Music Generation

**Abstract**

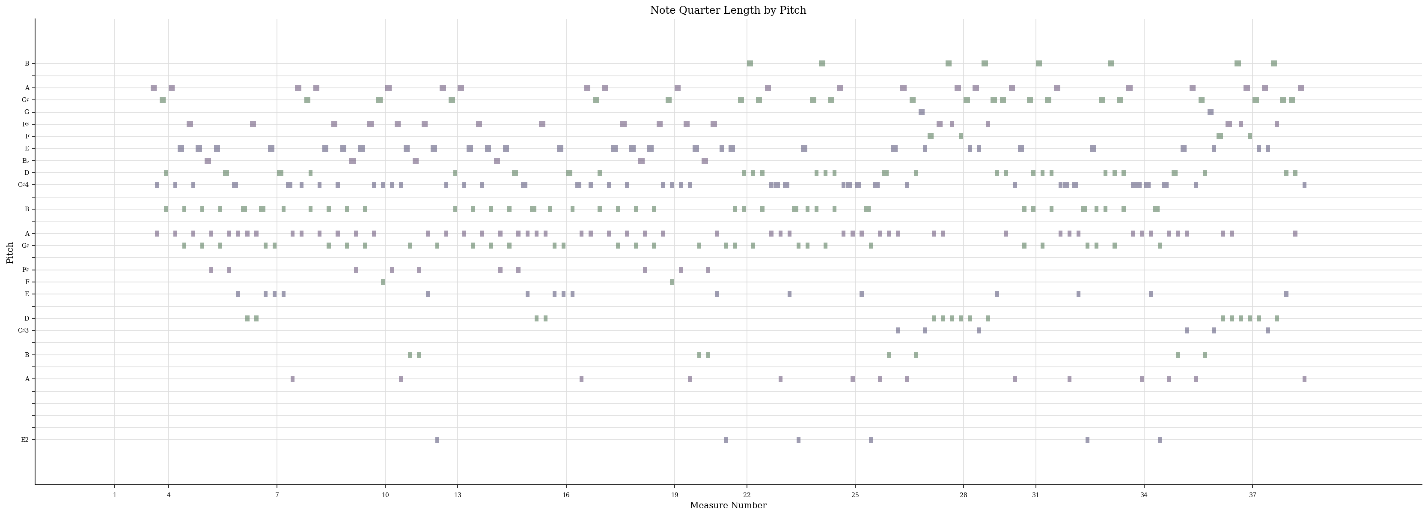
Utilizing recurrent neural net architecture to create original music based on multiple datasets. The end goal of this is to produce a model which produces a midi-to-midi model. A midi-to-midi model could be described as taking in midi data then generating midi files. Midi files are a standard way to store musical information used widely. An example of it would be what garage band uses to generate and store music those are midi files.

**State of the Art Solution**

There are a few different solutions that go about generating music. The first solution I found was the MUSNET model made by Open AI. MUSENET utilized the transformer architecture which takes advantage of the attention mechanisms. They encoded the midi files with the pitch, volume, instrument, and genre into one token. The transformer model was extremely large with 72 layers and 24 attention heads. The next solution is also from Open AI however the method they go about solving it is different. This model known as JukeBox it takes raw audio along with lyrical data as the training data. They take that Audio data and run it through a CNN. They then take the CNN and lyrical data through a transformer and then decode the transformer with another CNN into new generated audio. This varies heavily from MUSENET as the input data is with audio vs midi data. The final solution I found made a model on the MAESTRO dataset. They utilized transformers as well but the data process they went through went going from an audio to midi to audio model.

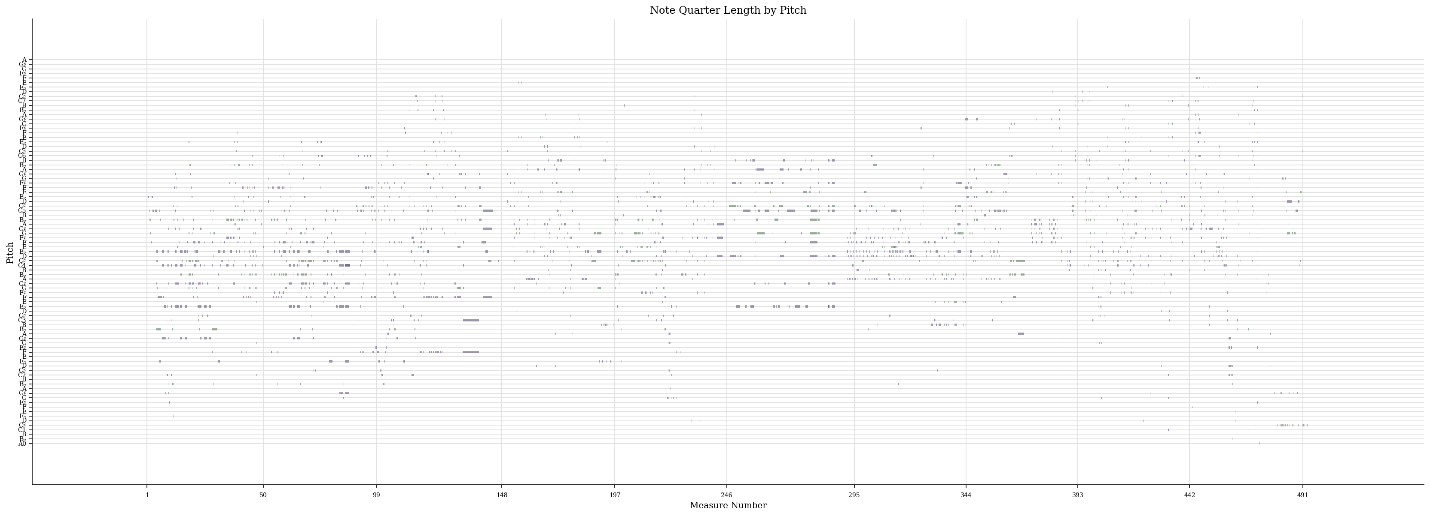
**Dataset explanation and Exploratory analysis**

There are 5 datasets that I used for this to see how different types of music and complexity can affect the learning rate of the model. The first dataset I used was the MAESTRO dataset mentioned earlier it is a set of classical piano virtuoso pieces. The total dataset is around 200 hours long with varying complexity and number of instruments but generally is more complex. The second dataset was from a beginner’s classical guitar website which I scraped for all the midi files. Overall, this music is less complex time wise, and it is all for one instrument. The third dataset was also scraped from a website which hosted Metal and Rock midi files from an assortment of bands. This wasn’t as complex as the MAESTRO however it has the added complexity of additional instruments. The fourth dataset is in similar complexity to the Metal/Rock dataset it is scraped from a website as well it is a set of Irish songs. The final dataset was scraped from a classical guitar website it all the music from a composer name Francisco Tárrega. Tárrega was a Spanish guitar player who wrote complex music for the guitar. A lot of his music took advantage of complex and quicker time that varies throughout each piece.

Fig 1 Example of a song from the Beginner’s Classical Guitar dataset

A picture containing graphical user interface

Description automatically generatedFig 2 Example of a song from the Metal/Rock dataset

Fig 3 Example of a song from the MAESTRO dataset

**Data Preprocessing**

For processing this data, I used the python package Music21. Music21 allows for turning normal midi files into textual data this is important so we can run the model. I used the same method on each dataset. For the music I had to eliminate or make constant certain parts of it. The first one I did was to make all the time constant. When I am referring to time in this case, I am referring to whether a note is a quarter note, eight note, whole note, or some other type. This can drastically change the way a song sounds to a person. The main reason I chose to do this is I couldn’t find a nice way to embed this into one string along with my pitch and octave. After making them all quarter notes I then transformed each song into one array where each datapoint is a string with the pitch and octave. After the array I created a mapping function for later and sequenced it and one-hot encoded it. The reason I chose one-hot encoding instead of embedding was when I started, I was having issues with embedding my data and the total sequences wasn’t too large, compared to a language dataset, due to the fact of holding the time constant.

**Model Explanation and Results**

The model used was a LSTM model utilizing CuDNNLSTM from keras to use the advantage of the higher performance of the graphics card. I also rented a server from Lambda Labs to train the model so I could run all of them quicker with better hardware than I have locally. The model consisted of a LSMT layer followed by a dropout layer then a relu activation layer followed by another LSTM layer, dropout layer, dense layer, and SoftMax activation layer. For this model I had it output a checkpoint every epoch in case I wanted to end the training early so the total number of epochs ran per model is different, but I had it set to 100 epochs as default.

A picture containing background pattern

Description automatically generated

A picture containing chart

Description automatically generatedFig 4 Model trained on the Beginner’s Classical Guitar dataset output as a music sheet

Fig 5 Model trained on the Beginner’s Classical Guitar dataset output

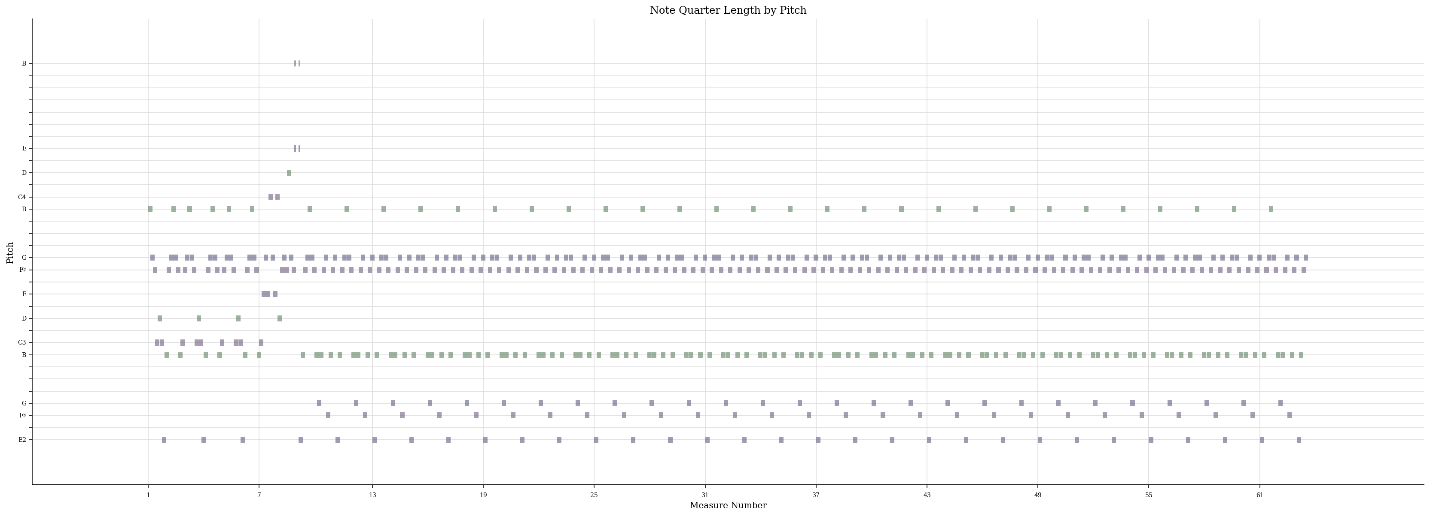


Fig 6 Model trained on the Metal/Rock dataset output

The result of each different model varies drastically and shows the importance of dataset chosen and how I processed the data. The ones shown above were the best performing results out of the 5 different models. There are the audio outputs and other model outputs not shown which can be listened to if you go to the GitHub linked at the bottom of the paper. The overall best performing model was from the beginner’s classical guitar dataset where the output resembles a song. This makes sense as compared to all the other datasets which had a problem this one fit the best for how I was processing my data. Due to it being a beginner’s guitar we see more constant time along with it being one instrument. The results from the metal/rock dataset were a more common occurrence where it repeats output due to poor generalization. This is large in part because that dataset along with the Irish and MAESTRO dataset have multiple instruments in it. Since the processing was not equipped well to deal with more than one instrument a lot of that information was lost upon training the models. Another output which we got from the Tárrega dataset shows the importance of the time and why removing it was a huge hinderance on our model. If you took a quick glance at the output, it would look promising but when you listen to it the resulting sounds make it seem like a complete jumble of notes. However, the reason for that is due to the lack of time it sounds random because all the notes are quarter notes when the songs it was trained from did not utilize quarter notes to that degree, so the time aspect of his music was lost in the model training.

**Future Improvements**

There are two different avenues for improvement. For basic improvements while keeping the basic LSTM structure one would be to do better preprocessing for the data. I would try to redo my encoding of time as well as include the option to support multiple instruments. I though more on how to implement this and perhaps including it in one string separated by periods/commas would be a way, if it were done this way swapping from one-hot encoding to embedding would be necessary. Another improvement would be to implement a N-gram similarity comparison at the end to better compare results instead of the current manual way. This way would give more concrete evidence if the model was overtraining and picking up one a certain song rather than generalizing over the dataset. I would also like to test different model structure and expanding the number of layers. There was also a website which hosted a bunch of more classical guitarists midi files which I found. In the future I would want to use that as it was a much larger amount of midi files compared to any of my other ones.

For more future long-term improvement would be to follow the guide of Open AI and to go the route of changing to a transformer model. This would obviously be the best for this problem due to the advantages attention mechanisms bring. I would try to follow the basic idea of what Open AI did with MUSENET. I would follow some of the same improvements mentioned earlier with embedding and on top of that I would like to add in genre and mix all the datasets together and see what happens. I would also like to see a way to include each song title and for music generation instead of using a random note using a title as starting point for generation.

**References and Code**

All source code along with datasets and where they were found from are linked in the GitHub.

<https://github.com/DChells/CPSC-393-Project>