

**Music Generation using Recurrent Neural Networks**

**Anuja Kapre**

**Raksha Kaverappa**

Index

Abstract 3

Music21 3

Exploratory Data Analysis 9

Feature Selection……………………………………………………………………………………………......11

LSTM for Music Generation 11

5.1 Model Generation 12

Results 14

Web Application 30

Reference 32

# **Abstract**

Since LSTMs have been used for generating patterns or sequences, they can be implemented to compose music. Here, the model treats each note or chord as an element in the sequence and predicts the next note. These LSTMs make use of sequeal information and efficiently learn via gradient descent. LSTMs are extremely useful to solve problems where the network must remember information for a long period of time as is the case in music and text generation.

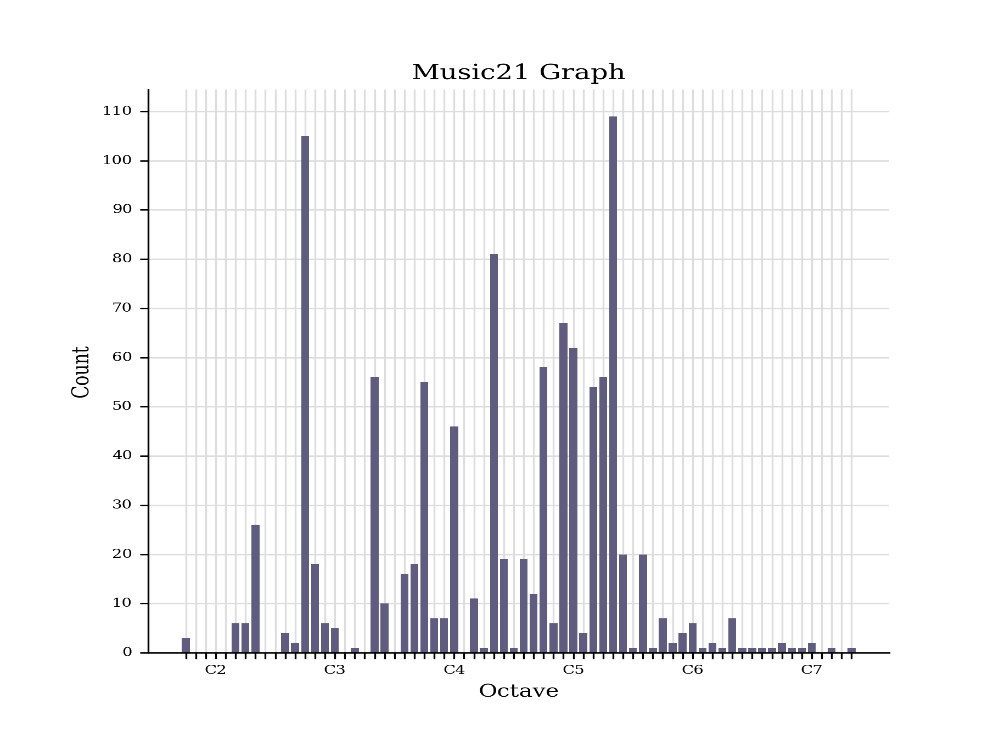
In this experiment, we have used music21 library to take input from midi files and convert them to numerical matrices which can be interpreted by the model. Our final output was a sequence of notes and chords generated after the model was trained for 500 epochs on 1 midi file.

# **Music21**

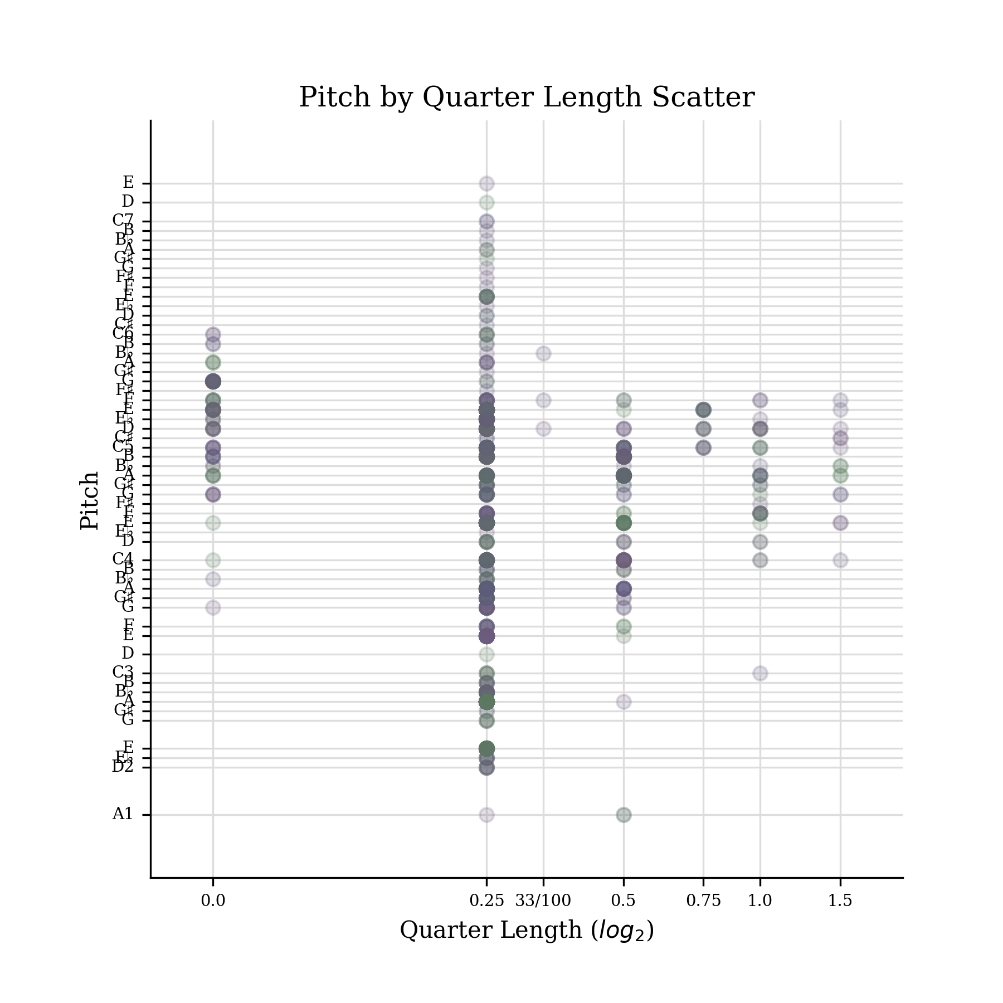
Music21 is a python-based toolkit for computer aided musicology. It can be used for answer questions from musicology, learn music theory, etc., among many other uses. Here we use it to generate and compose new music. Music21 parses the midi file using the and extract the notes and the chords from them. It provides an interface to acquire musical notations from midi files. It also helps us to create notes and chords object so that we can create our own midi files easily.

# **Exploratory Data Analysis**

The distribution of notes is as shown below



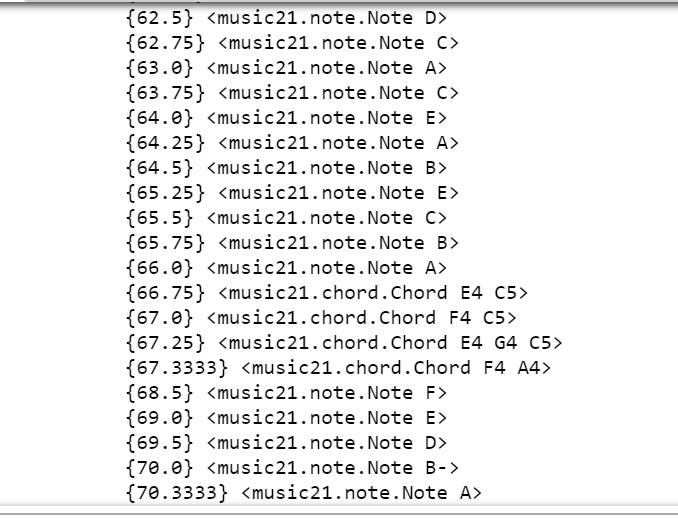
The pitch by quarter length scatter plot is shown below



These graphs tell us the distribution of notes through the midi files. They also help us understand how a composer would compose music by showing the sequence of notes.

# **Feature Selection**

Let us examine the data we are dealing with.



We can see that the midi file is a sequence of notes and chords. Each note has 3 components:

* Pitch-Note being played (A, B, C, D, E, F, G)
* Octave: Refers to which set of pitches you use on the piano
* Offset: Refers to where the note is located on the screen

A chord is a set of notes which are played together. Hence, extract these components of every note in the chord.

Notes usually have varying intervals between them. As we noticed that most of the noted had offset values close to 0.02, we are setting the offset value 0.02.

After selecting the features, we read the notes using music21 library and convert them to sequences to serve as input for our network.

We have provided the sequence length to be 500. So, the model would take the previous 500 notes to predict the next input.

Next, we normalize the input and one-hot encode the output and feed this to the model.

# **LSTM for Music Generation**

Using gating patterns, LSTMs can recognize and encode long-term patterns. These are extremely useful when the network must remember sequences for a long period of time as in the case of Music generation.

In our model, we have two LSTM layers. The first layer takes the network input and returns sequence of output. The second LSTM layer returns a matrix after processing the sequence.

We have one dropout layer to prevent overfitting of the model. We have one dense fully connected layer and provide softmax activation function to that layer.

We use categorical cross-entropy as our loss function and optimize our output using rmsprop.

We are using model checkpoints at every point so that we can stop the model at any point and we don’t lose trained model.

We must put the output of the trained model back to its state. Hence, we will put the load the same model and instead of training it again, we will load the weights. The output pattern is recorded into a single array. We extract the notes and convert them back to midi format.

## **Results:**

***Midi file:***

<github link>

***Accuracy: 97%***

***Loss: 0.12***

# **Web Application**

Using a simple Flask application we deployed the model on heroku server.

The webpage accepts four attributes

1. Name of the input file – These are predefined midi files
2. Name for the output file – Generated music files are automatically downloaded on the machine
3. Offset -- it decides the time gap between predicated patterns (between 0 to 1)
4. Sequence Length – It decided no of patterns predicted by the model (between 50 to 1000)

Link for the flask application: <http://musicgeneration.herokuapp.com/>

Generated Output File: <https://github.com/raksha592/Music-Generation-using-Recurrent-Neural-Networks/blob/master/Output/FinalOutput-10MIDI-0.02off100SeqLen.mid>

Parameters:

1. Name of the input file – 10MIDI
2. Name for the output file – FinalOutput-10MIDI-0.02off100SeqLen
3. Offset – 0.02
4. Sequence Length – 100