Symphony2vec:

Learning Musically Meaningful Embeddings for Orchestral Audio Data

Anna Yanchenko (aky5)

December 5, 2019

ECE 590 Final Project

Abstract

- Algorithmic composition is the use of statistical and machine learning models to generate new music
- However, there is currently no standard embedding approach for audio music data
- The **goal** of this project is to develop a musically meaningful latent representation of orchestral audio recordings that could be used in later generative modeling tasks.
- Adversarially learned inference (ALI) models are used to learn this embedding and evaluated on downstream composer classification task
- ALI models do not outperform baseline approaches on this classification task, but are able to recover some high-level musically meaningful features in the latent space.
- Overall, learning a meaningful embedding for audio data remains an open problem and challenging task.

Table of contents

Introduction

Project Motivation: Algorithmic Composition

- Algorithmic composition is the use of statistical and machine learning models to generate new music
- Impressive results have been achieved recently using deep learning approaches [22, 24, 11]
- Successful modeling of orchestral classical music is an important challenge with applications to many disciplines
- Orchestral classical music is a complex, multivariate time series that is highly structured, both temporally (in terms of melody) and vertically (in terms of harmony)
- Music is a long-memory process and labeled/annotated data can be expensive to obtain
- Similar problems are faced in many fields, such as medical applications, making challenges for algorithmic composition generally applicable

Musical Representation

- Despite increased research interest in machine learning for music, there is no clear consensus on the representation to use
- Audio data, in particular, can be challenging to work with and model [22, 11]
- In order to make progress in algorithmic composition, meaningful embedding of musical data must first be developed

Project Goals

The goal of this project is to develop a musically meaningful latent representation of orchestral audio recordings

- Adversarially Learned Inference (ALI) [10] is used to learn a representation for audio recordings
- Input data is mel-spectrogram representations
- Data consists of audio recordings of symphonies from 5 classical composers, recorded by the Berlin Philharmonic
- Learned representations are evaluated on the downstream task of composer classification, with performance comparison to various baselines
- ALI embeddings are compared to baseline embedding approaches like PCA and VAEs

Contributions

- Main contribution: one of first analyses (to my knowledge) of embedding classical *orchestral* audio data
- Much more complex setting than embedding popular songs, symbolic music data, or audio of a single instrument due to dependencies between instruments in orchestral data
- Classical music composer classification also has very little prior work
- Overall, project finds that learning a musically meaningful representation of orchestral audio data is still a challenging open problem that needs further research and is much different from learning about popular music and genre classification
- ALI models appear to be a promising approach with further architecture exploration
- Hopefully this project can serve as a baseline for future orchestral audio embedding work and composer classification

Related Works

Genre Classification

- More general task of genre classification of both symbolic and audio data has been previously considered by many different approaches [4, 7, 16, 21, 23, 27, 29]
- Work on classifying specific composers is more limited [6, 19]
- Most approaches use CNNs with spectrograms or Mel-frequency cepstral coefficients (MFCCs) as input data
- 70-80% accuracy achieved on genre classification problems

Latent Embeddings

Three primary existing approaches to learning latent embeddings for music or audio data.

1. Autoencoders: VAEs [2, 13, 17] and special-purpose hierarchical models for modeling long term structure in symbolic music data [24]
2. Word2vec or Seq2seq: Word2vec models adapted to genre classification [12, 25] and Seq2seq [28] approaches with Encoder-Decoder RNNs explored in the context of speech recognition [9, 8, 3]
    - Use the information at the current time step to predict the audio content within a window, both forwards and backwards in time
3. WaveNet: WaveNet model [22, 18] adapted to an Encoder-Decoder framework [11]
    - [11] is applied to short clips of raw audio data for one instrument and is successful in terms of reconstruction of the input audio signals

Adversarially Learned Inference Models

This project adapts the approach in [25] to learn a latent representation for *classical, orchestral* audio data using Adversarially Learned Inference (ALI) [10] models

- ALI is used by [25] to represent popular audio music, represented as spectrograms, and to classify the genre of these songs
- Want to focus on audio data, so don't consider approaches modeling symbolic data like [24] here
- Approaches like [22, 18, 11] that work on the raw audio data are likely too computationally intensive for this project and are potentially challenging to evaluate and interpret
- [25] find ALI to perform well for genre classification

Project Details

Data

- Project data consists of 10 second audio clips of symphonies from 5 different classical composers (manually compiled data set)
- Composers could be classified with fairly high accuracy by a musically-knowledgeable listener
- All recordings by the Berlin Philharmonic conducted by Herbert von Karajan to control for individual orchestra stylistic differences
- **5366** training examples and **3842** test examples

Table 1: Training and test set class balance for the 5 different composers considered. Both the training and test sets are fairly well balanced.

	Beethoven	Brahms	Haydn	Sibelius	Tchaikovsky
Training Set	0.231	0.115	0.253	0.171	0.230
Test Set	0.229	0.124	0.251	0.184	0.212

Input Data Representations

Three main data representations are explored for this project

1. Mel-Spectrograms - standard representation of audio signals and represents the power spectrum of the audio signal over the entire frequency range (on the mel-scale)
2. Chromagrams - aggregate the amplitude of each frequency bin in the spectrogram across octaves to give one amplitude for each pitch in the twelve tone scale [20]
3. Mel-frequency cepstral coefficients (MFCCs) - calculated by applying binning of the spectral vectors where the binning spacing is based on human perception of frequency [5]

Input Data Representation Details

- All data representations already musically meaningful and interpretable
- However, input representations are also high dimensional and may not be ideal as input features for generative deep learning model
- Temporal resolution for all representations is 0.023 seconds
- Input data dimensions - Chromagram: 12 x 431, MFCC: 13 x 431, Mel-spectrogram: 64 x 431
- All data normalized before training

Chromagram Representation

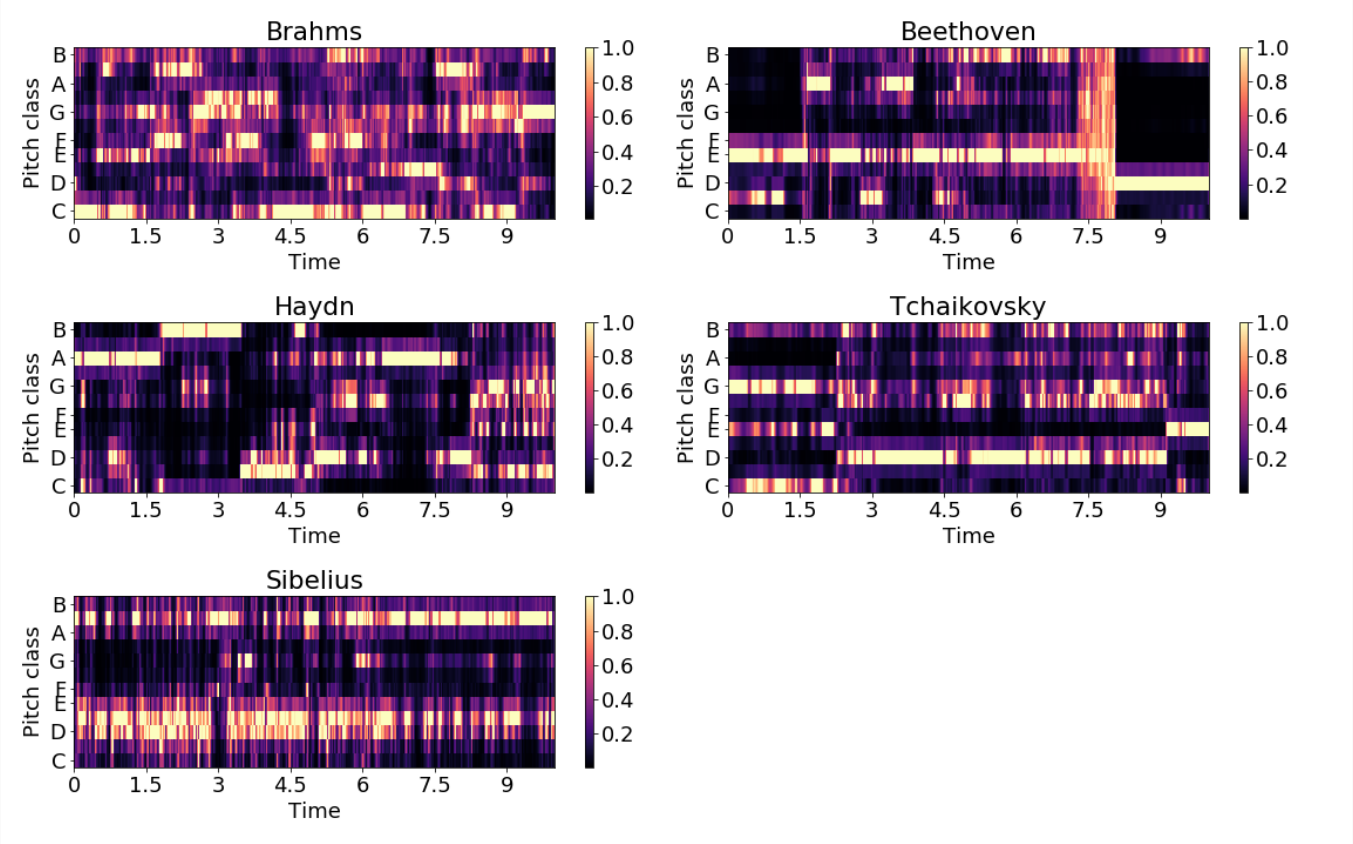

Figure 1: Example chromagrams for each of the 5 composers considered.

MFCCs Representation

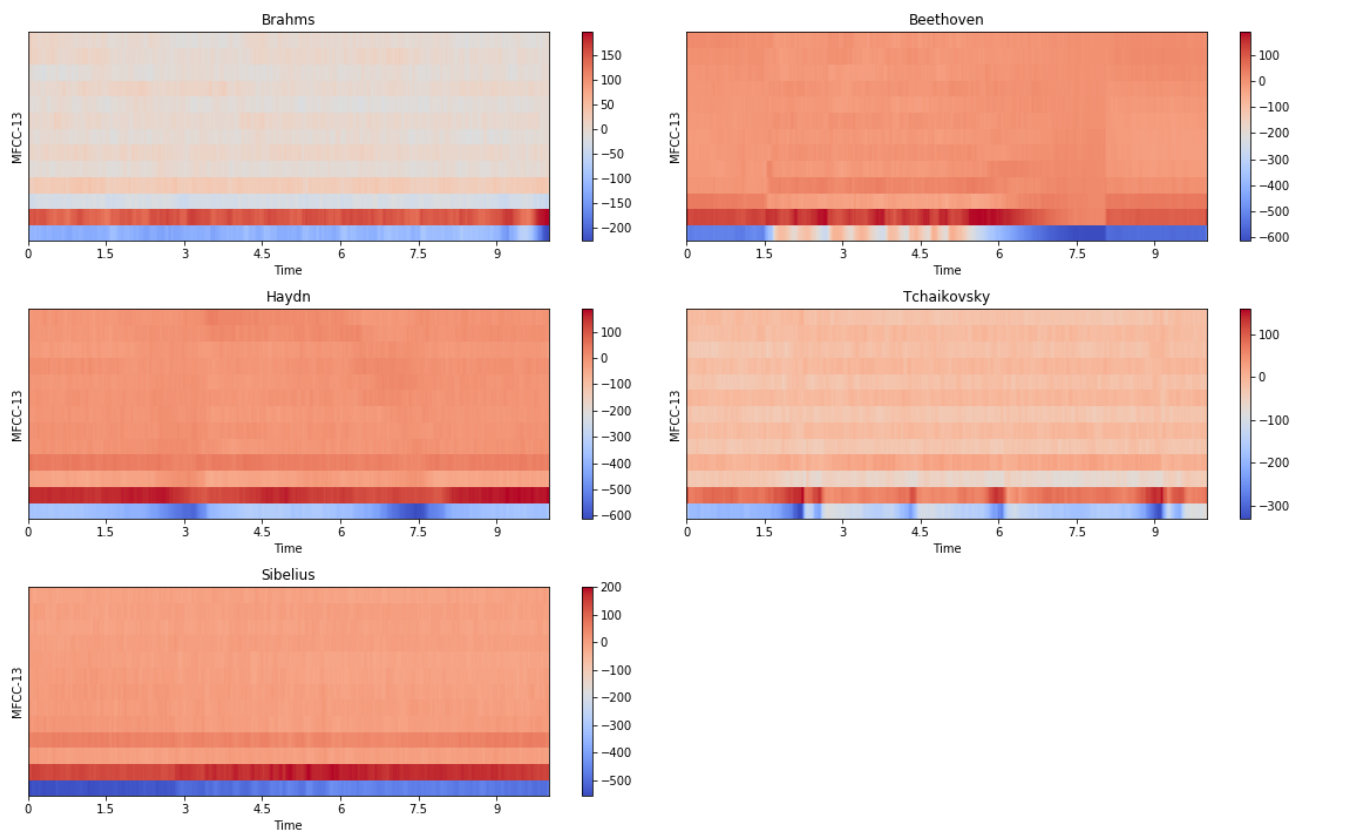

Figure 2: Example MFCC representations for each of the 5 composers considered.

Mel-Spectrograms Representation

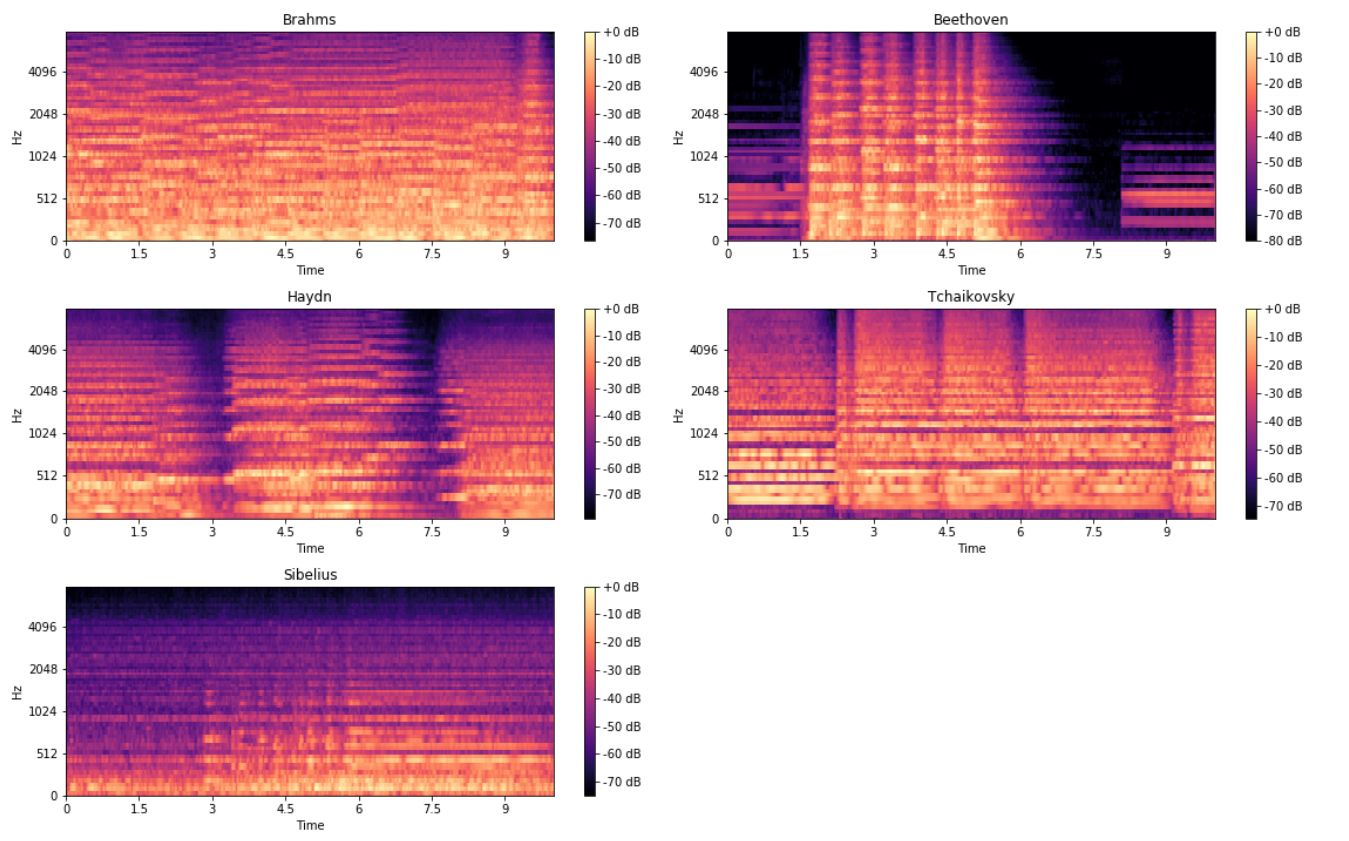

Figure 3: Example mel-spectrograms for each of the 5 composers considered. The intensity is scaled such that 0 dB is the loudest volume and the other intensities are relative to this maximum volume.

PCA Embedding of Input Features

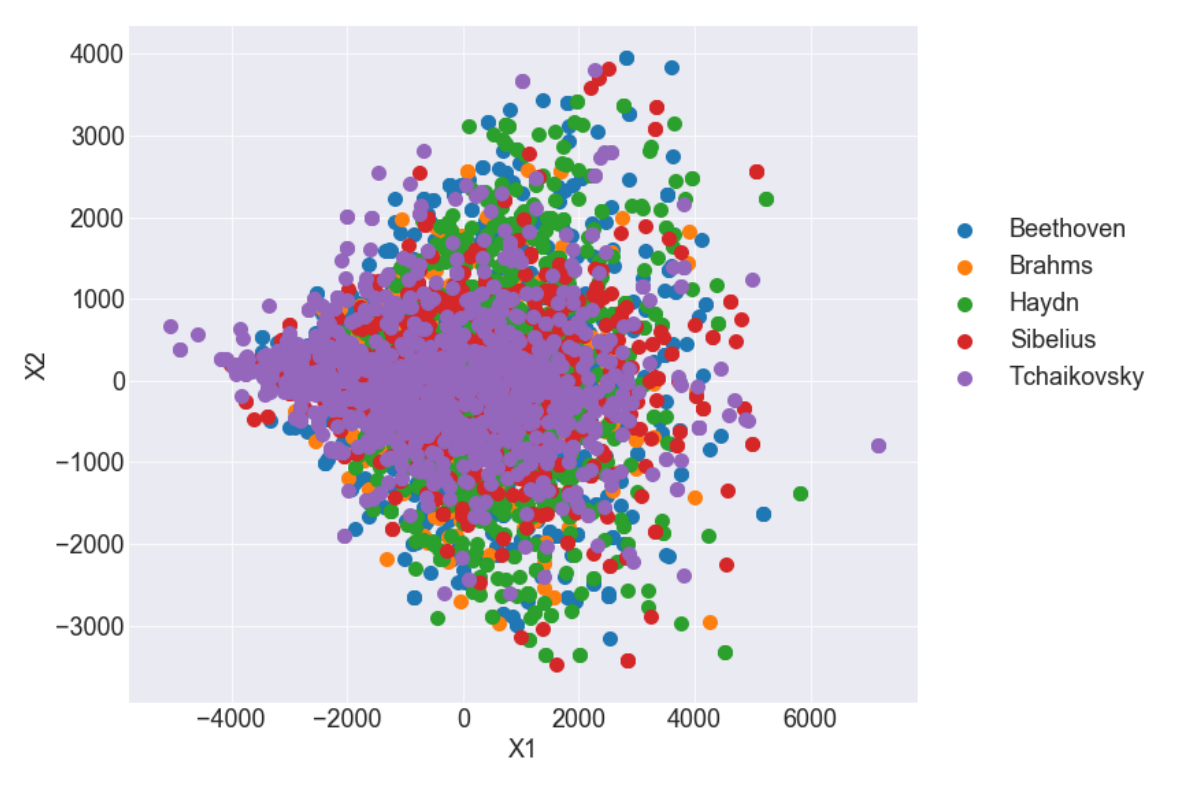

Figure 4: PCA embedding in 2-dimensions for the mel-spectrograms, colored by composer. The resulting embedding for the mel-spectrogram representation indicates that composer classification is likely to be a challenging task, as there is no clear separation by composer.

Classification Baselines

- This project uses different data then the related works described above
- Need to establish a classification baseline that can be compared to our proposed embedding methods
- Three types of classification baselines:
  1. Feed forward models
  2. Convolutional Neural Networks (CNNs)
  3. Long Short-Term Memory (LSTMs)

Feed Forward Architecture

Same architecture and hyper-parameter settings for all three data representations

- 4 fully connected hidden layers, each with ReLU activations
- First hidden layer has 1024 nodes, the second has 512 nodes, the third has 256 nodes and the fourth hidden layer has 64 nodes
- Output layer uses a softmax activation function with 5 classes, one for each composer
- Batch normalization performed after each of the hidden layers
- Cross entropy loss function
- Adam optimizer: learning rate of 0.0001, $\beta = (0.9, 0.999)$ and weight decay of $10e^{-5}$, 10 epochs.

CNN Architecture

Each input data representation has a slightly different dimension, requiring a slightly different CNN architecture. High-level details that are common across data representations are:

- Three convolutional layers, followed by two fully connected layers for classification.
- ReLU activations for all hidden nodes, softmax activation function for the output layer
- Two fully connected layers have 120 nodes and 84 nodes, output layer consists of 5 nodes for the composer classification
- Batch normalization after each of the convolutional layers
- Cross entropy loss function
- Adam optimizer: learning rate of 0.001, $\beta = (0.9, 0.999)$ and weight decay of $10e^{-5}$, 10 epochs.

CNN Specific Architectures

Table 2: CNN architecture details for the chromagram, MFCC and mel-spectrogram input data representations. The order of representation is chromagram/MFCC/mel-spectrogram. Parentheses indicate that the architecture differs for the height and width of the input representation. For example, kernel = (2,5) indicates that the kernel size is 2 in the vertical dimension and 5 in the horizontal direction.

Type of Layer	Architecture Details
Convolution	in channels = 1/1/1, out channels = 5/5/5, kernel = (2,5)/(4,5)/(4,3), stride = (2,1)/(1,1)/(2,2), padding = (6,1)/(1,1)/(1,1)
	Batch Normalization and Max pooling (1,2) (all representations)
Convolution	in channels = 5/5/5, out channels = 10/10/10, kernel = (2,4)/(2,4)/(4,4), stride = (2,2)/(2,2)/(2,2), padding = (6,2)/(6,2)/(1,1)
	Batch Normalization and Max pooling (1,2) (all representations)
Convolution	in channels = 10/10/10, out channels = 10/10/10, kernel = (2,4)/(2,4)/(3,4), stride = (2,2)/(2,2)/(2,2), padding = (6,1)/(6,1)/(1,1)
	Batch Normalization (all representations), Max pooling (2,2) for mel-spectrogram only
Fully Connected	120/120/120 nodes
Fully Connected	84/84/84 nodes
Fully Connected	5/5/5 nodes

LSTM Architecture

LSTM many-to-one model is the same for each input data representation

- Two hidden layers, five features in each hidden state
- Cross entropy loss function
- Adam optimizer: learning rate of 0.001, $\beta = (0.9, 0.999)$ and no weight decay
- Trained for 10 epochs, classification results did not improve with additional training
- Since LSTMs cannot handle input sequence lengths as long as the original data representations (431 time steps), the input data was split into shorter chunks of 36 time steps each, corresponding to roughly 0.8 seconds

Baseline Classification Results

Including additional information about the data structure did not greatly improve the classification accuracy. For all three data representations, the test set classification accuracy for the feed forward model was not significantly different from the CNN. The LSTM model, in particular, did not perform well for any input data representation: either a more complex time series model is required to classify this data with a high accuracy, or input data representations do not lend themselves to LSTM type models.

Table 3: Test set classification accuracy (as a percent) for each baseline classification model and input representation.

Representation	Feed Forward	CNN	LSTM
Chromagram	38.6257	36.8558	25.1171
MFCC	57.8605	52.8371	25.1171
Mel-spectrogram	52.3438	**55.7292**	25.2131

Baseline Embedding Approaches

- Baseline classification results similar between feed forward and CNN models
- Feed forward architectures used for all baseline embedding approaches, allows same architecture for each data representation
- Input to each model is a flattened version of the data representation
- Four baseline embedding approaches: autoencoder (AE), contractive autoencoder (CAE) [26], variational autoencoder (VAE) [15] and PCA
- Learned representation used as the input for a multi-class logistic regression classifier to classify the composer

Embedding Architectures

AE, CAE and VAE have the same encoder and decoder architectures. AE uses mean squared error loss, CAE uses contractive loss, VAE uses KL divergence.

- Encoder: 4 fully connected hidden layers, with 1024, 512, 256, 128 nodes, respectively, output of the encoder is 40 dimensional
- Decoder: 4 fully connected hidden layers, with 128, 256, 512 and 1024 nodes, respectively
- ReLU activation for all internal layers, final layer of encoder is linear and final layer of decoder uses tanh activation
- Adam optimizer: learning rate is 0.001, $\beta = (0.9, 0.999)$
- Weight decay of $10e^{-5}$ and 10 training epochs for AE and CAE
- No weight decay and 100 training epochs for VAE

Baseline Embedding Results

For all input data representations, autoencoder models did outperform PCA (with 40 dimensional embedding space). However, none of the models outperformed the baseline classification models in terms of composer classification accuracy.

Table 4: Test accuracy (as a percentage) for the learned representations (classified using multi-class logistic regression) for the AE, CAE, VAE and PCA models for each input data representation, all embeddings are 40 dimensional.

Data Representation	AE	CAE	VAE	PCA
Chromagram	39.43	**40.60**	38.73	35.89
MFCC	39.41	36.65	**45.45**	36.65
Mel-spectrogram	34.77	34.38	**36.85**	28.24

VAE Mel-Spectrogram Reconstruction

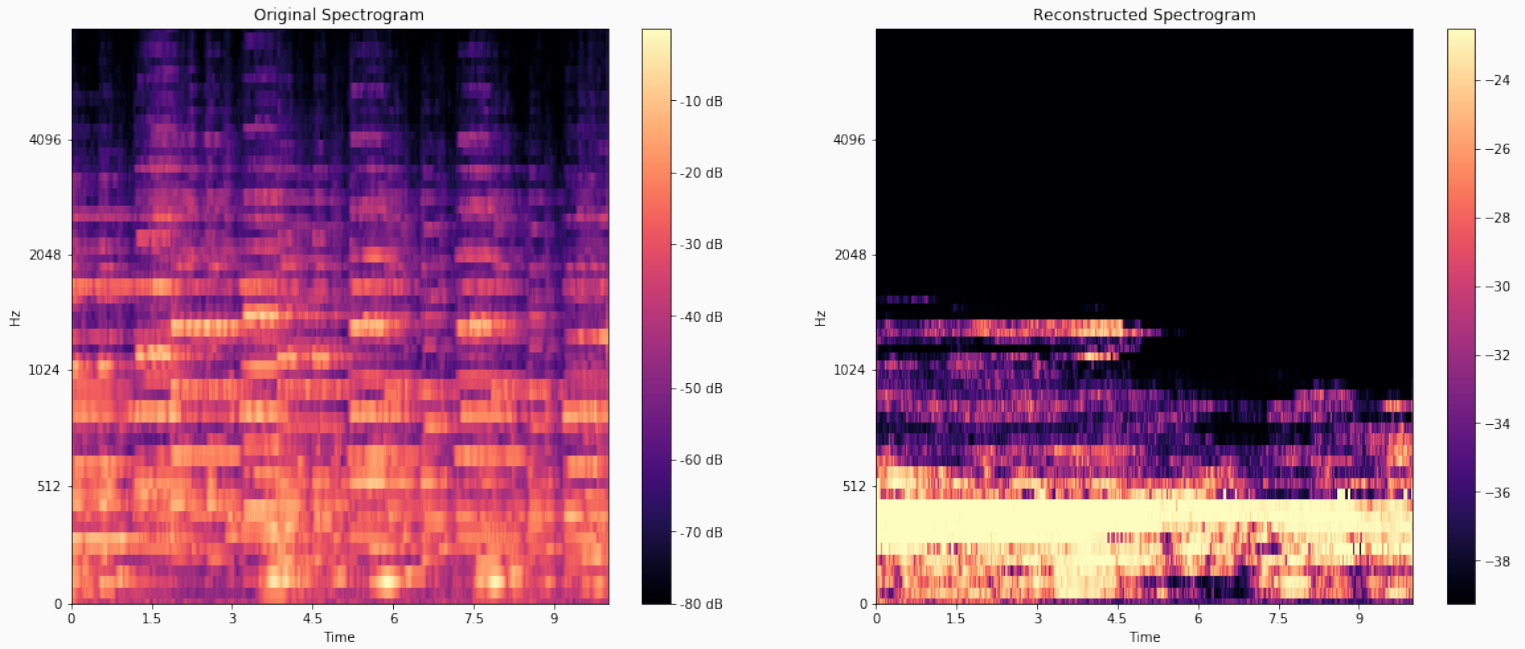

Figure 5: Original and reconstructed spectrograms for the VAE.

ALI Model Overview

- Main model considered for this project is the adversarially learned inference model (ALI) [10]
- [25] found ALI learned an embedding capable of clustering popular style songs by genre in a meaningful way
- ALI model uses an adversarial game to jointly learn a generation network and an inference network, where the generation network maps from stochastic latent variables to the data space (decoder) and the inference network maps from the data space to latent variables (encoder) [10]

ALI Model Description

Following [10], let $\mathbf{x}$ represent the observed data and $\mathbf{z}$ represent the corresponding latent variable. Joint distribution for the encoder is $q(\mathbf{x}, \mathbf{z}) = q(\mathbf{x})q(\mathbf{z}|\mathbf{x})$, where $q(\mathbf{x})$ is the empirical distribution of the data. Joint distribution for the decoder is $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})q(\mathbf{x}|\mathbf{z})$, where $p(\mathbf{z}) \sim N(0, \mathbf{I})$. ALI model objective is to match these two joint distributions, adversarial game is played to perform this matching with value function (Equation 1 from [10]):

$$\min_G \max_D V(D, G) = \mathbb{E}_{q(\mathbf{x})} [\log D(\mathbf{x}, G_z(\mathbf{x}))] + \mathbb{E}_{p(\mathbf{z})} [\log (1 - D(G_x(\mathbf{z}), \mathbf{z}))]$$

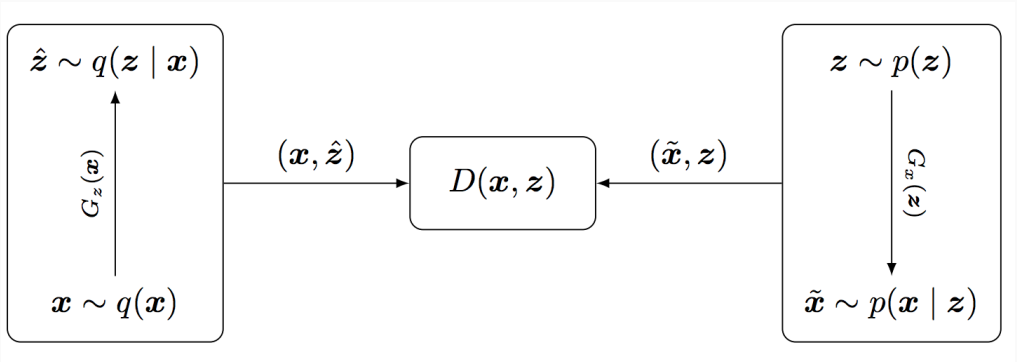

Figure 6: The game for the ALI model (image taken from [10]).

ALI Model Details

- Mel-spectrogram representation performed well across baselines, especially with CNN architecture, most information from musical perspective
- ALI model applied to mel-spectrogram representation only as a result
- Adversarial models can be hard to train, so same basic architecture as in original paper used here
- Mel-spectrograms are cropped to be of size 32 x 32 to match input data dimensions in [10] for stable training
- Binary cross-entropy loss, Adam optimizer with learning rate of $1e^{-5}$, $\beta = (0.9, 0.999)$, and no weight decay
- Trained for 50 epochs, initialized model weights, Leaky ReLU activation functions (negative slope of 0.01)
- Architecture very similar to Cifar10 model in [10], code adapted from [1]

Encoder and Generator Model Architecture

Table 5: ALI model architecture (following Table 3 of [10]). BN stands for batch normalization. All Leaky ReLU activation functions had a negative slope of 0.01. LD is the latent dimension and for this project, we considered models with LD = 256, 128 and 40.

Operation	Kernel	Strides	Feature Maps	BN?	Dropout	Nonlinearity
Encoder: $G_z(x)$ - 1 x 32 x 32 Input						
Convolution	5 x 5	1 x 1	32	Yes	0.0	Leaky ReLU
Convolution	4 x 4	2 x 2	64	Yes	0.0	Leaky ReLU
Convolution	4 x 4	1 x 1	128	Yes	0.0	Leaky ReLU
Convolution	4 x 4	2 x 2	256	Yes	0.0	Leaky ReLU
Convolution	4 x 4	1 x 1	512	Yes	0.0	Leaky ReLU
Convolution	1 x 1	1 x 1	512	Yes	0.0	Leaky ReLU
Convolution	1 x 1	1 x 1	2× LD	No	0.0	Linear
Decoder/Generator: $G_z(x)$ - LD x 1 x 1 Input						
Transposed Convolution	4 x 4	1 x 1	256	Yes	0.0	Leaky ReLU
Transposed Convolution	4 x 4	2 x 2	128	Yes	0.0	Leaky ReLU
Transposed Convolution	4 x 4	1 x 1	64	Yes	0.0	Leaky ReLU
Transposed Convolution	4 x 4	2 x 2	32	Yes	0.0	Leaky ReLU
Transposed Convolution	5 x 5	1 x 1	32	Yes	0.0	Leaky ReLU
Transposed Convolution	1 x 1	1 x 1	32	Yes	0.0	Leaky ReLU
Transposed Convolution	1 x 1	1 x 1	1	No	0.0	Sigmoid

Discriminator Model Architecture

Table 6: ALI model architecture (following Table 3 of [10]). BN stands for batch normalization. All Leaky ReLU activation functions had a negative slope of 0.01. LD is the latent dimension and for this project, we considered models with LD = 256, 128 and 40.

Operation	Kernel	Strides	Feature Maps	BN?	Dropout	Nonlinearity
$D(x)$ - 1 x 32 x 32 Input						
Convolution	5 x 5	1 x 1	32	No	0.2	Leaky ReLU
Convolution	4 x 4	2 x 2	64	No	0.2	Leaky ReLU
Convolution	4 x 4	1 x 1	128	No	0.2	Leaky ReLU
Convolution	4 x 4	2 x 2	256	No	0.2	Leaky ReLU
Convolution	4 x 4	1 x 1	512	No	0.2	Leaky ReLU
$D(z)$ - LD x 1 x 1 Input						
Convolution	1 x 1	1 x 1	512	No	0.2	Leaky ReLU
Convolution	1 x 1	1 x 1	512	No	0.2	Leaky ReLU
$D(x)$ - 1024 x 1 x 1 Input						
Concatenate $D(x)$ and $D(z)$ along the channel axis						
Convolution	1 x 1	1 x 1	1024	No	0.2	Leaky ReLU
Convolution	1 x 1	1 x 1	1024	No	0.2	Leaky ReLU
Convolution	1 x 1	1 x 1	1	No	0.2	Leaky ReLU

Experimental Results

Model Fit

Specified ALI model is fit with three different latent dimensions, 256, 128 and 40 (for comparison to the baseline embedding approaches). Overall, it appears that the model has converged

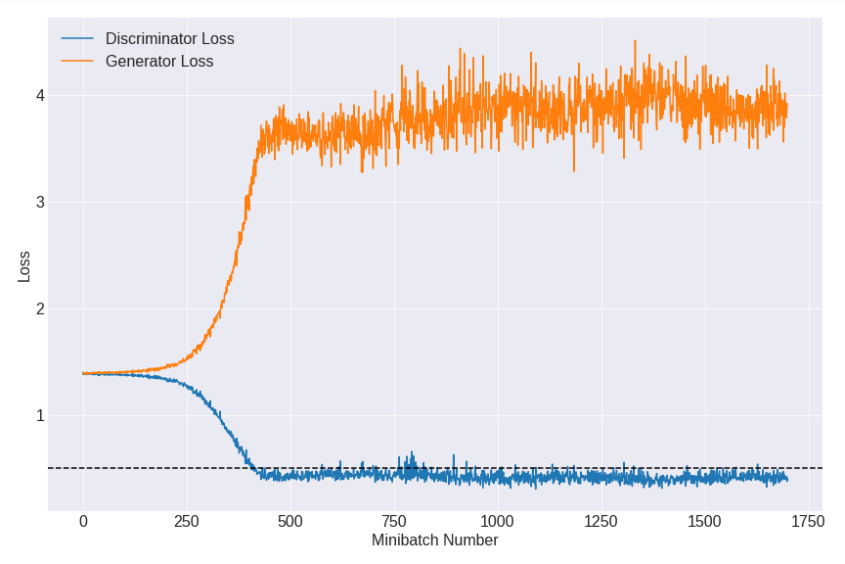

Figure 7: Generator and discriminator loss for the ALI model with a latent dimension of 128. The generator and discriminator both appear to have converged, although the generator loss is much larger than the discriminator loss. The discriminator loss is approximately 0.5, which usually indicates a stable and converged GAN [14].

Generated Mel-Spectrograms

ALI model able to generate mel-spectrograms that look more realistic than those generated by the VAE for all three latent dimensions. While the generated examples do not have the level of fidelity of the original mel-spectrograms, they do look approximately reasonable.

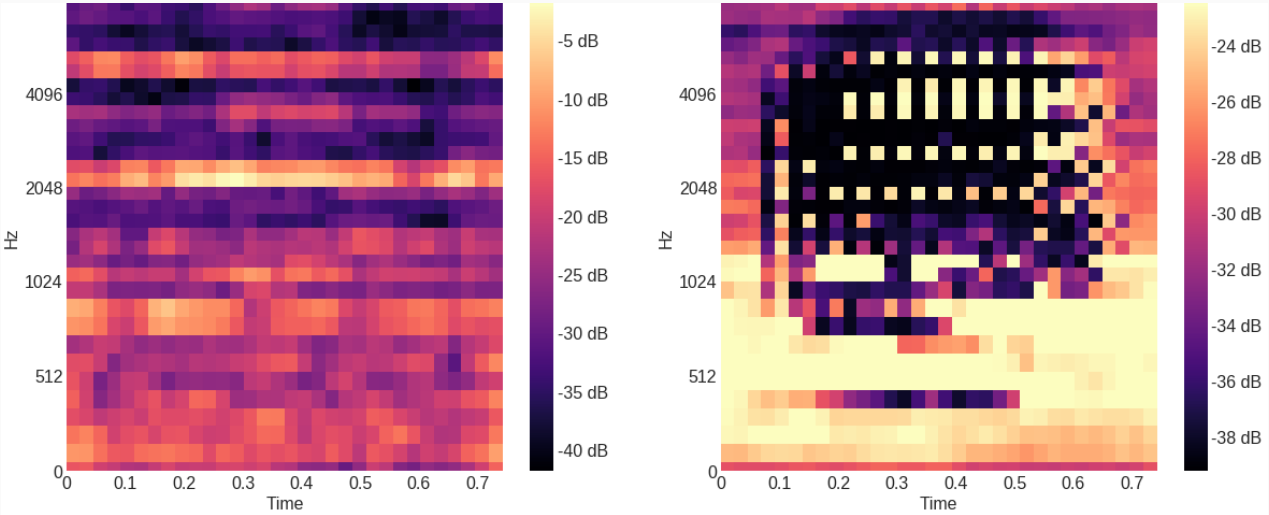

Figure 8: Original (left) and generated (right) mel-spectrograms from the ALI model with 128 and dimensional latent space.

ALI Composer Classification

ALI model does not result in embeddings that improve over the baseline methods, either the classification or embedding methods for composer classification, even when compared to baseline classification models using 32 x 32 input dimensions. A baseline feed forward, fully connected model (using the same architecture described above) achieves a classification accuracy of **35.81%**, while the ALI Encoder network as a stand alone classification model (with an additional output feed forward layer and softmax output activation for classification) achieves a classification accuracy of **40.63%**

Table 7: Multi-class logistic regression accuracy (as a percentage) for the learned ALI embeddings for three different latent space dimensions. LD is the dimension of the latent space.

	LD = 256	LD = 128	LD = 40
ALI Training Accuracy	38.24	35.25	33.21
ALI Test Accuracy	29.91	28.67	28.67

Latent Representation

- ALI model with a latent dimension of 128 does show differences in the generated mel-spectrograms when interpolating in the latent space
- Appears that latent vectors $\mathbf{z}$ closer to the origin represent mel-spectrograms where the volume across the entire spectrum is approximately uniform and close to the maximum volume of the recording (lighter colors represent larger values)
- Proposed interpretation: points in the latent space close to the origin represent parts in the audio where the full orchestra is playing at a loud volume, while points further from the origin represent points in the music where specific instruments play louder than others, or where the entire orchestra plays at a softer volume

Interpolation in the Latent Space

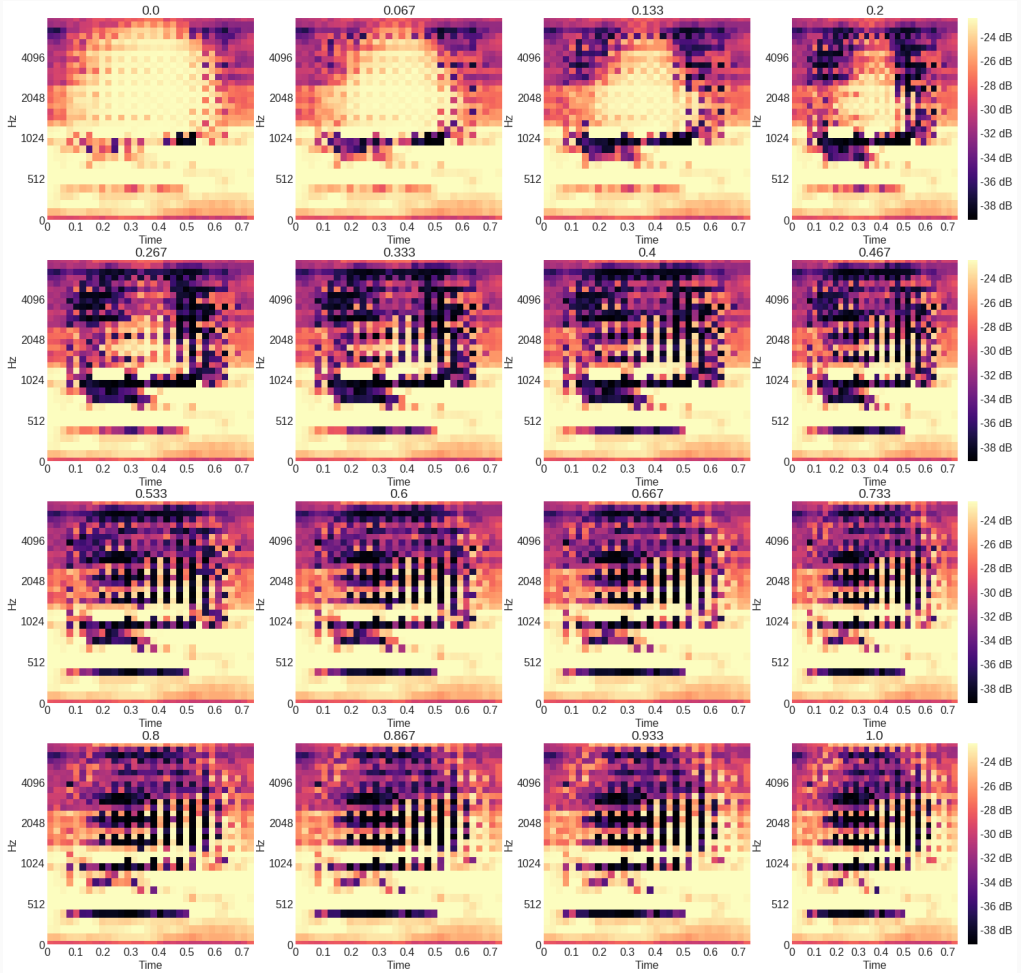

Figure 9: Interpolation in the latent space for the ALI model with latent dimension of 128. The interpolation runs from the origin to the point with coordinates all equal to 1 in the latent space. For the interpolation, each coordinate has the same value, which is the title of each subplot. For this model, latent points closer to the origin appear to represent mel-spectrograms which are louder over the entire spectrum than latent points further from the origin.

Discussion

- Overall, the ALI model is able to capture some general trends in the mel-spectrogram data and generate examples that look approximately like the input data
- However, the latent ALI representation does not outperform the baseline approaches in terms of composer classification accuracy
- While it is not necessary for a musically meaningful latent embedding to perform well on a downstream classification task, chosen embedding should beat baseline approaches
- ALI model seems like a promising direction for future work, but more model architecture exploration needed

Conclusions and Future Work

Conclusions

- Goal of this project was to learn a musically meaningful latent embedding of orchestral audio data using adversarially learned inference models.
- Eventually, successful audio embeddings could be utilized in generative deep learning models for algorithmic composition
- Learned embeddings were compared to baseline approaches on a downstream task of composer classification
- This project represents some of the first (to my knowledge) work on exploring data representations for classical, orchestral audio data
- Overall, the ALI model does not learn an embedding that outperforms baseline classification models for the composer classification task
- Learned latent space is somewhat interpretable from a musical perspective
- Generated mel-spectrograms produced by the generator do resemble the input data at a high level

Future Work

Further exploring ALI model seems like a promising direction for future work:

- Exploring generator and encoder architectures that more explicitly take into account the temporal and spatial structure of the data
- Exploring models that utilize all 10s of input data would likely provide a further improvement
- How to best incorporate temporal information/structure?
- Perhaps apply seq2seq type model in the context of ALI
- Overall, project reinforces that learning a meaningful audio embedding for any type of recording is a challenging task
- Much more work is needed to find a suitable approach for classical orchestral audio data and this project can perhaps serve as a baseline for what types of classification accuracies are achievable on full orchestral audio recordings with the models considered here.

References i


9310gaurav.
ali-pytorch: PyTorch Implementation of Adversarially Learned Inference (BiGAN).
GitHub.


Andreas Arzt and Stefan Lattner.
Audio-to-Score Alignment Using Transposition-Invariant Features.
19th International Society for Music Information Retrieval Conference, 2018.


Yusuf Aytar, Carl Vondrick, and Antonio Torralba.
SoundNet: Learning Sound Representations from Unlabeled Video.
CoRR, abs/1610.09001, 2016.

References ii


Roberto Basili, Alfredo Serafini, and Armando Stellato.
Classification of Musical Genre: A Machine Learning Approach.


Beth Logan.
Mel Frequency Cepstral Coefficients for Music Modeling.
In *In International Symposium on Music Information Retrieval*, 2000.


Zach C.
Deep Learning: Can we Use Computer Vision to Predict the Composer of Classical Music?, 2018.


Keunwoo Choi, György Fazekas, Mark B. Sandler, and Kyunghyun Cho.
Convolutional Recurrent Neural Networks for Music Classification.
CoRR, abs/1609.04243, 2016.

References iii


Yu-An Chung and James R. Glass.
Learning Word Embeddings from Speech.
CoRR, abs/1711.01515, 2017.


Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-yi Lee, and Lin-Shan Lee.
Audio Word2vec: Unsupervised Learning of Audio Segment Representations using Sequence-to-sequence Autoencoder.
CoRR, abs/1603.00982, 2016.


Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville.
Adversarially Learned Inference.
ICLR, 2017.

References iv


Jesse H. Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi.
Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders.
CoRR, abs/1704.01279, 2017.


Rajat Hebbar.
music2vec: Generating Vector Embeddings for Genre-Classification Task, 2017.


Jay Hennig, Akash Umakantha, and Ryan Williamson.
Sequence Generation and Classification with VAEs and RNNs.
ICML, 2017.

References v


Jason Brownlee.
How to Identify and Diagnose GAN Failure Modes.
https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/, July 2019.


Diederik P Kingma and Max Welling.
Auto-encoding variational bayes, 2013.


Haojun Li, Siqi Zue, and Jialun Zhang.
Combining CNN and Classical Algorithms for Music Genre Classification.
2018.

References vi


Yin-Jyun Luo and Li Su.
Learning Domain-Adaptive Latent Representations of Music Signals using Variational Autoencoders.
19th International Society for Music Information Retrieval Conference, 2018.


Rachel Manzelli, Vijay Thakkar, Ali Siahkamari, and Brian Kulis.
Conditioning Deep Generative Raw Audio Models for Structured Automatic Music.
19th International Society for Music Information Retrieval Conference, 2018.


Gianluca Micchi.
A Neural Network for Composer Classification.
19th International Society for Music Information Retrieval Conference, 2018.

References vii


Meinard Müller.
Fundamentals of Music Processing.
Springer, 2015.


Zain Nasrullah and Yue Zhao.
Music Artist Classification with Convolutional Recurrent Neural Networks.
CoRR, abs/1901.04555, 2019.


Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu.
WaveNet: A Generative Model for Raw Audio.
CoRR, abs/1609.03499, 2016.

References viii


Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra.
Multi-Label Music Genre Classification from Audio, Text and Images Using Deep Features.
18th International Society for Music Information Retrieval Conference, Suzhou, China, 2017.


Adam Roberts, Jesse H. Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck.
A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music.
CoRR, abs/1803.05428, 2018.


Brad Ross and Prasanna Ramakrishnan.
song2vec: Determining Song Similarity using Deep Unsupervised Learning.
2017.

References ix


Salah Rifai and Pascal Vincent and Xavier Muller and Xavier Glorot and Yoshua Bengio.
Contractive Auto-Encoders: Explicit Invariance During Feature Extraction.
Proceedings of the 28th International Conference on Machine Learning, 2011.


Rishi Sidhu.
Audio Classification Using CNN - An Experiment, 2019.


Ilya Sutskever, Oriol Vinyals, and Quoc V. Le.
Sequence to sequence learning with neural networks.
CoRR, abs/1409.3215, 2014.

References x


Alexandros Tsaptsinos.
Lyrics-Based Music Genre Classification using a Hierarchical Attention Network.
18th International Society for Music Information Retrieval Conference, Suzhou, China, 2017.