Music Generation using Generative Adversarial Networks (GAN)

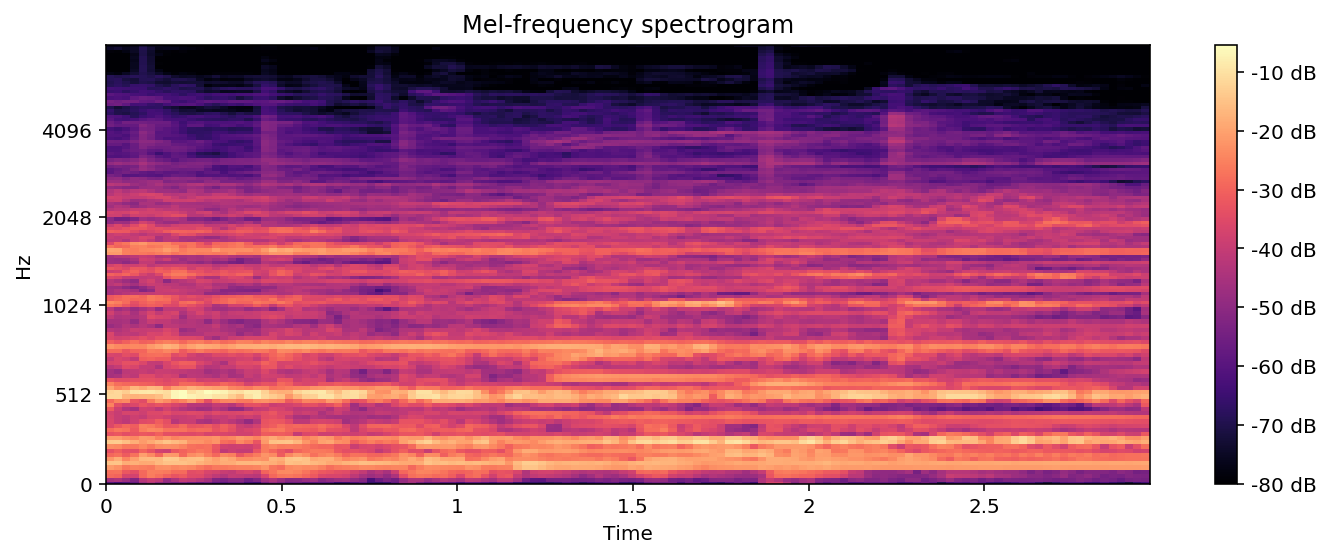
For my data project, I wanted to see how well GANs could create artificial classical music based on sound waves. There are several large classical music datasets online, and from further researching found the Maestro-V1 dataset containing 12,000 classical songs with up to 172 hours of music. However, upon spending close to 6 hours struggling to turn MIDI files into WAV files without any prior experience, I resorted to a simpler, larger dataset called the Free Music Archive, published on the Cornell Computer Science website, containing over 100,000 music MP3 tracks. I took about 8,000 music files to run on this specific problem. These are not all classical music, ranging across all genres and all types of musicians, however I felt it efficient for my analysis to understand GAN wavelength analysis and music generation.

This problem is definitely unsupervised as we are allowing the GAN to discriminate and generate artificial music from random noise. I don’t have a large background on this topic, however my previous background competing in piano competitions and learning of Spotify’s Convolutional NN music recommendation tool spurned my interest in potentially creating music from the raw music itself. I originally wanted to use actual wave length values, however after scanning existing research and questioning my peers, spectrogram graphs of wave lengths prove to be more accurate for this type of problem. Existing research uses similar convolutional networks for music generation and music interpolation.

As I explored the data, the music itself was put together very nicely for analysis. The 8000 songs were of the same length, each ~30 seconds long which made for easy access and manipulation. These two representations give different representations of the data, the top in decibels (*dB*) and the bottom in amplitude squared (*S*), where:

*dB = 10 \* log10(S / ref)*

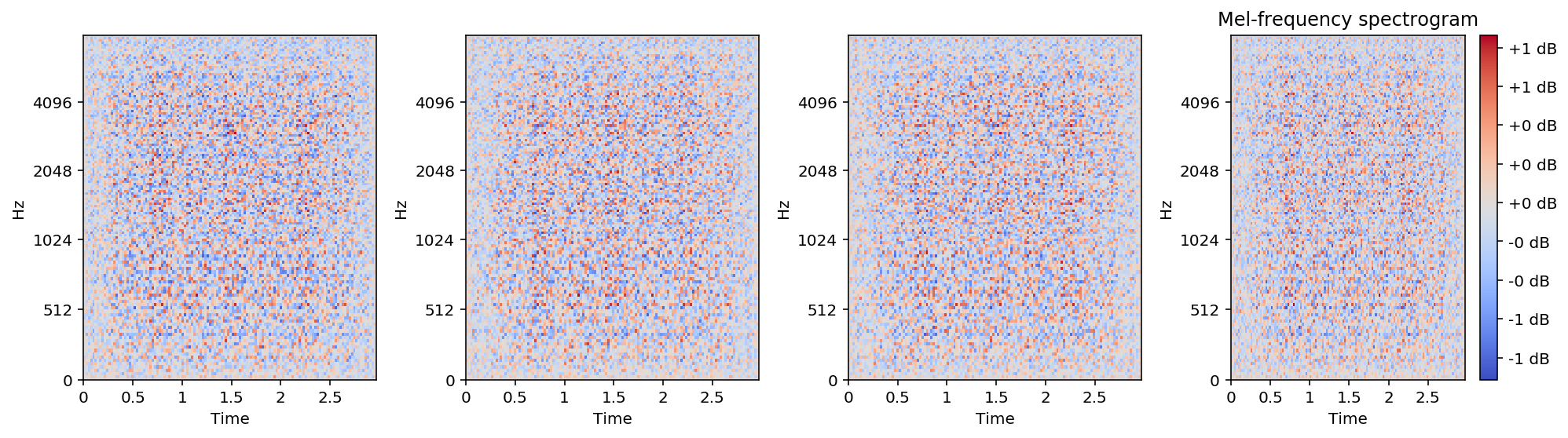
where *ref* is maximum value of *S,* helping normalize the values. We see in the decibels graph the melody lines scanning across the spectrogram, where notes are deciphered and picked up by the spectrogram. Each carry different information can be used for our analysis.





My approach used Convolutional Layers to predict upon 3-second spectrogram images of music files. I used a similar CNN structure to the GAN built in Lab 8 due to the data. Each MP3 file was taken and transformed into a spectrogram for ~3 seconds of music with image size 128 by 128. I then used 6 convolutional layers inside both my Generator and Discriminator classes, then normalizing each convolution using simple 2-d Batch Normalization. I calculated loss using BCELoss as it measures the loss between both the Generator and real values, allowing the model to train against this.

However, our results were very inconclusive. As seen below, our images for both decibels and amplitude squared showed no conversion and strong underfitting of the data.



Although the data owever it begs several questions whether the model isn’t picking up a majority of values in the spectrogram. In this case, I used the first model for my submission but focused more on the model not picking up important values in the data. After running the model with various epochs and batch sizes, it became apparent the model wasn’t seeing information readily available. Due to time limit constraints, I was unable to dive deeper into this, however creating the 128x128 arrays into color images could potentially yield greater results. Although my results are inconclusive, this model gives me a great base, and allows me to understand better how spectrograms work and interact to create music