Fusion of Musical Genres through Deep Learning: A ResNet-Based VAE Approach for Mel-Spectrogram Generation

Eric Chen
ECE176
A16420083

Abstract

The exploration of deep learning applications within audio domains has predominantly favored speech-related tasks, leading to significant advancements in speech recognition, synthesis, and natural language understanding. The primary focus of the research community has gravitated towards speech, driven by its vast array of commercial applications and immediate impacts on human-computer interaction. This disproportionate emphasis has left music generation, a domain ripe with creative and technical challenges, in a nascent stage. This study is inspired by the distinctive and captivating audio from the games developed by miHoYo and produced by their subsidiary music studio, HOYO-MiX. A notable collection of their composition is known for its exquisite fusion of traditional Chinese musical elements with modern electronic music, and this combination, which I will refer to as CN-EDM in this paper, serves as the foundational pillar of our exploration.

Disclaimer: The soundtracks utilized in this project are the property of their respective owners and copyright holders. I do not claim any ownership over the music used for model training. This project is strictly for educational and research purposes, without commercial intent or application. All music and related materials remain the intellectual property of their original creators.

1 Introduction

Traditionally, models like Recurrent Neural Networks (RNNs) and Transformers have been at the forefront of audio learning, leveraging their sequential data processing capabilities to capture the temporal dynamics of music. High-fidelity music generation models, such as OpenAI's Jukebox or Google DeepMind's WaveNet, have set benchmarks in generating music in various styles, demonstrating the potential of deep learning in creating complex audio sequences. However, these models often require substantial computational resources and extended training times, posing significant barriers for smaller projects like this one. Instead, we could approach the task from an image generation perspective.

This study adopts a Convolutional Variational Autoencoder (VAE) approach to music generation, informed by the visual analogy of mel-spectrograms in audio processing, where audio signals are transformed into visual representations that capture temporal and frequency information in a fashion similar to images. Compared to conventional spectrograms, mel specs has its frequency scale (y-axis) mapped on a logarithmic scale, which represents audio data in a form more consistent with how the human ear perceives audio frequencies.

The decision to pivot from conventional high-quality methods such as RNNs and Transformers to a convolutional VAE was driven by practical considerations of computational resources and time constraints in combination with the available. Despite the proven efficacy of these advanced models in generating nuanced and diverse musical compositions, their computational demands exceed the scope

of this project's available resources. Instead, by treating music generation as an image-processing task, the project leverages the convolutional VAE's efficiency in capturing the intricate patterns within the spectrograms of the target music style.

2 Related Work

The application of music genre fusion is uncommon in the scope of music generation, which means the lack of online repeatable benchmarks. Several related works have inspired the overall approach. The concept of high-fidelity music generation through mel spectrograms has been reviewed by Technische Hochschule Nürnberg and Hochschule für Musik Nürnberg in their recent report [1], and its feasibilities tested by Tracy Qian, Jackson Kaunismaa, and Tony Chung in their article published May 2022 [2]

Implementation methods have been adapted from two Github repositories: the first is a tensorflow MelSpecVAE by Moises Horta [3], which I referenced for the general structure of my VAE model, and the second is a VAE by Julian Stastny with a ResNet18-based encoder architecture designed for ImageNet [4].

3 Methodology

I am training a model from scratch using a custom dataset of raw audio or mp3 files that fit my target distribution. To simplify the collection process, I have obtained my songs from the music platform NetEase Cloud Music, which offers mp3 format downloads. In compliance with Fair Use and Copyright Laws, these songs will only be utilized in training for educational and research purposes.

With the audio files, they will need to be converted into spectrogram format and resized to equal dimensions. More details of the procedure are described in section 4.2.

Recall that the model's desired outcome is successfully combining musical features from distinct genres; in this context, a variational autoencoder would seem well-suited for the job, given its ability to interpolate between distributions. Specifically, the mel-spectrogram will be fed through a convolutional encoder, which down-samples the feature maps to a latent Gaussian Distribution determined by the latent mean and variance parameters. An output is a sample from the standard normal distribution and re-parameterized by the learned to mean and variance and the latent sample is up-sampled through the decoder and reshaped back into the array format for a spectrogram.

The loss function of a VAE comprises two parts: the Kullback-Leibler divergence regularization term and a reconstruction loss. Given the nature of our project, I chose to use the Mean Squared Error Loss (MSE Loss), similar to how image-based VAEs are trained.

As for the specific encoder-decoder architecture, I adopted an incremental method, starting with a basic 4-layer Convolutional Neural Network (CNN), then progressing to a more complex 6-layer VGG network, finally stopping at a ResNet18-based architecture, and thus reaching the limits of my computing capacity.

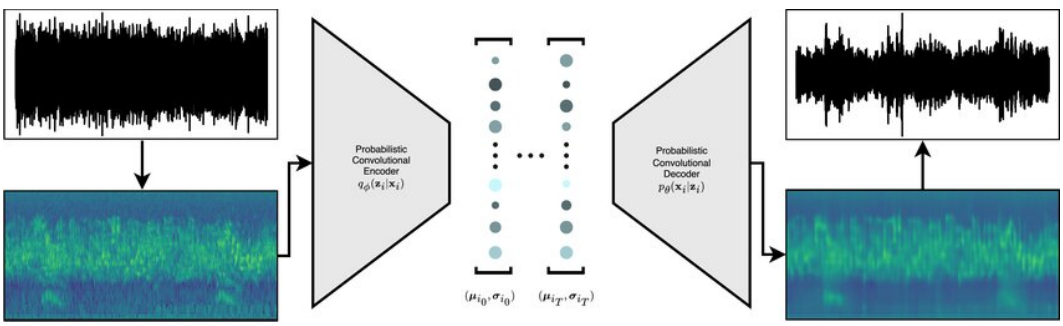

Figure 1: General Method Adapted

At the start of training, we will initialize the number of epochs to 100; the PyTorch framework is flexible enough to allow premature interruption in training; we can observe a few epochs and monitor the training procedure to terminate and adjust as needed.

The chosen optimizer is Adam without weight decay, leaving all parameters default initially. Adam's dynamic learning rate adjustment is advantageous, considering the complexity of the data.

4 Experimentation and Results

4.1 Dataset

Given the specificity of the intended task, where we combine features from classical Chinese and Western orchestral instruments and synths from electronic music, we must obtain sufficient samples to represent our desired latent space comprehensively. A breakdown of my final dataset is shown below:

Table 1: Audio Dataset

Genre	Number of Tracks
HOYO-MiX Albums	183
Western Orchestral	64
Classical Chinese	469
EDM	228
CN-EDM	284

The dataset comprises approximately 75 hours of audio data covering a range of genres, which will provide essential features to the overall model. Of the dataset, the 5 hours of data exclusively from the HOYO-MiX's relevant albums will be used to fine-tune the model pre-trained on the other data. The pre-training procedure intends to learn features of instrumentation and synths typical of CN-EDM fusion. Orchestral music represents only a minor proportion of the dataset because we require their instrumental features without having a substantial influence on the melodic part of the composition.

4.2 Data Preprocessing

As previously established, our input data will be in the form of mel-spectrograms. Figure 1 depicts a 15-second sample from my training dataset. In practice, such data is represented as a [1,m,n] array where m represents the number of frequency bins and n represents the timesteps influenced by the hop length and sampling rate when converting the original waveform. Naturally, the higher each value, the better the resolution. Each 'pixel' color within the mel-spectrogram is a decibel representation of the wave amplitude at the given timestep and frequency bin.

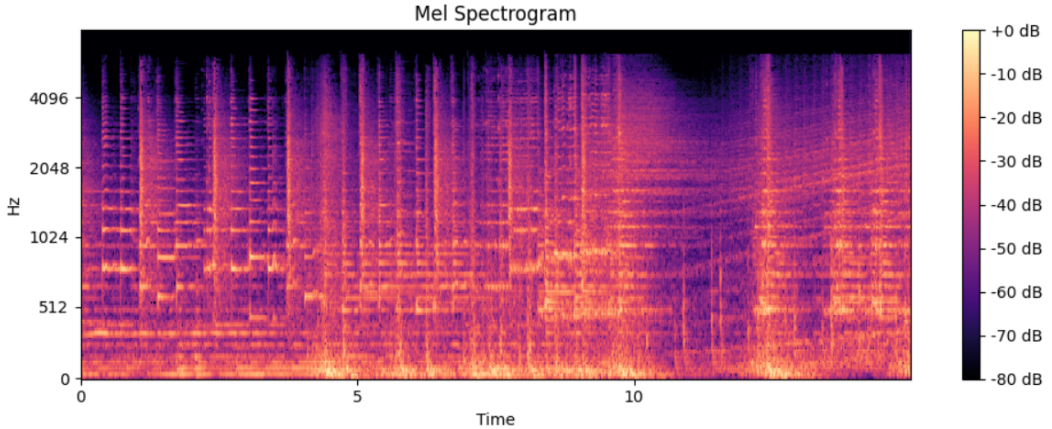

Figure 2: A mel-spectrogram of 15 second music clip

It should be highlighted that our dataset of raw audio contains tracks of different sampling rates and duration, which must be addressed since a VAE model only accepts inputs of equal dimensions. To resolve this issue, I used the Librosa Python library for audio processing to resample each audio file at a consistent rate of 44.1kHz and then split the data into 15-second segments with a 5-second overlap between adjacent samples in the same song. This procedure matches and also reduces the dimensionality of the input matrices. Having a 5-second overlap is a form of data augmentation that allows more samples to be trimmed from the limited audio and has a regularization effect during training.

4.3 Issues and Solutions

A number of issues occurred during the training iterations, the most significant of which was the exploding gradient problem, which led to NaN tensors being reconstructed and, therefore, interfering with the loss calculations. It turns out that even the default learning rate of 1e-3 is still too high for the dataset, resulting in significant overshooting. Reducing the rate to 1e-5 has effectively resolved this problem. In addition, I added small constants to the log and division calculations in the model to prevent similar issues in the future.

I also encountered bottlenecking during training when the loss was still high but converging very slowly. A potential solution is implementing a step decay in learning rate with a PyTorch scheduler. Once added, the model improved again, though it soon reached another bottleneck, suggesting that the issue is probably related to the model itself rather than optimization strategies.

4.4 Result Visualizations

Figures 3 and 4 below depict the generated mel-spectrograms randomly sampled from the latent distribution: We can observe that the samples are beginning to display features of a real music spectrogram, including periodic vacancies in specific frequency ranges and a higher proportion of lower frequencies. However, these samples are still far from the structured layout of the target.

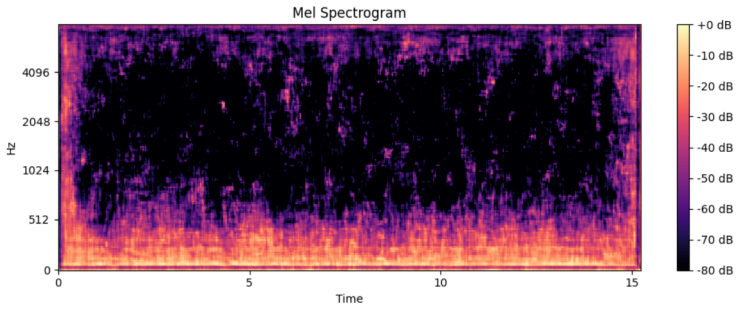

Figure 3: Generated Sample 1

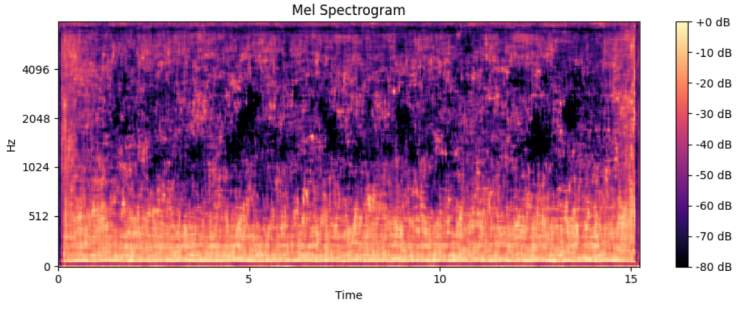

Figure 4: Generated Sample 2

5 Concluding Remarks

Over two weeks of periodic training sessions, the model has yet to converge appropriately despite efforts in implementing optimization strategies. I believe that we can appropriately conclude that this model is insufficient to reach the result we are expecting. It is likely that the ResNet18-encoder cannot identify all the necessary features. The ideal model should at least be able to enforce rhythmic integrity, as shown by distinct vertical bands in the spectrogram. We also need observable bright bands in the mid-range frequencies representing the dominant notes in the song; this is necessary to obtain any audible melody.

While there are existing cases of successful training with a model of similar capacity, the differences in datasets and resources must be considered, in particular, My experiment has been heavily restricted by the available memory which resulted in a relatively small latent space unable to fully capture the nuanced details of the mel-spectrograms. Alternatively, we could experiment with different convolution strategies than the square kernels used in ResNet18 to better account for the time dependencies of the waveform.

For further investigation, I could utilize cloud computing resources with more powerful hardware such as Google Colaboratory and see if I can obtain better results.

6 References

[1] A Survey of Music Generation in the Context of Interaction, arXiv, submitted by [Nürnberg et al], 23 Feb 2024, arXiv:2205.07319.

[2] cMelGAN: An Efficient Conditional Generative Model Based on Mel Spectrograms, arXiv, submitted by [Tracy Qian, Jackson Kaunismaa, Tony Chung], 5 May 2022, arXiv:2205.07319.

[3] Horta, Moises. MelSpecVAE. GitHub https://github.com/moiseshorta/MelSpecVAE.git.

[4] Stastny, Julian. VAE-ResNet18-PyTorch. https://github.com/julianstastny/VAE-ResNet18-PyTorch.git.