Symphony2vec: Learning Musically Meaningful Embeddings for Orchestral Audio Data

Anna K. Yanchenko

Abstract

Algorithmic composition is the use of statistical and machine learning models to generate new music and has received an increase in attention recently due to advances in deep learning. However, there is currently no standard embedding approach for audio music data, which is a necessary prior step for successful generative modeling. 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. We explore the use of adversarially learned inference models to learn this embedding and compare to baseline approaches. To my knowledge, this is some of the first work focusing on learning an embedding for *orchestral* audio data The learned embeddings are evaluated on a downstream task of composer classification. We find that the adversarially learned inference 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 a challenging task.

1 Introduction

Algorithmic composition is the use of statistical and machine learning models to generate new music. In recent years, algorithmic composition has seen an increase in research interest and has generated some impressive musical results [22, 24, 11]. The 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). There are complex dependencies between different instruments, and thus there is a high level of structure both within the part of a single instrument and between instruments. Music is a long-memory process, with long-term structure over multiple minutes in full orchestral works. Additionally, the modeling of music is challenging from a data perspective, as there is very little labeled data and it is expensive to obtain, necessitating unsupervised approaches. Audio data, in particular, while of high interest, is challenging to annotate and label in its raw form. These challenges are applicable to many fields beyond music composition.

However, despite the increased interest in algorithmic composition, there is currently no clear consensus on the representation to use for music in deep learning models and this remains an open research problem. Audio data, in particular, can be challenging to work with and model and has only recently received attention from a representation perspective, with work such as [22, 11], for example. In order to make progress towards the goal of realistic music generation by deep learning models, and by extension, to increase modeling capabilities in other fields such as speech recognition, there needs to be improvements in the representation of audio data for algorithmic composition.

The goal of this project is to develop a musically meaningful latent representation of orchestral audio recordings. We use Adversarially Learned Inference (ALI) [10] to learn a representation for audio recordings based on an input mel-spectrogram representation. We consider audio recordings of symphonies from five classical composers, recorded by the Berlin Philharmonic. The learned representation is evaluated on a downstream task of composer classification and the generated mel-spectrograms are qualitatively evaluated. The embedding learned by ALI is compared to baseline approaches, including Variational Autoencoders [15] and Principal Components Analysis.

The main contribution of this work is one of the first analyses (to my knowledge) of embedding classical *orchestral* audio data. In contrast to symbolic data or audio data of a single instrument, orchestral audio recordings contain complex dependencies between instruments. Additionally, it is an open problem as to how to separate instruments from an audio recording and there is not much prior work on classifying classical composers.

Overall, we conclude that learning a musically meaningful representation of orchestral audio data is still a challenging open problem that is much different from models for popular music and genre classification, in particular in terms of utilizing temporal information in a meaningful way. While ALI models seem like a promising approach for the task of audio representation, and the ALI model is able to recover some basic musical structure in the learned latent space, we find that this advanced method does not improve over a baseline VAE representation in terms of composer classification. The main challenge going forwards will be incorporating temporal information about the audio data that allows for a meaningful representation.

2 Related Works

To my knowledge, this is one of the first works exploring the embedding of orchestral audio data for use with deep learning models. However, there are many related works that consider learning an embedding of different types of music data, as well as work that examines classifying genre aspects of popular music.

The 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, 28, 30], though work on classifying specific composers is more limited [6, 19]. Many approaches use CNNs with spectrograms or MFCC features as input, though there is some debate as to the validity of such an approach [26]. There is much less prior work on classifying orchestral audio data for specific classical composers. The literature seems to suggest that most existing approaches on related genre classification problems are able to achieve a classification accuracy of 70-80%.

In terms of learning latent representations for music or audio data, there are three primary existing approaches. The first approach utilizes autoencoders, primarily in the VAE framework [2, 13, 17]. Google's Magenta project, in particular, has found success in developing a special-purpose hierarchical VAE, called MusicVAE, that is able to represent and generate symbolic piano music with long-term structure [24].

The second type of approach learns embeddings in a method similar to Word2vec, adapted for popular audio music and used for genre classification tasks [12, 25]. Additionally, Seq2seq [29] approaches with Encoder-Decoder RNNs have been explored in the context of speech recognition [9, 8, 3]. These types of approaches heavily utilize the temporal structure of the audio speech data, and similar to Word2vec, use the information at the current time step to predict the audio content within a window, both forwards and backwards in time.

Finally, more recent approaches use the success of the WaveNet model [22, 18] for generating raw audio, adapted to an Encoder-Decoder framework [11]. This model is explicitly autoregressive and takes the raw audio data as an input. The WaveNet Autoencoder used in [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, as well as meaningful interpolation in the latent space.

For this work, we follow [25] and use Adversarially Learned Inference (ALI) models [10] to learn an embedding. We choose not to explore approaches like MusicVAE [24] that work on symbolic musical data, as the primary interest for this work is audio data as input. However, 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 in terms of the musical application. Finally, we find that baseline RNNs do not work well for the data for this project (discussed in subsection 3.2) and thus do not consider Seq2seq type approaches, as we find that the temporal information for our data does not appear to be well represented by an RNN or LSTM. Thus, we consider an alternative approach, using ALI.

ALI is used by [25] to represent popular audio music, represented as spectrograms, and to classify the genre of these songs. The authors find that the learned ALI embedding outperforms a baseline PCA approach in terms of genre classification, and that the learned embeddings cluster logically in terms

of genres and sub-genres. This project adapts the approach in [25] to learn a latent representation for *classical, orchestral* audio data.

3 Project Details

In this section, we describe the data and the input data representations that are used for modeling. Next, several baseline classification and representation approaches are considered. Finally, we give an overview of ALI and describe the main architecture used for this work.

3.1 Data and Input Representations

The data for this project consists of 10 second, non-overlapping audio clips of symphonies from 5 different classical composers, Beethoven, Brahms, Haydn, Sibelius and Tchaikovsky. This data set was manually compiled from mp3 and CD recordings. These five composers represent different musical eras and styles, and could be classified with fairly high accuracy by a musically-knowledgeable listener. Thus, in theory, a deep learning approach should also be able to classify these composers with a reasonably high accuracy, with a sufficiently meaningful embedding. All recordings are by the Berlin Philharmonic under Herbert von Karajan to control for specific orchestra stylistic differences, which are not of interest for this work.

The data is split according to a 70%-30% training-test split, resulting in **5366** training examples and **3842** test examples. The dataset is fairly well balanced for both the training and test set Table 1.

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

Three main input audio representations are considered for this project; the chromagram, Mel-frequency cepstral coefficients (MFCCs) and the spectrogram. The chromagram is a simple representation that essentially extracts what notes are being performed by the orchestra at a given time. MFCCs are widely used in speech recognition [9, 8] and music information retrieval, and these coefficients capture aspects of the timbre, or quality of sound, of the orchestra, an important feature that is not captured with symbolic input music data. Finally, spectrograms contain the most information of the three data representations considered, and contain information about amplitude information across the entire spectrum.

The spectrogram is a standard representation of audio signals and represents the power spectrum of the audio signal over the entire frequency range. The spectrogram is found via the Short-Time Fourier Transform (STFT). The chromagram [20] aggregates the amplitude of each frequency bin in the spectrogram across octaves to give one amplitude for each pitch in the twelve tone scale. This aggregation is robust to differences in instrument balance and intonation. MFCCs are calculated by applying binning of the spectral vectors where the binning spacing is based on human perception of frequency. Lower frequencies are more important and the binning follows the mel-scale, which maps actual frequency to perceived pitch. The discrete cosine transform is applied to decorrelate the mel-spectral vectors, which results in cepstral vectors [5]. The spectrogram representation considered here is also on the mel-scale.

All three of these input data representations are already musically meaningful, in the sense that the representations can be interpreted in relation to the audio track and qualities of the orchestra and piece over time. However, each of these representations is high-dimensional and may not be ideal for a generative deep learning model directly. However, these three data representations seem like a logical input for a deep learning model that is capable of learning a lower-dimensional, while still musically-meaningful, embedding.

For all three representations considered, the temporal resolution is 0.023 seconds. There are 12 chroma vectors for each point in time, resulting in **12 x 431** dimensional matrix for each chromagram.

We consider 13 MFCC values, so each MFCC representation is a **13 x 431** dimensional matrix. Finally, we consider 64 mels for the mel-spectrogram, resulting in a **64 x 431** dimensional matrix for each spectrogram. All chromagrams, MFCCs and spectrograms are normalized prior to modeling. Example plots of the chromagrams Figure 1, MFCC representations Figure 2 and mel-spectrograms Figure 3 are shown below. For each of the sample representations, there are visual differences by composer.

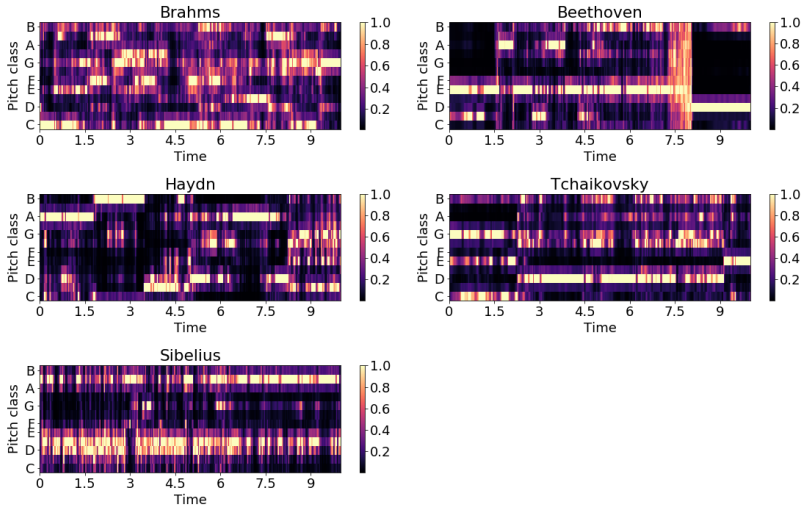


Figure 1: Example chromagrams for each of the 5 composers considered.

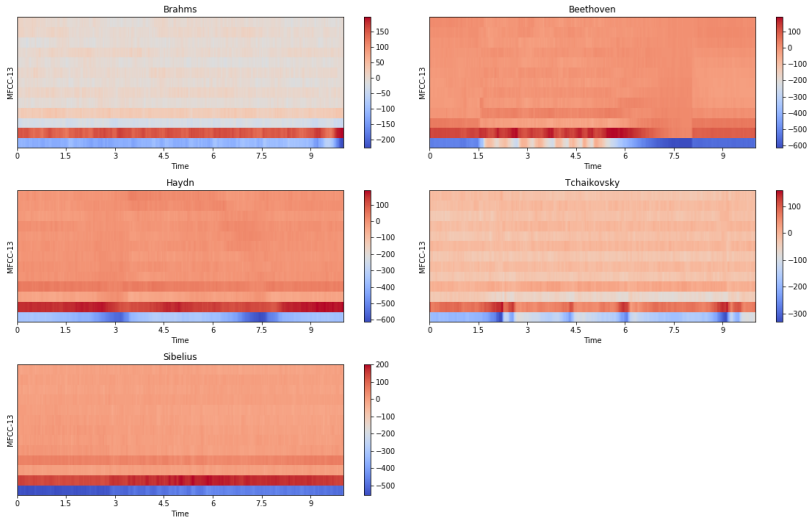


Figure 2: Example MFCC representations for each of the 5 composers considered.

Principal components analysis (PCA) can be used as an exploratory tool for embedding each data representation into a two-dimensional space for visualization. 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 Figure 4. Additionally, more complex methods than PCA and higher-dimensional embedding spaces are likely to be needed. Similar results are seen for chromagram and MFCC two-dimensional embeddings using PCA.

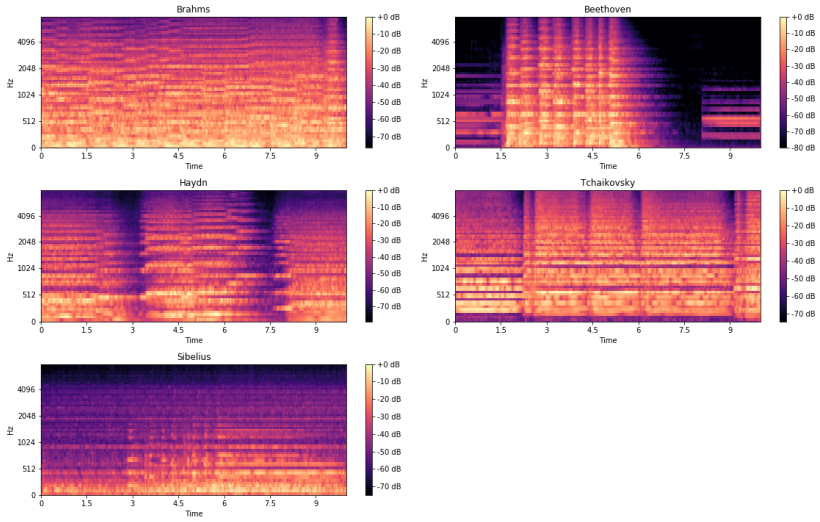


Figure 3: Example mel-spectrograms for each of the 5 composers considered. The intensity is scaled such that 0 dB is the loudest volume and the other intensities are relative to this maximum volume.

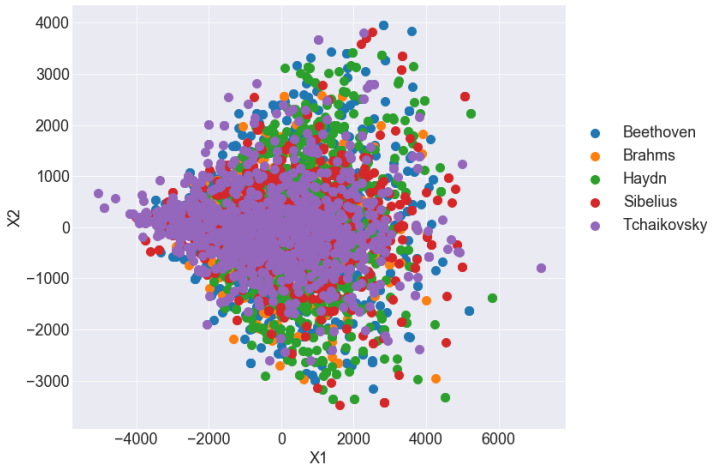


Figure 4: PCA embedding in 2-dimensions for the mel-spectrograms, colored by composer.

3.2 Classification Baselines

Since this project uses different data then the related works described above, we first need to establish a classification baseline that can be compared to our proposed embedding methods. We consider three types of classification baselines: a feed forward, fully connected model, a CNN and a LSTM. The CNN and LSTM models take into account the structure of our input data better than the feed forward network, as we expect both temporal and spatial structure in the input data. Relatively simple feed forward, CNN and LSTM models are considered to give a "naive" classification baseline.

Feed Forward Architecture For all 3 data representations, the feed forward neural network has 4 fully connected hidden layers, each with ReLU activations. The first hidden layer has 1024 nodes, the second has 512 nodes, the third has 256 nodes and the fourth hidden layer has 64 nodes. The output layer uses a softmax activation function with 5 classes, one for each composer. Batch normalization is performed after each of the hidden layers and the cross entropy loss function is used for all data representations. Additionally, the Adam optimizer is used with a learning rate of 0.0001,

$\beta = (0.9, 0.999)$ and weight decay of $10e^{-5}$. The model is trained for each data representation for 10 epochs.

CNN Architecture Each input data representation has a slightly different dimension, requiring slightly different CNN architectures for each input data type. All CNNs consist of three convolutional layers, followed by two fully connected layers for classification. ReLU activations are used for all hidden nodes, while a softmax activation function is used for the output layer. The two fully connected layers have 120 nodes and 84 nodes, respectively, while the output layer consists of 5 nodes for the composer classification. Batch normalization is performed after each of the convolutional layers for all architectures and the cross entropy loss function is used for all data representations. Additionally, the Adam optimizer is used with a learning rate of 0.001, $\beta = (0.9, 0.999)$ and weight decay of $10e^{-5}$. The model is trained for each data representation for 10 epochs.

The full CNN architectures for all three data representations are given in Table 2.

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 The LSTM architecture is the same for each input data representation. The LSTM is a many-to-one model that consists of two hidden layers, with five features in each hidden state. The cross entropy loss function is used with the Adam optimizer with a learning rate of 0.001, $\beta = (0.9, 0.999)$ and no weight decay. The model is again trained for 10 epochs and classification results did not improve with additional training. Additionally, since LSTMs cannot handle input sequence lengths as long as the original data representations (431 time steps), the input data representations were split into shorter chunks of 36 time steps each, corresponding to roughly 0.8 seconds.

Results For the naive classification models, including additional information about the data did not greatly improve the classification accuracy Table 3. 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. This suggests that either a more complex time series model is required to classify this data with a high accuracy, or that the input data representations themselves do not lend themselves to LSTM type models.

3.3 Embedding Baselines

Based on the results of the baseline classification methods, the performance of the feed forward model and the CNN was fairly comparable across input data representations. As a result, the embedding baseline models considered were all based on a feed forward architecture, which additionally allowed

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

for the same model architecture to be used for each input data representation. Thus, the input to each model was a flattened version of the data representation. The four baseline embedding approaches considered were an autoencoder (AE), a contractive autoencoder (CAE) [27], a variational autoencoder (VAE) [15] and PCA. For each embedding method, the learned representation was used as the input for a multi-class logistic regression classifier to classify the composer from the learned representation.

The architectures for the AE and the CAE are the same, with only the loss being different between the two models. For all data representations, the encoder for the AE and the CAE consists of four fully connected hidden layers, with 1024, 512, 256, 128 nodes, respectively. The output of the encoder is 40 dimensional. The decoder for the AE and CAE consists of four fully connected hidden layers, with 128, 256, 512 and 1024 nodes, respectively. All internal layers use the ReLU activation function. The final layer of the encoder is linear and the output layer of the decoder uses a tanh activation function. The loss function for the AE is the mean squared error, while the contractive loss is used for the CAE. Both the AE and CAE are trained with the Adam optimizer, with the standard learning rate of 0.001, $\beta = (0.9, 0.999)$ and weight decay of $10e^{-5}$ for 10 epochs each.

The VAE uses the same architecture for the encoder, four fully connected hidden layers, with 1024, 512, 256, 128 nodes, respectively, and the decoder, four fully connected hidden layers, with 128, 256, 512 and 1024 nodes, respectively. The dimension of the latent space is again 40 for the VAE. The loss function is the original KL divergence loss specified in [15]. Again, all internal layers use the ReLU activation function, the final layer of the encoder is linear and the output layer of the decoder uses a sigmoid activation function. The VAEs are trained with the Adam optimizer, with the standard learning rate of 0.001, $\beta = (0.9, 0.999)$ and no weight decay for 100 epochs.

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.

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

The AE, CAE and VAE models are also compared to PCA, where the embedding dimension was 40, just as for the deep learning models. For the chromagram input representation, the CAE gave the highest classification accuracy, while for the MFCC and mel-spectrogram representations, the VAE gave the best classification accuracy (Table 5). For all input data representations, the various autoencoder models did outperform PCA. However, none of the models outperformed the baseline classification models in terms of composer classification accuracy (Table 3).

While a musically-meaningful embedding does not necessarily require high accuracy on a composer classification task, the composer of a piece highly influences the style, dynamics and notes played for that piece, and thus composer classification is a relevant downstream task for learning a musically meaningful embedding. The results in Table 5 suggest that the learned embeddings for the AE, CAE and VAE models are learning something about the style of the input audio recordings, but not enough to separate composers with high accuracy.

The results of reconstructions by each decoder confirm these results. Example reconstructions for the mel-spectrogram representation are given for the CAE (Figure 5) and VAE (Figure 6). The reconstructions do generally resemble spectrograms, however, the reconstructions are clearly not as

detailed as the original inputs and are thus not expected to be good embeddings for achieving high classification accuracy on composer classification.

Table 5: 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.

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

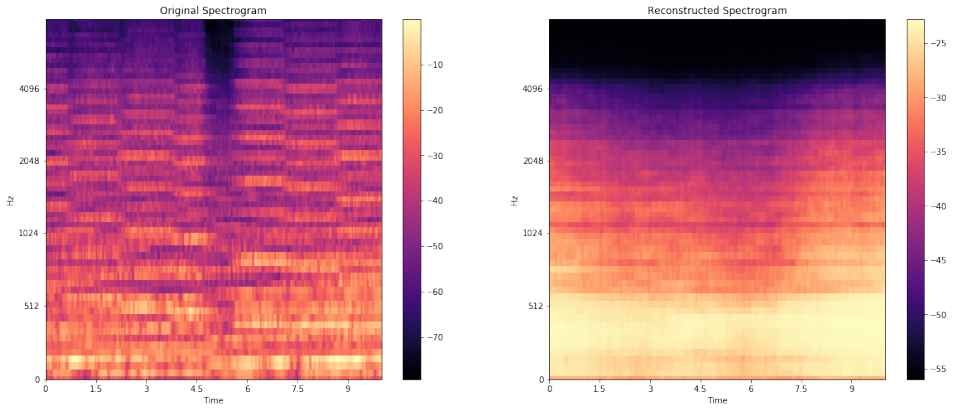


Figure 5: Original and reconstructed spectrograms for the CAE.

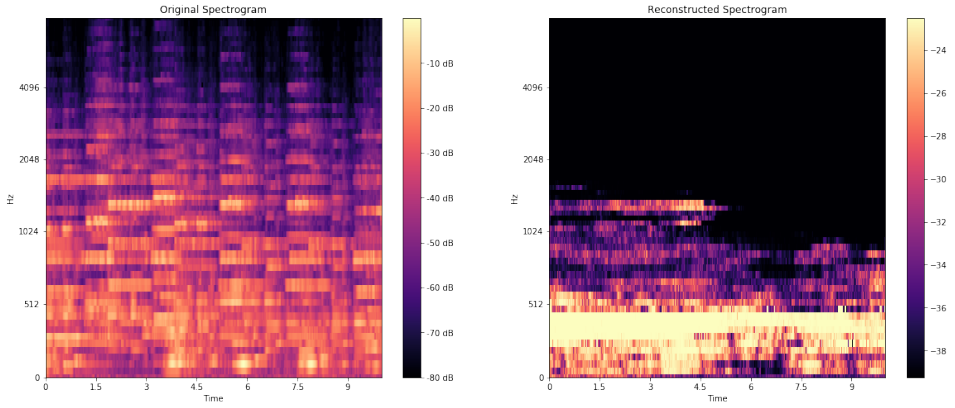


Figure 6: Original and reconstructed spectrograms for the VAE.

3.4 Proposed Model: Adversarially Learned Inference

The main model that we consider for this project is the adversarially learned inference model (ALI) [10], following [25], which found ALI to be capable of learning an embedding that could cluster popular songs by genre in a meaningful way. The 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]. The ALI model is similar to a generative adversarial network, and a discriminator is trained to separate joint samples of the data and latent variables from the encoder from joint data and latent variable samples from the decoder.

Following the notation of [10], let $\boldsymbol{x}$ represent the observed data and $\boldsymbol{z}$ represent the corresponding latent variable. Then, the joint distribution for the encoder is $q(\boldsymbol{x}, \boldsymbol{z}) = q(\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{x})$, where $q(\boldsymbol{x})$ is the empirical distribution of the data. The joint distribution for the decoder is $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})q(\boldsymbol{x}|\boldsymbol{z})$, where $p(\boldsymbol{z}) \sim N(0, \boldsymbol{I})$. The objective of the ALI model is to match these two joint distributions, and an adversarial game is played to perform this matching. The adversarial game is shown in Figure 7 and the value function for the game is (Equation 1 from [10]):

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

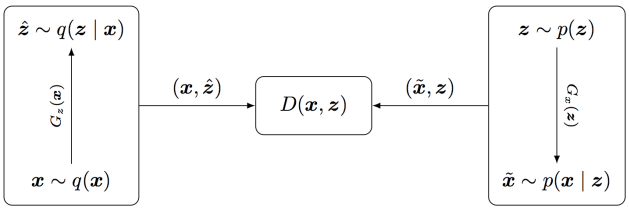


Figure 7: The game for the ALI model (image taken from [10]).

The ALI model is considered for several reasons. First, it allows for a latent representation of the input data to be learned without needing to specify an explicit sampling model, as is required for the VAE. The learned representation can be interpreted through interpolation in the latent space and by allowing for the reconstruction of input data and the generation of new data examples. Finally, the ALI model is very similar to a GAN, which has found great success in image generation recently. Since all of the input representations considered for this work can be treated like images, ALI seems like a promising approach to learning an embedding and allows for easy generation of new data.

Based on the results of the baseline classification and embedding models considered above, the mel-spectrogram generally performed the best of the three input data representations in terms of composer classification accuracy, and in particular, the classification accuracy for the mel-spectrogram was the highest when using a CNN for a classifier. The mel-spectrogram retains the most information of the three input data representations considered, and therefore seems the most useful as an input data representation for later generative modeling.

The ALI model is applied to the mel-spectrogram only for these reasons. Additionally, as adversarial models can be challenging to train, the mel-spectrogram input data is cropped to be 32 x 32 in dimensions, following the input data dimensions in [10]. This is a reduction in information and loses a good amount of temporal information, however, the ALI model architecture specified in [10] allowed for stable training of the model. When the basic ALI model was adapted to handle input of dimension 64 x 64, the model training was unstable. Future work involves examining different ALI model architectures to better handle the mel-spectrogram input dimensions.

The ALI model considered for this work uses the same architecture as the model for the Cifar10 data specified in the original paper [10], specified in Table 6. The model loss was the binary cross entropy loss and the Adam optimizer was used with a learning rate of $1e^{-5}$ and $\beta = (0.9, 0.999)$, with no weight decay. The model was trained for 50 epochs with a batch size of 64 and the model weights were initialized as N(0,0.02) for weights without batch normalization, as N(1, 0.02) for weights with batch normalization. The biases were initialized to 0. The Leaky ReLU activation functions all used a negative slope of 0.01. While the model architecture is very similar to [10] (except that for this work there is only one input channel for the image instead of three), the ALI model is trained from scratch for the mel-spectrogram input data. The model is implemented by adapting code from [1].

4 Experimental results

The 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 8). The discriminator and generator losses for all latent dimension are very similar to Figure 8, which shows the loss for just a latent dimension of 128. The losses are both stable, though the generator

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
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	$2\times$ 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
$D(x)$ - 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

loss is much higher than the discriminator loss, indicating that the generation of a latent embedding and new mel-spectrogram is much more challenging than discriminating real mel-spectrograms and their learned embedding from the fake data and corresponding embedding.

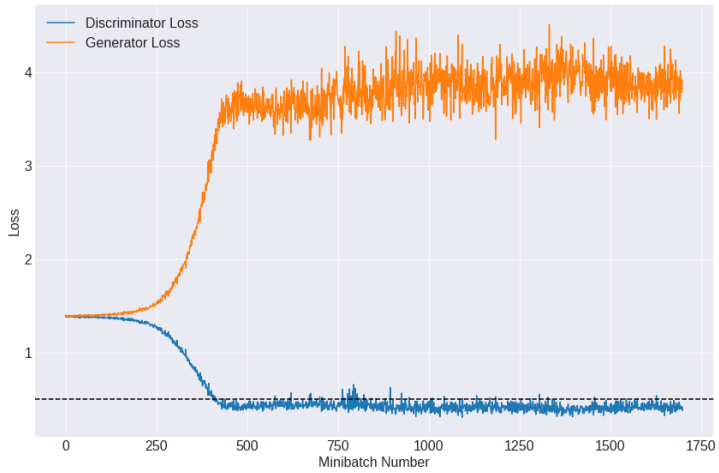


Figure 8: Generator and discriminator loss for the ALI model with a latent dimension of 128. The generator and discriminator both appear to have converged, although the generator loss is much larger than the discriminator loss. The discriminator loss is approximately 0.5, which usually indicates a stable and converged GAN [14].

The ALI model is additionally able to generate mel-spectrograms that look more realistic than those generated by the VAE for all three latent dimensions (Figure 9a). While the generated examples do not have the level of fidelity of the original mel-spectrograms, they do look approximately reasonable.

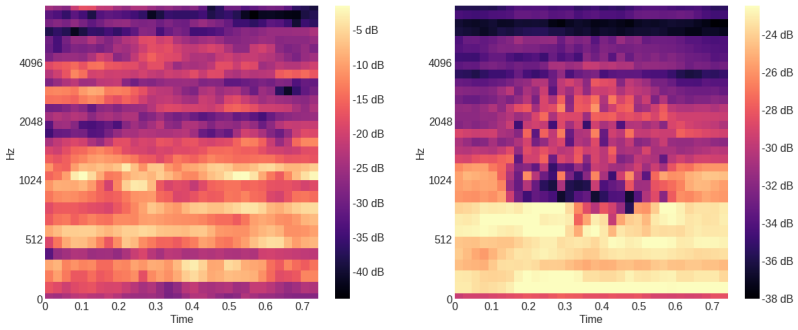
However, in terms of the downstream task of composer classification, the ALI model does not result in embeddings that improve over the baseline methods, either the classification or embedding methods (Table 7). Although the baseline models were all trained on the full mel-spectrogram (as opposed to a 32 x 32 input data size), even when we compare to baseline classification models that use the same input dimensions, the ALI model does not outperform these baselines. A baseline feed forward, fully connected model (using the same architecture described in subsection 3.2 but with the smaller 32 x 32 input data size) 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. Baseline approaches using the same dimensional data input were able to attain classification accuracies of 35.81% for the feed forward model and 40.63% for the ALI Encoder as a standalone CNN classifier.

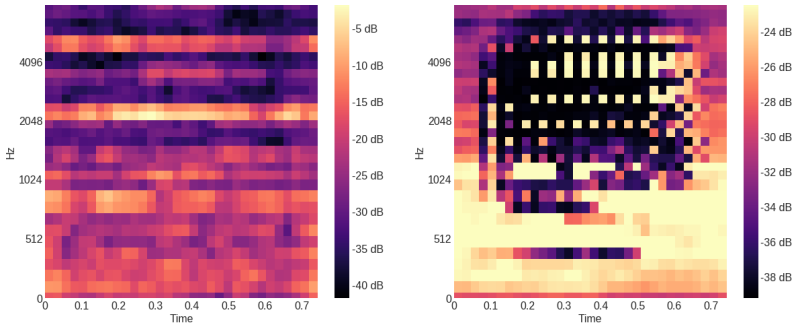
	LD = 256	LD = 128	LD = 40
ALI Training Accuracy	38.24	35.25	33.21
ALI Test Accuracy	29.91	28.67	28.67

The ALI model with a latent embedding space of dimension 256 appears to be suffering from mode collapse. For example, when interpolating in the latent space and comparing the generated images, all of the generated images appear very similar, regardless of the value of the latent vector $\boldsymbol{z}$ (Figure 10). 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.

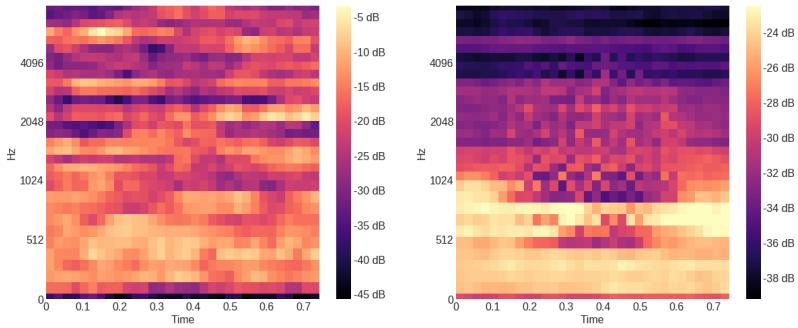
On the other hand, the ALI model with a latent dimension of 128 does show differences in the generated mel-spectrograms when interpolating in the latent space (Figure 11). For this ALI model, it appears that latent vectors $\boldsymbol{z}$ closer to the origin represent mel-spectrograms where the volume across



(a) LD = 256



(b) LD = 128



(c) LD = 40

Figure 9: Original (left) and generated (right) mel-spectrograms from the ALI model with (a) 256, (b) 128 and (c) 40 dimensional latent spaces.

the entire spectrum is approximately uniform and close to the maximum volume of the recording. The mel-spectrograms are normalized by the maximum volume of each recording, so lighter colors represent larger amplitudes. The mel-spectrograms in the first row of Figure 11 appear uniformly lighter in color than points in the bottom row, which are generated from latent $\boldsymbol{z}$ vectors that are further from the origin.

While it is difficult to interpret this trend specifically in terms of music, for the ALI model with a 128 dimensional latent space, it appears that 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.

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, it seems reasonable that the chosen embedding model should at least attain

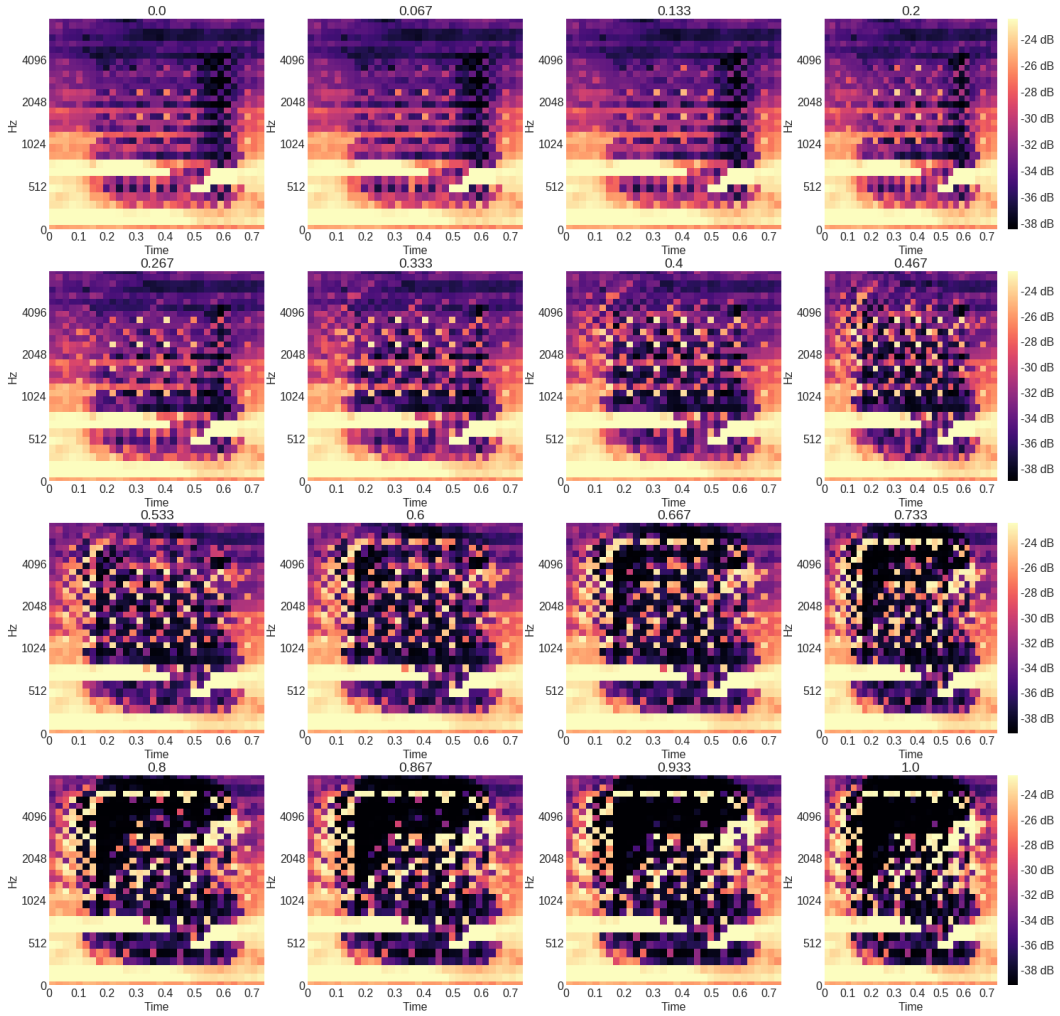


Figure 10: Interpolation in the latent space for the ALI model with latent dimension of 256. The interpolation runs from the origin to the point with coordinates all equal to 1 in the latent space. For the interpolation, each coordinate has the same value, which is the title of each subplot. This model appears to be suffering from mode collapse, as there is little to no change when interpolating in the latent space.

comparable performance to the baseline classification only approaches. This initial exploration of ALI models suggests that they may be a promising direction to pursue for embedding audio data, however, there needs to be a more thorough exploration of architectures and hyper-parameter tuning to achieve better classification performance.

5 Concluding remarks

The 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. The learned embeddings were compared to baseline approaches on a downstream task of composer classification and 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. However, the learned latent space is somewhat interpretable from a musical perspective and the generated mel-spectrograms produced by the generator do resemble the input data at a high level.

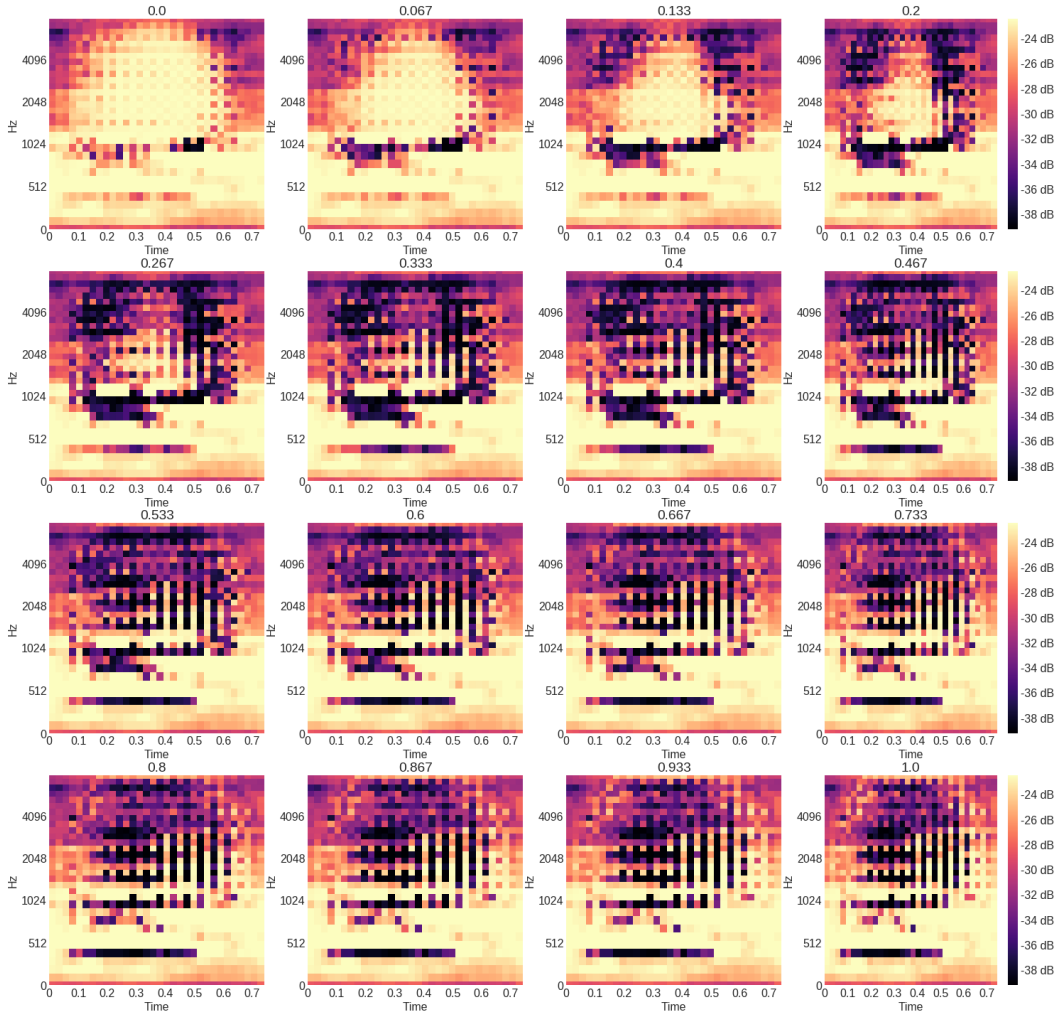


Figure 11: Interpolation in the latent space for the ALI model with latent dimension of 128. The interpolation runs from the origin to the point with coordinates all equal to 1 in the latent space. For the interpolation, each coordinate has the same value, which is the title of each subplot. For this 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.

Further exploring the ALI model seems like a promising direction for future work. In particular, exploring generator and encoder architectures that more explicitly take into account the temporal and spatial structure of the data would likely result in better latent representations. Additionally, the ALI model implemented for this project crops the input mel-spectrogram data, so exploring models that utilized all 10s of input data would likely provide a further improvement.

Finally, there remains much future work exploring how best to incorporate temporal information for these embeddings. The naive LSTM considered for this project performed the worst in terms of composer classification accuracy. Future work that considers a seq2seq type model in the context of ALI may be promising. Overall, though, this project reinforces that learning a meaningful audio embedding for any type of recording is a challenging task, but that the complex dependencies between instruments in orchestral music and the long term structure in classical music make this task particularly challenging for orchestral audio data. The classification accuracies found in this project were much lower than results found in the related work on genre classification of popular songs using similar approaches. Thus, 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

[1] 9310gaurav. ali-pytorch: PyTorch Implementation of Adversarially Learned Inference (BiGAN). GitHub.

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

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

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

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

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

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

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

[9] 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.

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

[11] 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.

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

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

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

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

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

[17] 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.

[18] 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.

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

[20] Meinard Müller. *Fundamentals of Music Processing*. Springer, 2015.

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

[22] 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.

[23] 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.

[24] 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.

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

[26] Daniel Rothmann. What's Wrong with CNNs and Spectrograms for Audio Processing?

[27] 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.

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

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

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