MELodyGAN: Exploring the Generation of Mel Spectrograms Using Generative Adversarial Networks

Victor Lee
University of Massachusetts Amherst
victorlee@umass.edu

Eric Wu
University of Massachusetts Amherst
ewu@umass.edu

Isaiah Baker
University of Massachusetts Amherst
ibaker@umass.edu

1. Introduction

Recently, there has been a lot of development in text-based image generators. Complex images and artistic creations are now able to be created by an average user simply from inputting a few lines of text into an online neural network. However, not as much progress has been made in the context of prompt-based music generation. Our inspiration lies in the success of these text-based image generators, and our motivation lies in the hope that we will be able to generate music in a similar fashion. We recognize, however, that fully creating a text-based music generator would require more time and experience then we currently have available. Thus in pursuit of generating music from text, we sought to implement a generative model that will allow users to create music from a list of specific genres.

Our classwork predominantly focused on machine learning in the context of image generation. In order to explore music generation in a similar way, we utilized the Mel Spectrogram image encoding of all audio files for training. Using these image encodings, we planned to implement Convolution Neural Networks (CNNs) on the images with the same techniques taught to us in class, and we specifically limited ourselves to the exploration of Generative Adversarial Networks (GANs).

GANs have what some might say is a bad reputation. Historically, they are a notoriously difficult to train right. A GAN is comprised of two main components: a generator network and a discriminator network. The goal of the generator network is to create inputs that will ”fool” the discriminator network into classifying them as being from the real dataset, while the goal of the discriminator is to minimize the possibility of classifying samples from the generator as being real. These two neural networks must not only be optimized themselves, but be optimized to work well with each other. The choice of working with GANs as our generative model for Mel Spectrograms is that GANs have the potential for producing much clearer and more realistic-looking images compared to VAEs, which is important for generating Mel Spectrograms of appropriate clarity so that they can be properly reconverted back to audio without sounding corrupted.

Our group had little-to-no prior experience in music generation, and realistic music generation proved to be a significantly more complex task than realistic image generation.

Our first approach was to train a GAN solely on the images of Mel Spectrograms provided within the dataset. We hoped that the musical structure and patterns represented within these Mel Spectrograms could be captured and recreated by our GAN. We implemented a classic Convolutional Neural Network (CNN) structure as the backbone for our Generator and Discriminator, which normally performed well in the task of image classification. However, this set up proved to perform quite poorly with regard to musical creation. We speculated that this may be due to a limitation on the information being given to the GAN. Just training on Mel Spectrograms alone perhaps misses key information needed for a more realistic representation of the underlying musical genre. After realizing this potential limitation, we set out again to give our GAN more room to work with.

Our second approach involved using the raw audio provided by the dataset instead of the images of Mel Spectrograms. We created the Mel Spectrograms ourselves using PyTorch’s torchaudio library. Not only did this provide a richer representation of audio, it also made it easier to convert Mel Spectrograms back into audio files. We then trained our GAN on the Mel Spectrograms converted from raw audio, and were able to obtain results that resembled certain aspects of the underlying musical genre. Our second approach yielded better results than the first approach, but generated Mel Spectrograms still weren’t too representative of a given genre of music.

Our third approach involved modifying the underlying neural network model, and importantly changing the last layer of the generator network from a Tanh layer to a ReLU. This improved generated image respresentations to be significantly more visually appealing, but audio conversions of these images still left much to be desired.

Although our GAN was able to pick out certain characteristics of the underlying musical genre, we will elaborate on some shortcomings of our GAN training, and expand upon future ways to approach similar styles of training.

2. Problem Statement

2.1. Dataset

We utilized the GTZAN Dataset. This dataset contains 1,000 audio files evenly distributed amongst 10 genres of music stored in .wav audio files, each 30 seconds long, and .png images of the audio converted into Mel Spectrograms. Compared to other potential datasets (MusicCaps and AudioSet for example), the GTZAN Dataset has an explicit number of genres (10), and sufficient training samples for each genre (100), which makes it easier to work with for training our GAN. Additionally, we split each file into three different pieces, and thus had 300 10-second training samples to work with as data for each genre.

In our first approach to training our GAN, we utilized the direct .png images for training. On our second and third approach, we utilized .wav audio that was directly converted to Mel Spectrograms for training.

2.2. Expected Results

For our first approach, we expected our untrained GAN to have the ability to learn from the GTZAN Dataset Mel Spectograms. It should be able to learn patterns within the images and represent a finished model that is able to generate an original Mel Spectrogram *image* for a piece of music based on a given genre from user input.

For our second and third approaches, we expected our untrained GAN to have the ability to learn the GTZAN Dataset .wav audio encodings. These encodings are the direct representation of Mel Spectrograms, rather than an image representation of a Mel Spectrogram (idea expanded upon further in Technical Approach). The model should be able to learn patterns within the Mel Spectrogram, and it should ultimately be capable of generating an original Mel Spectrogram for a piece of music based on a given genre from user input.

Once spectrograms have been generated, they will then be converted back into a 10-second audio file that can be played.

2.3. Evaluation

The GAN is comprised of two primary components: the generator and the discriminator (idea expanded upon further in Technical Approach).

Note that our primary concern is the performance of our generator. Our evaluation of the generator will lie in its ability to output a spectrogram, that when converted to a 10-second audio file, can retain certain characteristics of the user-specified genre. We are not necessarily evaluating how "good" of a file the music will sound in terms of musical composition, but rather we are primarily evaluating what "essence" and features of a given musical genre is extracted and produced within the generated Mel Spectrogram and corresponding audio conversion.

Our evaluation of the discriminator will lie in how successfully it can distinguish between a "real" image of spectrogram-encoded audio (GTZAN data) and a generated image (produced from the generator) utilizing Least-Squared GAN Loss (idea expanded upon further in Technical Approach).

3. Technical Approach

3.1. Vision

The ultimate goal of our project was to create a generative model for spectrogram images that can represent a given genre of music. To this end, we implemented a deep convolutional generative adversarial network (DCGAN) that can find a representation of a specified genre. Previous work with DCGANs has demonstrated that these types of models can be effective at extracting patterns and features within training images for realistic image generation [Radford, Metz, Chintala]. We trained the generator and discriminator components of the DCGAN until the generated Mel Spectrograms and associated audio converged to a genre representation. Model parameters were updated according to a gradient-descent based approach using the least squares loss function (LS-loss) (expanded upon later in Loss Function and Learning).

3.2. Approach

We chose to use PyTorch as our framework for our implementations of our GAN architecture, due to our experience with the API from the course along with its easy to use optimizations of loss functions and back propagation. With familiarity of CNNs and implementing architecture for GANs trained with images from coursework, our first approach was using 4 x 288 x 432 images of Mel Spectrograms provided by the GZTAN dataset and training a GAN to generate realistic looking images of Mel Spectrograms. Images were converted into NumPy arrays through the use of PIL (Python Image Library) before being converted into Tensors. We believed that the images of Mel Spectrograms captured musical structure and motifs that could be represented and reproduced by our GAN. This quickly proved to be a naive direction for music generation.

As a representation of audio, a Mel spectrogram groups similar audio frequencies, which are modeled using Fast Fourier Transforms, into clustered bins in the Mel Scale, which is an alternative measure of distance between frequencies based on audibility to the human ear rather than the standard unit of frequencies based on wavelengths, Hz. The Mel scale is then plotted on the y-axis, with the x-axis representing frames (which is proportional to time $*$ sampling rate), thus encoding information of audio frequencies that would be perceivable by humans over a period of time.

Unfortunately, as a raw image of pixels, a Mel Spectrogram no longer captures the exact values of frequencies with bins plotted over set frames in time, and without a proper correlation of color to frequency value, an image of a Mel Spectrogram cannot be reconverted back into audio. Realistic generated images of Mel Spectrograms with our original approach would ultimately remain just as images.

As such, our second approach involved using the raw audio provided by the same dataset instead of the images of Mel Spectrograms. We created the Mel Spectrograms ourselves using PyTorch's torchaudio library to recreate the sound information from the Mel Spectrograms that would be lost when simply presenting their plots as images. The added benefit of this approach was that we could also use the torchaudio API to convert Mel Spectrograms back into audio files to be played, as well as load representations of audio directly as Tensors.

3.3. Audio Preprocessing

30 second audio clips could first be converted into waveform representations, which with a sampling rate of 22050 (from each audio clips metadata from wav format) resulted in $30 * 22050 = 661500$ single dimensional Tensors (single dimension over one audio channel). Waveforms can then be converted into a lower capacity representation in the form of Mel Spectrograms, using 2048 Fast Fourier Transforms to model the waveforms, 128 bins to group the frequencies produced using Fast Fourier Transforms, and a hop length of 512, to reduce the points in the wave length from 661500 to $661500//512 = 1293$, for $(1, 128, 1293)$ dimensional representation of the Mel Spectrogram (1 dimension of channels, 128 bins, and 1293 frames). To reduce the scale of the Mel Spectrogram, we ultimately ended up slicing each 30 second audio clip from the dataset into three 10-second audio clips, and used those as our training samples, which reduces the frame dimension of the Mel Spectrograms down to $1293/3 = 431$, for a $(1, 128, 431)$ dimension Tensor for the Mel Spectrogram of a single 10 second audio-clip from the dataset.

The full dataset of 10 unique genres of music, each with 100 30-second audio clips, would then contain 300 10-second audio clips per class, resulting in a $(10, 300, 1, 128, 431)$ dimension Tensor storing Mel Spectrogram information for all audio-clips from the GZTAN Dataset. Additionally, every 300 samples from each genre was then split into batch sizes of 10 for use during training the GAN, resulting in a final Tensor of dimension $(10, 30, 10, 1, 128, 431)$ holding all Mel Spectrogram information for each audio clip from every genre.

3.4. Generator

The generator component of our GAN architecture is a fully convolutional neural network inspired by the DCGAN model. Our Generator uses the same number of convolution layer blocks as the DCGAN model, where the first three transposed convolution blocks follow the same structure of (Transposed Convolution, Batchnorm, non-linear function), with adjusted kernel (filter) sizes and strides to allow for the generation of a $(1, 128, 431)$ image representing a Mel Spectrogram encoding audio information, and hidden channel dimension sizes of $[64, 32, 16]$. The final convolution block is not followed by Batchnorm, but still includes a nonlinear function unlike the DCGAN Generator, with this design choice being intentional as the values in a Mel Spectrogram are only positive values (representing the presence and strength of a frequency). As illustrated later, changing the final nonlinear layer can result in drastic changes in generated outputs.

To enable the concept of fully convolutional layers, the random noise vector z of some dimension d, (chosen to be 100 in our implementation), which is generated from a normal distribution, is transposed so that the d random values represent the values across d channels of a 1 x 1 image, as done so in the DCGAN model. This d x 1 x 1 random noise image is then passed through the layers in our Generator model to result in a 1 x 128 x 431 image representing a Mel Spectrogram, which can then be passed as training data for the Discriminator or converted back to waveform and displayed as audio.

Here are the generators we explored in our second and third approach:

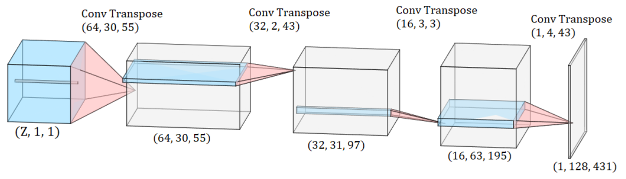

Figure 1. Second Approach architecture for the generator. Utilizes Transposed Convolution operations to upscale the initial noise vector to an image of shape 128×431 with 1 channel.

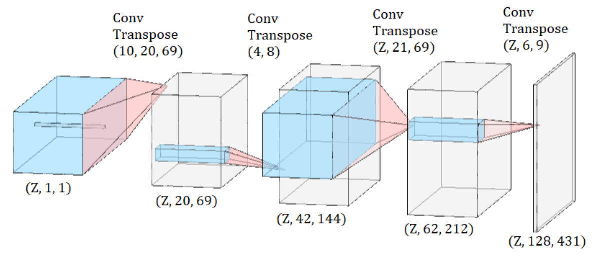

Figure 2. Third Approach architecture for the generator (Final Model). 128×431 with 1 channel.

3.5. Discriminator

Similar to the Generator, the Discriminator component of our GAN Architecture is also a fully convolutional neural network using the similar design choices to the DCGAN Discriminator. The first three convolutional blocks follow the same structure of (Convolution, Batchnorm, non-linear function) and hidden channel dimension sizes of $[64, 32, 16]$. The final convolution block is not followed by any non-linear functions, and convolves to a final $(1, 1, 1)$ dimension image representing the prediction score on the input Mel Spectrogram as real or fake. All convolutional layers have adjusted kernel (filter) sizes and strides compared to the DCGAN model to allow for the transformation of a $(1, 128, 431)$ dimension Mel Spectrogram to a $(1, 1, 1)$ prediction value.

Here are the discriminators we explored in our second and third approach:

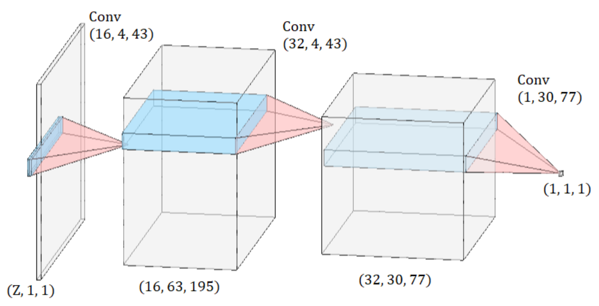

Figure 3. Second Approach architecture for the discriminator.

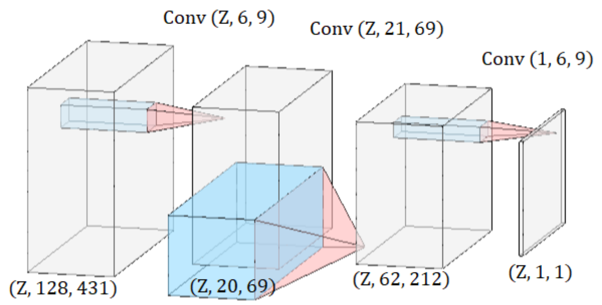

Figure 4. Third Approach architecture for the discriminator (Final Model)

3.6. Loss Function and Learning

The Generator and Discriminator components are trained in tandem in order to provide feedback and improve both models. This prevents one half of the GAN from training too fast and becoming too dominant, which will prevent the other half of the model from improving. The goal when optimizing the Generator model is for it to create realistic Mel Spectrograms that will cause the Discriminator to label them as being genuine, and that the converted audio from its generated Mel Spectrograms is identifiable as the genre the GAN is trained upon. On the other hand, the goal when optimizing the Discriminator model is for it to improve its ability to accurately discern between genuine Mel Spectrograms from the dataset and artificially produced Mel Spectrograms from the Generator.

To do so, each epoch, both models are trained once on separate batches of data: Batch sizes of 10 generated Mel Spectrograms are used per epoch for evaluating the Generator, and batch sizes of 10 generated Mel Spectrograms + 10 real Mel Spectrograms from the dataset are used per epoch for evaluating the Discriminator. With 30 batches per epoch, this results in the Generator and Discriminator models updating 30 times per epoch.

Both sets of data for the Generator and Discriminator models are passed through the Discriminator to return a prediction value from 0 to 1 indicating the ”realness” of the Mel Spectrogram fed through the Discriminator, with 0 corresponding to fake, and 1 corresponding to real.

The Generator model is evaluated using a loss function between the predicted score of ”realness” by the Discriminator model on the generated Mel Spectrograms compared to a ground truth of 1 indicating ”real” for the generated Mel Spectrograms, as the goal for the Generator is to produce realistic looking Mel Spectrograms that will fool the Discriminator.

On the other hand, the Discriminator is evaluated on the combined set of generated and real Mel Spectrograms evaluated on a loss function of the predicted scores of the Mel Spectrograms and ground truth labels of 1 for real Spectrograms and 0 for fakes, as the goal of Discriminator is to correctly separate fake Mel Spectrograms from the Generator from the real Mel Spectrograms provided from the dataset.

Both neural networks scored using the Least Squares loss (LS-loss) to improve stability in training by avoiding vanishing gradient problems that might occur when using loss functions such as Binary Cross Entropy (BCE loss) or sigmoid cross entropy. LS-loss function has been shown to achieve greater stability in training GANs by mitigating the effects of the vanishing gradient problem by other loss functions such as the sigmoid cross entropy (Mao, LSGAN). In that same paper, the authors report two primary benefits of using the least squares loss: the least squares loss will influence the generator to create samples closer to the decision boundary associated with a particular type of image, and it also reduces the effect of vanishing gradients by penalizing all generated samples regardless of their distance to the decision boundary (Mao, LSGAN).

Parameters for the Generator and Discriminator model are also updated alternating, once per epoch, with gradient descent using the Adam optimizer from PyTorch, with default learning rate of $1e-3$, and betas $= (0.5, 0.999)$. To avoid back propagation through the Generator parameters when updating the parameters for the Discriminator model, we call detach() on the Tensors of the fake Mel Spectrograms generated by the Generator. This ensures that parameters for the Generator are only updated through the loss computed for the Generator, and not additionally through the loss computed for the Discriminator.

3.7. Expected Output

After sufficient epochs of training the Generator and Discriminator components of the GAN, we expect to reach a point where the 10-second audio converted from the generated spectrogram images are recognizable to the given genre. The quality of the musical composition is left entirely to human evaluation.

4. Results

4.1. First Approach Results

The results were so abysmally inaccurate, that they are not worth talking about, and will not be referenced any further in the paper.

4.2. Second Approach Results

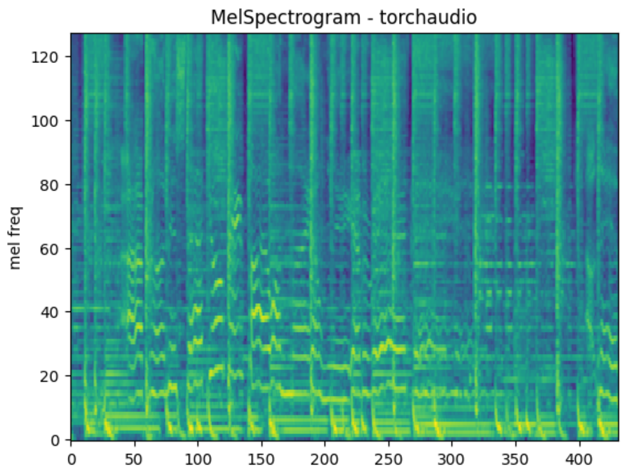

Figure 5. Torchaudio creation of Mel Spectrogram for a piece of real pop audio; .wav from the GTZAN dataset

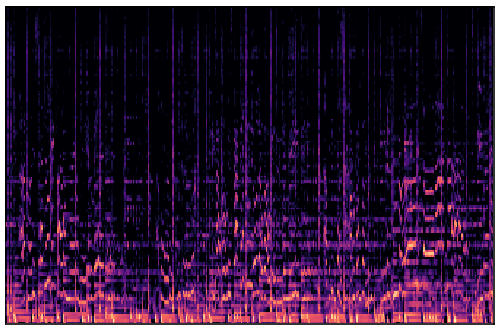

Figure 6. Corresponding Mel Spectrogram for a piece of real pop music from the GTZAN dataset; .png from the GTZAN dataset

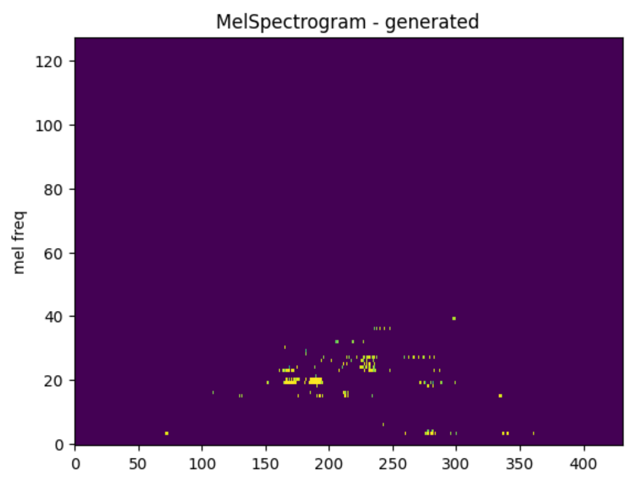

Figure 7. Mel Spectrogram generated by the generator component of our GAN for a piece of music in the pop genre.

Here, figures 5, 6, and 7 depict the spectrograms for two pieces of music within the 'pop' genre: figures 5 and 6 depict a spectrogram representation for a piece of real pop music from GTZAN, and figure 7 depicts a spectrogram for a piece of music in the same genre created by the generator. One may observe that the spectrograms in figures 6 and 7 do not closely resemble each other; more specifically, the generated piece of music appears to be more sparse in sounds, tones, and frequencies when compared to the spectrogram for a real piece of pop music.

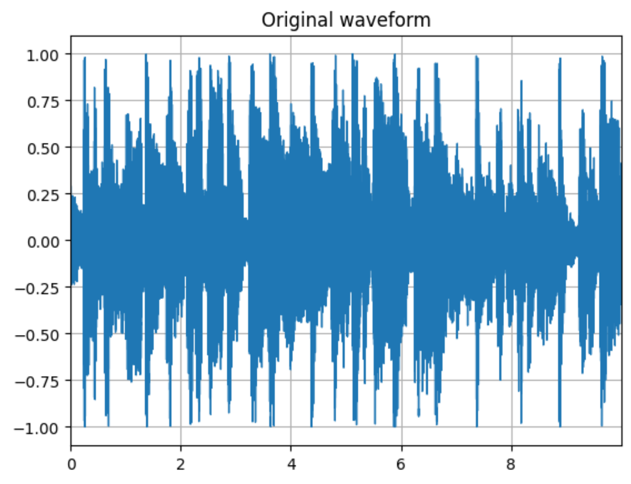

Figure 8. Wave form representation of the piece of real pop music

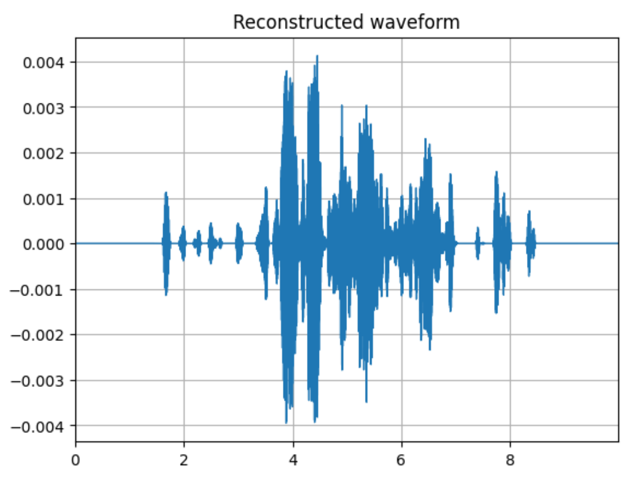

Figure 9. Wave form representation of the piece of generated pop music

In addition to comparing spectrograms, one may also compare the waveform representations of the generated and real pieces of music. For this specific example from the pop genre, one may observe from the waveforms that the genuine pieces of music tend to more uniformly distribute sound across the time interval when compared to the generated pieces of music. In addition, one may also observe that there is a broader range of frequencies for the genuine piece of music when compared to the generated piece.

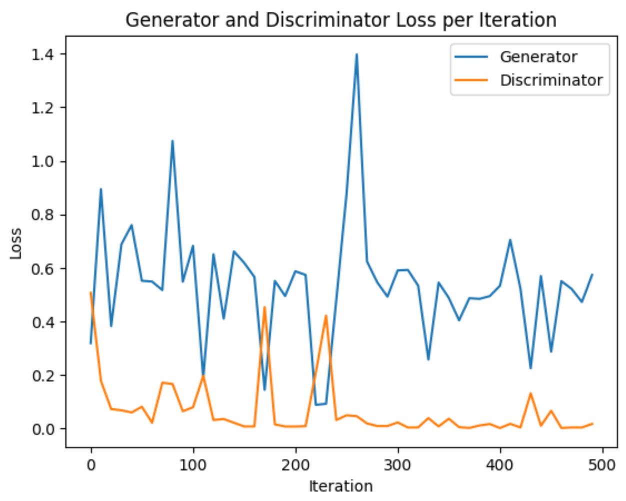

Figure 10. Comparison of generator and discriminator losses for each training iteration

Figure 5 depicts the loss per iteration between the generator and discriminator on the real and generated images. Here, one may observe that the generator loss was consistently higher than the discriminator, whereas the discriminator generally experienced lower loss when compared to the generator.

4.3. Final Model (third approach)

Our third model made a few modifications to layer kernels (as illustrated above in section Technical Approach) providing a different architecture than the model used in our second approach. It also made a change in the last layer of the generator, as implementing ReLU in place of Tanh for non-linearity seemed to significantly improve visual results.

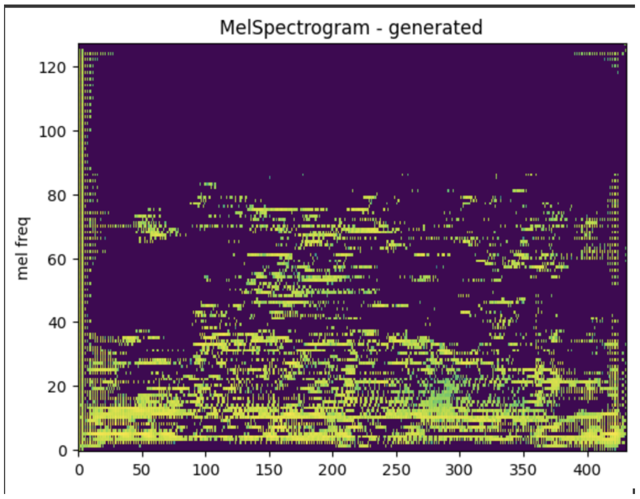

Figure 11. Spectrogram generated by the generator component of our Final GAN Model for a piece of music in the pop genre.

As you can see, the generated Spectrogram in Figure 11 relates more similarly to those seen in Figure 6 and 7. This generated piece of music appears to be much less sparse in sounds, tones, and frequencies when compared to previous model.

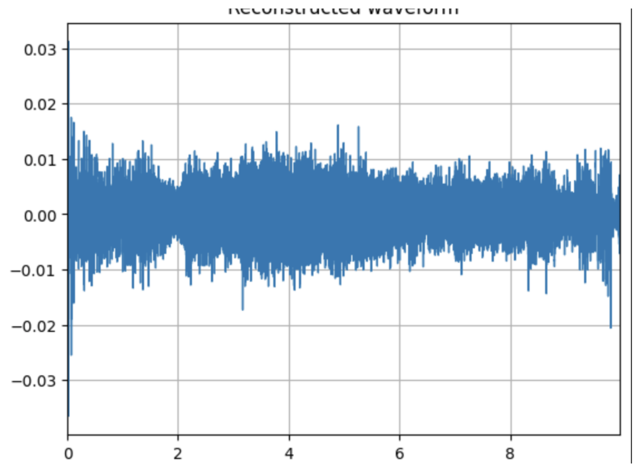

Figure 12. Wave form representation of the piece of generated pop music

The waveform representations of the generated Spectrogram resolves some of the problems seen in the previous model. The waveforms are now more evenly distributed across the time interval. However, there is still a limited range with regard to the wave length for this generated music when compared to a piece of real music.

5. Analysis / Discussion

5.1. Second Approach: GAN Performance

In our second approach, when comparing our generated spectrograms versus the spectrograms of the real pieces of music from the GTZAN dataset, one may observe that the generator's output differed vastly from the pieces of music present within the GTZAN set. As mentioned in the Results section, the samples generated by our generator often prioritized the production of sparse pieces of music with low frequencies, while the genuine pieces of music often had broader ranges of frequencies; from figures 8 and 9, one may observe that the genuine pieces of music often had normalized frequencies within the range of -1 to 1, while the normalized frequencies of the sample music were within the range -1×10^{-3} to 1×10^{3}. One may also observe that the majority of the frequencies the generator did create were often clustered in the middle section of the time interval for the piece of music, whereas in a genuine piece of music sound was more uniformly distributed within the given time interval.

The outputs of our generator are likely connected to how it was trained with the discriminator during training time. From figure 10, one will observe that the generator loss often struggled to keep up with the discriminator; more specifically, the generator loss was consistently higher and failed to converge over the 500 training iterations, whereas the discriminator appeared to possess very low loss and converged to a value close to 0. This disparity between the generator and discriminator ultimately suggests that the discriminator was able to effectively distinguish between examples of music from the GTZAN dataset and examples created by the generator, and as a result the generator was provided with very little feedback as to which types of images would be effective in "fooling" the discriminator into classifying generated spectrograms as being genuine. Furthermore, one may also infer from figures 7 and 9 that the types of music that produced spectrograms that could minimize loss were sparse pieces with low frequency notes and sound concentrated in the middle of the time interval. Additionally, it should also be noted that samples generated from the pop genre were more likely to contain elements of a melody and slightly more evenly spaced notes; this was likely due to the fact many of the samples in the GTZAN dataset labelled as pop music did not contain as many complex features and noises when compared to other genres of music in the dataset.

5.2. Third and Final Approach: GAN Performance

In our third approach, our generated spectrograms were visually a lot more similar to their real image counterparts. However, the relation between generator and discriminator performance was actually fairly similar to that of our second approach. This may suggest that although changing the model may have helped, the main difference in output performance was more so due to a change from Tanh to ReLU as a last layer of activation.

5.3. Overall Approach: Modelling Issues

In addition, the major discrepancy in loss values between the generator and discriminator could suggest that modelling music as a visual Mel spectrogram may be too reductive for the task of generating music. For example, since Mel spectrograms represent music and sounds as visual groupings of fourier transforms, a convolutional neural network may not necessarily be directly applicable to finding trends and structures within the fourier transforms visualized by the spectrograms. Modeling wavelengths directly, instead of the Mel spectrograms, may prove to have better output performance. In addition, convolutional neural networks don't necessarily consider the sequential nature of music and the temporal relationship between a series of notes and sounds, and as a result our GAN model may struggle to learn and extract features from pieces of music with varying tempos and beats per minute.

6. Conclusion

In this paper, we explored and experimented with different architectures for deep convolutional generative adversarial networks for the task of generating spectrograms for music that could then be converted back to audio. Although our model was not able to recreate spectrograms similar to the inputs in the GTZAN dataset, our model was ultimately able to produce spectrograms that demonstrated a very basic understanding of the sounds and features present within the original data.

In terms of further work, one possible improvement we could make to the model is to standardize the tempo and beats per minute of the music used to train the GAN. Since tempo and beats per minute can vary drastically between two different pieces of music even within the same genre, this likely caused our model to fail in generalizing the features for the music in each genre because critical motifs and information about a genre may be distorted as a result of differences in tempo and beats per minute. In addition, another possible improvement that could be made to the model is to utilize a different architecture that can take advantage of the sequential and temporal nature of music like a transformer model or a recurrent neural network.

7. References:

- Radford, Metz, Chintala. https://arxiv.org/abs/1611.04076
- Mao et. al. https://arxiv.org/abs/1611.04076
- Agostinelli et. al. https://arxiv.org/pdf/2301.11325.pdf
- Alexandre Défosse https://arxiv.org/abs/2306.05284

8. Video

https://drive.google.com/file/d/1tgydEp2mOqNeoKSIL6SN4FJ96-XPQIGy/view?usp=share_link