Transfer Learning for Underrepresented Music Generation

by

Anahita Doosti Sanjani

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

The increasing popularity of Deep Neural Networks (DNN) has led to their application to many domains, including Music Generation. However, these large DNN-based models are heavily dependent on their training dataset, which means they perform poorly on musical genres that are out-of-distribution (OOD) for that dataset. This heavily limits these systems' practical use and essentially requires the model to be retrained on a large dataset containing a musical genre in order to recreate it. In many domains, transfer learning has been effective at adapting an existing model to a new target dataset of much smaller size by training for a much shorter period. However, such an approach remains underexplored in the domain of music generation. To investigate the viability of this approach, we explored different genres that might represent OOD genres for a DNN-based music generator. Consequently, we identified Iranian folk music as an example of such a genre of music. This was in line with the fact that this genre has a melodic structure different from music based on Western music theory principles. We then proceeded to collect a dataset of Iranian folk music and utilize it to explore different methods of transfer learning to improve the performance of MusicVAE, a large generative music model with a DNN architecture. We identify a transfer learning approach that allows us to efficiently adapt MusicVAE to the Iranian folk music dataset, which indicates a potential for the future generation of underrepresented music genres.

Acknowledgements

Firstly, I would like to express my deepest gratitude to my supervisor, Professor Matthew Guzdial. I could not have completed this work without his guidance and encouragement. I am eternally grateful for the understanding and unwavering support I was shown during a very challenging time in my personal life.

Additionally, I would like to extend my thanks to the many people who have helped me during my journey as a researcher; My committee examiners, Professor Pierre Boulanger and Professor Levi Lelis as well as Professor Janelle Harms, whose kind advice helped me navigate my program. Moreover, I am grateful to the people at Amii who have created a very supportive community of Artificial Intelligence researchers, of which I have been very fortunate to be a part . I also would like to acknowledge and thank my friend and fellow UoA student, Pouneh Gorji, who passed away in the tragic plane crash of January 2020. Without her kind encouragement, I would not have embarked on this academic journey at the University of Alberta.

Last, I would like to express my appreciation and gratitude for my family and friends. They have always been my source of strength and motivation. I have been very fortunate to have their unconditional love and support which has helped me through all manner of challenges throughout my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The automatic generation of music by means of computer systems can be traced back to 1957, when Newman Guttman generated a 17 second long melody called “The Silver Scale” at the Bell Laboratories. The first score was composed by a computer in the same year, called “Illiac Suite” at the University of Illinois Urbana-Champaign (UIUC) [2]. Most of the traditional automatic music generation systems were algorithmic systems that ultimately relied heavily on expert musical knowledge, rule/grammar-based [12] algorithms being an example [10]. Stochastic models like Markov chains [19] are also a traditionally common method. With the reemergence of Neural Networks (NNs) in recent years, there has been much interest in revisiting the task of music generation using such large machine learning models. The hope is that their superior computational power can learn long-term and high-order dependencies and also eliminate the need for expert domain knowledge [2]. Characteristically, these models are trained on a very large amount of data. Yet, realistically no matter how large, these datasets cannot be exhaustive. Datasets are usually limited to the types of music that are more popular. Particularly in case of working with symbolic music data, musical performances need to be transcribed in a particular format. Therefore any large symbolic dataset is usually gathered from online sources, which naturally results in popular music genres being more represented. Moreover, new music is constantly being created and new genres of music will organically emerge. Consequently this leads to a key weak point; these models are not as successful in generating music for genres outside their learned distribution and this distribution is closely based on their training datasets.

In recent years, there are have been a few prominent approaches to music generation, most of which involve the use of neural networks [4]. For instance, sequence-based approaches are popular in this field due to their ability to learn long-term dependencies in musical pieces. Multiple studies have combined sequence-based models such as Long short-term memory (LSTM) Recurrent Neural Networks with autoencoders and achieved good results [22]. Alternatively, Generative Adversarial Networks (GAN) have been employed to generate novel music [30]. These approaches have mainly attempted to create a general generative model, which is not customizable and is highly dependent on the composition of the training dataset. These models typically are also trained from scratch. However, given the data imbalance across different genres, approaches like transfer learning that can adapt knowledge from one domain to another might be useful. Despite this, few examples of prior work explore this approach. One such example attempts to test a pre-trained Generative Adversarial Networks (BinaryMuseGAN) with traditional Scottish music and improves its performance using finetuning [18]. To the best of our knowledge, finetuning is the only transfer learning approach that has been applied to DNN music generation [27].

For the purposes of this thesis, we conducted a study on Google Magenta's MusicVAE model [24]. This model is trained on an enormous dataset of about 1.5 million unique MIDI files collected from the web. Observing different resources for MIDI files, which are usually fan-made annotations of popular songs, we can see that automatic and indiscriminate data collection would result in an unbalanced dataset in terms of genre diversity. This is due to the fact that popular chart-topping songs are much more likely to be annotated in the MIDI format. Therefore, we could assume that MusicVAE will not be good at generating more specific and less mainstream genres of music. Exploring several different genres, we discovered that this is indeed true for Iranian folk music which was picked due to it being very different than popular Western music in structure and due to the author's familiarity with it. Subsequently, we surveyed several different methods of transfer learning to adapt MusicVAE to this particular genre using a minimal amount of new data, such as fine-tuning and Conceptual Expansion Monte Carlo Tree Search(CE-MCTS) [17]. Based on the accuracy of the model on reconstructing musical samples, we observed that CE-MCTS offers an improvement in quality especially in test accuracy. Finetuning all layers, performed only slightly better than MusicVAE itself and other methods lead to a poor performance.

Throughout the span of this thesis, pursuing our aforementioned motivations, we were interested in examining the following research questions:

1. How good are large Deep Learning music generation models at reconstructing music of different genres?

2. Is it possible to specialize such a model for a specific out-of-distribution (OOD) genre?

3. More specifically, is it possible to improve MusicVAE's performance on Iranian music through transfer learning methods using a small amount of data?

4. What is the effects of the melody extraction algorithm used, on the model's performance?

Based on the results of our research, we can infer that a large music generation model like MusicVAE struggles in reconstructing music from OOD genres such as Iranian folk music. Using a CE-MCTS transfer learning approach, we were able to improve the reconstruction accuracy of MusicVAE on a new dataset of 100 Iranian folk songs. We also venture that, based on our subjective opinion, music generated by this approach is more similar to the target genre in comparison to other methods.

This thesis includes six different chapters. After this Introduction, we continue with the background information essential to our work in chapter 2. In chapter 3, we introduce the target model of our study, MusicVAE, in more detail and proceed to explain the different approaches used in this study. Subsequently, we go through our different experiments and their setup. We also discuss various datasets used and our evaluation protocol. We conclude chapter 3 by presenting the results and following up with their analysis and comparison. Finally, in chapter 4 we examine the implications of our results and the conclusions from our experiments and observations. We also discuss how this work can be expanded upon in the future.

Chapter 2

Background

This chapter provides the reader with the background information necessary in understanding the presented work in this thesis. We begin by introducing artificial neural networks in 2.1. In 2.1.1 and 2.1.2, we introduce recurrent neural network architectures that enable learning from sequence data. Sections 2.1.3 and 2.1.4, explain autoencoders and their modified version, variational autoencoders. The MusicVAE [24] model used in this study is a variational autoencoder that uses LSTMs as its principle components. In 2.2, we introduce the concept of transfer learning, which is the focus of this research. Next, in 2.3, we cover conceptual expansion Monte Carlo tree search (CE-MCTS) which is a transfer learning method we employed in our work. In section 2.4, we go over the basics of music generation and most recent methods for generating music using deep learning methods and subsequently in 2.5, we detail the Music Generator used in this research, MusicVAE, its setup and data, prepossessing, and training processes. Finally in 2.6, we give the reader some background on Melody extraction.

2.1 Artificial Neural Networks

Artificial Neural Networks(ANN) are the foundations of some of the most prominent modern developments in the Artificial Intelligence and Machine Learning fields. These networks, as is apparent by their name, are inspired by the mechanics of the neurons in the brain. A simple architecture of an artificial neuron is shown in 2.1.

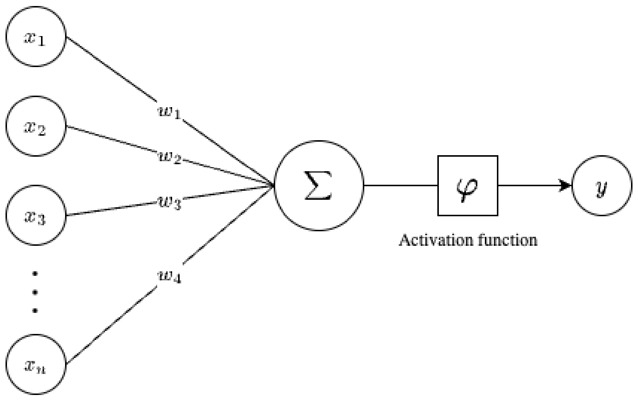


Figure 2.1: An artificial neuron

As it is depicted in 2.1, each neuron receives a number of input values such as $x_1, x_2, \cdots, x_n$, each with separate weights $w_1, w_2, \cdots, w_n$, and returns an output y by feeding the weighted sum of the inputs to a function known as the activation function φ which is usually a non-linear function with certain properties.

$$y = \varphi(\sum_{i=1}^{n} x_i * wi)$$

A neuron can receive feature data values as its inputs and output the target prediction or value for a machine learning task such as classification or regression. However with more complex data the need for more than one computational unit arises. Artificial Neural Networks consist of a number of interconnected neurons organized in layers.

A Deep Neural Network (DNN) has at least three layers; an input layer, at least one hidden layer, and an output layer. The nodes in each layer are connected to the nodes in the next layer via weighted directed edges hence the network forms a weighted directed graph.

To use a neural network, we need to first train it. In many cases, a dataset is first split into three separate sections: a training set, a validation test and a test set. Then, we feed the training set to the network in small batches. The output is compared to the expected output using a loss function and the error is used to update the weights of the network using backpropagation, which involves the process of calculating and propagating gradients backwards through the network, starting from the output layer towards the input layer.

Gradient refers to the vector of partial derivatives that indicates the direction and magnitude of the steepest ascent or descent in a multi-dimensional space during optimization. The methods that can be used to the gradient may differ, but optimizers like Gradient Descent, Stochastic Gradient Descent, Adagrad and ADAM are some of the most popular. This process is then repeated for a number of iterations on the entirety of the training set until the the network converges to its final weights.

The validation set is used to measure convergence after the training or to tune the network's hyperparameters before the training and the test set is used for the final evaluation of the network on unseen (through the training process) data.

2.1.1 Recurrent Neural Networks

Conventional neural networks, as introduce in section 2.1, are not well-equipped to work on sequential data, such as in tasks like translation or word prediction as they cannot convey the temporal nature of the data. For example When translating one word in a sentence the previous words in that sentence are essential in order to make the correct choice for a translated word. Recurrent Neural Networks (RNNs) are a variation of neural networks in which a neuron not only receives the current input but also the output from the previous input, which presumably carries knowledge from all the previous inputs in the sequence.

For training these networks a variation of backpropagation called backpropagation through time is used which backpropagates the error with respect to every time step by unfolding the network through time.

The major issue that arises from using gradient-based algorithms to train an RNN is called the "vanishing gradient". Since these algorithms backpropagate the gradient by multiplication, in a long sequence the gradient could become so small as to be negligible as it travels back through previous time steps. Therefore the network is unable to maintain coherency through the full length of the sequence and retain long-term dependencies in the input.

2.1.2 Long Short-Term Memory

To remedy the long-term dependency issues of RNNS, a new variation called Long Short-Term Memory (LSTM) was introduced by Hochreiter et al. in 1997 [11]. LSTMs modify a normal RNN by adding a cell state to it. The cell state carries essential information that the network needs to remember throughout the entire sequence and is updated by simple linear operations that are regulated by different gates. This helps ensure that information can be carried forward from the beginning to the end of the sequence as needed. The internal architecture of an LSTM cell is given in figure 2.2.

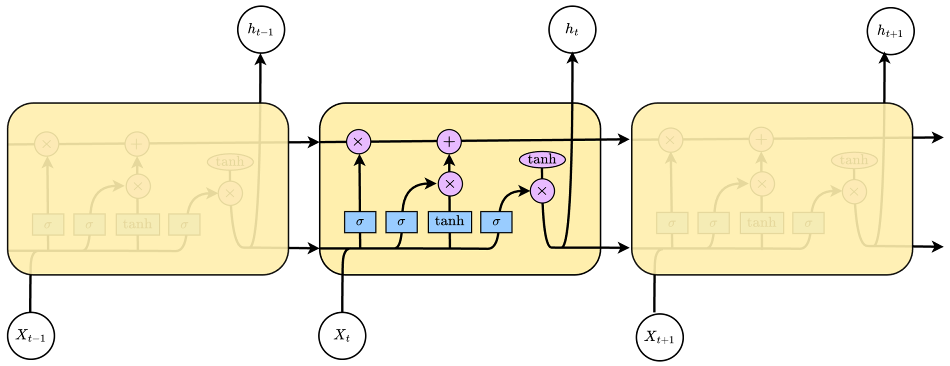


Figure 2.2: An LSTM cell

When processing input, an LSTM cell first needs to decide how much of the previous cell state it wants to keep. This is determined by the "forget gate layer" which is a Sigmoid layer that receives the input and the previous output. The old cell state is then multiplied by the output of this gate which falls between 0 (forget everything) and 1 (remember everything).

Next, an LSTM cell decides how much of the input information needs to be retained in the cell state. This part is comprised of the "input gate layer" which is another Sigmoid layer which decides which parts of the input the cell should remember and a tanh layer that creates a vector of the input values that the input gate will choose from. These two are multiplied and the result is then added to the cell state.

At last, we need to produce the output from the LSTM cell. Since the updated cell state now contains the information from the current input and the past time steps, the cell state is multiplied by another Sigmoid layer of the input that again acts as a selector of the parts of the cell state that we want as the output.

2.1.3 Autoencoders

Autoencoders are a type of neural network that derive a representation of a dataset by attempting to recreate it. These networks are extremely versatile, used in a variety of tasks such as data compression and dimensionality reduction, anomaly detection and translation; with their subsequent variants such as Variational Autoencoders (VAE) and Vector Quantized Variational Autoencoders (VQ-VAE) used as powerful generative models.

An autoencoder is made out of an encoder and a decoder network. An input X of N dimensions is fed to the encoder and its output of M dimensions is then fed to the decoder, which will then calculate an output Y of N dimensions. Since in many cases reducing the dimensionality of the data is of interest, M is usually less than N and therefore the output of the encoder is a more compact representation of X. The output of the encoder is commonly called the encoding or the latent vector.
Since the network's objective is to recreate Y from X the loss is essentially a measure of how well the network has managed to compress the input while losing minimal information.

2.1.4 Variational Autoencoders

As we discussed in the previous section 2.1.3, autoencoders map each data point to a point in the latent space. By sampling points close to these encodings of data points and feeding them to the decoder we may be able to generate new samples that resemble our original dataset. However, there is no guarantee that this will work because decoding a point in the latent space might not lead to a valid point in the sample space, as the latent space is not regularized. This greatly diminishes the generative power of the autoencoder. This however can be solved if we try to impose a regularization constraint on the network. This idea led to the development of Variational Autoencoders (VAEs)[15].

VAEs manage to be extremely powerful tools for generation of data by imposing a secondary constraint on a conventional autoencoder; the network not only wants to accurately recreate a given sample but instead of encoding the sample as a point in the latent space, it is encoded as a Gaussian distribution. A point is then sampled from that distribution and fed to the decoder.

To enforce this regularization the loss function includes both a reconstruction loss and the KL divergence of the distribution, which is a measure of how close a distribution is to a Gaussian. The trade off between these two terms are important. Concretely the total loss term of a VAE for N total data points is $\sum_{i=1}^{N} l_i$ and each l_i is defined as follows,

$$l_i(\theta, \phi) = -\mathbb{E}_{z\sim q_\theta(z|x_i)}\left[\log p_\phi(x_i|z)\right] + \mathbb{KL}(q_\theta(z|x_i)\|p(z))$$

where θ and ϕ are the weights and the biases of the encoder and the decoder models respectively. The first term is the reconstruction error under the latent variable z or expected negative log-likelihood of the data point. The second term is the KL divergence of the encoder's distribution and $p(z)$ which is a standard normal distribution.

Ultimately, their highly regularized latent space makes the VAE's able to not only generate new samples by randomly sampling from the latent space but also allows spatial manipulation of these samples or the interpolation of different samples to have meaningful results.

2.2 Transfer Learning

Transfer Learning is an area of Machine Learning that aims to improve the performance of an ML system in a certain target domain by using the knowledge extracted from a different yet similar domain called the source domain. This is especially useful when we do not have enough data in the target domain, or when we would rather decrease training time by using prior knowledge learnt.

According to C. Tan et al. [28], we can categorize deep transfer learning into 4 types:

- Instance-based: In which some instances of the source domain can be transformed using appropriate weights to approximate instances of the target domain and used to supplement the training data.
- Mapping-based: In which both instances of the source and the target domains are mapped into a new domain space, in which the instances of the two domains may be more similar.
- Adversarial-based: In which an adversarial approach is used to find a new representation for the data that is ”discriminative for the main learning task and indiscriminate between the source domain and target domain” in order to facilitate a good transfer.
- Network-based: In which a network trained on the source domain is fully or partially reused on the target domain. The basis for this method is that idea that the front layers of a DNN act as more general feature extractors, therefore they can be reused. Given a DNN pre-trained on some source data, the network is then trained on the target data while a number of the front layers (usually every layer other than the last few layers) of the network are frozen. This means that their weights remain unchanged during the transfer and the more fundamental feature extractors remain intact. We usually say the network pre-trained network was **fine-tuned** on the target data. Alternatively we can use parts of the pre-trained DNN and add new layers to the end before fine-tuning. A common example is using large image recognition models such as VGG that have been trained on large image datasets and adding new layers to finetune this kind of model to a specific image classification task.

2.3 Conceptual Expansion Monte Carlo Tree Search

Conceptual Expansion Monte Carlo Tree Search (CE-MCTS) is a model specialization approach introduced by Mahajan and Guzdial [17] in order to model a specific individual's behavior on a certain task. In their case, there is an initial model predicting behavior for other individuals but there is little to no data available for the target individual. This approach appears to work better for this task than standard transfer learning methods while requiring less data.

CE-MCTS makes use of the classic Monte Carlo Tree search (MCTS) method while using a concept called Conceptual Expansion to define the search space by combining existing knowledge. Assuming the knowledge is in the form of a neural network, they use the equation below to create a new network, expanded upon the previous one. Here, CE^W is a weight in the new model and $F = f_1, f_2, \ldots, f_n$ represent the weights of the previous model which get multiplied pairwise by a corresponding alpha value filter.

$$CE^W(F, \alpha) = \alpha_1 * f_1 + \alpha_2 * f_2 + \cdots + \alpha_n * f_n \tag{2.1}$$

Note that a particular weight f can appear an arbitrary number of times with different alphas which means this representation is unbounded.

Subsequently, the aforementioned CE is used iteratively to create and search a tree structure based on the MCTS algorithm. Each node in this tree will be a CE representation of a neural network model. Each node has a fitness score based on a combination of the network's accuracy on the primary and secondary tasks. The root node is the initial model.

As with standard MCTS, the algorithm starts by traversing the tree based on a given policy. This first step is called Selection. In our case, the algorithm uses an ϵ-greedy policy. This means at each iteration a random number in the range $[0, 1]$ is generated. If this number is greater that ϵ, the algorithm traverses to the best child node (e.g. the one with the highest fitness), otherwise it chooses to explore the space by randomly creating and adding a new node. The number of steps taken to explore the space resulting in new nodes in the tree is adjusted by a parameter called the rollout length. Adding a new node to the tree is a step called Expansion. After reaching the end of a rollout a portion of each new child's fitness is backpropagated to its parent using a discount factor. This process is repeated for a number of iterations and and each rollout starts from the root node.

To create a child node, Mahajan and Guzdial employed four functions that create new models by manipulating the f and α values of the parent node. The functions are as follows:

- Function 1: Multiply a randomly selected index of a randomly selected α by a random scalar in the range $[-2, 2]$.
- Function 2: Multiply a randomly selected α by a random scalar in the range $[-2, 2]$.
- Function 3: Swap the values of two randomly selected α and f.
- Function 4: Add a randomly selected α and f values to the CE approximation.

Ultimately, three of the best performing models are selected and their average performance is reported. The strategy for the final model selection can vary based on domain specific information.

2.4 Music Generation

Music generation using machine learning involves using models trained on music data to generate novel music. There have been different types of music generation systems in recent years that vary in their generation approach, data representation used and functionality [14].

Models are mostly designed to use one of three types of input data: waveform, spectrogram, or symbolic music data. The type of input not only affects the choice of the model but also directly affects the results. Models based on waveform music, assuming it's not digitally generated, can capture more unique characteristics of sound and play, but also may produce more undesirable audio artifacts. Spectrograms are visual representations of the waveform signal through time and enable different types of models and analysis. Symbolic music however can be more precise as notes are discrete events described by different attributes like tone, length, etc. using a format such as MIDI. This allows a more particular analysis of melodies and generation results but at the price of losing the natural sound and more subtle performance qualities. Of course, the training data may also be accompanied by different music domain information such as chord info, lyrics, etc. to enhance or alter the generated results. In this work, our focus is on symbolic music in the MIDI format, because the model we based this research on, MusicVAE is trained with symbolic data [14].

In terms of generation approaches, more algorithmic and procedural methods along with rule/grammar-based methods and Markov chains have been in use for a long time [12]. These methods rely on musical knowledge (music theory and music composition to varying degrees. In this thesis we are more interested in the effectiveness of deep learning models for this task. There are many recent papers applying deep learning models for music generation tasks with very little need for expert knowledge. However, these methods instead rely on large amounts of data.

The prevalent neural network and deep learning approaches to music generation utilize a variety of architectures. WaveNet [21] is a specifically modified convolutional neural network (CNN) that can generate raw audio waveforms. The architecture used is inspired by PixelCNN [20] which is a generative network for images. WaveNet was used primarily for text-to-speech achieving state-of-the-art results, however the researchers utilize the network for other tasks such as music generation which produces aesthetically pleasing musical fragments. However as is the case with most music generation models, WaveNet struggles with creating longer pieces with long-term structure.

Recurrent Neural Networks (RNN) and their variants such as Long Shot-Term Memory (LSTM) have been heavily utilized in music generation. Due to the sequential nature of music, these networks seem to be a natural choice for researchers, due to their capability to discover latent temporal structures in sequential data. Some of the instances of this are Deep Bach [9] and Google Magenta's Performance RNN [22]. In subsequent research from Magenta, LSTMs were combined with a Variational Autoencoder (VAE) to create MusicVAE [24], a network that generates symbolic musical sequences in the MIDI format and is more successful in creating longer pieces by utilizing a hierarchical decoding method. We use MusicVAE as the basis of this research, as it is publicly available and trained on a very large dataset of music. Open AI's Jukebox [6] uses a different type of autoencoder called Vector-Quantized Variational Autoencoder (VQ-VAE) along with Transformer networks in a yet another hierarchical architecture to generate waveform music. Jukebox can also generate vocals if provided with unaligned lyrics. A distinct approach has been the use of Generative Adversarial Networks (GAN), as in MidiNet [30] and MuseGAN [7], which are feed-forward neural networks trained adversarially to generate music that cannot be distinguished from human generated music by the network.

Despite the great progress being made in this field, there are still many challenges remaining. We briefly describe some of these challenges as it relates to high-level score generation according to [14].

- **Structure:** Current methods consistently struggle to generate music with long-term structure. Even though there have been improvements in generating lengthier pieces of music such as in [24], these pieces still do not come close to the full length of the song and they lack the patterns and the repetitive structures that are expected in a piece of music. Importantly this includes having an thematically appropriate closure to a piece rather than an abrupt one, which rarely occurs in computer-generated music.

- **Creativity:** It can be argued that Music generated by highly data-driven deep learning models does not create novel and interesting music but an interpolation of the examples in its training dataset. Improving and even determining the creativity of a ML-generated musical piece remains an obstacle in this field.

- **Style:** The style of computer-generated music in the available systems remains largely dependant on the style of their training data. Creating a universal framework for music generation that can be used to generate music of different genres is desirable but unexplored. This challenge is central to this thesis.

- **Evaluation:** Evaluating the quality of generated music is very difficult due to multiple factors. The metrics used by different researchers are very diverse, making it hard to compare outputs across different generation systems. Furthermore, there is no correlation between qualitative and quantitative metrics of evaluation, causing many discrepancies in the the implications of these metrics. Finally evaluating music generation is largely reliant on human subject studies and a broadly agreed upon automatic evaluation method of music based on computational models currently does not currently exist.

2.5 MusicVAE

MusicVAE [24] is a hierarchical recurrent variational autoencoder music generation model developed by Google Magenta. This model generates new melodies after being trained on melodies extracted from a dataset of symbolic music. For MusicVAE, the authors employ a VAE model as it has been proven effective in encoding natural data into semantically meaningful latent representations. However, there lies a challenge in the fact that this type of model is not often used with sequential data and VAEs struggle to model sequences with long-term dependencies as is seen in musical data.

A VAE that uses an autoregressive encoder and decoder model such as an LSTM, can help with the aforementioned problem but can cause "posterior collapse" [3] in the VAE. This is because the autoregressive decoder is powerful enough to effectively ignore the latent code. Therefore, the KL divergence term of the loss function can be trivially set to zero, meanwhile the model is not

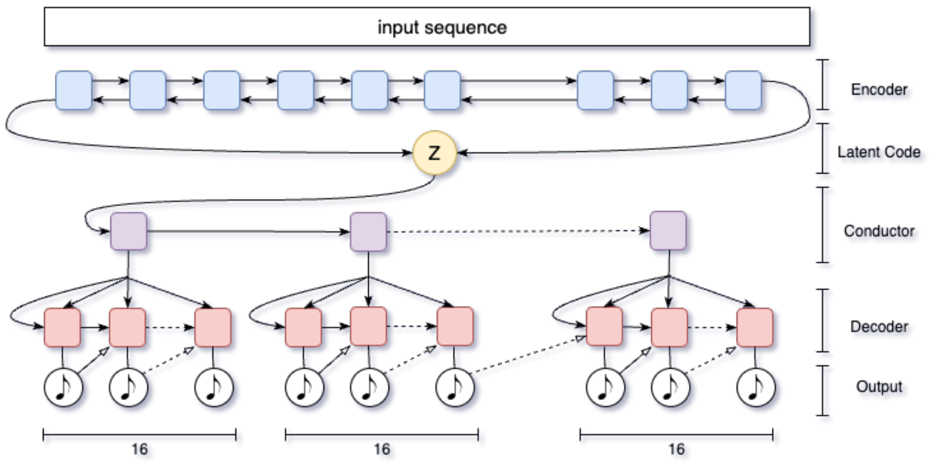


Figure 2.3: MusicVAE's architecture

functioning as a autoencoder anymore.

MusicVAE circumvents posterior collapse by limiting the effective scope of the decoder. This is implemented by adding an intermediate LSTM layer called the "conductor" that receives the latent code and generates an embedding for each bar. The main LSTM than uses this embedding to decode one bar at a time. The authors then show that while a flat model would perform well when generating 2-bar melodies, this hierarchical architecture outperforms it when generating and reconstructing 16-bar melodies.

Different experimental setups are explore in the paper, including 2-bar melody, 16-bar melody.

2.5.1 Data and Preprocessing

MusicVAE works with symbolic music in the MIDI format. The data used to train MusicVAE was collected from the web and is comprised of about 1.5 million unique publicly available MIDI files.

MIDI Format

MIDI is short for Musical Instrument Digital Interface, which is a communication protocol between hardware/musical instrument and a computer system. A digital musical device such as a synthesizer that is connected to a computer via MIDI cables and a MIDI interface can record a performance as a sequence of events through the MIDI format. The main events in Standard MIDI are KeyON, KeyOff and they correspond to the note activation and release on a virtual keyboard. Control events are also included to capture the changes in the instrument settings. Each key event has attributes like pitch (0-127; 60=middle C), and velocity (0-127). Each event belongs to one of 16 channels. Each channel can represent a different instrument allowing MIDI files to transcribe polyphonic music.

Preprocessing

At the preprocessing stage, files with a non-4/4 time signature are discarded. Then the notes in each file are quantized to sixteen 16th notes per bar, by taking the tempo into account. The files in the dataset may be polyphonic (may contain notes being played at the same time) but a melody is a monophonic (without overlapping notes) string of notes. Thus, a sliding window on *n*-bars (n is either 2 or 16 depending on the setup) is used to extract all unique monophonic 2-bar sequences that contain a at most one bar of consecutive rests.

2.5.2 Training

MusicVAE is trained using the Adam optimizer, with a learning rate annealed from 10^{-3} to 10^{-5} with exponential decay rate of 0.9999. The batch size is 512 and the training runs for 50k gradient updates with the 2-bar setup and 100k with the 16-bar setup. The loss is cross entropy against the ground-truth output with scheduled sampling [1] (This technique randomly replaces some tokens with model predictions during training to lessen exposure bias) for the 2-bar model and teacher forcing (next step prediction) for the 16 bar model.

2.6 Melody Extraction

A melody is a succession of pitches that forms a tune. Melody is often the most recognizable and memorable pattern of a song. Most musical pieces often contain polyphony, meaning that more than one pitch can be heard at a time. Melody extraction seeks to extricate the underlying monophonic melody from the other instrumentation in a song. Melody extraction is considered a difficult music information retrieval task [13] The bulk of the research on this topic works with music as a signal. For symbolic data, the most popular method uses a naive approach called the “skyline algorithm” [29]. This goes through a polyphonic sequence and at each time step picks the note with the highest pitch. It also truncates a current high note if it’s sustained long enough to overlap with an even higher note. This algorithm is predicated on the assumption that the higher notes have a greater possibility of belonging to the main melodic line. There are variations to this algorithm but they are not shown to be particularly superior in their performance. [13]

Chapter 3

Experimental Setup

In this chapter, we cover the setup and results of our experiments into applying transfer learning to music generation models. We start with an overview of the task at hand and our objectives (3.1). In 3.2, we briefly re-introduce MusicVAE, the model that is the basis for our experiments and research. We provide details on the data used to train MusicVAE, its preprocessing and training. Next in 3.3, we expand on our analysis into how MusicVAE works with different genres or styles of music. We then introduce a new dataset that we used for our experiments based on the results of that analysis. The transfer learning approaches used in this research are explained in 3.5, followed by our evaluation approach and results. In 3.10, we discuss our observations and qualitative assessments of the resulting generations. 3.11 discusses the effects of melody extraction on this model. Lastly, we end this chapter with our takeaways.

3.1 Task Overview

A deep generative model trained on a dataset of musical data is able to generate new musical sequences. However the type of music used for the training stage is a significant determinant of the type music the model will generate. Throughout this work, we examine the ability of a large DNN music generation system to produce certain out of distribution (OOD) genres of music. With this in mind, we chose Google Magenta's MusicVAE as the model on which to base our study. Next, we analyzed its performance on different genres and in order to identify OOD genres. Ultimately, by surveying a number of different transfer learning methods, we attempted to improve the systems performance on an OOD genre with minimal new data from that genre.

3.2 MusicVAE

MusicVAE [24] is a VAE-based music generation model by Google Magenta. We used this model as the basis of our experiments as it has been trained on an enormous dataset of 1.5 million unique MIDI files and the resulting weights are publicly available. This model has two modes: 2-bar and 16-bar generation. For the purposes of this research, we used the 2-bar configuration due to training speed, and lower requirements in terms of data and computational resources. For this setting, MusicVAE consists of a bidirectional LSTM encoder and a Categorical LSTM decoder.

3.3 Genre Analysis

MusicVAE as a music generation model boasts a very impressive performance. Specifically, in the 2-bar melody setup it achieves 95.1% over its test dataset. As mentioned in 1, we wanted to know how this performance varies across different types/styles of music. Our hypothesis was that MusicVAE would do poorly for OOD music, which relates to our first research question from the introduction.

To examine this question, we analyzed the model's performance on four experimental datasets of 10 songs each. Two factors were taken into account when selecting the MIDI files. All files are fan-made recreations of songs in MIDI and made publicly available on the web.

First we tried to select music that was created after the publication of the MusicVAE paper [24]. This ensures they were not used in training the model. Since the main dataset is not publicly available. We have no other way of determining this. Second, these datasets should each represent genres of music that we feel sound decidedly different from a melodic standpoint.

Melodies can differ in many ways such as contour, range and scale and these characteristics are different across different genres [5].

Our four datasets are as follows:

- A collection of synth pop songs: These are songs from a 2021 Netflix comedy special, Inside by Bo Burnham, and musically fall into the synth pop category. This dataset serves as a comparison point to the others, since we expect the model to perform well on this genre.

- A collection of Iranian songs: Having arisen from a region with a long standing history of composing music with independent roots from Western music, Iranian melodies are distinctly different in structure. The more traditional pieces of Iranian music adhere to a unique musical system, which is quite different from modern Western music theory. These songs were collected from Farsi-speaking internet spaces, therefore we can assume they were not part of MusicVAE's training data.

- A collection of video game music: This collection consists of NES (Nintendo Entertainment System) or NES-like video game music. This type of music has to fit certain criteria. It has limited polyphony as only 3 simultaneous notes can be played on the NES. It is designed to loop seamlessly so it can be repeated indefinitely, therefore it does not have arranged beginning and end sections as musical pieces typically do.

- A collection of horror movie scores: Horror movie soundtracks are a great source of highly diverse music. These are designed to build suspense and create a sense of foreboding. Musically this genre frequently use dissonant notes or chords, atonality (not having a clear scale), sudden changes of tempo, and so on to induce a sense of eeriness and dread. The main challenge here is that there are very few midi sources in this category and the ones available are mostly from very famous scores. Therefore, we cannot be certain that pieces like these were not in the original MusicVAE dataset.

Dataset	Accuracy(%)
Synth pop	95.83
Iranian folk	43.75
Video game	84.38
Horror score	87.92

Table 3.1: MusicVAE accuracy on 4 datasets of different genres

We include the results in 3.1. In these experiments, we fed melody sequences extracted from the songs into the model with pre-trained weights and reported the accuracy of the reconstruction of those exact sequences. As we expected, the model performed best on the first dataset both quantitatively and qualitatively. We observed that the model was able to reconstruct the melody samples from the synth pop dataset almost perfectly. The 95.83% accuracy is in line with what was reported for the test accuracy on the original MusicVAE dataset. On the other hand, when it came to the other three datasets, the model often would only produce the very first note of a melodic sequence followed by a lengthy silence. Incorrect notes were also a frequent observation. These shortcomings were very pronounced with the Iranian music dataset. Predictably, the accuracy is noticeably lower for the other three datasets, with the Iranian dataset standing at a mere 43%, a major drop in performance compared to the rest.

Although these datasets are too small to fully represent these genres, the results signaled that there might be merit in examining them further. We specifically focused on the Iranian folk music dataset and posed a question: Is it possible to improve MusicVAE's performance on Iranian music through transfer learning methods using a small amount of data? This relates to the second and third research questions covered in the introduction. In order to answer these questions, we gathered a new dataset of Iranian folk MIDI files and attempted to improve the model's performance via transfer learning, using this dataset.

3.4 Iranian Folk Music Dataset

As mentioned in the previous section, we collected an additional dataset of Iranian folk music. This dataset consists of 100 MIDI files from a variety of sources. Some songs were gathered from Farsi-speaking websites/forums and some were downloaded from `musescore.com` which is a free sheet music sharing website. These files contain different instruments and varying levels of polyphony.

3.5 Transfer Learning Approaches

We have employed three different transfer learning approaches to train MusicVAE to better represent and recreate our Iranian folk music dataset:

- Finetuning all layers
- Finetuning the last layer
- Conceptual expansion Monte Carlo tree search (CE-MCTS)

In the following sections, we go over each method in more detail. These methods were chosen because they allow us to use available knowledge in the form of MusicVAE pre-trained weights and adapt it to the domain of Iranian folk music by using a small dataset, namely the one discussed in the previous section.

There are many types of transfer learning, such as instance-based, mapping-based, network-based and adversarial-based (see 2.2). Instance-based learning would require re-weighting samples from the source domain to fit the target domain and mapping based learning would require learning a mapping from both the domains to a third latent space. However, we do not have access to the source domain data and even if we did, these methods require a similar amount of target and source data. For many genres of music collecting the require amount of data in itself would be very challenging. Moreover these methods do not decrease the training time nor the resources needed for training. Similarly, training adversarially would require a significantly large amount of data, which we lack [28].

Alternatively, we focus on network-based methods. These methods are predicated on the assumption that much like the human brain, the initial layers of a deep neural network act as feature extractors of increasing complexity and that these features are versatile and not limited to a source domain. This would mean by altering the last layers of a network we can adapt it to a target domain [28]. In our research, this would suggest that MusicVAE originally extracted universal features that appear in any genre of music during the initial training process.

We also employed a knowledge distillation approach called student-teacher learning [25]. In this approach we trained a MusicVAE (student network) on Iranian music, by using a combination of its loss and the loss of another MusicVAE with pre-trained weights (teacher network). Training this model proved very time consuming and we did not achieve desirable outcomes through it. Thus we do not include its results.

For comparison to the transfer learning approaches we utilize two evaluation baselines:

- Non-transfer Baseline, for which we train a randomly initialized MusicVAE on the Iranian music dataset alone.
- Zero-shot approach Baseline, which uses the pre-trained weights of MusicVAE with no additional training on the Iranian music dataset.

3.5.1 Finetuning All Layers

As explained in 2.2, finetuning is a network-based transfer learning method. This means instead of training MusicVAE from its initial (random) weights, we used the pre-trained MusicVAE weights provided for the 2-bar melody setup and then proceeded to train the full network using the Iranian music dataset. This could allow us to make use of previously learned knowledge in form of pre-trained weights and alter them through transfer learning to fit our new target domain/genre using minimal data and training time. In this baseline, we chose to not freeze any of the model's weights. However, we predicted that this would likely result in "catastrophic forgetting", meaning that by allowing the entire model to be finetuned, the model loses or forgets essential knowledge that it had previously learned. In this case, it would likely lose the "universal features" that it learned to reconstruct the original training dataset.

3.5.2 Finetuning the Last Layer

In this baseline, we again finetuned the network using the pre-trained MusicVAE weights except this time we froze all weights except the last layer. The intuition behind this is that we would like to keep the low level, "universal"feature extraction in the initial layers intact, while changing the last layer which is responsible for producing the final output based on these extracted features. We hoped that this baseline would be less susceptible to "catastrophic forgetting". This approach to finetuning is more commonplace such as in [16].

There could be an argument to the contrary, if the datasets are so drastically different that the features extracted from the initial dataset are not useful for reconstructing the target data. In our case, this question boiled down to the musical similarity of Iranian and Western music and which we cannot answer without expert knowledge. We anticipated that our experimental results could help answer this question empirically.

3.5.3 Conceptual Expansion Monte Carlo Tree Search

In this baseline, we use a model specialization approach called Conceptual Expansion Monte Carlo Tree Search (CE-MCTS) [17]. As we detailed in 2.3, this method was introduce to specialize a model trained on general data for an individual's behavior and its results surpassed other transfer learning methods on two different tasks. In this method the search space is represented using Conceptual Expansion (CE), and searched through using a Monte Carlo tree search, which results in two properties; (1) The search space is unbounded and (2) we can take larger steps while exploring the space, unlike normal finetuning. This is ideal for our task and working on discrete data, it is most likely for this method to not get stuck in local optima unlike finetuning. Also, this approach relies on a smaller amount of data on the secondary task than is usually common for transfer learning because the researchers developed it to specialize to one individual at a time. This parallels our problem, given the size of our Iranian music dataset is proportionally very small in comparison to MusicVAE's original training data.

To apply this approach to our task, we used the same four default neighbor functions. (See 2.3.) Each node in the Monte Carlo Tree Ssearch is a MusicVAE model, with the root being loaded with pre-trained weights and each subsequent child being a copy with its weights altered randomly by one of the neighbor functions. At each iteration there are 10 rollouts and at each rollout there is an $\epsilon = 0.5$ chance of choosing either exploration or exploitation. If exploration is chosen the rollout has a length of 5 nodes. The fitness used for each node is accuracy and at the end of each exploration each child node's fitness is added to its parent's accuracy by a discount factor of 0.3. Since each node in the tree is a distinct instance of a MusicVAE network, this approach uses up a lot of memory as it builds out the tree. Due to our limitations, we set the root to the best node found through exploitation and pruned the tree, keeping only the subtree of the new node. At the end, we kept the top three nodes with the highest fitness/accuracy.

3.5.4 Non-transfer Baseline

In this baseline, we trained the MusicVAE model without using the pre-trained weights. This was done without any transfer learning methods and explores the possibility of building the music generation model entirely based on our OOD dataset. However, we expected that this method would prove to be ineffective due to our significantly smaller dataset of 100 MIDI files, as opposed to the original ≈ 1.5 million files used to train MusicVAE.

3.6 Zero-shot Baseline

In this baseline, we use the publicly available pre-trained weights provided for MusicVAE [24]. This baseline is the only one that does not involve any training or finetuning on our part and shows the ability of the original MusicVAE model in reconstructing Iranian folk music.

3.7 Experiment Evaluation

Ultimately, we need to be able to concretely evaluate and interpret each approach in order to answer the question we previously posed: Will transfer learning methods help improve the model's performance on an specific OOD dataset such as the Iranian folk music dataset? To do so, we randomly split the dataset (D) into 5 folds of equal size and performed a 5-fold cross-validation. ($\bigcup_{i=0}^{5} F_i = D$, $\bigcap_{i=0}^{5} F_i = \emptyset$) This means that for each fold F_i, we trained the model on $D - F_i$ and then tested it on F_i. This is due the fact that given the small size of the dataset a single random train-test split might coincidentally be a favorable split and lead to in misleading results. The metric used in the evaluation is the accuracy of the model in correctly reconstructing input sequences, namely classifying the note being played at each step in the sequence.

It is clear that this method of evaluation ignores the ability of the model to actually generate new musical sequences, which is the main objective of such a music generation model. However, evaluating the quality of generation would require a human subject study. Moreover, given our specific dataset, the study's subject would need to be at least familiar with Iranian folk music. Give this constraint, we lacked the time and resources to conduct such a study. Nonetheless, the author is familiar with Iranian folk music and therefore can serve as an initial evaluator (see section 3.10).

3.8 Experimental Details

The training for every baseline was performed for 2000 steps. The learning rate is the same as the original MusicVAE at 10^-3 for the non-transfer baseline and $5 * 10^-4$ for the finetuning baselines. The batch size is 8. For the CE-MCTS baseline, there are 10 iterations with 10 rollout of length 5 each. The discount factor is 0.3 and the epsilon is 0.5. We keep the number of nodes in the MCTS tree limited to 100. Other parameters of MusicVAE remain unchanged from the original paper [24].

3.9 Results

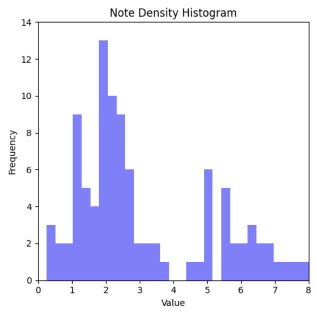
Approach	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
MusicVAE	93.75	90.62	84.37	87.50	87.50	88.75 ± 3.56
Non-transfer	68.75	68.75	68.75	68.75	68.75	68.75 ± 0
Finetune (all layers)	87.50	84.37	78.12	78.12	75.00	80.62 ± 5.13
Finetune (last layer)	96.87	90.62	93.75	100.00	100.00	96.25 ± 4.08
CE-MCTS	98.96	94.80	98.97	94.84	100.00	97.52 ± 2.49

Table 3.2: Training accuracy percentage of each baseline

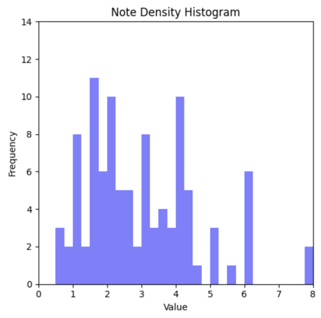
Approach	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
MusicVAE	90.62	87.50	75.00	37.50	90.62	76.24 ± 22.60
Non-transfer	37.50	37.00	37.50	6.25	53.12	34.27 ± 17.09
Finetune (all layers)	81.25	68.75	25.00	34.37	65.62	55.00 ± 24.06
Finetune (last layer)	96.87	96.87	65.62	40.62	93.75	78.75 ± 25.04
CE-MCTS	93.75	97.9	83.34	51.07	93.77	83.97 ± 19.17

Table 3.3: Test accuracy percentage of each baseline

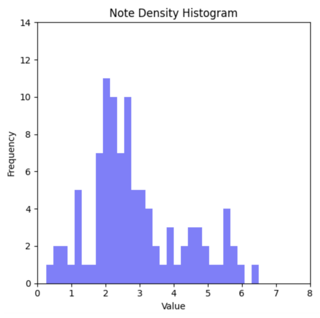
Tables 3.2 and 3.3 contain our training and test accuracy results respectively. Overall we can observe that CE-MCTS outperforms other methods in both training and test accuracy. Finetuning on the last layer is a close second. Although these two methods perform similarly during training, CE-MCTS is better at reconstructing the test data. This is especially evident in fold 4 as CE-MCTS achieves the best test accuracy on this fold out of all the approaches. Throughout every baseline, the fourth fold consistently proved challenging for the models, but upon further examination, we recognized that only fold 4 clearly has a large number of high note density outliers (shown in 3.1).



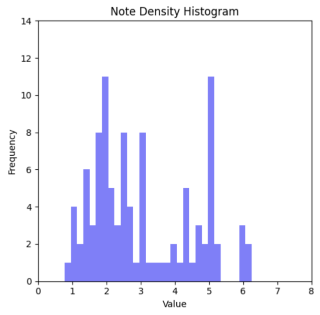
(a) Fold 1



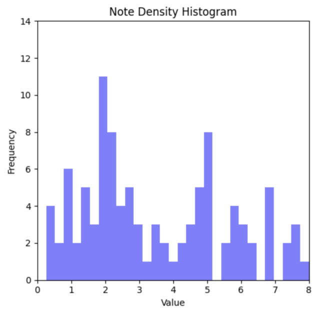
(b) Fold 2



(c) Fold 3



(d) Fold 4



(e) Fold 5

Figure 3.1: Each figure depicts the histogram for the note density of each sample in a fold. Note density is calculated by dividing the number of notes in a sample by the total time in seconds.

As we hypothesized, finetuning on all layers produces inferior results to finetuning only the last layer. In fact, it seems that the original pre-trained MusicVAE outperforms this method. This is likely due to catastrophic forgetting, which means by changing the weights of the entire network we lose valuable feature extractors that exist in MusicVAE. As for the non-transfer method, it is not surprising that the small quantity of data available is unable to effectively train the network. The initial dataset size used to train MusicVAE is 15,000 times larger than our dataset.

As previously shown, CE-MCTS outperforms both the pre-trained MusicVAE and last layer finetuning on test accuracy. Therefore we can deduce by making bigger changes including changes to the feature extractors in earlier layers, CE-MCTS is able to create better feature extractors for the target dataset and based on the results is not catastrophically forgetting useful features. This implies that the feature extractors present in the frozen layer in the finetuning baseline and the zero-shot learning baseline are not completely sufficient when working on Iranian folk music. We speculate that CE-MCTS can accomplish this because it can make more targeted changes to individual learned features using conceptual expansion. This flexibility allows for more useful features to be created that may be more helpful for training on Iranian music.

3.10 Qualitative Analysis

In this section, we provide a qualitative evaluation of each baseline's generation results. Below, we provide figures for the examples of generated melodies by each approach. These examples were chosen by an author with expertise in Iranian music because they were generally representative of the characteristics of each baseline's generations. This is obviously, highly subjective and susceptible to confirmation bias. As explained in 3.7, a study with expert participants is needed to make reliable assertions about the quality of generation.

The examples used were generated by using one of the top performing models in each category. In each corresponding figure, the x-axis is representing time in seconds, limited to 4 seconds which is the length of all 2-bar generations. The y-axis represent the pitch for the notes in the melody in the MIDI format which is between 0 to 127. Each red rectangle in the figure represents a continuous note.

Figure 3.2 represents a typical melody generated by using the pre-trained MusicVAE. The notes in this melody sound harmonious and follow a somewhat cohesive progression. They also gradually move from a higher pitch to a lower one, spanning somewhat evenly across the melodic range (distance between the highest and the lowest pitch). In the next Figure 3.3, we observe a melody generated by the no-transfer model. The differences between these two melodies are clear. The notes in 3.3 are scattered and disjunct and it sounds like a random collection of notes being played. This is consistent with the results from this baseline as the model does not seem to have learned the sample space properly. Figure 3.4, shows a melody generated by the model finetuning all layers. Melodies generated using this model are very sparse and many times end abruptly at the beginning of the interval. While they do not sound as arbitrary as the melody in 3.3, they often do not sound very cohesive or meaningful.

The last two figures 3.5 and 3.6, were generated using the model finetuning the last layer and the CE-MCTS model respectively. In the author's subjective opinion, samples generated by these two models sound more similar to, and evocative of, the type of melodies present in Iranian folk music. This is hard to qualify but here we point out a number of characteristics commonly seen in traditional and folk Iranian (Persian) music according to [8] and [26].

- Melodies has a narrow register (pitch range).
- Melodic movement is often achieve with conjunct steps.
- There is an emphasis on cadence, symmetry, and repetition of musical motifs at varying pitches.
- Rhythmic patterns are generally kept uncomplicated and rhythmic changes are infrequent.
- The tempo is often fast, with dense ornamentation. Similar to this, it is common to see repetitive and rapid use of the same note/pitch.

As it is shown in both figures, the register is more limited locally and patterns that repeat the same note are prominent in these melodies.

Based on our subjective analysis and some of the characteristics mentioned above, we predicted that metrics such as a high note density and a low average note length might be useful in confirming our qualitative views, thus we have included them in table 3.4. We can see that CE-MCTS and MusicVAE have the highest note density, with CE-MCTS having a lower standard deviation. As for average note length, finetuning for all layers has the lowest value by a proportionally big margin. The other baselines have very similar values, with CE-MCTS having the least average note length.

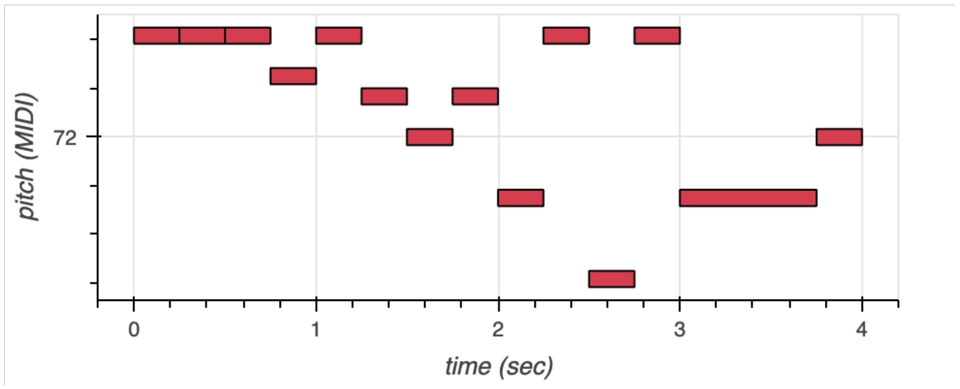


Figure 3.2: Visualization of a melody generated by the pre-trained MusicVAE model

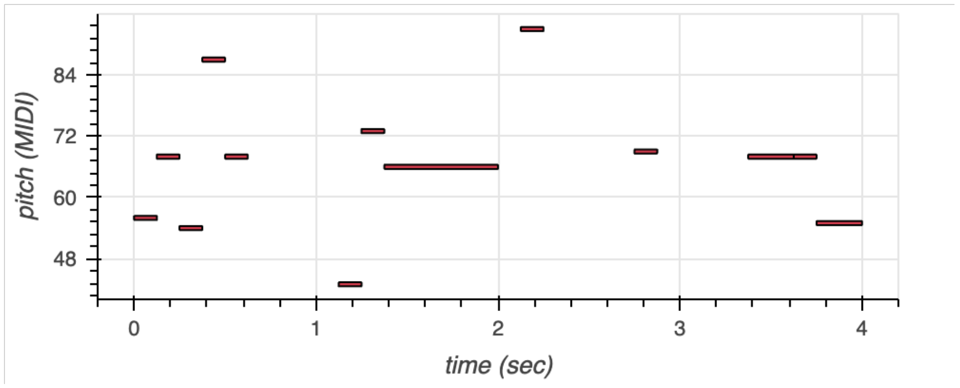


Figure 3.3: Visualization of a melody generated by the non-transfer model

3.11 Melody Extraction

In this section we discuss one important aspect of MusicVAE's pipeline that we were not able to sufficiently evaluate. As mentioned in preprocessing, melody extraction (See 2.6) is a part of MusicVAE's data pipeline. From the reconstruction viewpoint, how melody is extracted may not seem to matter as the VAE will strive to encode and recreate the same sequence it's been given. This is further bolstered by the fact that accuracy is the one quantifiable metric for the performance of MusicVAE.

However, we argue that the melody extraction algorithm used plays a crucial part in the way musical data is presented to the model and how it generates novel melodies. Thus it should not be overlooked. This is especially important when we are trying to make the generation resemble a certain target genre

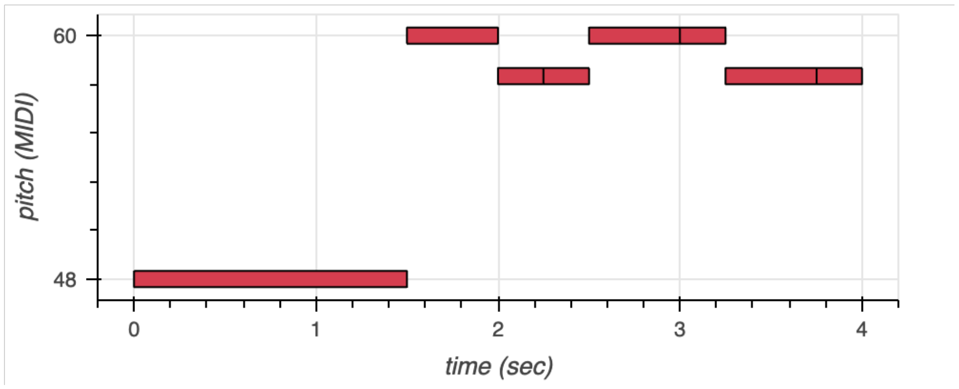


Figure 3.4: Visualization of a melody generated by the model finetuning all layers

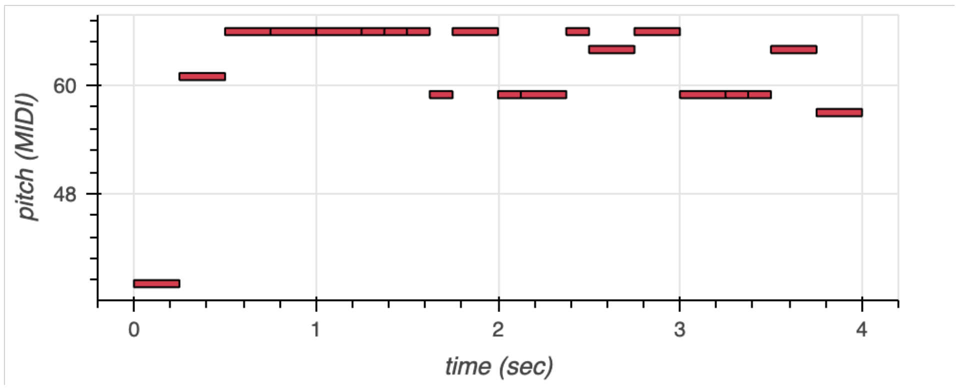


Figure 3.5: Visualization of a melody generated by the model finetuning the last layer

or style. In this case the quality of generations depend on the ability of the melody extraction algorithm to pick up the correct notes that form melodies. In this section, we further examine MusicVAE from this aspect.

MusicVAE uses the skyline algorithm on each channel of the MIDI file separately. Essentially, extracting melodies for each instrument. However, we found that extracting from the combination of all channels in one polyphonic stream resulted in melodies that better represent the piece as a whole. This is, of course, a purely subjective assessment. Interestingly, while training the model using this second version of the algorithm, we observed a drop in performance in reconstructing OOD datasets.

Ultimately, this implies that a more in depth analysis of the effects of different melody extraction strategies may benefit the quality and style specificity

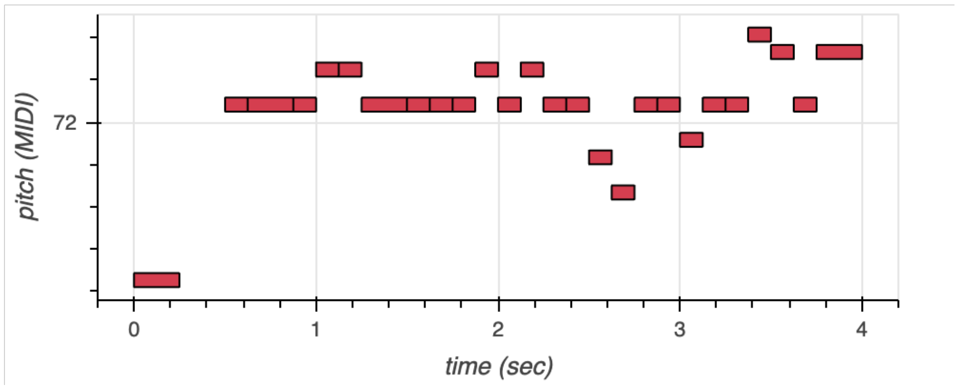


Figure 3.6: Visualization of a melody generated by the CE-MCTS model

Approach	note density	note length
MusicVAE	2.43 ± 1.47	0.48 ± 0.37
Non-transfer	2.01 ± 1.06	0.47 ± 0.29
Finetune (all layers)	1.95 ± 0.85	0.32 ± 0.15
Finetune (last layer)	2.01 ± 0.94	0.49 ± 0.37
CE-MCTS	2.44 ± 1.28	0.44 ± 0.31

Table 3.4: The note density and average note length for each baseline

of the generated music and could be explored further in the future. In more recent works, such as [23] the use of deep learning methods have been used in extracting melody from waveform data. A similar approach using symbolic music could be examined, although collecting a dataset of sufficient size for this task would be challenging and will require domain specific knowledge.

3.12 Takeaways

In this chapter we started by overviewing the music generation task and our main goal of improving an existing music generation model's performance on out-of-distribution (OOD) genres of music using minimal data and training. After summarizing the structure of the base model used for this research (MusicVAE), we investigated how it performs on a few OOD genres of music, discovering that in contrast to its performance on synth pop music, it particularly struggles with Iranian folk music. We then explored different transfer learning methods in order to improve MusicVAE's performance on a newly collected Iranian folk music dataset. Based on our results, we observed that by using CE-MCTS, a new approach based on the combination of conceptual expansion and Monte Carlo Tree Search, we are able to produce much improved reconstructions of this genre of music. Additionally, even though it is extremely difficult to conclusively opine on the quality of the model's generated music, our qualitative analysis implies subtle improvements and differences in structures of the generated data.

Chapter 4

Conclusion

This chapter concludes this thesis by going over its main implications and takeaways (4.1). 4.2 provides discussion on how this research can be expanded upon in the future. Lastly, we present our closing thoughts in 4.3.

4.1 Implications

Throughout this work we attempted to improve MusicVAE's performance on OOD genres, and through attempting and comparing different transfer learning methods, discovered that CE-MCTS is better at reconstructing samples from our dataset of Iranian folk music, particularly if there are a greater number of high note density outlier samples present. The results of this research imply two main points. First, that big music generation models are not good at representing OOD music. These models are very dependant on the diversity of their datasets which for publicly available MIDI files, are very skewed towards Western mainstream pop music. Second, taking Iranian folk music as our example of OOD music, we can improve a music generation model's ability to generate or reconstruct this type of music by transfer learning using a small dataset and that non-backpropagation method like CE-MCTS can outperform finetuning by exploring the search space in larger steps and combining existing knowledge (in the form of model weights) via Conceptual Expansion.

4.2 Future Work

Although the results obtained throughout this research are positive, we cannot make a strong claim that our conclusions hold generally across different genres, neither can we claim our conclusions hold for different generation systems. There are many aspects and questions that require further probing, in order to form more concrete conclusions.

As discussed previously 3.7, to fully evaluate performance for any genre we would need a study with human experts in that genre. This would be an essential next step in further expansion of this work.

Moreover, we believe that by doing a more rigorous hyperparameter optimization (e.g. by doing a hyperparameter sweep), we can explore and better understand the limitations of each transfer learning and how different parameter such batch size, learning rate, latent code size, and KL divergence related parameters may affect the quality of both reconstruction and generation. Also, MusicVAE offers a larger model that generates 16-bar melodies as opposed to 2-bar. This would indeed be a more computationally costly task and likely will require more data but on the other hand with a longer sequence length, it will be much easier to recognize distinct pattern and musical styles which is extremely difficult and very speculative in a 2-bar setting.

Ultimately, as we discussed in 3.11, melody extraction plays a role in the quality of generation in a music generation model. A comprehensive exploration of different extraction methods and their subsequent effects will likely be helpful in improving the model's ability to generate better representations of the target genre.

4.3 Closing Thoughts

In this work, we set out to examine the ability of large music generation models to generate music from out of distribution genres. Our endeavors were focused on answering a number of questions. (1) We wished to know how different genres can affect the reconstructive performance of music generation models.

Upon evaluating on small datasets of 4 different genres, using MusicVAE as our model, we observed that these models perform poorly on Horror movie scores, Video Game music and Iranian folk music. In contrast, the model performed well on Synth-pop music. (2) Next, we sought to determine if it is possible to specialize such a model to a specific OOD dataset. We collected an additional dataset of 100 Iranian folk music songs and utilized this data to improve the model's performance via transfer learning. We explored a variety of approaches, and achieved improvements using CE-MCTS. We also wanted to investigate the effects of the melody extraction algorithm on the model's performance. Although we made preliminary explorations into melody extraction methods, we are unable to make assertions without further probing.

References

[1] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *Advances in neural information processing systems*, vol. 28, 2015.

[2] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep learning techniques for music generation–a survey," *arXiv preprint arXiv:1709.01620*, 2017.

[3] X. Chen, D. P. Kingma, T. Salimans, *et al.*, "Variational lossy autoencoder," *arXiv preprint arXiv:1611.02731*, 2016.

[4] M. Civit, J. Civit-Masot, F. Cuadrado, and M. J. Escalona, "A systematic review of artificial intelligence-based music generation: Scope, applications, and future trends," *Expert Systems with Applications*, vol. 209, p. 118 190, 2022, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.118190. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422013537.

[5] R. DeLone and G. E. Wittlich, "Melody: Linear aspects of twentieth-century music," in *Aspects of twentieth-Century music*. Prentice-Hall, 1975, pp. 270–301.

[6] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[7] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[8] H. Farhat, *The dastgah concept in Persian music*. Cambridge University Press, 2004.

[9] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: A steerable model for bach chorales generation," in *International Conference on Machine Learning*, PMLR, 2017, pp. 1362–1371.

[10] D. Herremans, C.-H. Chuan, and E. Chew, "A functional taxonomy of music generation systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–30, 2017.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] S. Holtzman, "Using generative grammars for music composition," *Computer music journal*, vol. 5, no. 1, pp. 51–64, 1981.

[13] C. Isikhan and G. Ozcan, "A survey of melody extraction techniques for music information retrieval," in *Proceedings of 4th Conference on Interdisciplinary Musicology (SIM'08), Thessaloniki, Greece*, 2008.

[14] S. Ji, J. Luo, and X. Yang, "A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions," *arXiv preprint arXiv:2011.06801*, 2020.

[15] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2022. arXiv: `1312.6114 [stat.ML]`.

[16] J. Kunze, L. Kirsch, I. Kurenkov, A. Krug, J. Johannsmeier, and S. Stober, "Transfer learning for speech recognition on a budget," *arXiv preprint arXiv:1706.00290*, 2017.

[17] A. Mahajan and M. Guzdial, "Modeling individual humans via a secondary task transfer learning method," in *Federated and Transfer Learning*, R. Razavi-Far, B. Wang, M. E. Taylor, and Q. Yang, Eds. Cham: Springer International Publishing, 2023, pp. 259–281, ISBN: 978-3-031-11748-0. DOI: `10.1007/978-3-031-11748-0_11`. [Online]. Available: `https://doi.org/10.1007/978-3-031-11748-0_11`.

[18] F. Marchetti, C. Wilson, C. Powell, E. Minisci, and A. Riccardi, "Convolutional generative adversarial network, via transfer learning, for traditional scottish music generation," in *Artificial Intelligence in Music, Sound, Art and Design: 10th International Conference, EvoMUSART 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 10*, Springer, 2021, pp. 187–202.

[19] J. McCormack *et al.*, "Grammar based music composition," *Complex systems*, vol. 96, pp. 321–336, 1996.

[20] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," *CoRR*, vol. abs/1606.05328, 2016. arXiv: `1606.05328`. [Online]. Available: `http://arxiv.org/abs/1606.05328`.

[21] A. v. d. Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[22] S. Oore, I. Simon, S. Dieleman, and D. Eck, "Learning to create piano performances," in *NIPS 2017 Workshop on Machine Learning for Creativity and Design*, 2017.

[23] K. S. Rao, P. P. Das, *et al.*, "Melody extraction from polyphonic music by deep learning approaches: A review," *arXiv preprint arXiv:2202.01078*, 2022.

[24] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International conference on machine learning*, PMLR, 2018, pp. 4364–4373.

[25] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, *Fitnets: Hints for thin deep nets*, 2015. arXiv: `1412.6550 [cs.LG]`.

[26] I. C. Society, *Iranian classical music*, `https://www.iranchamber.com/music/articles/iranian_classical_music.php`, [Accessed: April 27, 2023], n.d.

[27] J. Svegliato and S. Witty, "Deep jammer: A music generation model," *Small*, vol. 6, p. 67, 2016.

[28] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, Springer, 2018, pp. 270–279.

[29] A. L. Uitdenbogerd and J. Zobel, "Manipulation of music for melody matching," in *Proceedings of the sixth ACM international conference on Multimedia*, 1998, pp. 235–240.

[30] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.