**Music Composition using LSTM**

Avik Pal1, Biprarshi Dutta1, and Snigdha Bhagat2

avikpal62@gmail.com

1 Department of Electronics and Communication Engineering,

Visvesvaraya National Institute of Technology, Nagpur

**Abstract.** Sequence modeling is now a major field of research in the latest deep learning applications like speech recognition, music generation, sentiment classification, and DNA sequence analysis. In this paper, we focus on music composition where the task is to take some existing music data then train a model such that it can extrapolate and generate new music sequences. It should not simply be duplicated from the training data. It has to understand the patterns of music to generate new music. We propose here a Long Short Term Memory (LSTM) Recurrent Neural Network (RNN) model for musical note composition. We have modified the model from [4] by adding an extra LSTM layer. The data we feed the network is of Musical Instrument Digital Interface (MIDI) format. The various music components in a MIDI file is discussed. The pre-processing of this data before the training and architecture of the model is also discussed. The model after examining produces good melodious music. The output was checked for similarity by taking the Euclidean distance with the training dataset, to ensure we have a distinct composition. We have also used our model to extrapolate a given piece of music which could be beneficial for musicians.

**Keywords:** Deep learning, LSTM, Time-Series modeling, Music generation.

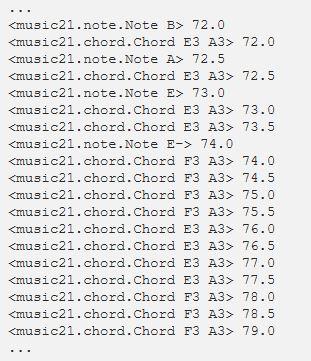
**1 Introduction**

Music is the art of ordering tones and sounds in succession, and in combination such that they produce a melodious symphony. Notes are the discrete building blocks of music that help improve performance and analysis. Chords are a group of notes (also called pitches) played simultaneously.

The musical symbols representing notes and chords in printed form is called sheet music. Here, music is represented by a sequence of musical notes, each separated by a space. This can be used to represent both a single instrument and multi-instrument music. Our key task is to represent music as a time series data which will be used as input to the RNN.

The MIDI file format stores the sequence of changes in notes and chords. The notes and chords are also called the objects of a MIDI file. It contains several types of information such as the note played, how hard the note is pressed and how long the note is held. The MIDI data is transmitted for all the notes simultaneously in case of multiple notes.

The Music21 package in python is used for manipulating MIDI files. It gives an interface to get the music notes of MIDI files. It also helps to create chords and notes with which we can create MIDI files.



**Fig. 1.** The output of contents stored in the midi file, read using Music21 with the offset of the object behind it.

As can be seen from figure 1 and most of the dataset, the most common time interval between notes in the MIDI files is 0.5. Therefore, we simplify the data and model by disregarding the varying offsets in the list of possible outputs.

This paper covers software implementation of the proposed model, related work and technical challenges faced by researchers in this domain.

**2 Related Work**

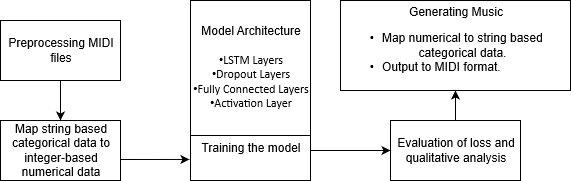
Automatic music composition was first studied in 1988 by Lewis and Todd, who proposed the use of neural networks. A multi-layer perceptron was utilized by Lewis[2] for his algorithmic way to actuate tune composition. Though a Jordan auto-regressive neural system (RNN) was utilized by Todd[1] to produce music consecutively, this model is as yet legitimate after numerous years.

Even though RNN's are used widely for time series modeling they face the problem of vanishing or exploding gradients. They cannot connect information from previous steps to the current step. Hochreiter et al.[6] (1991) and Bengio, et al.[7] (1994) came up with a solution. They proposed the use of LSTM (Long short-term memory) cells which remember long-term dependencies.

In 2014, Dieleman[3] and colleagues researched on what is called End-to-end learning for music audio. They directly processed waveforms for the task of music audio tagging. Kotecha and Young[5] in their paper presented a Bi-axial LSTM network to generate polyphonic music. Sigurður Skúli[4] in his article used a two-layered LSTM model to generate music. Our model is inferred and modified from his article.

**3 Methodology**

We have used an LSTM network because of its ability to process and predict time-series data which is music in our case. We process the MIDI files to obtain the sequence of notes. However, the input to the network is numerical rather than string-based music notes. Hence, we create a dictionary to map the string data to integer-based numerical data. Once training is complete, the predicted numerical data is converted back to notes. The stream of notes is used to create a new MIDI file. The generated file is then checked for similarity with the training dataset to analyze the uniqueness of the predicted musical piece. The flow of work is shown in figure 2.



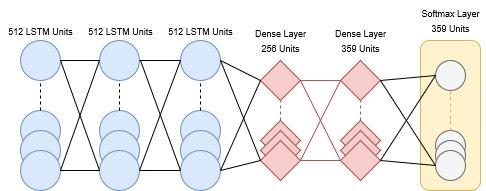
**Fig. 2.** Flowchart of work

**3.1 Implementation**

The model from [4] uses a 2-layered LSTM network. We modify this model by adding a third LSTM layer which gives significantly better results. We have also increased the dropout rate to prevent overfitting of the predicted output with the training dataset.

We start by extracting objects from all the MIDI files in the dataset. The notes are placed in a list. If the object is a chord, we split it into notes and append to the notes list. We then create a dictionary to map the notes to integers. These integers are then normalized to adjust their values to a smaller scale. The input sequence initially consists of the first 100 notes (1 to 100) followed by the next sequence of 100 notes (2 to 101) and so on. This sequence is fed to the LSTM network.

The mapping and correlation between notes, the patterns followed by the neighboring musical notes and their duration is learned by the LSTM layers from the training dataset. We added dropout layers to avoid overfitting. We then add dense layers next to dropout layers to connect the previous LSTM layers and reshape to our list of notes (359). Finally, the softmax activation layer is added to the model. This determines which neurons (LSTM cells) should be activated during training. The model architecture is shown in figure 3.



**Fig. 3.** Model Architecture

After training is complete, the model is now capable of generating a new sequence of musical notes. A random index is picked from the dictionary of notes as the start point. This helps in getting different results each time on re-evaluating the prediction network. The same mapping function created while training is used to map integers back to notes. The probability of occurrence of a musical note at the current time is being predicted by the LSTM network. The current output at any instant depends on all the previous 100 outputs.

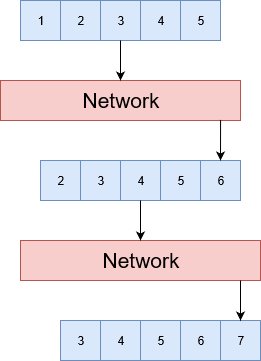
We generated 700 notes which is three minutes of output notes and also provides the network with sufficient data to form a tune. For each note, we provide a sequence to the network. The initial sequence is that of the start index notes. For later sequences, the first note is removed and the output of the previous iteration is appended at the end of the sequence as shown in figure 4. We then collected all the network outputs into a single array. The decoded output can be either a note or a chord. If it is a note it is left as it is. If it is a chord, the string is split into a note array. We next loop through each note while creating note objects for them. A chord object is then created which contains these notes.

After every iteration, we append the note/chord object created to a list and increase the offset by a constant 0.5 as was discussed in section 1. We then create a Music21 Stream object after obtaining the output generated. Finally, the MIDI file is created.

We have used the categorical cross-entropy (CE) loss function. The CE Loss is defined as,

CE = -∑ ti ×log(si)

where ti and si are the true value and the generated value for each class i in the notes list. We have used Adam optimizer for our purposes. Adam makes use of the average of the second moments of the gradients. This optimizer is the most used in deep learning because it achieves good results fast.



**Fig.4.** Prediction input sequence

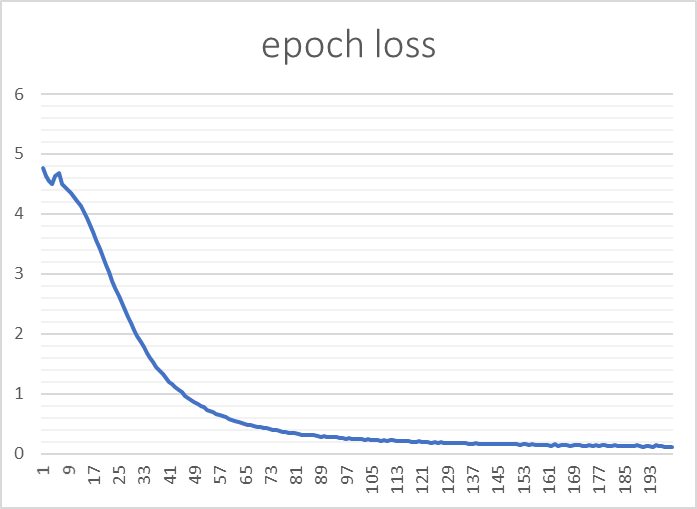
From the predicted output musical note sequence, we also find the minimum similarity distance with the existing dataset which subsequently showed that unique and random patterns of musical notes are generated in every prediction which is substantially different from the training dataset. Euclidean distance is used to find the similarity between the patterns created by our network and the training dataset. This works because Euclidean distance is the L2 norm.

**3.2 Dataset**

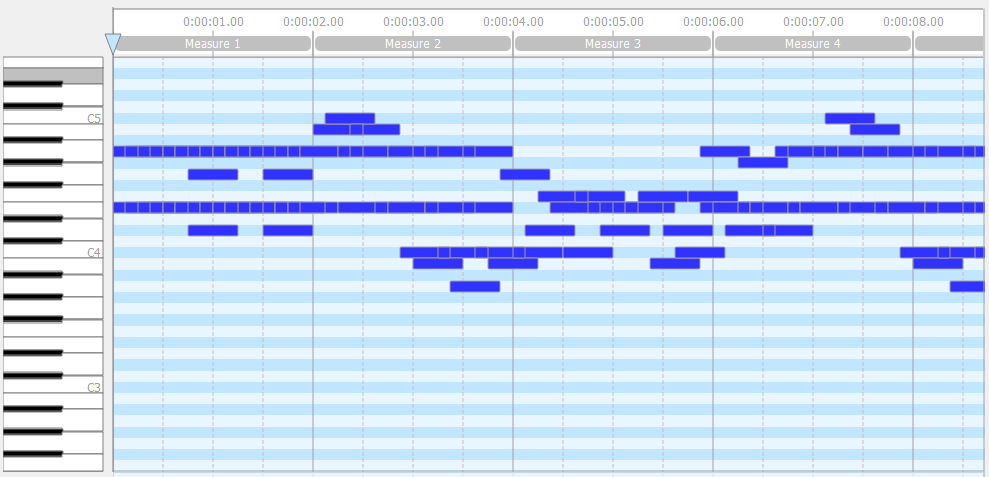
For our dataset, we picked Final Fantasy music due to the very distinct and beautiful melodies that the majority of their albums have, the sheer amount of pieces that exist and the use of a single instrument for recording them. This dataset consists of a wide range of musical tones varying from light background music to emotionally intense pieces.

**4 Results**

After training the model we obtain the graph of loss versus the number of epochs as shown in the figure. 5. The loss is observed to gradually decrease as the number of epochs increases. At the starting point, the loss was approximately 4.772 but as we move further, loss quickly converges and then becomes steady and gets reduced approximately to 0.1169 at the 200th epoch. This loss is significantly lower than the loss we had initially indicating the network is trained properly. Figure 6 shows the newly generated music in MIDI format when opened in MIDI Editor software.

****

**Fig. 5.** Graph of loss per epoch after 200 epochs of training



**Fig. 6.** Output MIDI file viewed in MIDI Editor software

**4.1. Qualitative Analysis**

The music composed by the model has a decent rhythm and creates stretches of melody after some interval as found in good music. These stretches of notes have a local sense of coherence. The network having been trained for 25 hours, was checked for similarity with the training dataset using Euclidean distance. The minimum Euclidean distance with the training dataset was found out to be 4.45 indicating that the output is not directly replicated from the training dataset. Five compositions were generated using our network and were reviewed by a group of 20 people thereby giving us good results based on consensus. The composed music can be heard in the link [8].

**5 Conclusion**

In this paper, a three-layer LSTM model capable of learning harmonic and melodic rhythmic probabilities from MIDI files of the Final Fantasy soundtrack was developed. Our model gives significantly better results as compared to the 2 layer model used by [4]. The output was checked for similarity with the training dataset using Euclidean distance to ensure we have a distinct composition. The underlying logic and method of training and generation of algorithmic music were presented. Further, the outputs of the model were analyzed quantitatively and qualitatively. One application of this is the extrapolation of music used by musicians to predict subsequent notes given an a priori input sequence of notes. This has been included in the code link [8].

**References**

1. Todd, 1988 — “A sequential network design for musical applications” in Proceedings of the Connectionist Models Summer School.

2. Lewis, 1988 — “Creation by Refinement: A creativity paradigm for gradient descent learning networks” in International Conference on Neural Networks.

3. Dieleman & Schrauwen, 2014. “End-to-end learning for music audio” in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

4. Sigurður Skúli, “How to generate music using an LSTM neural network in Keras”

<https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5>

5. Nikhil Kotecha, Paul Young, 2018 - “Generating Music using an LSTM Network”. arXiv:1804.07300v1 [cs.SD]

6. Sepp Hochreiter and Jürgen Schmidhuber, “Long Short-term Memory” in Neural Computation 9(8):1735-80 · December 1997.

7. Y. Bengio, P. Simard, P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", IEEE transactions on neural networks, vol. 5, pp. 157-166, 1994.

8. <https://github.com/PalAvik/MusicCompositionLSTM>