Major Project Report

Music Generation

Jerin Joseph

Mayank Bhandari

Computer Science Engineering

National Institute of Technology, Delhi

2022

Music Generation

A Report submitted

In partial fulfilment for the Degree of

B. Tech

In

Computer Science

By

Jerin Joseph
Mayank Bhandari

181210025
181210030

Pursued in

Department of Computer Science and Engineering

National Institute of Technology

Delhi, 2022

CERTIFICATE

This is to certify that the project report entitled **Music Generation** submitted by **Jerin Joseph** and **Mayank Bhandari** to the National Institute of Technology, Delhi, in partial fulfilment for the award of the degree of **B. Tech in (Computer Science and Engineering)** is a bona fide record of project work carried out by them under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree or diploma.

……………………….

Dr. Anurag Singh

Department of Science and Engineering

DECLARATION

I declare that this project report titled **Music Generation** submitted in partial fulfilment of the degree of **B. Tech in (Computer Science and Engineering)** is a record of original work carried out by me under the supervision of **Dr. Anurag Singh**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

NIT, Delhi	Jerin Joseph	Mayank Bhandari
15th Oct, 2021	181210025	181210030

ACKNOWLEDGEMENTS

I would like to express my thanks to my mentor **Dr. Anurag Singh** who guided me in choosing the perfect topic for the project. I would also like to thank my professors and all the faculty members who taught me during this period whose lessons helped me understand the workings of this project. I would finally like to thank my institute for giving me this opportunity to showcase my skills in this project and thus helped me learn more during this period of time.

Jerin Joseph

Mayank Bhandari

ABSTRACT

Music generation is nowadays widely used in the creative industry. Developers around the world are trying to create better AI which generates better music and are also trying to make such AI more accessible to music artists. This AI helps musicians to create better music and gives them inspiration to create new beats and rhythms through its generated music. Even though the algorithm is not perfected to create masterpieces, there have been certain AIs created which make music and whose soundtracks are released to the public. AI which generates music also helps ordinary people who don't have any background in music to create beats and rhythms much easier instead of learning an instrument.

Most of the music generated AI uses an LSTM (Long Short-Term Memory) neural network to generate music. This neural network is a type of Recurrent Neural Network, which uses a feedback system where the output of the previous input will be part of the new input. This method is most suitable for music generation since a particular note is played based on the notes played before to create a song or a rhythm. Many music pieces are required to train the model to predict the next set of notes with increased accuracy. Initially a set of scales are used to train such models and fine tune the variables to create better results. After obtaining suitable results from the initial set of inputs more songs are added into the model to increase its accuracy. The inputs however are not in the form of a wave or a midi file. The inputs are in the form of a piano roll which essentially is a matrix of a particular set of notes in different intervals of times. The piano roll shows whether the notes are played or not in a particular interval of time.

Many people fear that in the distant future, AI generated music will dominate the industry and beat other music artists. These statements are far from the truth since music artists not only create music but they also have live concerts for their fans who adore their music and play their favourite songs live for the audience. Essentially music artists become celebrities and thus AI generated music cannot steal their fame through its generated music since the listeners will miss live concerts with their favourite celebrities. AI generated music has more benefits to offer both for the people who create music and for the people who love to listen to music and thus it is important to continue the study of AI generated music and to create better AI technology which would create music that inspires people.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ NOTATIONS

AI	Artificial Intelligence
BPM	Beats per Minute (Tempo)
BPS	Beats per Second (Tempo in seconds)
CNN	Convolutional Neural Network
E.g.	For example
Etc.	Etcetera
LSTM	Long Short-Term Memory
MIDI	Musical Instrument Digital Interface
ML	Machine Learning
NLP	Natural Language Processing
RNN	Recurrent Neural Network
TPB	Ticks per Beat (Resolution)
TPS	Ticks per Second

CHAPTER 1
INTRODUCTION

1.1 Aim and Scope of Music Generation

Music Generation is widely used in the creative industry. Its main aim is to develop an AI which generates rhythms, beats, riffs etc. that inspires other creators to add those rhythms or beats and create a masterpiece. It's not intended to replace music artist but help them create better music. AI generated music can also be integrated in many music software which help users to create their own music even if the users are not experienced with any instruments. It contributes in making the creation of music more accessible and easier.

1.2 Motivation

Music Generation has always been an interesting subject. It is capable of changing the way music is perceived, in a world where people are taught to learn to play several instruments to create a song, this technology helps to make an AI create a song played in guitar, piano, or any other instruments.

Many companies have managed to create AI which were able to generate music which does not sound like it was created by an AI. Flow Machine, created by Sony, is an AI tool which allows composers to create music of different styles through its music generation algorithms. AIVA is another AI which released multiple copyrighted music. Magenta created by Google is one of the most used music generation models and is being used in many applications which help composers create music faster with the help of the automated process.

CHAPTER 2
STATE OF THE ART/ LITERATURE REVIEW

2.1 Models for Music Generation

Many studies have tried various ML algorithms to generate music successfully. The main principle of generating a particular set of notes at an interval of time is through randomization. The music generated using randomization is called stochastic music. There are various models which are used to implement a proper randomization of notes at an interval of time.

2.1.1 Wave Net using CNN

One such model used is the Wave Net model which is a deep learning model where a particular soundtrack is given to the system as input and then using that soundtrack, the model creates a random sequence of notes. This model is usually used for NLP. This model is based on a type of CNN Model. CNN models are usually used in the field of computer vision and not for audio. Using the original CNN model will not be able to create music efficiently and thus a feed-back system is required.

2.1.2 LSTM

Another common model for the generation of music is the LSTM model which is a type of RNN model. RNN models are usually suitable for music generation since such models have a feedback system and the previous outputs are used for the next input. In many studies, it shows that the preferred model is the LSTM model since it has the ability to carry forward the previous sequence of inputs for a long amount of time[2]. This feedback system helps in generating better music since the notes generated are connected with the previous notes played. The LSTM model has various gates like the forget gate and the input gate which helps to send the previous outputs as feedback.

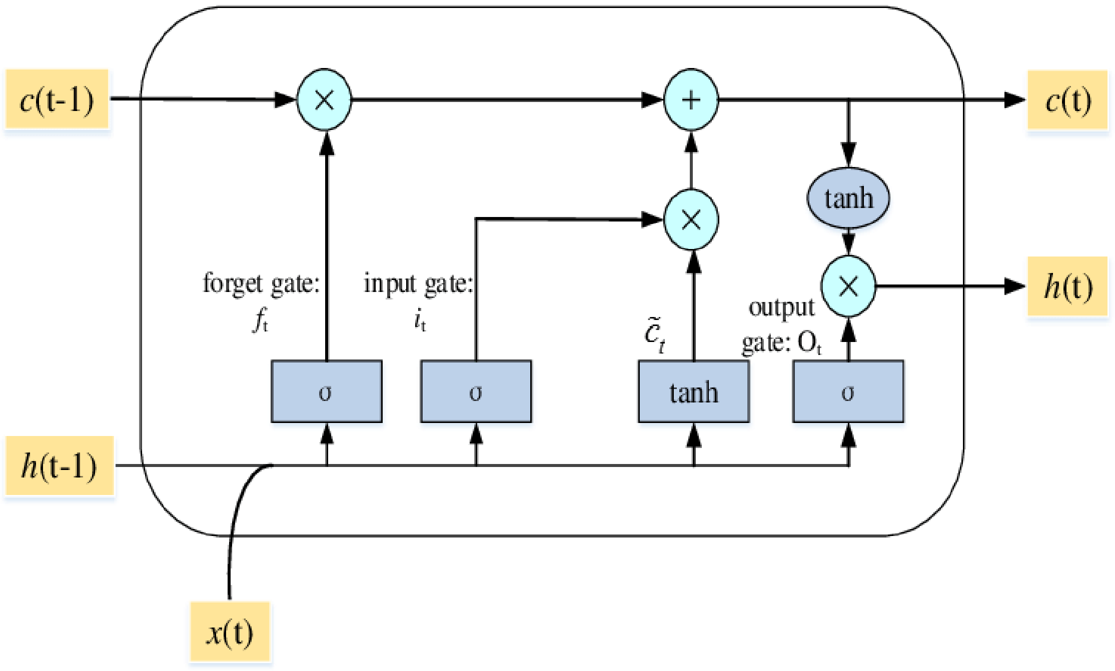


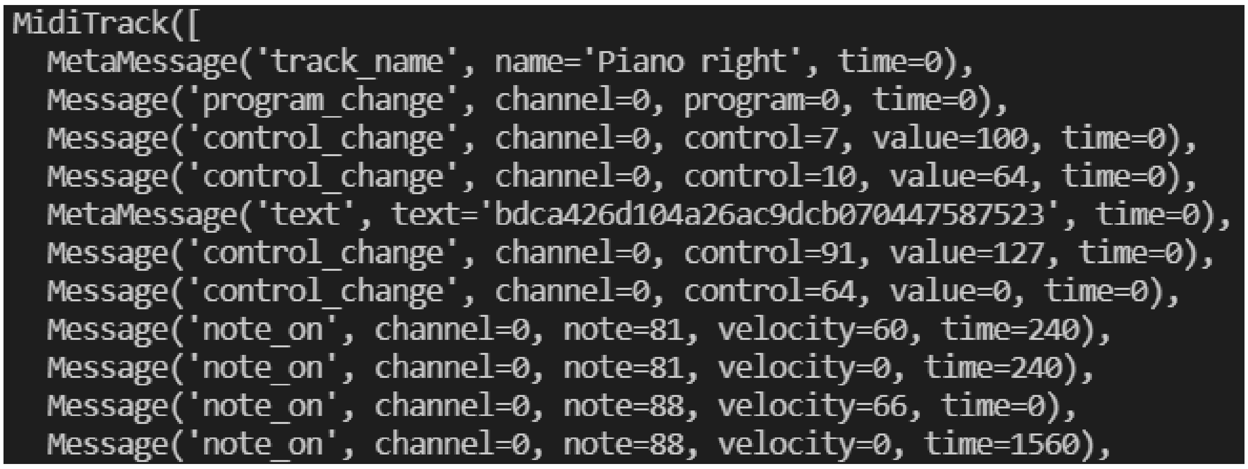
Figure 2.1: Structure of an LSTM Model

2.2 Input for the Music Generation Model

Another challenge in the generation of music is to find out the most suitable form for the input. Generally, the input for any music generation model are the soundtracks. However, the challenge comes in representing the input in the best way possible for the model. The common forms of soundtracks are .wav, .mp3, .flac etc. Adding the soundtracks with these formats directly will not help in any way and cannot be used for the model. These formats are based on the audio waves. A format which consists of certain semantics is important for the generation process.

2.2.1 MIDI Format

MIDI is a widely used audio format around the world. It is used to connect multiple instruments together and is used to merge sounds from different instruments into one. It does not store its data as an audio wave, but instead, stores it based on the notes and the time it is played with its velocity(amplitude). A MIDI file has certain headers which tell about the soundtrack. It consists of the name of the audio file, the resolution of the audio, and a list of all the tracks. A track in a MIDI file is a set of events that consist of the sounds in one piece. If we consider a MIDI file of the piano instrument, there would usually be two tracks, one for the left hand and another for the right. These tracks contain multiple "note_on" and "note_off" events which tell when a note is pressed and when a note is released respectively[5]. These events also have other data like the note being pressed, the time it waits to execute the event, and the velocity at which the event is executed. Other events in a track are "set tempo" and "end_of_track" events which tell the tempo of the track and the end of track respectively.



```
MidiTrack([
  MetaMessage('track_name', name='Piano right', time=0),
  Message('program_change', channel=0, program=0, time=0),
  Message('control_change', channel=0, control=7, value=100, time=0),
  Message('control_change', channel=0, control=10, value=64, time=0),
  MetaMessage('text', text='bdca426d104a26ac9dcb070447587523', time=0),
  Message('control_change', channel=0, control=91, value=127, time=0),
  Message('control_change', channel=0, control=64, value=0, time=0),
  Message('note_on', channel=0, note=81, velocity=60, time=240),
  Message('note_on', channel=0, note=81, velocity=0, time=240),
  Message('note_on', channel=0, note=88, velocity=66, time=0),
  Message('note_on', channel=0, note=88, velocity=0, time=1560),
```

Figure 2.2: Data in a MIDI Track

2.2.2 Piano Roll

MIDI files alone still cannot be added as an input for the model. However, the data in a MIDI file can be converted into a piano roll which is suitable for the model. A piano roll is another format which is more semantic than the rest of the formats reviewed. It shows all the notes played at every interval of time. It can be represented in a matrix of 1s and 0s. The piano roll matrix can be made by making the y-axis depict the note played and the x-axis depict the time interval at which the set of notes are played[4]. It is easier for an LSTM model to be trained using a matrix of 1s and 0s. The 1s represent the notes played and the 0s represent the notes not played. Piano Roll matrices are created by traversing every track in a MIDI file. While traversing a track it checks if the event is a "note_on" event or a "note_off" event, it also extracts the time and note from the event and based on the event and the note, it sets that note to 1 or 0 in every time interval for the time duration of the interval.

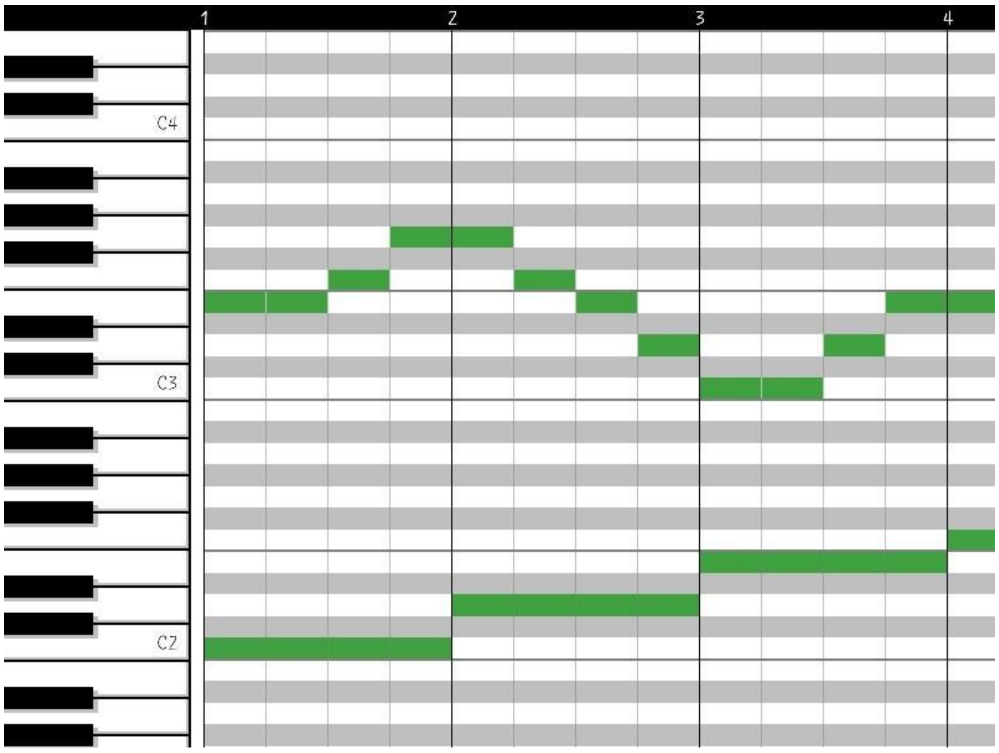


Figure 2.3: Piano Roll (The x axis is the time interval and y axis are the notes played)

Table 2.1: Piano Roll Matrix (The keys C4, D#4 and E4 are played from time intervals 0-3)

Notes	**Time Interval**					
	0	**1**	**2**	**3**	**4**	**5**
60 (C4)	1	1	1	1	0	0
61 (C#4)	0	0	0	0	0	0
62 (D4)	0	0	0	0	0	0
63 (D#4)	1	1	1	1	0	0
64 (E4)	1	1	1	1	0	0

CHAPTER 3
METHODOLOGY

3.1 Data Collection

The data for the model should be in MIDI format since MIDI formats are easier to be converted into piano rolls which later would be the input for the model to be trained. Since the data are used for training the model, the dataset would have a basic form of music. All the dataset used is for the piano instrument.

3.1.1 Scales Dataset

The scales dataset is obtained free from “feelyoursound.com/scale-chords/" website. This dataset is chosen as scales is one of the basic forms of rhythm in music theory. The scale dataset contains all the scales from a major and minor to g major and minor for all the common octaves in the piano instrument. These dataset help train the model with the basic rhythm.

3.1.2 Classical Music Dataset

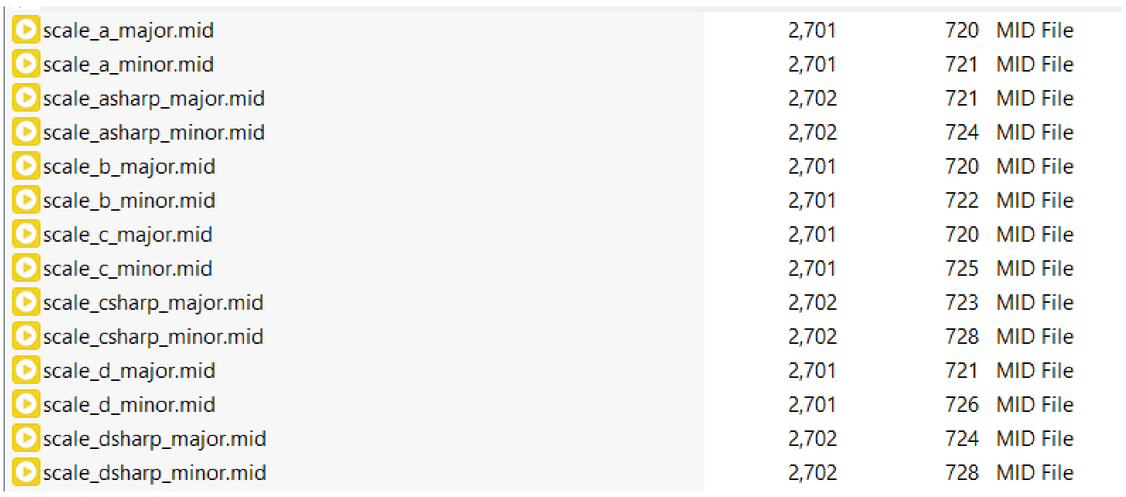
The classical music dataset is obtained from Kaggle. This dataset consists of masterpieces from famous musicians. The songs are from Albeniz, Bach, Mozart, Schubert and from many other musicians. There are a total of 295 songs from this dataset and it will help the model to create complicated beats.

These two datasets are the initial training datasets for the model, depending on the progress of the model being trained, more datasets would be added to generate music with a better accuracy and a better range.

3.1.3 Giant Midi Piano Dataset

The giant midi piano dataset was added later during the training phase of the project. This dataset was added as the current dataset had fewer files for the proper training of the music generation model.

Addition of the dataset added a total of 472 midi files for the training of the model.



scale_a_major.mid	2,701	720	MID File
scale_a_minor.mid	2,701	721	MID File
scale_asharp_major.mid	2,702	721	MID File
scale_asharp_minor.mid	2,702	724	MID File
scale_b_major.mid	2,701	720	MID File
scale_b_minor.mid	2,701	722	MID File
scale_c_major.mid	2,701	720	MID File
scale_c_minor.mid	2,701	725	MID File
scale_csharp_major.mid	2,702	723	MID File
scale_csharp_minor.mid	2,702	728	MID File
scale_d_major.mid	2,701	721	MID File
scale_d_minor.mid	2,701	726	MID File
scale_dsharp_major.mid	2,702	724	MID File
scale_dsharp_minor.mid	2,702	728	MID File

Figure 3.1: Scale Dataset

3.2 Simulation Setup

An LSTM model is used to generate music. This model is trained with multiple input files consisting of various datasets. The initial step is to obtain the required input. The required midi files for training the model are first converted in its equivalent piano roll format. Then after obtaining the piano rolls for all the songs, the data is added into a single list. To begin the training process, the list of piano rolls is divided into several sequences. A sequence consists of a list of all the note values for a particular range of time duration. The training dataset is created by taking data from the piano roll from a range of time duration as the input sequence and then taking the data from the next time duration as the output sequence. These input sequence and output sequence are used to train the model and create the heuristic functions within the model. After training the model, a song is added into the model and it generates its own output sequence which later will be converted to piano roll and then converted to its final midi file. This midi file is the generated music.

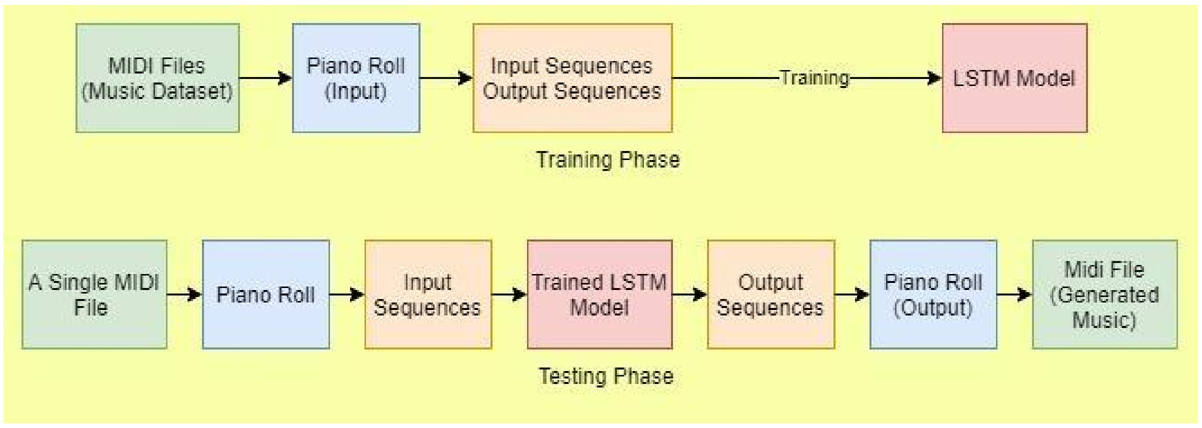


Figure 3.2: Music Generation Steps

3.2.1 MIDI to Piano Roll Conversion

The MIDI to piano roll conversion function's code is written in python. The code utilises the Mido library and the NumPy library. The Mido library helps to extract the contents of the MIDI file and parse through various tracks in the file. It also helps convert piano rolls into MIDI files by creating events and saving them into tracks and saving the tracks into a MIDI file. The NumPy library is very useful for manipulating arrays and hence makes it easier for the conversion process.

The MIDI files are converted into Mido objects. These objects act as the MIDI data required. The resolution of the song which is the TPB (Ticks per Beat) are stored in the MIDI data. Ticks is a unit of time which is standardly used in MIDI files. Ticks are usually very less compared to seconds. Beats also can be considered as a unit of time. It usually is comparable with seconds and tells the rhythmic timing of the song played. After obtaining the resolution, the tracks in the midi data are traversed. These tracks contain tempo which is extracted for further use. Tempo represents the BPM (Beats per minute). However, tempo stored in MIDI tracks are represented as microseconds per beat, which is converted to BPM. Using the resolution and tempo, the TPS (Ticks per second) can be obtained. TPS is very important as it tells the relation between ticks and seconds, where ticks are the standard unit of time for the MIDI file and seconds are the standard unit of time for a person.

$$\text{TPS} = \text{Resolution (TPB)} * \text{Tempo in seconds (BPS)} \qquad \text{(eq. 3.1)}$$

To create the piano roll from the MIDI data, a suitable time slice must be chosen. A time slice represents a column in the piano roll matrix. The ticks per time slice tells how many ticks consist of a column in the matrix. While traversing a track, the total ticks are obtained, this is done by taking the sum of all the time data in each event in a track. The total ticks constitute the whole x axis of the piano roll. This x axis must be divided into time slices based on the ticks in a time slice. A suitable value is taken as the seconds per time slice. Based on the value chosen, the ticks per time slice changes. With the ticks per time slice and the total time slice, the total number of columns for the piano roll can be obtained.

$$\text{Ticks per Time Slice} = \text{TPS} * \text{Seconds per Time Slice} \qquad \text{(eq. 3.2)}$$

$$\text{Number of Time Slices (Columns)} = \text{ceil (Total Ticks/ Ticks per Time Slice)} \qquad \text{(eq. 3.3)}$$

The events in all the tracks are traversed and for every “note_on” event with a velocity > 0 (amplitude is not 0), the value for that note at that time interval is 1 and for all the other values, the value for the same is 0. The matrix data is obtained for all the tracks and converged to get the final piano roll.

3.2.2 Piano Roll to Sequence Conversion

After the conversion of the MIDI file into its equivalent piano roll. The piano rolls for all the MIDI files are split into sequence for training. An array of input sequence and an output sequence is generated from the piano rolls. A sequence length of 50 is considered in this project for the input and output sequence. It means that 50 columns where each column acts as a time slice is used in the sequence and the rows for the sequence consist of the note array with all the notes played during that time slice.

After generation of the sequences, the input sequence and output sequence matrix are shuffled. The shuffling of the sequence is performed to obtain better training results.

3.2.3 Model Training

After obtaining the inputs in the form of sequences, the model training phase begins. In this phase, multiple training iteration is done to obtain the best variables for the training which would eventually lead to better accurate prediction.

After training the model, another input sequence is added into the trained model from the already existing dataset. The model uses the input sequence and predicts the output sequence. The output sequence then is converted into its piano roll format and then back to a MIDI file. The audio generated sounds better if the model is better trained.

The model used in the project is a Sequential LSTM Model which takes the input as a sequence and gives the output as a sequence[3]. The model layers contain an encoder and a decoder where the encoder layers are used to extract the main features of the input sequence and the decoder sequence.

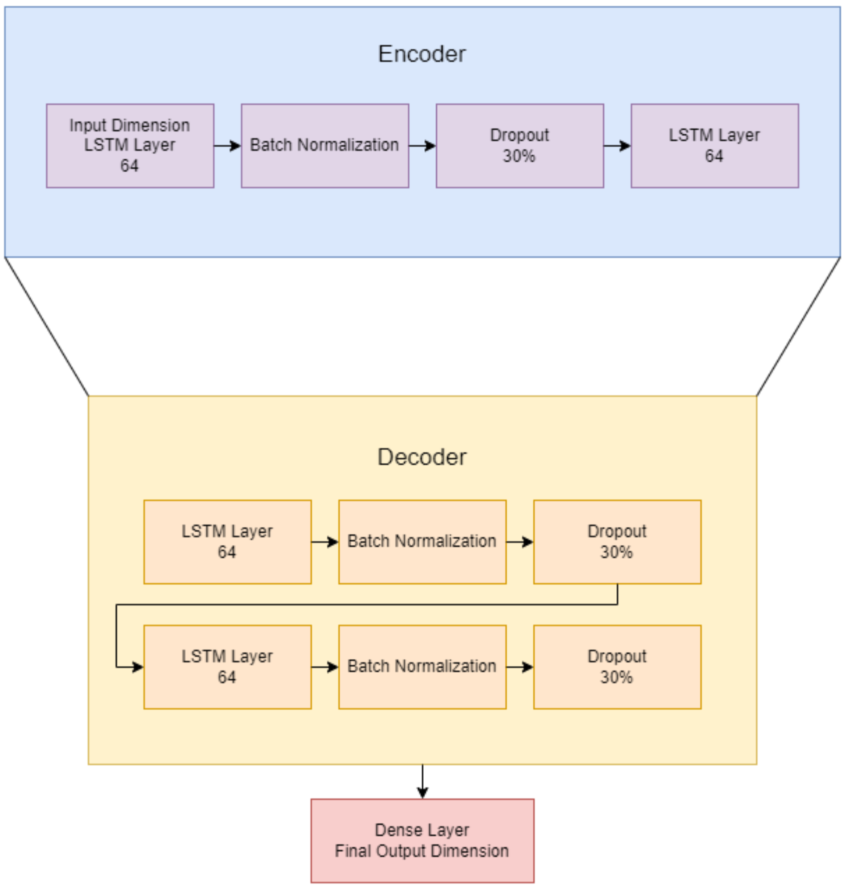


Figure 3.3: Encoder-Decoder LSTM network used for the model

An encoder-decoder model was the optimal model network for the project since the dataset used for the project is less to use more complicated models. Although there are models which produce better results and are more efficient like the C-LSTM models, it requires a different set of input and a huge variety of audio files for training with a high processing power for the training process.

The encoder consists of two LSTM layers with a dropout of 30 percent to prevent overfitting and batch normalisation is used to make the training more effective. The data passing through the encoder is used to reduce the feature size to extract the important data. This data is then passed to the decoder with 2 more LSTM Layers followed with dropout and batch normalisation layers. At the end of the decoder the data is passed through the Dense to fit the resultant data in the final output sequence dimension.

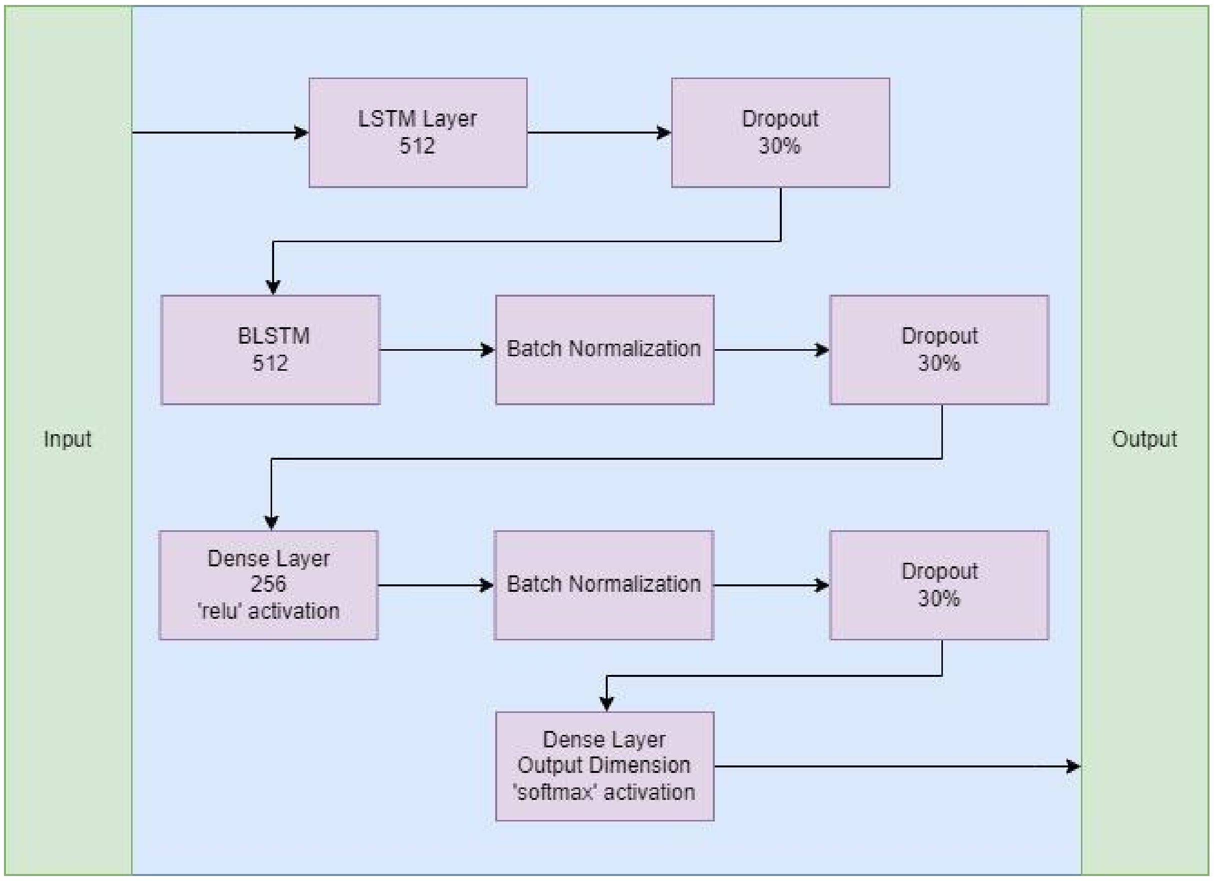


Figure 3.4: Enhanced LSTM network used for the model

We tried creating our own (enhanced version) model, which takes **input** and passes through different layers as shown in above figure 3.4. Unlike the previous model, we here have a **Dense layer with "relu" activation.**

The optimizer used in the model is an Adam () optimizer and the loss function used is categorical cross entropy. The optimizer and the loss function help in speeding up the learning rate and decreasing the loss.

The accuracy of the model is not supposed to be high, but is supposed to be good enough to predict its own sequence and thus creates a generated music. The number of epochs chosen is 50 since after the completion of 50 epochs no change in the accuracy is noted.

3.2.4 Model Generation

After the training phase is completed, an audio file is selected randomly. This midi file is converted to its equivalent piano roll which is then converted into a sequence format and acts as the input for model prediction.

The saved model is loaded and used for prediction to obtain an output sequence. The input sequence given to the model will make it generate a new output sequence and since its accuracy is not that high, the generated output sequence will be completely new from the input sequence but its pattern will be based on the input sequence and all the audio files used for training the model.

After the generation of the output sequence, the output sequence is converted to its equivalent piano roll. The output sequence generated has data ranging from 0 to 1 and a threshold of 0.1 is used so values greater or equal to 0.1 is considered as 1 and values less than 0.1 is considered as 0. This is done as the piano roll only contains 0 or 1 as its values.

After obtaining the piano roll, the piano roll is converted to its final midi file by creating multiple midi messages based on the value of the piano roll matrix. The code checks all the notes played in a time slice and creates a midi "note_on" message for all the respective notes. The final generated file is then saved in the output folder.

CHAPTER 4

RESULTS

The project consists of obtaining multiple midi files as dataset, then converting them into its equivalent piano rolls and then later convert them into input training sequence and output training sequence for the model. An LSTM model is used with an encoder-decoder model network. After training a random file is used as the test input and is passed through the model as a sequence test input. The model creates a sequence test output which is converted to a midi file and saved as output. During the input creation, model training and model generation phase, many arbitrary values were used to obtain the adequate results. Some of the arbitrary values chosen in the code are as follow:

Table 4.1: Arbitrary Values (These values could change based on the progress of the project)

Variable	**Value**
Seconds per Time Slice (column)	0.02
Highest note that can be read	105 (A7)
Lowest note that can be read	21 (A0)
Velocity (Amplitude)	65
Ticks per Time Slice (for converting output piano rolls into output MIDI file)	1
Input Sequence Length	50
Output Sequence Length	50
Epochs	50
Batch Size	64
Dropout Percentage	30 %
Output Dimensions for LSTM layers	64

4.1 Piano Roll Creation

The generation of piano roll from a midi file function helped to generate training values for the model properly. To check the working of the function, a classical song by Albeniz was added into the code and was converted into its equivalent piano roll. This piano roll was then added to the function which converts the piano roll into its MIDI file as output. The original MIDI file and the output MIDI file audio were played and both sound similar.

Although in the output MIDI file there were some syncing problems between the right hand and left hand it still sounded similar to the original audio. The piano roll obtained was plotted into a graph using matplotlib library.

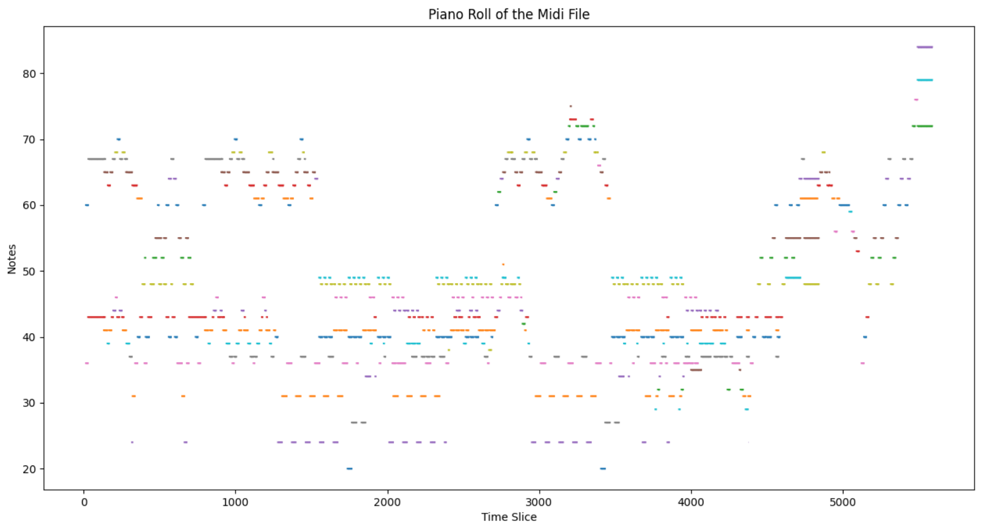


Figure 4.1: Albeniz Classical Song’s Piano Roll

It is really difficult to tell whether the piano roll for the Albeniz audio file is accurate. The C Major MIDI file is converted into its piano roll and plotted as a graph. This audio file is easier to test with since it is one of the basic rhythms in music. The audio file consists of the C Major scale from the central octave, the chord version of the C Major in multiple octaves, and different versions of the C Major with minute tweaks.

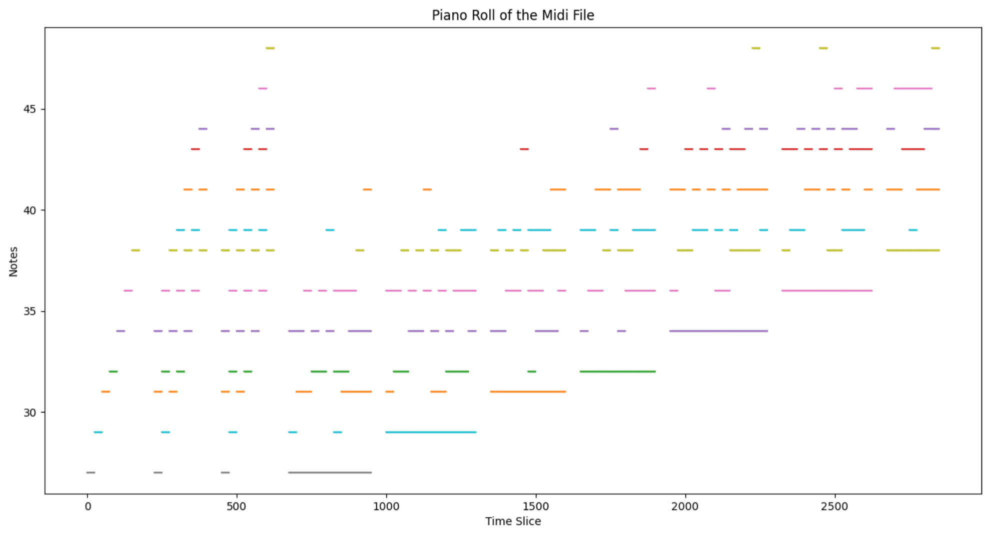


Figure 4.2: C Major Scale's Piano Roll

4.2 Model Training

The model training phase took an enormous amount of GPU power to run. Keras was used to create the model and cuDNN and CUDA libraries helped to run the model faster with the help of the Nvidia GTX 1660Ti graphics card. The amount of loss during the model training phase was 8.5 which could be improved by taking more data for training or by optimizing the model in other ways. However, the generated sound does contain patterns throughout the song and does contain continuity throughout the song.

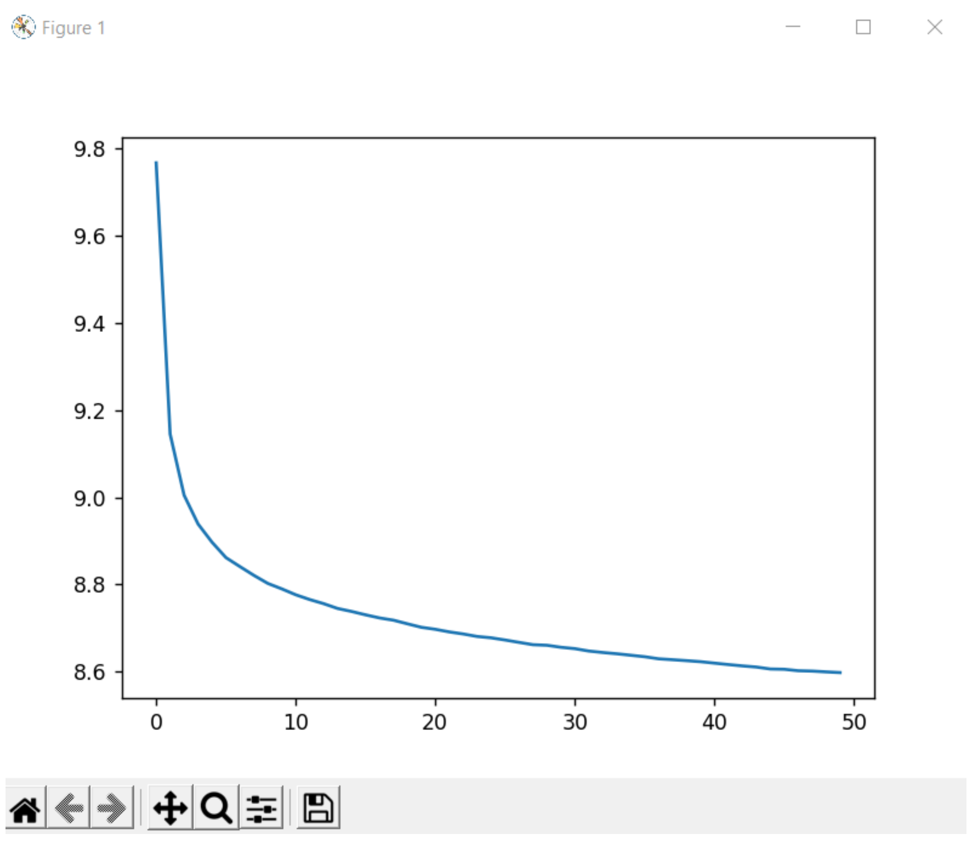


Figure 4.3: LSTM Model Loss Function (X-axis: Epochs, Y-axis: Loss)

This is the **loss function curve** for the 1st model we created. It can be seen clearly that we were able to **reduce loss-value from 9.8% to 8.6% in just 50 epochs.**

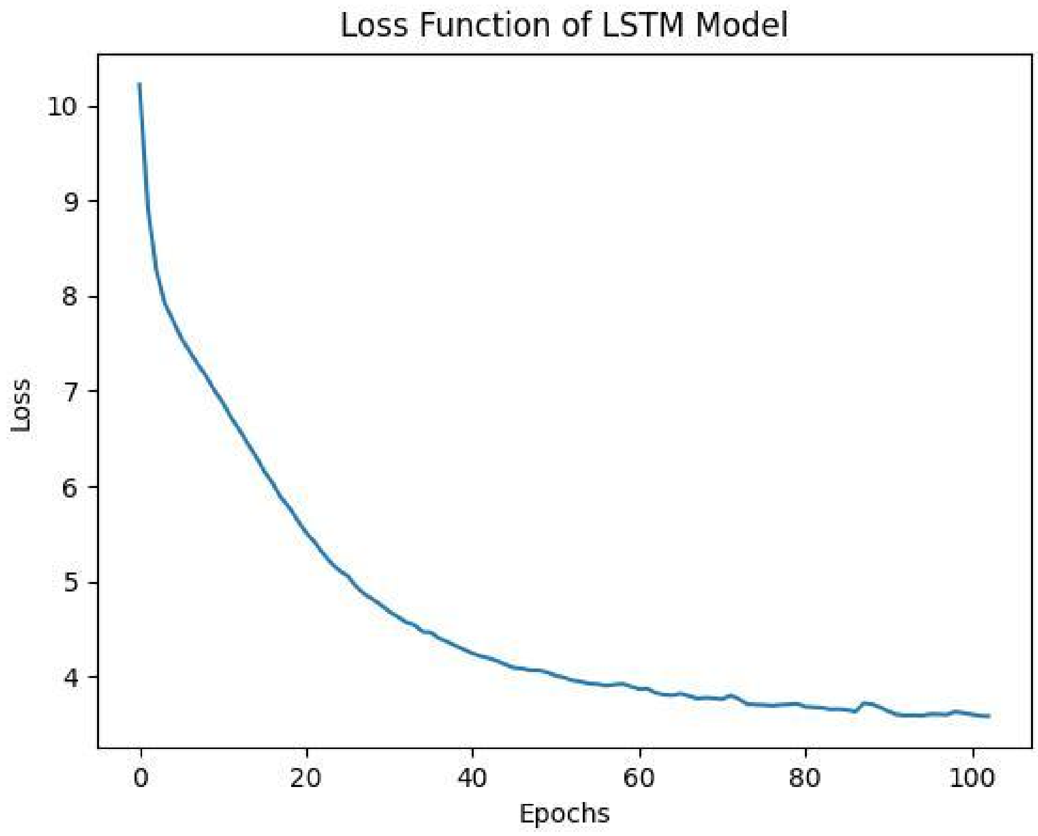
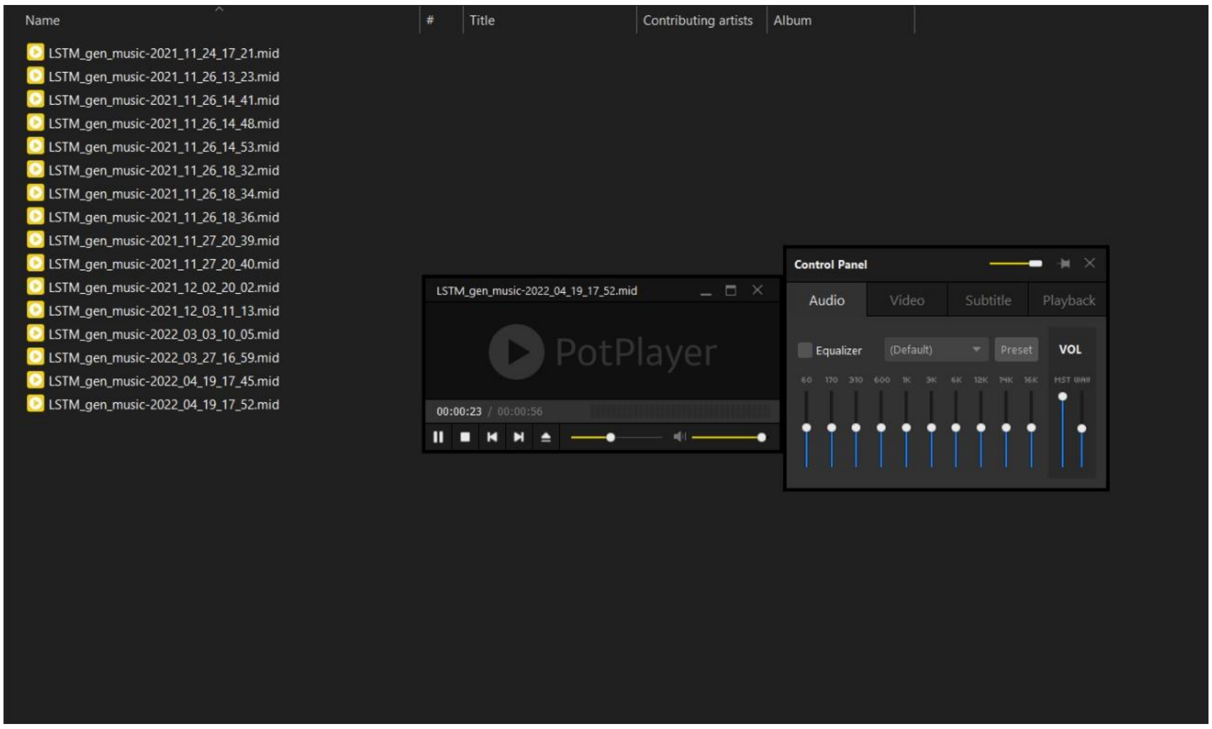


Figure 4.4: Enhanced LSTM Model Loss Function (X-axis: Epochs, Y-axis: Loss)

This is the improved model (**Enhanced LSTM Model**). Here we significantly reduced **loss-value** from **10% to 4% in less than 100 epochs**.

4.3 Model Generation

The generation of songs is done by giving a random song as input to the model which tries to predict the notes in the input creating a new song. The newly generated midi files are added in an output folder. Although the song doesn't sound like proper music, it does contain patterns generated by the AI. The patterns in the music are continuous and sometimes might not sound harmonic.



Name | # | Title | Contributing artists | Album

```
LSTM_gen_music-2021_11_24_17_21.mid
LSTM_gen_music-2021_11_26_13_23.mid
LSTM_gen_music-2021_11_26_14_41.mid
LSTM_gen_music-2021_11_26_14_48.mid
LSTM_gen_music-2021_11_26_14_53.mid
LSTM_gen_music-2021_11_26_18_32.mid
LSTM_gen_music-2021_11_26_18_34.mid
LSTM_gen_music-2021_11_26_18_36.mid
LSTM_gen_music-2021_11_27_20_39.mid
LSTM_gen_music-2021_11_27_20_40.mid
LSTM_gen_music-2021_12_02_20_02.mid
LSTM_gen_music-2021_12_03_11_13.mid
LSTM_gen_music-2022_03_03_10_05.mid
LSTM_gen_music-2022_03_27_16_59.mid
LSTM_gen_music-2022_04_19_17_45.mid
LSTM_gen_music-2022_04_19_17_52.mid
```

LSTM_gen_music-2022_04_19_17_52.mid

Figure 4.5: LSTM Model Generated Audio File

The generated songs can be accessed in the GitHub repository attached at the end of the report. The audio files are in an output folder.

CHAPTER 5

CONCLUSION

Although the model will be able to generate music, it may not sound as good compared to other man-made songs. However, it is still a step forward in reaching the goal where an AI can make music that sounds better than most of the existing songs. It may not be possible to make rhythms sound better than a most masterpiece, but in the distant future, music generation AI will definitely create soundtracks that sound different and would inspire many other creators to go in a different direction in composing music. It would give ideas and show the potential of music and how it can be different even after a long period of time. The applications which contain music generation AI will help ordinary people to create music, solely based on their creative thinking without the constraints of learning a new instrument.

REFERENCES

Research Papers

1. Nayebi, A., & Vitelli, M. (2015). Gruv: Algorithmic music generation using recurrent neural networks. Course CS224D: Deep Learning for Natural Language Processing (Stanford).
2. Huang, Y., Huang, X., & Cai, Q. (2018). Music Generation Based on Convolution-LSTM. Comput. Inf. Sci., 11(3), 50-56.
3. Mangal, S., Modak, R., & Joshi, P. (2019). Lstm-based music generation system. arXiv preprint arXiv:1908.01080.
4. Shi, Z., Arul, K., & Smith III, J. O. (2017). Modeling and Digitizing Reproducing Piano Rolls. In ISMIR (pp. 197-203).
5. Raffel, C., & Ellis, D. P. (2016, August). Extracting Ground-Truth Information from MIDI Files: A MIDIfesto. In ISMIR (pp. 796-802).

Figures

- Fig 2.1: Basic architecture of the LSTM Model

  Retrieved from hindawi.com/journals/complexity/2021/9903518/
- Fig 2.3: Example of a Piano Roll

  Retrieved from researchgate.net/figure/Example-of-symbolic-piano-roll

Dataset

- Soumik Rakshit (2019). Classical Music MIDI. Kaggle
  Retrieved from kaggle.com/soumikrakshit/classical-music-midi
- Webpage. Free Scale Chords Pack. FeelyourSoul
  Retrieved from feelyoursound.com/scale-chords
- Giant Midi Piano Dataset
  Retrieved from github.com/bytedance/GiantMIDI-Piano

Other References

- Mido Documentation
  Retrieved from mido.readthedocs.io/en/latest/
- MIDI Note Numbers and Centre Frequencies
  Retrieved from inspiredacoustics.com/en/MIDI
- MIDI to Piano Roll using Numpy Array
  Retrieved from medium.com/analytics-vidhya/convert-midi-to-numpy

APPENDIX I

CODE FOR THE PROJECT

The code for the project is all in the GitHub repository attached at the end of this appendix. The output file in the GitHub repository contains some of the generated music files.

GitHub Repository: github.com/jerinjoseph121/Music-Generation

APPENDIX II

PLAGIARISM REPORT

Due to lack of resources to check for plagiarism, the plagiarism report has not been added.

APPENDIX III

FORMULAS USED

EQUATION	**EQUATION NUMBER**
TPS = Resolution (TPB) * Tempo in seconds (BPS)	3.1
Ticks per Time Slice = TPS * Seconds per Time Slice	3.2
Number of Time Slices (Columns) = ceil (Total Ticks/ Ticks per Time Slice)	3.3