others. (2007). Greedy Layer-Wise Training of Deep Networks. In *Advances in Neural Information Processing Systems (NIPS)* (Vol. 19, p. 153). 40.

Berg-Kirkpatrick, T., Andreas, J., & Klein, D. (2014). Unsupervised Transcription of Piano Music. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 1538–1546). 114 and 117.

Berthelot, D., Schumm, T., & Metz, L. (2017). BEGAN: Boundary Equilibrium Generative Adversarial Networks. *arXiv preprint arXiv:1703.10717*. 48.

Bertin, N., Badeau, R., & Vincent, E. (2009). Fast Bayesian NMF Algorithms Enforcing Harmonicity and Temporal Continuity in Polyphonic Music Transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 24.

Bertin, N., Badeau, R., & Vincent, E. (2010). Enforcing Harmonicity and Smoothness in Bayesian Non-Negative Matrix Factorization Applied to Polyphonic Music Transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(3), 538–549. 24.

Bittner, R. M., Mcfee, B., Salamon, J., Li, P., & Bello, J. P. (2017). Deep Salience Representations for F0 Estimation in Polyphonic Music. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 23–27). 19, 26, 54, 57, 92, and 113.

Bittner, R. M., Salamon, J., Tierney, M., Mauch, M., Cannam, C., & Bello, J. P. (2014). MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (Vol. 14, pp. 155–160). 53, 59, and 67.

Blaauw, M., & Bonada, J. (2017). A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs. *Applied Sciences*, *7*(12), 1313. 30.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022. 42.

Boccardi, F., & Drioli, C. (2001). Sound Morphing With Gaussian Mixture Models. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)* (pp. 6–9). 76.

Böck, S., & Schedl, M. (2011). Enhanced Beat Tracking with Context-Aware Neural Networks. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. 54.

Böck, S., & Schedl, M. (2012). Polyphonic Piano Note Transcription with Recurrent Neural Networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1–4). 26.

Boersma, P. (1993). Accurate Short-Term Analysis of the Fundamental Frequency and the Harmonics-To-Noise Ratio of a Sampled Sound. In *Proceedings of the Institute of Phonetic Sciences* (Vol. 17, pp. 97–110). 22.

Bosch, J. J., & Gómez, E. (2014). Melody Extraction in Symphonic Classical Music: a Comparative Study of Mutual Agreement Between Humans and Algorithms. In *Proceedings of the Conference on Interdisciplinary Musicology*. 53.

Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. (2012). Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In *Proceedings of the International Conference on Machine Learning (ICML)*. 26, 94, and 100.

Bregman, A. S. (1990). *Auditory Scene Analysis*. MIT Press. 27.

Brown, G. J., & Cooke, M. (1994). *Computational Auditory Scene Analysis* (Vol. 8). 27.

Brundage, M., Avin, S., Clark, J., Toner, H., Eckersley, P., Garfinkel, B., ... Amodei, D. (2018). The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation. *arXiv preprint arXiv:1802.07228*. xi and 12.

Burraston, D., Edmonds, E., Livingstone, D., & Miranda, E. R. (2004). Cellular Automata in MIDI based Computer Music. In *Proceedings of the International Computer Music Conference (ICMC)* (Vol. 4, pp. 71–78). 31.

Caetano, M., & Rodet, X. (2010). Automatic Timbral Morphing of Musical Instrument Sounds by High-Level Descriptors. In *Proceedings of the International Computer Music Conference (ICMC)* (pp. 254–261). 76.

Caetano, M., & Rodet, X. (2013). Musical Instrument Sound Morphing Guided by Perceptually Motivated Features. *IEEE Transactions on Audio, Speech, and Language Processing*, *21*(8), 1666–1675. 76.

Camacho, A., & Harris, J. G. (2008). A Sawtooth Waveform Inspired Pitch Estimator for Speech and Music. *The Journal of the Acoustical Society of America*, *124*(3), 1638–1652. 22 and 60.

Carreira-Perpiñán, M. A., & Hinton, G. E. (2005). On Contrastive Divergence Learning. *Artificial Intelligence and Statistics*, 17. 42.

Casey, M. A., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., & Slaney, M. (2008). Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceedings of the IEEE*, *96*(4), 668–696. 16.

Celma, Ó. (2010). *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer. 10.

Cemgil, A. T., Kappen, B., & Barber, D. (2003). Generative Model Based Polyphonic Music Transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 25.

Cemgil, A. T., Kappen, B., Desain, P., & Honing, H. (2000). On Tempo Tracking: Tempogram Representation and Kalman filtering. *Journal of New Music Research*, *29*(1967), 259–273. 19.

Cemgil, A. T., Kappen, H. J., Member, S., Barber, D., Member, S., & Barber, D. (2006). A Generative Model for Music Transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(2), 679–694. 2 and 117.

Chemilier, M. (2001). Improvising Jazz Chord Sequences by Means of Formal Grammars.

Journee d'Informatique Musicale, 121–126. 31.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 2172–2180). 48.

Cheng, T., Mauch, M., Benetos, E., & Dixon, S. (2016). An Attack/Decay Model for Piano Transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 20.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 36, 37, and 119.

Cho, T., Weiss, R. J., & Bello, J. P. (2010). Exploring Common Variations in State-of-the-Art Chord Recognition Systems. *Sound and Music Computing*, 11–22. 20.

Choi, K., & Cho, K. (2019). Deep Unsupervised Drum Transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 117.

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., & Choo, J. (2017). StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. *arXiv preprint arXiv:1711.09020*. 49.

Chollet, F. (2015). *Keras: The Python Deep Learning Library*. *https://keras.io/*. 58.

Chomsky, N. (1966). *Topics in the Theory of Generative Grammar*. Mouton. 31.

Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-Based Models for Speech Recognition. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 577–585). 37.

Cogliati, A., & Duan, Z. (2015). Piano Music Transcription Modeling Note Temporal Evolution. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 429–433). 20.

Cont, A. (2006). Realtime Multiple Pitch Observation using Sparse Non-Negative Constraints. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 206–211). 23.

Cook, P. R. (2002). *Real Sound Synthesis for Interactive Applications*. CRC Press. 11.

Costantini, G., Todisco, M., & Perfetti, R. (2013). NMF Based Dictionary Learning for Automatic Transcription of Polyphonic Piano Music. *WSEAS Transactions on Signal Processing*, *9*(3), 148–157. 23.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2017). Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, 53–65. 96.

Dannenberg, R. B. (1985). An On-Line Algorithm for Real-Time Accompaniment. In *Proceedings of the International Computer Music Conference (ICMC)* (pp. 193–198). 10.

Dannenberg, R. B., & Hu, N. (2003). Polyphonic Audio Matching for Score Following and Intelligent Audio Editors. In *Proceedings of the International Computer Music Conference (ICMC)* (pp. 27–33). 10.

Daskalakis, C., Ilyas, A., Syrgkanis, V., & Zeng, H. (2018). Training GANs with Optimism. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 51.

Davy, M., Godsill, S., & Idier, J. (2006). Bayesian Analysis of Polyphonic Western Tonal Music. *The Journal of the Acoustical Society of America*, *119*(4), 2498–517. 25.

Davy, M., & Godsill, S. J. (2003). Bayesian Harmonic Models for Musical Signal Analysis. *Bayesian Statistics*, 1–16. 25.

de Cheveigné, A., & Kawahara, H. (2002). YIN, a Fundamental Frequency Estimator for Speech and Music. *The Journal of the Acoustical Society of America*, *111*(4), 1917–1930. 22 and 54.

Denton, E. L., Chintala, S., Szlam, A., Fergus, R., & Others. (2015). Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 1486–1494). 46.

Dessein, A., Cont, A., & Lemaitre, G. (2010). Real-Time Polyphonic Music Transcription With Non-Negative Matrix Factorization and Beta-Divergence. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 489–494). 24.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*. 136.

Ding, C., Li, T., & Peng, W. (2008). On the Equivalence Between Non-Negative Matrix Factorization and Probabilistic Latent Semantic Indexing. *Computational Statistics and Data Analysis*, *52*(8), 3913–3927. 24.

Dixon, S. (2000). On the Computer Recognition of Solo Piano Music. In *Proceedings of the Australasian Computer Music Conference* (pp. 31–37). 23.

Doersch, C. (2016). Tutorial on Variational Autoencoders. *arXiv preprint arXiv:1606.05908*. 45.

Donahue, C., Li, B., & Prabhavalkar, R. (2017). Exploring Speech Enhancement with Generative Adversarial Networks for Robust Speech Recognition. *arXiv preprint arXiv:1711.05747*. 29.

Donahue, C., McAuley, J., & Puckette, M. (2018). Synthesizing Audio with Generative Adversarial Networks. *arXiv preprint arXiv:1802.04208*. 30 and 47.

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., & Yang, Y.-H. (2017). MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 34–41). 96.

Dosovitskiy, A., & Brox, T. (2016). Generating Images with Perceptual Similarity Metrics based on Deep Networks. *Advances in Neural Information Processing Systems (NIPS)*, *1*(c), 1–9. 93.

Downie, J. S., Hu, X., Lee, J. H., Choi, K., Cunningham, S. J., & Hao, Y. (2014). Ten Years of MIREX: Reflections, Challenges, and Opportunities. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 657–662). 22.

Dozat, T. (2016). Incorporating Nesterov Momentum into Adam. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 39.

Duan, Z., Pardo, B., & Zhang, C. (2010). Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(8), 2121–2133. 67.

Dubnowski, J. J., Schafer, R. W., & Rabiner, L. R. (1976). Real-Time Digital Hardware Pitch Detector. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *24*(1), 2–8. 21.

Dubois, C., & Davy, M. (2005). Harmonic Tracking Using Sequential Monte Carlo. In *IEEE/SP Workshop on Statistical Signal Processing* (pp. 1292–1297). 117.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(Jul), 2121–2159. 39.

Durrieu, J. L., David, B., & Richard, G. (2011). A Musically Motivated Mid-Level Representation for Pitch Estimation and Musical Audio Source Separation. *IEEE Journal on Selected Topics in Signal Processing*, *5*(6), 1180–1191. 28.

Emiya, V., Badeau, R., & David, B. (2010). Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(6), 1643–1654. 21 and 83.

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., & Roberts, A. (2019).

GANSynth: Adversarial Neural Audio Synthesis. *arxiv preprint arXiv:1902.08710*. 30 and 96.

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., & Norouzi, M. (2017). Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. *arXiv preprint arXiv:1704.01279*. 67, 73, and 78.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., & Bengio, S. (2010). Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research*, *11*(Feb), 625–660. 40.

Esling, P., Chemla-Romeu-Santos, A., & Bitton, A. (2018). Bridging Audio Analysis, Perception and Synthesis with Perceptually-Regularized Variational Timbre Spaces. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 77.

Everitt, B. S., & Hand, D. J. (1981). *Finite Mixture Distributions*. Chapman and Hall. 41.

Ewert, S., Pardo, B., Müller, M., & Plumbley, M. D. (2014). Score-Informed Source Separation for Musical Audio Recordings: An Overview. *IEEE Signal Processing Magazine*, 116–124. 28.

Ewert, S., & Sandler, M. (2016). Piano Transcription in the Studio Using an Extensible Alternating Directions Framework. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(11), 1983–1997. 24.

Ewert, S., & Sandler, M. B. (2017). An Augmented Lagrangian Method for Piano Transcription Using Equal Loudness Thresholding and LSTM-Based Decoding. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (pp. 146–150). 24.

Ezzat, T., Meyers, E., Glass, J. R., & Poggio, T. (2005). Morphing Spectral Envelopes Using Audio Flow. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)* (pp. 2545–2548). 76.

Fan, Z.-C., Lai, Y.-L., & Jang, J.-S. R. (2017). SVSGAN: Singing Voice Separation via Generative Adversarial Network. *arXiv preprint arXiv:1710.11428*. 29.

Farbood, M., & Schoner, B. (2001). Analysis and Synthesis of Palestrina-Style Counterpoint Using Markov Chains. In *Proceedings of the International Computer Music Conference (ICMC)*. 31.

Fedus, W., Rosca, M., Lakshminarayanan, B., Dai, A. M., Mohamed, S., & Goodfellow, I. (2018). Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence At Every Step. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 51.

Fernández, J. D., & Vico, F. (2013). AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research, 48,* 513–582. 11 and 31.

Fuentes, B., Badeau, R., & Richard, G. (2013). Harmonic Adaptive Latent Component Analysis of Audio and Application to Music Transcription. *IEEE Transactions on Audio, Speech, and Language Processing*. 24.

Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A Neural Algorithm of Artistic Style. *arXiv preprint arXiv:1508.06576*. 28 and 36.

Gaussier, E., & Goutte, C. (2005). Relation between PLSA and NMF and Implications. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 601–602). 24.

Ghias, A., Logan, J., Chamberlin, D., & Smith, B. C. (1995). Query By Humming: Musical Information Retrieval in An Audio Database. *Proceedings of the ACM International Conference on Multimedia, 6,* 110–113. 10.

Glorot, X., & Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 41.

Goertzel, B., & Pennachin, C. (2007). *Artificial General Intelligence*. Springer. 33.

Goodfellow, I. (2016). NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv preprint arXiv:1701.00160*. 96 and 100.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 2672–2680). xi, 12, 45, 96, 97, 100, and 102.

Goto, M., Hashiguchi, H., Nishimura, T., & Oka, R. (2002). RWC Music Database: Popular, Classical and Jazz Music Databases. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 287–288). 59.

Grace, K., Salvatier, J., Dafoe, A., Zhang, B., & Evans, O. (2018). Viewpoint: When Will AI Exceed Human Performance? Evidence from AI Experts. *Journal of Artificial Intelligence Research, 62,* 729–754. 33.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). DRAW: A Recurrent Neural Network For Image Generation. In *Proceedings of the International Conference on Machine Learning (ICML)*. 43.

Gregor, K., Danihelka, I., Mnih, A., Blundell, C., & Wierstra, D. (2014). Deep AutoRegressive Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*. 43.

Grey, J. M. (1977). Multidimensional Perceptual Scaling of Musical Timbres. *The Journal of the Acoustical Society of America, 61*(5), 1270–7. 77.

Griffin, D., & Lim, J. S. (1984). Signal Estimation from Modified Short-Time Fourier transform. *IEEE Transactions on Audio, Speech, and Language Processing, 32*(2), 236–243. 18.

Grindlay, G., & Ellis, D. P. W. (2009). Multi-Voice Polyphonic Music Transcription Using Eigeninstruments. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (pp. 53–56). 24, 114, and 115.

Grindlay, G., & Ellis, D. P. W. (2010). A Probabilistic Subspace Model for Multi-Instrument Polyphonic Transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 21–26). 24.

Grosche, P., Muller, M., & Kurth, F. (2010). Cyclic Tempogram - a Mid-Level Tempo Representation for Music Signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 19.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems (NIPS)* (Vol. 30, pp. 5769–5779). 47.

Hamanaka, M., Hirata, K., & Tojo, S. (2013). Computational Music Theory and Its Applications to Expressive Performance and Composition. In *Guide to Computing for Expressive Music Performance* (pp. 205–234). 31.

Harte, C. A., & Sandler, M. B. (2005). Automatic Chord Identifcation using a Quantised Chromagram. In *Proceedings of the AES Convention* (pp. 1–21). 19.

Harte, C. A., Sandler, M. B., & Gasser, M. (2006). Detecting Harmonic Change in Musical Audio. In *Proceedings of the ACM workshop on Audio and Music Computing Multimedia* (p. 21). 19.

Hartmann, W. M. (1997). *Signals, Sound, and Sensation*. Springer. 21.

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., ... Eck, D. (2018). Onsets and Frames: Dual-Objective Piano Transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 26, 79, 92, 93, 94, 95, 102, 103, 104, 114, 118, and 121.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-z. A., Dieleman, S., ... Brain, G. (2019). Enabling Factorized Piano Music Modeling and Generation with the MAESTRO dataset. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 78, 89, 103, 104, and 110.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing

Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp. 1026–1034). 39 and 41.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). 36.

Heittola, T., Klapuri, A., & Virtanen, T. (2009). Musical Instrument Recognition in Polyphonic Audio Using Source-Filter Model for Sound Separation. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 327–332). 28 and 115.

Hershey, J. R., Chen, Z., Le Roux, J., & Watanabe, S. (2016). Deep Clustering: Discriminative Embeddings for Segmentation and Separation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 31–35). IEEE. 116.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*. 50 and 100.

Hinton, G. (2012). *rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude*. 39.

Hinton, G., Osindero, S., & Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation, 18*(7), 1527–1554. 26 and 42.

Hinton, G. E., Sejnowski, T. J., & Ackley, D. H. (1984). *Boltzmann Machines: Constraint Satisfaction Networks that Learn* (Vol. 9; Tech. Rep. No. CMS-CS-84-119). 42.

Hjelm, R. D., Jacob, A. P., Che, T., Cho, K., & Bengio, Y. (2018). Boundary-Seeking Generative Adversarial Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 49.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735–1780. 36.

Hofmann, T. (1999). Probabilistic Latent Semantic Analysis. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)* (pp. 289–296). 24.

Hsu, C.-L., & Jang, J. S. R. (2010). On the Improvement of Singing Voice Separation for Monaural Recordings Using the MIR-1K Dataset. *IEEE Transactions on Audio, Speech, and Language Processing, 18*(2), 310–319. 67.

Hua, K. (2018). Do WaveNets Dream of Acoustic Waves? *arXiv preprint arXiv:1802.08370*. 29.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Hawthorne, C., Dai, A. M., … Eck, D. (2019). Music Transformer. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 32 and 136.

Huang, P. S., Chen, S. D., Smaragdis, P., & Hasegawa-Johnson, M. (2012). Singing-Voice Separation from Monaural Recording using Robust Principal Component Analysis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 57–60). 28.

Humphrey, E. J., & Bello, J. P. (2012). Rethinking Automatic Chord Recognition with Convolutional Neural Networks. In *Proceedings of the International Conference on Machine Learning and Applications and Workshops (ICMLA)* (Vol. 2, pp. 357–362). 54.

Humphrey, E. J., Glennon, A. P., & Bello, J. P. (2011). Non-Linear Semantic Embedding for Organizing Large Instrument Sample Libraries. In *Proceedings of the International Conference on Machine Learning and Applications and Workshops (ICMLA)* (Vol. 2, pp. 142–147). 77.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 448–456). 40 and 58.

Isik, Y., Le Roux, J., Chen, Z., Watanabe, S., & Hershey, J. R. (2016). Single-Channel Multi-Speaker Separation Using Deep Clustering. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)* (pp. 545–549). 116.

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1125–1134). 49, 98, 100, and 103.

Itoyama, K., Goto, M., Komatani, K., Ogata, T., & Okuno, H. G. (2011). Simultaneous Processing of Sound Source Separation and Musical Instrument Identification using Bayesian Spectral Modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3816–3819). IEEE. 115.

Jansson, A., Humphrey, E., Montecchio, N., Bittner, R., Kumar, A., & Weyde, T. (2017). Singing Voice Separation With Deep U-Net Convolutional Networks. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 745–751). 28.

Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., & Yan, S. (2015). Deep Learning with S-shaped Rectified Linear Activation Units. *arXiv preprint arXiv:1512.07030*. 39.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., … Kavukcuoglu, K. (2018). Efficient Neural Audio Synthesis. *arXiv preprint*

arXiv:1802.08435. 30 and 43.

Kalingeri, V., & Grandhe, S. (2016). Music Generation with Deep Learning. *arXiv preprint arXiv:1612.04928*. 29.

Karpathy, A., & Fei-Fei, L. (2017). Deep Visual-Semantic Alignments for Generating Image Descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 37.

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *Proceedings of the International Conference on Learning Representations (ICLR)*. xi, 12, and 47.

Karras, T., Laine, S., & Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4401–4410). xi, 12, 96, and 137.

Kassler, M. (1966). Toward Musical Information Retrieval. *Perspectives of New Music*, *4*(2), 59. 16.

Kelz, R., Dorfer, M., Korzeniowski, F., Böck, S., Arzt, A., & Widmer, G. (2016). On the Potential of Simple Framewise Approaches to Piano Transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. xii, 26, 92, and 96.

Kelz, R., & Widmer, G. (2017). An Experimental Analysis of the Entanglement Problem in Neural-Network-based Music Transcription Systems. In *Proceedings of the AES Conference on Semantic Audio*. 27.

Khadkevich, M., & Omologo, M. (2009). Use of Hidden Markov Models and Factored Language Models for Automatic Chord Recognition. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 561–566). 20.

Kim, J. W., & Bello, J. P. (2019). Adversarial Learning for Improved Onsets and Frames Music Transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 92.

Kim, J. W., Bittner, R., Kumar, A., & Bello, J. P. (2019). Neural Music Synthesis for Flexible Timbre Control. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 176–180). 75, 86, and 110.

Kim, J. W., Salamon, J., Li, P., & Bello, J. P. (2018). CREPE: A Convolutional Representation for Pitch Estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 53 and 92.

Kim, T., Cha, M., Kim, H., Lee, J. K., & Kim, J. (2017). Learning to Discover Cross-Domain

Relations with Generative Adversarial Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*. 49.

Kindermann, R., & Snell, J. L. (1980). *Markov Random Fields and Their Applications* (Vol. 16) (No. 4). American Mathematical Society. 42.

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)* (pp. 1–15). 39, 58, 103, and 122.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improving Variational Inference with Inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems (NIPS)*. 45.

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 44 and 77.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-Normalizing Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*. 40.

Klapuri, A. P. (2003). Multiple Fundamental Frequency Estimation Based on Harmonicity and Spectral Smoothness. *IEEE Transactions on Speech and Audio Processing*, *11*(6), 804–816. 20.

Klapuri, A. P., & Davy, M. (2006). *Signal Processing Methods for Music Transcription*. Springer. 2.

Kleene, S. C. (1951). *Representation of Events in Nerve Nets and Finite Automata*. 33.

Kodali, N., Abernethy, J., Hays, J., & Kira, Z. (2017). On Convergence and Stability of GANs. *arXiv preprint arXiv:1705.07215*. 51.

Koushik, J., & Hayashi, H. (2016). Improving Stochastic Gradient Descent with Feedback. *arXiv preprint arXiv:1611.01505*. 39.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 1097–1105). 36.

Larochelle, H., & Murray, I. (2011). The Neural Autoregressive Distribution Estimator. In *Proceedings of the International Conference on Machine Learning (ICML)* (Vol. 15, pp. 29–37). 43 and 94.

Le Roux, J., Kameoka, H., Ono, N., & Sagayama, S. (2010). Fast Signal Reconstruction Frommagnitude Stft Spectrogram Based on Spectrogram Consistency. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. 18.

LeCun, Y. (2016). *Unsupervised Learning*. NYU. 11.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, *521*(7553), 436–444. 26 and 95.

LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., ... Vapnik, V. (1995). Comparison of Learning Algorithms for Handwritten Digit Recognition. In *Proceedings of the International Conference on Artificial Neural Networks* (Vol. 60, pp. 53–60). 35.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., ... Shi, W. (2017). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 90.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788–791. 23.

Lee, D. D., & Seung, H. S. (2001). Algorithms for Non-Negative Matrix Factorization. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 556–562). 23 and 95.

Leglaive, S., Badeau, R., & Richard, G. (2016). Multichannel Audio Source Separation with Probabilistic Reverberation Priors. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(12), 2453–2465. 28.

Lerdahl, F., & Jackendoff, R. (1983). *Generative Theory of Tonal Music*. MIT Press. 31.

Li, Q., Myaeng, S. H., & Kim, B. M. (2007). A Probabilistic Music Recommender Considering User Opinions and Audio Features. *Information Processing and Management*, *43*(2), 473–487. 10.

Li, S. (2017). Context-Independent Polyphonic Piano Onset Transcription with an Infinite Training Dataset. *arXiv preprint arXiv:1707.08438*. 117.

Li, Y., & Wang, D. (2007). Separation of Singing Voice From Music Accompaniment for Monaural Recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, *15*(4), 1475–1487. 28.

Linnainmaa, S. (1970). The Representation of the Cumulative Rounding Error of an Algorithm as a Taylor Expansion of the Local Rounding Errors. *Master's Thesis (in Finnish), University of Helsinki*, 6–7. 38.

Liu, M.-Y., & Tuzel, O. (2016). Coupled Generative Adversarial Networks. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 469–477). xi and 12.

Liutkus, A., Rafii, Z., Badeau, R., Pardo, B., & Richard, G. (2012). Adaptive Filtering for Music/Voice Separation Exploiting the Repeating Musical Structure. In *Proceedings of*

the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 53–56). 28.

Logan, B. (2000). Mel Frequency Cepstral Coefficients for Music Modeling. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 19, 67, and 77.

Long, J., Shelhamer, E., & Darrell, T. (2018). Fully Convolutional Networks for Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 6810–6818. 99.

Lostanlen, V., & Cella, C.-E. (2016). Deep Convolutional Networks on the Pitch Spiral for Music Information Recognition. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 612–618). 113.

Lu, C. C., & Tseng, V. S. (2009). A Novel Method for Personalized Music Recommendation. *Expert Systems with Applications*, *36*(6), 10035–10044. 10.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., & Bousquet, O. (2017). Are GANs Created Equal? A Large-Scale Study. *arXiv preprint arXiv:1711.10337*. 48.

Luo, Y., Chen, Z., Hershey, J. R., Le Roux, J., & Mesgarani, N. (2017). Deep Clustering and Conventional Networks for Music Separation: Stronger Together. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 61–65). 116.

Magno, T., & Sable, C. (2008). A Comparison of Signal-Based Music Recommendation To Genre Labels, Collaborative Filtering, Musicological Analysis, Human Recommendation, and Random Baseline. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 161–166). 10.

Mahendran, A., & Vedaldi, A. (2016). Visualizing Deep Convolutional Neural Networks using Natural Pre-Images. *International Journal of Computer Vision*, *120*(3), 233–255. 36.

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., & Smolley, S. P. (2017). Least Squares Generative Adversarial Networks. *Proceedings of the International Conference on Computer Vision (ICCV)*, 2794–2802. 48 and 97.

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., & Smolley, S. P. (2018). On the Effectiveness of Least Squares Generative Adversarial Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 48.

Marolt, M. (1999). A Comparison of Feed Forward Neural Network Architectures for Piano Music Transcription. In *Proceedings of the International Computer Music Conference (ICMC)*. 26.

Marolt, M. (2004). A Connectionist Approach to Automatic Transcription of Polyphonic Piano Music. *IEEE Transactions on Multimedia*, *6*(3), 439–449. 26.

Maron, M. E. (1961). Automatic Indexing: An Experimental Inquiry. *Journal of the ACM*, *8*(3), 404–417. 41.

Martin, K. D. (1996). Automatic Transcription of Simple Polyphonic Music. *The Journal of the Acoustical Society of America*, *100*(399), 2813. 23.

Mauch, M., Cannam, C., Bittner, R., Fazekas, G., Salamon, J., Dai, J., ... Dixon, S. (2015). Computer-Aided Melody Note Transcription Using the Tony Software: Accuracy and Efficiency. In *Proceedings of the First International Conference on Technologies for Music Notation and Representation* (p. 8). 53.

Mauch, M., & Dixon, S. (2014). pYIN: A Fundamental Frequency Estimator Using Probabilistic Threshold Distributions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 659–663). 22, 54, 59, 60, 67, 69, and 94.

Mauch, M., & Ewert, S. (2013). The Audio Degradation Toolbox and Its Application To Robustness Evaluation. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 60.

McFee, B., Humphrey, E. J., & Bello, J. P. (2015). A Software Framework for Musical Data Augmentation. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 248–254). 73, 130, and 135.

McFee, B., Kim, J. W., Cartwright, M., Salamon, J., Bittner, R. M., & Bello, J. P. (2019). Open-Source Practices for Music Signal Processing Research: Recommendations for Transparent, Sustainable, and Reproducible Audio Research. *IEEE Signal Processing Magazine*, *36*(1), 128–137. 66.

Mcfee, B., Raffel, C., Liang, D., Ellis, D. P. W., Mcvicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and Music Signal Analysis in Python. In *Proceedings of the Python in Science Conference* (pp. 18–25). 121.

Mcleod, A., & Steedman, M. (2018). Evaluating Automatic Polyphonic Music Transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 42–49). 27.

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., ... Bengio, Y. (2017). SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 29 and 43.

Mescheder, L., Nowozin, S., & Geiger, A. (2017). The Numerics of GANs. In *Advances in*

Neural Information Processing Systems (NIPS). 51.

Miranda, E. R., Al, J., & Eds, B. (2007). *Evolutionary Computer Music*. 31.

Miron, M. (2018). Source Separation Methods for Orchestral Music: Timbre-Informed and Score-Informed Strategies (Doctoral dissertation, Pompeu Fabra University, Barcelona).
28.

Mirza, M., & Osindero, S. (2014). Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*. 48 and 97.

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral Normalization for Generative Adversarial Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 47.

Miyato, T., & Koyama, M. (2018). cGANs with Projection Discriminator. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 49.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*. 37.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Ostrovski, G. (2015). Human-Level Control Through Deep Reinforcement Learning. *Nature*, *518*(7540), 529–533. 37.

Mohamed, S., & Lakshminarayanan, B. (2017). Learning in Implicit Generative Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 51.

Moorer, J. A. (1977). On the Transcription of Musical Sound by Computer. *Computer Music Journal*, *1*(4), 32–38. 2 and 23.

Mor, N., Wolf, L., & Polyak, A. (2019). A Universal Music Translation Network. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 78.

Nagarajan, V., & Kolter, J. Z. (2017). Gradient Descent GAN Optimization is Locally Stable. In *Advances in Neural Information Processing Systems (NIPS)*. 51.

Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 807–814). 39.

Nam, J., Ngiam, J., Lee, H., & Slaney, M. (2011). A Classification-Based Polyphonic Piano Transcription Approach Using Learned Feature Representations. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 175–180). 25 and 26.

Nayebi, A., & Vitelli, M. (2015). GRUV: Algorithmic Music Generation using Recurrent Neural Networks. *Stanford CS224d Class Project*. 29.

Nesterov, Y. (1983). *A Method for Solving the Convex Programming Problem with Convergence Rate O(1/k⊠)* (Vol. 269). 39.

Ng, A. (2011). Sparse Autoencoder. *Stanford CS294a Lecture notes*. 43.

Ni, Y., McVicar, M., Santos-Rodriguez, R., & De Bie, T. (2012). An End-to-End Machine Learning System for Harmonic Analysis of Music. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(6), 1771–1783. 93 and 94.

Niedermayer, B. (2008). Non-Negative Matrix Division For The Automatic Transcription Of Polyphonic Music. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 544–549). 24.

Noll, A. M. (1967). Cepstrum Pitch Determination. *The Journal of the Acoustical Society of America*, *41*(2), 293–309. 21.

Nowozin, S., & Cseke, B. (2016). f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems (NIPS)*. 47.

Odena, A., Olah, C., & Shlens, J. (2016). Conditional Image Synthesis With Auxiliary Classifier GANs. *arXiv preprint arXiv:1610.09585*. 48.

Ono, N., Miyamoto, K., Kameoka, H., & Roux, J. L. (2010). Harmonic and Percussive Sound Separation and Its Application to MIR-Related Tasks. In *Advances in Music Information Retrieval* (pp. 213–236). Springer. 28.

Oramas, S., Nieto, O., Barbieri, F., & Serra, X. (2017). Multi-Label Music Genre Classification from Audio, Text, and Images Using Deep Features. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 23–30). 19 and 20.

Orio, N., Lemouton, S., Schwarz, D., & Schnell, N. (2003). Score Following: State of the Art and New Developments. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)* (pp. 36–41). 10.

Osaka, N. (1995). Timbre Interpolation of Sounds using a Sinusoidal Model. In *Proceedings of the International Computer Music Conference (ICMC)* (pp. 408–411). 76.

Oudre, L., Grenier, Y., & Févotte, C. (2009). Template-Based Chord Recognition: Influence of the Chord Types. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 153–158). 20.

Ozerov, A., Philippe, P., Bimbot, F., & Gribonval, R. (2007). Adaptation of Bayesian Models

for Single-Channel Source Separation and its Application to Voice/Music Separation in Popular Songs. *IEEE Transactions on Audio, Speech, and Language Processing*, *15*(5), 1564–1578. 28.

Pascual, S., Bonafonte, A., & Serrà, J. (2017). SEGAN: Speech Enhancement Generative Adversarial Network. *arXiv preprint arXiv:1703.09452*. 29 and 49.

Patel, A. D. (2008). *Music, Language, and the Brain*. Oxford university press. 31.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Vol. 8) (No. 1). 41.

Peeling, P. H., Cemgil, A. T., & Godsill, S. J. (2010). Generative Spectrogram Factorization Models for Polyphonic Piano Transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(3), 519–527. 24.

Peeters, G., Giordano, B. L., Susini, P., Misdariis, N., & McAdams, S. (2011). The Timbre Toolbox: Extracting Audio Descriptors from Musical Signals. *The Journal of the Acoustical Society of America*, *130*(5), 2902–2916. 77.

Pejrolo, A., & Metcalfe, S. B. (2017). *Creating sounds from scratch: A Practical guide to music synthesis for producers and composers*. Oxford University Press. 76.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., & Courville, A. (2017). FiLM: Visual Reasoning with a General Conditioning Layer. *Proceedings of the AAAI Conference on Artificial Intelligence*. 80 and 119.

Pesek, M., Leonardis, A., & Marolt, M. (2017). Robust Real-Time Music Transcription with a Compositional Hierarchical Model. *PLoS ONE*, *12*(1). 25.

Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., ... Miller, J. (2018). Deep Voice 3: 2000-Speaker Neural Text-to-Speech. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 78 and 79.

Piszczalski, M., & Galler, B. A. (1977). Automatic Music Transcription. *Computer Music Journal*, *1*(4), 24–31. 2 and 23.

Plumbley, M. D., Abdallah, S. A., Bello, J. P., Davies, M. E., Monti, G., & Sandler, M. B. (2002). Automatic Music Transcription and Audio Source Separation. *Cybernetics and Systems: An International Journal*, *6*, 603–627. 27.

Poliner, G. E., & Ellis, D. P. (2006). A Discriminative Model for Polyphonic Piano Transcription. *EURASIP Journal on Advances in Signal Processing*, 1–16. 25, 93, and 105.

Polyak, B. T. (1964). Some Methods of Speeding Up the Convergence of Iteration Methods. *USSR Computational Mathematics and Mathematical Physics*, *4*(5), 1–17. 39.

Pons, J., Nieto, O., Prockup, M., Schmidt, E. M., Ehmann, A. F., & Serra, X. (2018). End-to-End Learning for Music Audio Tagging at Scale. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 20.

Prenger, R., Valle, R., & Catanzaro, B. (2019). WaveGlow : A Flow-Based Generative Network for Speech Synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3617–3621). 30 and 89.

Rabiner, L. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, *77*(2), 257–286. 41.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434v2*, 1–16. xi, 12, and 46.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. *OpenAI Blog*. 136 and 137.

Raffel, C., Mcfee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., & Ellis, D. P. W. (2014). mir_eval: A Transparent Implementation of Common MIR Metrics. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 367–372). 60 and 105.

Rafii, Z., & Pardo, B. (2013). REpeating pattern extraction technique (REPET): A simple method for music/voice separation. *IEEE Transactions on Audio, Speech, and Language Processing*, *21*(1), 71–82. 28.

Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marcal, A. R., Guedes, C., & Cardoso, J. S. (2012). Optical Music Recognition State-of-the-Art and Open Issues. *International Journal of Multimedia Information Retrieval*, *1*(3), 173–190. 31.

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the Convergence of Adam and Beyond. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 39.

Rezende, D. J., & Mohamed, S. (2015). Variational Inference with Normalizing Flows. In *Proceedings of the International Conference on Machine Learning (ICML)* (Vol. 37). 45.

Riedmiller, M. (1994). Advanced Supervised Learning in Multi-layer Perceptrons - From Backpropagation to Adaptive Learning Algorithms. *Computer Standards and Interfaces*, *16*, 265–278. 35.

Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 833–840). 44.

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*(3), 400–407. 38.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (pp. 234–241). Springer International Publishing. 28.

Rosenblatt, F. (1957). *The Perceptron, a Perceiving and Recognizing Automaton (Project Para)*. Cornell Aeronautical Laboratory. 34.

Ross, M. J., Shaffer, H. L., Cohen, A., Freudberg, R., & Manley, H. J. (1974). Average Magnitude Difference Function Pitch Extractor. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *ASSP-22*(5), 353–362. 22.

Rowe, R. (2003). *Machine Musicianship*. 14.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Chapter 8. Learning Internal Representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations* (pp. 354–361). 36.

Rumelhart, D. E., Hinton, G. E., & Williams., R. J. (1986). Learning Representations by Back-Propagating Errors. *Nature, 323*(6088), 533–538. 38.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision, 115*(3), 211–252. 49 and 50.

Russell, S. J., & Norvig, P. (2009). *Artificial Intelligence: a Modern Approach (3rd Edition)*. Pearson. 13.

Saito, S., Kameoka, H., Takahashi, K., Nishimoto, T., & Sagayama, S. (2008). Specmurt Analysis of Polyphonic Music Signals. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(3), 639–650. 19.

Sakaue, D., Itoyama, K., Ogata, T., & Okuno, H. (2013). Robust Multipitch Analyzer against Initialization based on Latent Harmonic Allocation using Overtone Corpus. *Journal of Information Processing*, *21*(2), 246–255. 25.

Salamon, J., Bittner, R. M., Bonada, J., Bosch, J. J., Gomez, E., & juan pablo Bello. (2017). An Analysis/Synthesis Framework for Automatic F0 Annotation of Multitrack Datasets. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 71–78). 54, 59, and 67.

Salamon, J., Gómez, E., Ellis, D. P. W., & Richard, G. (2014). Melody Extraction from Polyphonic Music Signals: Approaches, Applications, and Challenges. *IEEE Signal Processing Magazine, 31*(2), 118–134. 60.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016).

Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 2226–2234). 46 and 50.

Saruwatari, H., Kawamura, T., Nishikawa, T., Lee, A., & Shikano, K. (2006). Blind Source Separation Based on a Fast-Convergence Algorithm Combining ICA and Beamforming. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(2), 666–678. 28.

Schörkhuber, C., & Klapuri, A. P. (2010). Constant-Q Transform Toolbox for Music Processing. In *Proceedings of the Sound and Music Computing (SMC) Conference* (pp. 3–64). 19.

Shelhamer, E., Long, J., & Darrell, T. (2017). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 640–651. 36.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., ... Wu, Y. (2018). Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2–6). 30, 78, 79, and 110.

Shepard, R. N. (1964). Circularity in Judgments of Relative Pitch. *The Journal of the Acoustical Society of America*, *36*(12), 2346–2353. 9.

Sigtia, S., Benetos, E., Boulanger-Lewandowski, N., Weyde, T., d'Avila Garcez, A. S., & Dixon, S. (2015). A Hybrid Recurrent Neural Network For Music Transcription. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 26.

Sigtia, S., Benetos, E., Cherla, S., Weyde, T., d'Avila Garcez, a., & Dixon, S. (2014). An RNN-based Music Language Model for Improving Automatic Music Transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 53–58). 32.

Sigtia, S., Benetos, E., & DIxon, S. (2016). An End-to-End Neural Network for Polyphonic Piano Music Transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(5), 927–939. 26, 93, and 100.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Others (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, *529*(7587), 484–489. 37.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*. 36.

Slaney, M., Covell, M., & Lassiter, B. (1996). Automatic Audio Morphing. In *Proceedings of*

the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1001–1004). 76.

Smaragdis, P. (2009). Relative-Pitch Tracking of Multiple Arbitrary Sounds. *The Journal of the Acoustical Society of America, 125*(5), 3406–13. 24.

Smaragdis, P., & Brown, J. C. (2003). Non-Negative Matrix Factorization for Polyphonic Music Transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (pp. 177–180). 23.

Smaragdis, P., Raj, B., & Shashanka, M. (2006). A Probabilistic Latent Variable Model for Acoustic Modeling. In *Advances in Neural Information Processing Systems (NIPS)*. 24.

Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., ... Wattenberg, M. (2019). TensorFlow.js: Machine Learning for the Web and Beyond. In *Proceedings of the SysML Conference*. 70.

Smolensky, P. (1986). Chapter 6. Information Processing in Dynamical Systems: Foundations of Harmony Theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations* (pp. 194–281). 42.

Sønderby, C. K., Caballero, J., Theis, L., Shi, W., & Huszár, F. (2017). Amortised MAP Inference for Image Super-resolution. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 49 and 51.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research, 15*(1), 1929–1958. 40, 58, and 100.

Stoller, D., Ewert, S., & Dixon, S. (2017). Adversarial Semi-Supervised Audio Source Separation applied to Singing Voice Extraction. *arXiv preprint arXiv:1711.00048*. 29.

Sturm, B. L. (2013). Classification Accuracy is Not Enough: On the Evaluation of Music Genre Recognition Systems. *Journal of Intelligent Information Systems*, *41*(3), 371–406. 59 and 130.

Subakan, C., & Smaragdis, P. (2017). Generative Adversarial Source Separation. *arXiv preprint arXiv:1710.10779*. 29.

Sukhbaatar, S., Szlam, A., Weston, J., & Fergus, R. (2015). End-To-End Memory Networks. In *Advances in Neural Information Processing Systems (NIPS)*. 37.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the Importance of Initialization and Momentum in Deep Learning. *30th International Conference on Machine Learning, ICML 2013*, 2176–2184. 39.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with

neural networks. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 3104–3112). 37.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction (2nd Edition)*. MIT press Cambridge. 37.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–9). 36.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 50.

Talkin, D., Kleijn, W. B., & Paliwal, K. K. (1995). A Robust Algorithm for Pitch Tracking (RAPT). In *Speech Coding and Synthesis* (pp. 495–518). Elsevier Science. 22.

Teng, Y., Zhao, A., & Goudeseune, C. (2017). Generating Nontrivial Melodies for Music as a Service. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 32.

Theis, L., & Bethge, M. (2015). Generative Image Modeling Using Spatial LSTMs. In *Advances in Neural Information Processing Systems (NIPS)*. 43.

Theis, L., van den Oord, A., & Bethge, M. (2016). A Note on the Evaluation of Generative Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 49.

Thickstun, J., Harchaoui, Z., Foster, D., & Kakade, S. M. (2018). Invariances and Data Augmentation for Supervised Music Transcription. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 27 and 133.

Thickstun, J., Harchaoui, Z., & Kakade, S. (2017). Learning Features of Music from Scratch. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 83, 121, and 130.

Thomé, C., & Ahlbäck, S. (2017). Polyphonic Pitch Detection With Convolutional Recurrent Neural Networks. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 26.

Tikhonov, A., & Yamshchikov, I. P. (2017). Music Generation with Variational Recurrent Autoencoder Supported by History. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR)*. 32.

Tolonen, T., & Karjalainen, M. (2000). A computationally efficient multipitch analysis model. *IEEE Transactions on Speech and Audio Processing*, *8*(6), 708–716. 19.

Ullrich, K., & Van Der Wel, E. (2017). Music Transcription With Convolutional Sequence-To-Sequence Models. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*. 26.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. *arXiv preprint arXiv:1609.03499*. 29, 43, 77, 81, and 118.

van den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel Recurrent Neural Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*. 43 and 94.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., & Kavukcuoglu, K. (2016). Conditional Image Generation with PixelCNN Decoders. *Advances in Neural Information Processing Systems (NIPS)*. 94.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., ... Hassabis, D. (2018). Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In *Proceedings of the International Conference on Machine Learning (ICML)*. 89.

van den Oord, A., Vinyals, O., & Kavukcuoglu, K. (2017). Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems (NIPS)*. 45 and 136.

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*(2008), 2579–2605. 88.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is All You Need. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 5999–6009). 136.

Vercoe, B. (1984). The Synthetic Performer in the Context of Live Performance. In *Proceedings of the International Computer Music Conference (ICMC)* (pp. 199–200). 10.

Vincent, E., Bertin, N., & Badeau, R. (2010). Adaptive Harmonic Spectral Decomposition for Multiple Pitch Estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(3), 528–537. 24.

Vincent, E., Bertin, N., Gribonval, R., & Bimbot, F. (2014). From Blind to Guided Audio Source Separation: How Models and Side Information Can Improve the Separation of Sound. *IEEE Signal Processing Magazine*, *31*(3), 107–115. 27.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 1096–1103). 43.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., ... Silver, D. (2019). Grandmaster Level in StarCraft II using Multi-Agent Reinforcement

Learning. *Nature, 575*(7782), 350–354. 1.

Viterbi, A. J. (1967). Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, *13*(2), 260–269. 69.

Vogl, R., Dorfer, M., Widmer, G., & Knees, P. (2017). Drum Transcription via Joint Beat and Drum Modeling using Convolutional Recurrent Neural Networks. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference* (pp. 150–157). 19.

von dem Knesebeck, A., & Zölzer, U. (2010). Comparison of Pitch Trackers for Real-Time Guitar Effects. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. 22.

Wang, Q., Zhou, R., & Yan, Y. (2018). Polyphonic Piano Transcription with a Note-Based Music Language Model. *Applied Sciences*, *8*(3), 470. 26.

Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., ... Others (2017). Tacotron: A Fully End-to-End Text-To-Speech Synthesis Model. *arXiv preprint arXiv:1703.10135*, *2017-Augus*, 4006–4010. 29 and 30.

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, *13*(4), 600–612. 49.

Wang, Z., Simoncelli, E. P., & Bovik, A. C. (2003). Multi-Scale Structural Similarity for Image Quality Assessment. In *Proceedings of the IEEE Asilomar Conference on Signals, Systems and Computers* (Vol. 2, pp. 9–13). 50.

Wang, Z. Q., Roux, J. L., & Hershey, J. R. (2018). Alternative Objective Functions for Deep Clustering. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 686–690). IEEE. 116.

Weninger, F., Kirst, C., & Bungartz, H.-j. (2013). A Discriminative Approach to Polyphonic Piano Note Transcription Using Supervised Non-Negative Matrix Factorization. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6–10). 25.

Werbos, P. J. (1982). Applications of Advances in Nonlinear Sensitivity Analysis. In *System Modeling and Optimization* (pp. 762–770). Springer. 38.

Wessel, D. L. (1979). Timbre Space as a Musical Control Structure. *Computer Music Journal*, 45–52. 77.

Wu, J., Zhang, C., Xue, T., Freeman, W. T., & Tenenbaum, J. B. (2016). Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *Advances in Neural Information Processing Systems (NIPS)*. 47.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*. 37.

Wu, Y.-T., Chen, B., & Su, L. (2019). Polyphonic Music Transcription with Semantic Segmentation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 166–170). 116.

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv preprint arXiv:1505.00853*. 39.

Yang, L.-C., Chou, S.-Y., & Yang, Y.-H. (2017). MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation using 1D and 2D Conditions. *arXiv preprint arXiv:1703.10847*. 32 and 96.

Yeh, C., Roebel, A., & Rodet, X. (2010). Multiple Fundamental Frequency Estimation and Polyphony Inference of Polyphonic Music Signals. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(6), 1116–1126. 21.

Yoshii, K., & Goto, M. (2012). A Nonparametric Bayesian Multipitch Analyzer Based on Infinite Latent Harmonic Allocation. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(3), 717–730. 25.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent Neural Network Regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 109.

Zeiler, M. D. (2012). ADADELTA: an Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*. 39.

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). mixup: Beyond Empirical Risk Minimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 100.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. (2017a). StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks. *arXiv preprint arXiv:1710.10916*. 47 and 49.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. (2017b). StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 49.

Zhang, W., Chen, Z., Member, S., Yin, F., & Zhang, Q. (2018). Melody Extraction From Polyphonic Music Using Particle Filter and Dynamic Programming. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(9), 1620–1632. 113.

Zhao, J., Mathieu, M., & LeCun, Y. (2017). Energy-based Generative Adversarial Network.

In *Proceedings of the International Conference on Learning Representations (ICLR)*. 48.

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2223–2232). 49.

Zubizarreta, M. L. (1998). *Prosody, Focus, and Word Order*. MIT Press. 54.