בס״ד

The Hebrew University of Jerusalem

The Rachel and Selim Benin School of Computer Science and Engineering

The Department of Cognitive and Brain Sciences

6124 - Seminar Paper

Class Transfer in Music Using LORD

Name: Avi Marsden

ID: 312267529

Advising Professor: Prof. Shmuel Peleg

Date of Submission: 15.07.2021

1 INTRODUCTION 3

1.1 Music Theory Background 4

1.1.1 Pitch 4

1.1.2 Rhythm and Meter 4

1.1.3 Melody 5

1.1.4 Timbre 5

1.1.5 Musical Genre 6

1.2 Deep Learning Theory Background 6

1.2.1 The Music Class Transfer Problem 6

1.2.2 Modelling the Transformations 7

2 METHOD AND RESULTS 10

2.1 LORD 10

2.1.1 Components 11

2.2 Audio Reconstruction 13

2.3 Music Genre Transfer 13

2.3.1 Dataset 13

2.3.2 Network Configurations 15

2.3.3 Loss Functions 16

2.3.4 Full-Sized Spectrogram Reconstruction 16

2.3.5 Results and Discussion 17

2.4 Music Instrument Transfer 17

2.4.1 Dataset 17

2.4.2 Full-sized spectrogram Reconstruction 18

2.4.3 Results and Discussion 18

3 CONCLUSION 19

4 REFERENCES 20

Abstract

Learning to transfer between classes for a set of observations while preserving their content is an important task for artificial intelligence. We studied the viability of using latent optimization for class representation disentanglement for class transfer in the domain of music by training and inferring using the semi-amortized version of LORD. LORD - Latent Optimization for Representation Disentanglement – is a method using latent optimization to disentangle class and content representations within a set of images. In the original LORD paper, the authors use the LORD method to disentangle class and content representations for images and to use those representations to transfer between classes for a given image while preserving its content. We use LORD for music class transfer with varying degrees of success.

1 Introduction

Objects in the real world can be observed to have different attributes, of which some are permanent, for example those that pertain to the identity of the object, or class identity, and of which some are transitory – the content of the object, for example the pose of the object. A significant task in artificial intelligence is to transfer between class identities of an object while preserving all other transitory features - for example, changing the gender of a face, while keeping its general features, such as eye shape and colour, nose shape and general face structure, and the only features that change are those that change the identity of gender – e.g. strength of jaw or sharpness of features.

A popular domain of research in deep learning is style transfer of images – generating an output image with a different style from the input image while preserving its contents. The subject of this project is class transfer in music, using latent optimisation for learning class representations and an encoder to learn content representations, for which we use the semi-amortised version of the Latent Optimisation for Representation Disentanglement (LORD) architecture [1]. While originally designed for class and content representation disentanglement in images, we experiment with LORD in a new domain – audio, specifically on music spectrograms. We attempted two tasks – music genre transfer and music instrument transfer. In the music genre transfer task we show that a few thousand samples do not suffice as training data for the network to learn class representations of genre. In the music instrument transfer task we show that this version of LORD learns class representations well and learns content representations moderately well.

In this paper, we give a brief background of music theory relating to our work (sec. 1.1), followed by a brief foundation of deep learning theory necessary for our work (sec. 1.2). We then describe the model architecture used for both tasks (sec. 2.1) and present the methods and result for music genre transfer (sec 2.3) followed by the methods and results for music instrument transfer (sec. 2.4), and conclusions (sec. 3).

1.1 Music Theory Background

Music is composed of sounds, arranged through certain aural elements, of which included are pitch, melody, rhythm, and volume [2].

We will briefly discuss several of the elements of music that pertain to our project and their relation to our work, namely the way they are expressed in the features of the spectrogram of a given musical piece.

1.1.1 Pitch

Pitch is the perceived lowness or highness of a tone and allows its placement on a frequency-related scale in sounds that have a frequency that is stable enough to be distinguished from noise [3]. Pitch can be quantified as a frequency; however it must be noted that pitch is considered by theorists to be a subjective psychoacoustical property of sound rather than a purely objective one [4]. For a musical performance, musical instruments are tuned according to a pitch standard, for example the A above middle C is usually set at 440 Hz.

In the spectrogram of a sound, a higher pitched sound correlates with lighter colours (i.e. higher amplitudes) for a set amount of timesteps in the higher frequency banks, and conversely – lower pitched sounds correlate with lighter colours (higher amplitudes) in the lower frequency banks. the pitch of a sound can thus be visualised in the spectrogram as a visual feature.

1.1.2 Rhythm and Meter

Rhythm is the sequential arrangement of sounds and silences in time. It is the way in which one or more unaccented (non-emphasised) beats are grouped in relation to an accented (emphasised) one [5]. Meter measures the groupings of beats into recurring patterns, called bars [6]. In the spectrogram of a piece of music, the rhythm can be seen as recurring durations of high and low amplitudes across time, while the meter can be seen as amplitude durations groupings across time.

1.1.3 Melody

Melody is a series of musical tones grouped together such that they are perceived as a single entity. In its literal sense, melody is composed of a sequence of pitch and rhythm and is seen as the foreground to the background accompaniment, or rather the main part of the musical piece. With regards to spectrograms, melody may be seen as the overlying pattern of different amplitudes of the frequencies that pertain to the foreground of pitch and rhythm in the musical piece.

1.1.4 Timbre

When two different musical instruments play at the same pitch and volume, they sound different to humans. This is principally due to the instruments' difference in timbre, which is the phenomenon that allows us to differentiate between two different instruments or voices, and is the quality of sounds often described in terms like bright, dull, shrill, etc. Timbre is mainly determined by two components:

1. Harmonics - the relative balance of overtones produced by an instrument - all the frequencies of a sound greater than the fundamental frequency of the sound which is the lowest frequency in the sum of harmonically related frequencies of a sound (i.e. all higher frequencies are integer multiples of the fundamental frequency). For example, when the tuning note of an orchestra is played, each instrument produces a combination of 440 Hz, 880 Hz, 1320 Hz, 1760 Hz and so on, and each instrument is unique in the different combination of those frequencies it produces.

   This balance can be observed in the spectrogram of a musical sound as the relative colours (amplitudes) of a set of harmonically related frequencies in a given time step.
2. The envelope (the smooth curve outlining the extremes of the oscillating signal) of the sound produced, or the overall amplitude structure of a sound. For example, the first blast of the player's lips on a trumpet mouthpiece is characteristic of the sound of a trumpet, which can be observed as a sudden increase in amplitude of the signal, while a slower increase in amplitude at the beginning of the signal of a violin sound is more characteristic of a violin.

   The envelope of a sound can be observed in its spectrogram as the structure of the colours (amplitudes) across time, in the way the colour increases, is sustained and decreases across timesteps. Note that this feature is continuous along the time axis, but not along the frequency axis [image]

1.1.5 Musical Genre

The genre of a musical piece may be defined by the musical techniques involved in producing it, its cultural context, and the content and spirit of the themes of the music. The Italian musician Franco Fabbri proposed that musical genre is a set of musical events whose course is governed by a definite set of socially accepted rules [7]. A musical event can be defined as any type of activity performed around any type of event involving sound. Fabbri's definition is now considered to be normative [8].

1.2 Deep Learning Theory Background

1.2.1 The Music Class Transfer Problem

As described, given a collection of musical tracks and their classes, our goal is to transfer the class of a given musical track to a different class from the collection while retaining all other information of the musical track. To this end, we disentangle the information concerning the class of a musical track from all information of the track that must remain unchanged when the track is transferred between classes, which we term the content of a track, and apply the class representation of a musical track belonging to a different class to the given track, while preserving its content. Let us formalise this process:

Let us denote the space of all possible tracks as $\mathcal{D}$. Assume we are given a collection of n musical tracks $x_1, x_2, \dots, x_n \in \mathcal{X} \subseteq \mathcal{D}$. Each musical track $x_i \in \mathcal{X}$ possesses a class label $y_i \in [k]$. We assume that every musical track belongs to a single class, and that $k \ll n$ so several musical tracks share a single class label. Let us define $\mathcal{Y}$ as the latent space of class representations and $\mathcal{C}$ as the latent space of content representations. We define the class representation $e_{y_i} \in \mathcal{Y}$ of a track x_i as only all the information that is shared by all images of the same class, and the content representation $c_i \in \mathcal{C}$ of a track x_i as only all information that remains unchanged when a track is transferred between classes. We define the style representation of a track s_i as all other information not included in either the class or content representations, and we denote the latent space of style representations as S.

We want to find the transformations:

$$Y: [k] \to \mathcal{Y}, \qquad C: \mathcal{X} \to \mathcal{C}$$
$$s.t. \; for \; all \; x_i \in \mathcal{X}, \qquad Y(y_i) = e_{y_i} \; C(x_i) = c_i \qquad \textit{Eq. 1}$$

In other words, $Y(y_i)$ is the class representation of x_i, and $C(x_i)$ is the content representation of x_i. In addition, in order to generate the transferred track, we want to find a transformation parameterised by θ:

$$G_\theta: \mathcal{Y} \times \mathcal{C} \to \mathcal{D}$$
$$s.t.\ G_\theta(e_{y_i}, c_i) = x_i$$

Eq. 2

And such that for all $e \in \mathcal{Y}, c \in \mathcal{C},$ the class representation of $G_\theta(e, c)$ is e, and the content representation of $G_\theta(e, c)$ is c. In other words, the transformation G_θ – the generator – must successfully reconstruct a track from its class and content representations, in addition to generating from any class and content representations a synthesised track that preserves the class and content representations.

Therefore, we can conclude that given two tracks $x_i, x_j \in \mathcal{X}$ and their class labels y_i, y_j, using the above transformations, we can transfer the class of x_i to the class of x_j while preserving the content of x_i, denoting the synthesised track $x_{y_i \to y_j}$:

$$Y(y_j) = e_{y_j}, \qquad C(x_i) = c_i$$
$$G_\theta\left(e_{y_j}, c_i\right) = x_{y_i \to y_j}$$

Eq. 3

And according to the definition of G_θ, the class representation of $x_{a \to b}$ is e_{y_j} and the content representation is c_i.

1.2.2 Modelling the Transformations

We have seen that finding the transformations Y, C and G will enable us to transfer music class. We use Deep Neural Networks (DNNs) to learn and model these transformations.

1.2.2.1 Content Representation

We saw in sec. 1.1 that many principal features of a piece of music can be observed as visual features in the spectrogram of the musical piece. We therefore attempt to treat the problem of finding C, or the content encoder, as an image class transfer problem by transforming each music track to its spectrogram and applying deep learning methods on the spectrograms.

We use a convolutional neural network (CNN) to learn the transformation C. Convolutional layers are especially effective at learning and extracting visual features because they can model linear translation-equivariant systems [9], which is a requirement we have of C:

1. translation-equivariance - translation in space of the input leads to translation in space of the output: if L is the function, x the input to the function and $L[x(t_1, t_2)] = x'(t_1, t_2)$ then:

$$L[x(t_1 - T_1, t_2 - T_2)] = x'(t_1 - T_1, t_2 - T_2) \qquad \textit{Eq. 4}$$

[10]

There are several music features that are translation-equivariant: For example, a series of notes that compose a melody compose the same melody even when they are shifted to a different frequency (i.e. they are at a different scale/octave) or to a different time interval. Rhythm and meter too, do not change whether they are shifted (as a whole) between time intervals or across frequencies. [cite] Therefore if these features appear in two spectrograms (or within the same spectrogram) at a different time step, or at different frequencies, e.g. the visual features associated with a specific rhythm, they must be incorporated by C as the same information. Therefore C is required to be translation-equivariant at the stage of feature extraction.

2. Linearity – additivity and homogeneity – for inputs $x_1(t_1, t_2)$, $x_2(t_1, t_2)$ such that $L(x_1(t_1, t_2)) = y_1(t_1, t_2), L(x_2(t_1, t_2)) = y_2(t_1, t_2)$ it holds that:

$$\begin{aligned} L[x_1(t_1, t_2) + ax_2(t_1, t_2)] &= L[x_1(t_1, t_2)] + aL[x_2(t_1, t_2)] \\ &= y_1(t_1, t_2) + ay_2(t_1, t_2) \end{aligned} \qquad \textit{Eq. 5}$$

Where a is a constant. Several music features remain the same when the amplitudes of the music track are multiplied (i.e. the volume is changed). For example, a melody only becomes louder when the amplitudes are multiplied, but it remains the same melody. Rhythm and meter also remain the same when the amplitudes are uniformly multiplied, as does pitch – the frequencies remain the same, only the amplitudes change. Therefore if the amplitudes of a feature in a spectrogram is multiplied by a constant (i.e. the sound is simply louder), the feature must be transformed by C as the same information before the multiplication. Therefore C is required to be linear at the stage of feature extraction.

When the input passes sequentially through several convolutional layers, each layer outputs a feature map, which is a tensor of predefined depth of visual features that the convolution operation extracted. In order to extract features of increasingly lower resolution (i.e. that span a greater visual field in the input), we apply the convolution every s strides – for example, if $s = 2$ we apply the convolution every 2 pixels in the spectrogram, thus reducing each dimension (width and height) by half and reducing the size of each filter in the feature map by

a quarter, eventually reaching a feature map of high dimensional depth but low dimensional width and height, which contains features of very low resolution, or in terms of the original spectrogram – features that span over the entire spectrogram rather than just a small portion of it.

Note that if our CNN – which is to model C – were entirely comprised of convolutional layers, this would render the network capable of only modelling linear functions; restricting the network to modelling only linear functions greatly inhibits its expressive power, because C is not a linear transformation. Therefore, it is necessary to add activation layers that break linearity (in our case ReLU), in addition to adding more than one Fully Connected layer at the end of the network, also with non-linear activation layers. In addition to breaking linearity, these fully-connected layers can learn and model relationships between features, for instance associating several features of pitch and a specific meter to incorporate the concept of a certain melody into the resulting content representation c_i.

1.2.2.2 Class Representation

Recall that for each $x_i \in \mathcal{X}$ we have a class label $y_i \in [k]$. Therefore, the task of learning the class representation is a supervised learning task. In their work, Gabbay and Hoshen optimise directly over the latent class and content representations in order to achieve highly disentangled class and content representations; optimising directly over the latent vectors (representations) means that for each input – in the case of content, each image, and in the case of class, each class label – a latent vector is kept. Initially these latent vectors or randomised, and during learning, the values of each latent vector are directly optimised with relation to minimising some loss function. Gabbay and Hoshen show that this latent optimisation achieves state-of-the-art results in representation disentanglement. In addition, they show that when latent optimisation is performed only on class representations, but content representations are learned by an encoder, class information is leaked into the content representations, but far less than if both representations are learned using encoders.

In music, the requirement for near-absolute disentanglement should be relaxed, because often (depending on the definition of the class), we may desire including style information, which varies according to class – for example certain attributes of a melody (style) are changed based on the instrument (class) playing it. We therefore use latent optimisation on class labels to reach class representations and an encoder on the spectrograms to reach content representations – this is the semi-amortised version of LORD.

1.2.2.3 Generator

Generative networks model probability distributions and can generate novel samples from the distributions [11]. G receives vectors from the latent space and applies transformations on the vectors to create novel samples. The network modelling the generator function G is also a CNN and will be described in detail in sec. 2.1.1.4.

2 Method and Results

2.1 LORD

As described, the semi-amortised version of LORD is comprised of three main components: content encoding, class embedding and generator (see Fig. 1, and see below for detailed descriptions of these components). During training, the spectrograms are fed to the content encoder (see sec. 2.1.1.1), which outputs a vector for each spectrogram – the content code. In parallel, the genre label for each of those spectrograms is fed to the class embedding (see sec. 2.1.1.2), which outputs a vector for each track – the class code.

The class code is passed through a class modulation component (see sec. 2.1.1.3), which prepares the class code for the Adaptive Instance Normalization layer in the generator (explained below). The class code and the content code are then passed to the generator (see sec. 2.1.1.4), which attempts to reconstruct the spectrogram image according to the given class code and content code.

The loss of the network is measured between the input spectrogram and the reconstructed spectrogram.

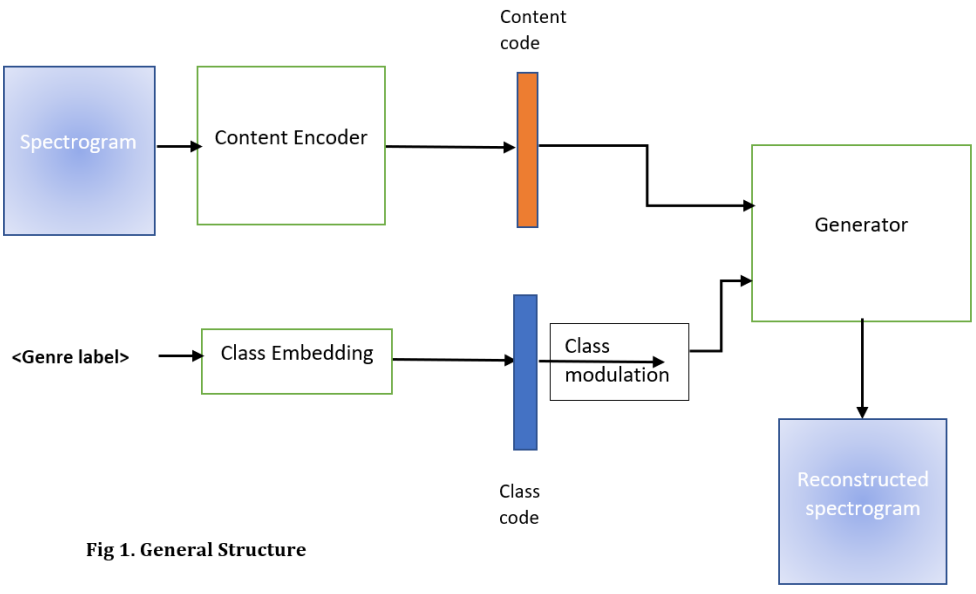


Fig 1. General Structure

2.1.1 Components

2.1.1.1 Content Encoder

The content encoder (see Fig. 2) consists of five convolutional layers. The kernel of the first convolutional layer is of shape $(7,7)$ and of stride $(1,1)$. The remaining four convolutional layers have a kernel size of $(4,4)$ and a $(2,2)$ stride. After the convolutional layer, the tensor is flattened and three fully-connected layers follow, outputting the content code. Every layer is followed by a Leaky ReLU activation layer, except for the last fully-connected layer which is followed by a Gaussian Noise regulariser.

The content encoder is the amortized component of the LORD model. Note that in the unamortized configuration of LORD, the content encoder is replaced by a content embedding layer.

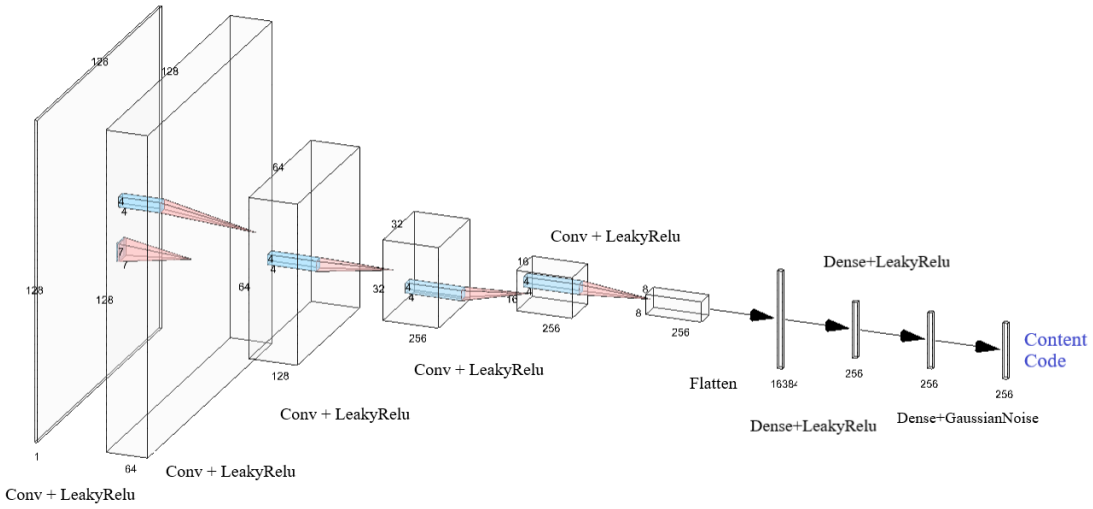
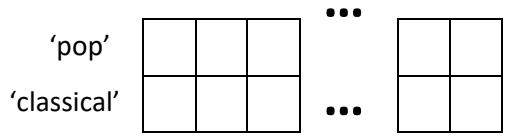


Fig 2. Content Encoder

2.1.1.2 Class Embedding

An Embedding layer keeps a table of parameters of shape $(number\ of\ classes, class\ dimension)$, where the class dimension is the predefined size of the class code vector, i.e. for every unique class in the dataset, a vector of size $class\ dimension$ is kept. Essentially, this table is a lookup table that outputs a vector according to the given class. The values, or the parameters of the vectors, are learned and optimized directly during training (in our case using Stochastic Gradient Descent – the Adam optimiser) such that the loss – which is computed between the input image and the reconstructed image – is minimised. The vector outputted is the class code, and the class code

is shared by all images of the same class. These class codes are latent vectors, encapsulating information about each class in latent space.



The Class Embedding component is the non-amortized component of the LORD model.

2.1.1.3 Class Modulation

The purpose of the Class Modulation component is to prepare the class code for Adaptive Instance Normalization (AdaIN). The class modulation hyperparameters are $AdaIN\ dimensions$ and $number\ of\ AdaIN\ layers$. $number\ of\ AdaIN\ layers$ vectors are constructed by fully connected layers (of size $2 \cdot AdaIN\ dimension$) applied to the class code. The vectors are then concatenated and reshaped to shape $(number\ of\ AdaIN\ layers, AdaIN\ dimensions, 2)$. Each AdaIN vector is fed separately into the generator to be combined with the content code using an AdaIN layer (discussed in sec. 2.1.1.4).

2.1.1.4 Generator

The generator receives the content code, passes it through three fully-connected layers, reshapes it to shape $\left(\frac{h}{2^n}, \frac{w}{2^n}, d\right)$, where h, w are the height and width of the original spectrogram, n is the number of AdaIN layers, and d is the AdaIN dimension. Then, for $number\ of\ AdaIN\ layers$ iterations, the tensor passes through a block consisting of an upsampling layer $(2,2)$, a convolutional layer (kernel size $(3,3)$, $AdaIN\ dimension$s filters), leaky ReLU activation, and an Adaptive Instance Normalization layer. The function of the AdaIN layer is to combine the content code c with the class code y by normalizing them such that per-channel mean and variance of the content code is aligned to match those of the class code:

$$AdaIN(c, y) = \sigma(y)\left(\frac{c - \mu(c)}{\sigma(c)}\right) + \mu(y) \qquad \textit{Eq. 6}$$

In each iteration i, the class code used is the i-th tensor of shape $(AdaIN\ dimensions, 2)$ from the output of Class Modulation.

After the above iterations, the tensor passes through another convolutional layer (64 filters, (5,5) kernel), leaky ReLU activation, and finally another convolutional layer (1 filter, (7,7) kernel) followed by sigmoid activation.

The generated spectrogram is then compared against the input spectrogram to calculate the loss of the network.

2.2 Audio Reconstruction

We reconstruct the waveform from the generated spectrogram by transforming the generated spectrogram from the decibel scale to the power scale, multiplying the mel-spectrogram by the pseudo-inverse of the mel filterbank, and use the Griffin-Lim algorithm [12] to recover the phase information (in configurations in which phase information was not included, see sec. 2.3.1).

2.3 Music Genre Transfer

2.3.1 Dataset

For music genre transfer, we used the "medium" Free Music Archive (FMA) dataset [13]. Each track in the dataset is a 30s MP3 track, extracted from the middle of the original full track, and every track is categorised by its artist into one or more genres. Genres in the dataset are ordered hierarchically, however every track in the dataset used belongs to exactly one of 16 "root" genres, thus resulting in a single label for each track. The training data consists of music belonging to only two top genres. Genre pairs experimented with were instrumental-pop, classical-pop and rock-folk.

A mel-spectrogram is generated from every 30s MP3 track by converting the track to a waveform signal, applying short-time-Fourier-transform to the waveform signal, and multiplying the spectrogram by a mel filterbank. The mel-spectrogram is then converted to decibel scale, resulting in a spectrogram of shape 128×1280. At this point, we experimented on several configurations of the training dataset to train the network.:

1. Partitions of shape 128×128 (~3 seconds), chopped consecutively from the full spectrogram. This has the advantage over the following dataset configurations that each spectrogram partition has unique contents, because repetition of data within a dataset may bias the model towards the training data, causing overfitting and reducing its ability to generalise well. Because this resulted in a smaller dataset, the training loss function was not minimised well during training.

2. Partitions of shape 128×128, chopped from each spectrogram with 25% overlap between consecutive partitions. This had the advantage of increasing the size of the dataset but had the disadvantage of running the risk of overfitting. Indeed, this configuration reached lower training loss than the previous configuration.
3. Identical to the previous configuration, but with 50% overlap. This of course decreased training loss, but increased overfitting.
4. Incorporating phase information – we expected that this would improve the performance of the network since amplitude spectrograms are missing phase information. Phase information has been shown to lead to improved effectiveness of training CNNs on spectrograms when incorporated into training data [14]. In this context, the following configurations were tried:
    a. Two-channel spectrograms where the first channel is the amplitude information, and the second channel is the phase information. We split the phase information into its sine and cosine components and apply the mel-transform to each, thus generating two mel-spectrograms for the phase. We then derive the angle from both resulting spectrograms, which becomes the second channel (where the first channel is the mel-transformed amplitude spectrogram) and use MSE cyclic loss (e.g. where $L(\pi, -\pi) = 0$ and 0 and π are the furthest distance possible). The network seemed unable to learn the phase information. A reason for this may be that CNNs have trouble learning cyclic information (For example, convolution being an edge detector, may recognise a change in the image from $-\pi$ to π as a large change even though according to the loss function $L(\pi, -\pi) = 0$.
    b. Three-channel spectrograms, where the first channel is the amplitude, and the second and third channels are the mel-transformed cosine and sine of the phase information. This way, we presented the phase information to the network as non-cyclic, to facilitate correct learning of the convolutional layers. The results were inconclusive but were more promising than the previous configuration.

    We speculate that introducing phase to the network failed because phase information appears visually as noise and lacks any coherent visual features, which the encoder is optimised to find. Additionally, visual features of the amplitude spectrogram were observed to be leaked into the phase spectrogram. These problems may be overcome by increasing the dataset size.

Note that for every configuration, the relevant genre label was kept for each individual partition.

To reduce intra-class variation, we tried to apply clustering to the dataset samples of each class and used the resulting cluster labels as class labels in the dataset, the cluster labels acting as subclasses to the original classes. This was performed as described in the LORD paper, by running every sample through the first convolutional layer of the VGG16 network [15] and using the k-means clustering algorithm [16] to cluster over the results. However, this resulted in small sample sizes for each subclass, from which the network was unable to learn the class codes.

2.3.2 Network Configurations

We trained LORD both with and without modification to the network itself and tried different dimensions for the class and content representations vectors; we tried many different variations of these dimensions, from 128 dimensions to 2048 dimensions, however the best results were found when using lower dimensions, e.g. 128 dimensions for the content code and 256 dimensions for the class code. A possible explanation is that though higher dimension vectors can represent more information, lower dimension vectors act as a bottleneck and help the network encode only the most essential information.

We also attempted applying many modifications to the network itself, among which were:

1. Increasing the number of units in the fully-connected layers of the encoder and generator from 256 to 512. This did not decrease training loss, nor did it improve generated music quality.
2. Increasing the kernel size in the convolutional layers. We speculated that larger images (the spectrograms being 128×128 as opposed to 64×64 in the LORD paper) have larger features and therefore require larger kernels to extract such features. This decreased the effectiveness of the network.
3. Doubling the number of filters for each convolutional layer. We speculated that firstly, larger images contain more features and so require larger feature maps, and secondly, music contains more content characterising features than images. This too, did not decrease training loss nor improve generated music quality, either because the assumptions were false, or perhaps because the network architecture could not handle such an increase in number of filters.

2.3.3 Loss Functions

We tried several different loss functions and loss function combinations during training, among which were:

1. Mean Square Error (MSE)
2. Mean Absolute Error (MAE) – this usually resulted in generated audio with a prominent robotic influence. MAE as a loss function tends to cause the network to ignore outlying data, which may be a detrimental property in the music domain.
3. Different combinations of MSE and MAE – these generated superior results over MSE and MAE separately, perhaps because of the balance of ignoring outlying data, characteristic of MAE, and overcorrecting for outlying data, characteristic of MSE.
4. Perceptual Loss [17] – this had significantly inferior results over the previous loss function regarding generated music quality. This may be because perceptual loss measures perceptual similarity between the input and the output by running them through the first layer of a trained VGG16 network and computing MAE between the results. Since VGG16 is trained only on images of real-world objects, the network may be unable to learn the visual features of a spectrogram.
5. Combination of Perceptual Loss, MSE and MAE – generated audio was of similar quality to that of the MSE and MAE combination, with a slight improvement.

2.3.4 Full-Sized Spectrogram Reconstruction

To stitch together the small 128×128 shaped, generated spectrograms to a full-length track we attempted several methods:

1. Simple concatenation of consecutive generated spectrograms – this resulted in audible points of discontinuity, or “jumps” occurring every few second in the reconstructed waveform.
2. Combining overlapping regions of consecutive spectrograms along the time axis using a Gaussian mask – given two consecutive input spectrograms I_i, I_{i+1} both of shape $m_1 \times m_2$, assume they overlap by n pixels along the time axis. We denote their generated spectrograms I'_i, I'_{i+1}. Let us denote $\hat{I}_i, \hat{I}_{i+1}$ as the $m_1 \times k$ overlapping regions of each generated spectrogram, respectively. We compute $2n$ binomial coefficients normalised to $[0,1]$, denoted $a_1, \dots, a_{2n}$, and compute the sum of the combination of the two generated spectrograms in the overlapping areas:

$$for\ each\ t_1 \in [m_1], t_2 \in [n],$$
$$\hat{I}_{[\hat{I}_i,\hat{I}_{i+1}]}(t_1,t_2) = (1-a_{t_2})\hat{I}_i(t_1,t_2) + a_{t_2}\hat{I}_{i+1}(t_1,t_2) \qquad \textit{Eq. 7}$$

Note that we only used the first n coefficients $a_1, \dots, a_n$, thus using the left side of a Gaussian curve in $[0,1]$ as a mask for the overlapping regions of the two spectrograms. This resulted in smooth transitions in the generated waveform, however audio quality was compromised.

2.3.5 Results and Discussion

An issue that arose in all the experiments conducted in the domain of music genre transfer was that even when the model managed to learn the identity transform decently well, it was unable to come close to genre transfer. This may be due to the complexity of learning a clear representation of a given genre with the data as it is; Firstly, intra-genre variation is high – for example, pop music has a wide range of styles and subgenres that do not necessarily have much in common. Secondly, the definition of what makes a piece of music belong to a certain genre may be too complex. At the very least a far larger dataset would be required for successfully learning such a concept.

2.4 Music Instrument Transfer

In consideration of simplifying the task at hand, we attempted instead the task of music instrument transfer; thus the class is defined as the musical instrument playing the track, the content is defined as the melody, and the task is to transfer a track of solo music from one instrument to another while preserving the content.

Loss Function We use a combination of Perceptual Loss, MSE and MAE.

2.4.1 Dataset

Two different datasets were used for this task:

1. Medley_solos_db [18] – a collection of 21572 3-second solo instrument music clips, each played by a single instrument out of eight: Clarinet, distorted electric guitar, female singer, flute, piano, tenor saxophone, trumpet, and violin. Each clip is labelled by its respective instrument.
2. YouTube downloaded dataset – ~12 hours of solo music were downloaded from YouTube, containing ~6 hours of piano solo music and ~6 hours of violin solo music. The tracks were chopped on silences, and the resulting clips were further chopped into

3-second solo instrument music clips. Each clip was labelled by its respective instrument – piano or violin. Thus more samples were available for each class.

Here too, we apply STFT to the waveform signal and multiply the spectrogram by a mel filterbank. The mel-spectrogram is then converted to decibel scale.

For the test dataset, a solo piano music track (not appearing in the train dataset) was downloaded from YouTube, transformed to a mel-spectrogram as described, which was then sliced to spectrograms of shape 128×128, with $\frac{127}{128}$ overlap – i.e. each spectrogram has a single unique timestep, while the remaining 127 timesteps are shared with its neighbouring spectrograms.

In addition, every spectrogram was assigned its respective instrument label.

2.4.2 Full-sized Spectrogram Reconstruction

For the YouTube dataset, the spectrograms are outputted by the network as small 128×128 spectrograms. To combine the clips and to create a full-sized track, we consecutively concatenated the single unique timestep from each consecutive spectrogram, thus creating a full-sized spectrogram. When reconstructing the waveform using the method described in sec. 2.2., this resulted in low quality audio. The results were improved drastically when a larger number of timesteps from each spectrogram were concatenated to create the full-sized spectrogram.

2.4.3 Results and Discussion

The results of training on either dataset were promising, far better than the results of the genre transfer task.

When trained on the medley_solos_db dataset, all instruments transferred well to piano (albeit not perfectly), and not just a simple transfer of melody - i.e. the notes played with the right hand – the transfer also included the support contributed by the left hand (i.e. the background support). Piano seemed to be the best learned class, because the transfers to other instruments were not as successful. Perhaps this was because the discretisation of the notes in a piano is easy for the model to learn, while more "continuous" instruments are more troublesome. Additionally, the melody of an audio clip tended to be preserved between transfers, especially when it was a simple melody (i.e. only a small number of notes). It is important to note that there was a clear correlation between instruments that have less appearances in the dataset and how badly they are learned, but the opposite correlation is not necessarily true - the violin class has 2900 clips but the transfers to the violin class were not so

successful, though a transfer had clearly been done. Still, these problems may be solved with a larger dataset, because the network was certainly on the right track towards learning the required latent representations.

When trained on the YouTube dataset, the identity transform was learned by the network well. the music instrument transfer was far more successful, both from piano to violin and vice-versa. The generated music could be clearly identified as belonging to the target class. Melody was moderately well-preserved between transfers, though not perfectly so, while simpler melodies were transferred better between classes compared to more complex melodies.

The efficacy of the latter dataset over the former dataset as the training data may be attributed to the increased number of training samples for each class, thus enabling the network to learn the class representations better. This might explain the fact that generated clips could be better identified as belonging to the target class in comparison to class transfer when trained on the former dataset.

3 Conclusion

We used several configurations of the semi-amortised version of LORD to achieve music genre transfer. The identity transform was somewhat learned, however we conclude that due to the complexity of the concept of genre, the network failed to learn class representation for genre, and so failed on the genre transfer task. To simplify the task of class representation, we redefined the class transfer task to musical instrument transfer, thus the network must only learn the class representation of a musical instrument, which may have less intra-class variation than the class representation of a musical genre. Class transfer in this task was successful and produced generated music that could be clearly classified to the target class; however content (melody) preservation was only partially successful. In future works, we suggest expanding the size of the datasets used, which could greatly improve results.

In this paper, we have shown that latent optimisation for class representation disentanglement is a viable method for class transfer in audio. Based on this work, we suggest the following research directions: Firstly, experimenting with the described method on far larger datasets, while also experimenting with and without phase information on the larger datasets. Secondly, when intra-class style variation is high, training a classifier on a relevant dataset to discriminate between the classes and using the feature maps of the classifier both as a component of the loss function and as a pre-processing stage for clustering.

4 References

[1] A. Gabbay and Y. Hoshen, “Demystifying inter-class disentanglement,” in *International Conference on Learning Representations (ICLR)*, 2020.

[2] C. V. Palisca and I. D. Bent, “Theory, theorists,” 2001. [Online]. Available: https://www.oxfordmusiconline.com/grovemusic.

[3] H. S. Powers, The harvard dictionary of music, 4th ed., Harvard University Press, 2003, pp. 499-502.

[4] W. M. Hartman, Signals, sound, and sensation, New York: Springer, 1997, pp. 145, 284, 287.

[5] G. W. Cooper, G. Cooper and L. B. Meyer, The rhythmic structure of music, Chicago: University of Chicago press, 1963.

[6] B. Benward, Music in theory and practice, 7th ed., vol. 1, Boston: McGraw-Hill, 2003.

[7] F. Fabbri, “A theory of musical genres: two applications,” *Popular Music Perspectives*, p. 52, 1981.

[8] A. F. Moore, “Categorical conventions in music discourse: style and genre,” *Music & Letters*, vol. 82, no. 3, pp. 432-442, 2001.

[9] I. Goodfellow, Y. Bengio and A. Courville, Deep learning, Cambridge: MIT Press, 2016.

[10] A. V. Oppenheim and R. W. Schafer, Digital signal processing, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1975.

[11] I. Goodfellow, Y. Bengio and A. Courville, Deep learning, Cambridge: MIT Press, 2016.

[12] D. Griffin and J. Lim, “Signal estimation from modified short-time fourier transform,” in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1984.

[13] M. Defferrard, K. Benzi, P. Vandergheynst and X. Bresson, “FMA: A dataset for music analysis,” in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[14] L. Guo, L. Wang, D. Jianwu, H. Guan and X. Li, “Speech emotion recognition by combining amplitude and phase Information using convolutional neural network,” in *Interspeech*, Hyderabad, 2018.

[15] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations (ICLR)*, 2015.

[16] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[17] Y. Hoshen and J. Malik, “Non adversarial image synthesis with generative latent nearest neighbors,” in *CVPR*, 2019.

[18] R. M. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi and T. Kell, “Mirdata: software for reproducible usage of datasets,” in *International Society for Music Information Retrieval (ISMIR)*, 2019.