SSY340-Deep Machine Learning
Generate Music using a LSTM Neural Network
Group-110

1st Hariharan Gopinath
Systems, Control and Mechatronics
Chalmers University of Technology
Gothenburg, Sweden
hargop@student.chalmers.se

2nd Soundarya Jai Prakash
Computer science – algorithms, languages and logic
Chalmers University of Technology
Gothenburg, Sweden
soua@student.chalmers.se

***Abstract*—Nowadays, deep machine learning seems to be playing a significant role in many aspects of almost every field like image detectors, computer vision, medical and business analytic. But a multitude of the available models has been designed for image processing/ classification or text-processing where mostly a variant of CNN or feed-forward neural network is used to extract more in-depth features. However, there are many other realistic problems, where further research need to be carried out on deep machine learning models to address some prominent issues. One of these domains (which require further consideration) is that of music generation. Traditionally, music was treated as an analogue signal and was generated manually. However, in recent years, music is conspicuous to technology that can generate a suite of music automatically without any human intervention. To accomplish this task, we need to overcome some technical challenges to successfully generate a composition of notes having continuity and unity. This is where we can try for different Deep learning frameworks on sequential inputs other than images.**

***Index Terms*—Music, MIDI, Neural networks, RNN, LSTM.**

I. Introduction

Music has been an integral part of all our lives, be it in the form of art , therapy or just mere passing of time. It's not just one genre of music that people are interested in. With different taste comes different genre of music which has been constantly expanding based on the generation and interest of people. The one thing that has been constant in bringing different groups of people all over the world has been music and therefore there has never been a dead end to this field which has ever made it obsolete. Having said that, there are many of us who love music and use it everyday as a way of communication with people but haven't been blessed enough to compose music of our own. Having come across Deep learning, gave us the opportunity to do the same now. Without knowing how to play certain instruments and also not knowing the theory of music, just using certain frameworks and techniques even we were able to compose music of our own. There is a lot of research done in generating words but not much seems to be done when it comes to automatic music generation. Therefore we chose this topic to bring two our interests of deep learning in the generation of automatic music together.

II. Aim

Having fed music tracks initially by the user, the aim of this project is to generate music which is associated with the notes and chords which were previously fed. A variant of Recurrent Neural networks, Long Short Term Memory Model has been used to capture the long term dependencies in the input sequence which has been fed by the user. The data has been first pre-processed and many of the features of the music data has been picturized in the form of graphs. The training is done on songs or music in the form of MIDI data set. Further the evaluation is done on the audio midi files that has been decoded.

III. Background

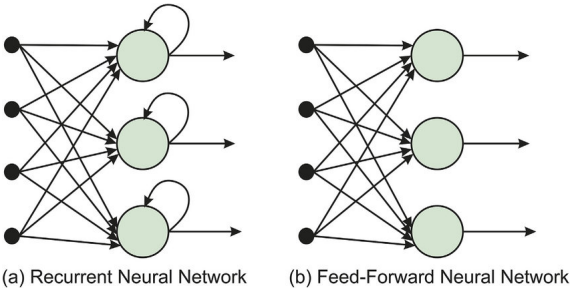

Fig. 1. RNN vs Feedforward NN.

For our problem, choosing a correct model will plays a major role. In the deep learning and NLP task, there are different types of models we can use like feedforward neural networks, Convolutional neural network, recurrent neural network, and the long short-term memory network. In all these model, the group of neurons forms a neural networks and the neurons takes the input, extract the features and gives out the output to the next neurons as shown in figure 1 Feed forward neural networks works well for the various tyeps of application like image classification, detection, etc,. But for the sequence prediction, we can use the feed forward and the Convolutional neural network since those model are do not

consider for previous inputs, which makes it not very useful for the sequence prediction problems. So, the RNN will make it better to work on it.

The Recurrent Neural Networks can remember most of their recent calculations and can use them in addition to an input to generate another new output by using a feedback loop which allows a small memory structure to be added to the model. But the RNN would fail sometimes due to 'Vanishing Gradient' problem. Vanishing gradient is when the gradient of the loss function decays exponentially over time and due to this our model keeps forgetting more and more information faster. The picture of RNN in figure 2

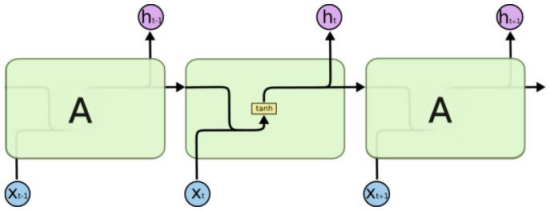

Fig. 2. The repeating module in a standard RNN contains a single layer.

To overcome the above mentioned problem, Long Short Term Memory neural networks was formed to allow the initial inputs to travel further down into the network without losing them inbetween. Here when the network is deciding the output it has to go with for a particular input and also decide which part of the results is less or more important. This is done as they have 'memory cells' added to the structure, which has gates and decides when a particular information enter the memory, or when it exits or what the output could be. The gates here are the input gate, update gate decides the information which can be passed on from the hidden states to the next state and the forget gate lets the information from the previous state pass through if the value is closer to 1 otherwise stops it. The structure of the gates can be seen in figure 4 For updating the parameters in the LSTM activation functions like the sigmoid activation function and the tanh activation functions have been used. The picture of RNN in 3

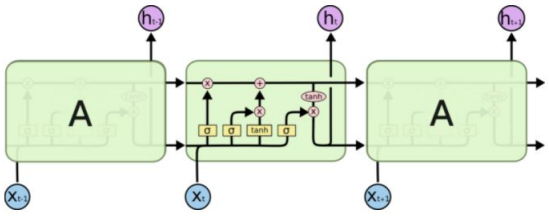

Fig. 3. The repeating module in a LSTM contains four interacting layers

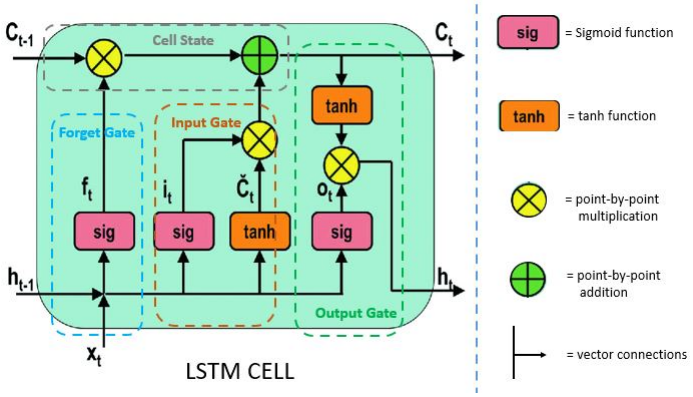

Fig. 4. Gates in LSTM Module

IV. Data collection and preprocessing

A. Data

For the dataset we have used used piano music from the soundtrack of Final fantasy as it seems to have distinct melodies that is present in majority of the music pieces. We are using MIDI dataset here as the other datasets of music like the mp3, au etc are raw audio files with a lot of noise which messes up the pre process. For examining the data we are going to be using here, Music21 a python toolkit which is used to get the musical notations of a MIDI file. To make our own midi files in the final music creation process it helps us by creating the Note and Chord objects from the dataset here. The Note object consists of : 1)Pitch - frequency of the sound represented with the letters [A, B, C, D, E, F, G] 2)Octave - the set of pitches which is used on the piano here 3)Offset - the location of the note in the piece. The Chord objects behaves as a container for the set of notes that is being played at a given time.

B. Data Presentation

Having learnt about data and its insights, we are now going to use it for pre-processing and the training of the network. Here we can also see the pitch characteristic of our input music along with the notes and its frequency. Firstly, Our songs are represented by the piano roll as seen in figure 5. The x-axis of the piano roll represent the time steps and the y axis represent the pitch possibilities for the entire songs. For this, we have used a sampling rate of 16 times per time set.

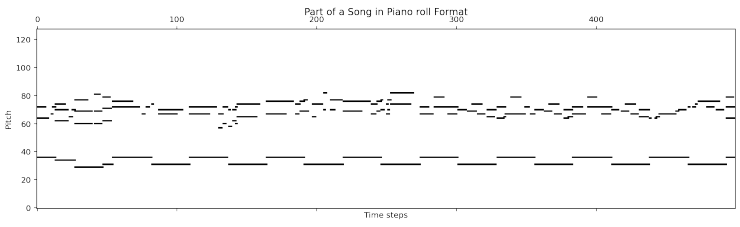

Fig. 5. midi pitch distribution

Secondly, we can represent the number of occurrence or maximum occurrence of the each notes or chords in the whole

piano songs. When we took all the songs in the training set, there are around 350 notes and it is difficult to visualize all of them and therefore we use a some small amount of training songs and check the notes occurrence in these song. This method can be used to compare it with the output song that we are going to generate with the LSTM network.

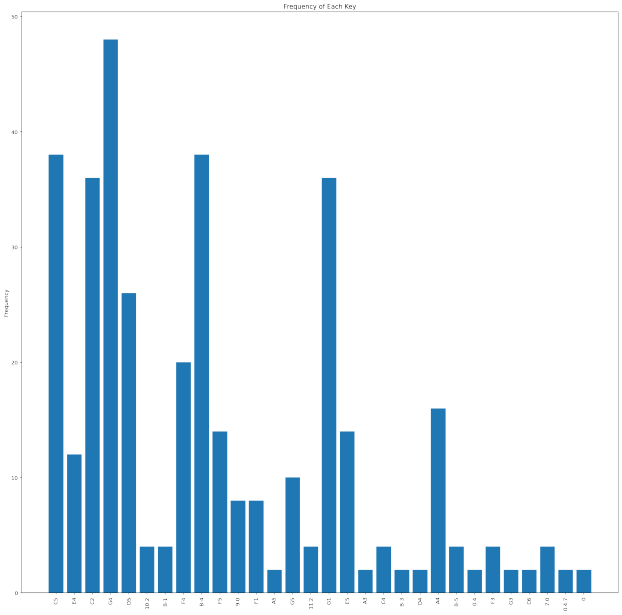

Fig. 6. Notes frequency

From the above figure 6, we can see that notes G1,G4,c5 etc occurs most in the songs.

C. Pre processing

Whenever we want to solve a problem using AL and machine learning, it has to be reduced to a fundamental problem which can be understood by humans and also solvable for the computer. For the image classification and the detection problem, we will reduce the image complexity to a simple form of less channel or readable image. For our problem, music generation, it involves sequence prediction. These sequence predictions are used in many problems like machine translation, speech recognition and NLP. So our problem can be reduced to a sequence prediction task where we are processing and predicting the sequential of music notes.

The Music21 library converts the MIDI files into notes/chords/rest object with the instrument it is associated with and the rest duration. Then an array was created to store the notes/chord/rest combination of all the MIDI songs and then split into a 100 notes sample songs. Then as the LSTM understands numerical data better than any other, the note/chord/rest-duration combination was converted into unique numerical values. For this, we have to create a mapping function which should convert the string based notes and chords to the numerical data.

For the training the 100 notes, sample was fed into the LSTM and the prediction of the next note based on the previous 100 sequence length that we have taken and the note which was got back was compared with what the actual note was supposed to be.

Now, we have to create the input sequence for the network and the output corresponding to that. All the notes are also passed into the mapping functions for each iterations and the one hot encoded output labels are obtained for the input sequence.

V. Model

The picture of the LSTM can be seen below 7

```
The network for training the model is created now and the structure now looks like
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 128)          66560
_________________________________________________________________
dropout (Dropout)            (None, 100, 128)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 128)          131584
_________________________________________________________________
flatten (Flatten)            (None, 12800)             0
_________________________________________________________________
dense (Dense)                (None, 256)               3277056
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 373)               95861
_________________________________________________________________
activation (Activation)      (None, 373)               0
=================================================================
Total params: 3,571,061
Trainable params: 3,571,061
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7. LSTM model structure

As mentioned in the picture above, there are four different types of models used in this project. Firstly, the LSTM layer - This is an RNN layer which returns either a sequence of output or a matrix after taking in a sequence of input. The first LSTM layers dimensions is equal to its networks input size. We have also used a parameter called the input shape here which tells the network about the shape of the data it is going to be training. Followed by the Dropout layer, which is used for regularization. This is done at the training to prevent overfitting as it updates a fraction of the inputs to 0. In our model it is set as 0.3 as it was found to be optimal. Next, we have the fully connected layers also called as the Dense layers which connects every single input node to its corresponding output node. The dimensions of the last dense layer in our project shows us the number of unique output layers required. Lastly, the Activation function layer which determines the activation function used to calculate the desired output. Here in our case we use the Softmax optimisation. Here we make use of the categorical cross entropy loss and pass each iteration through this as every output here belongs to a single class but we are here dealing with atleast two classes. As far as the optimizer is concerned, after trying out a couple of optimizers we settled with the Adam Optimizer as it ended up giving us good results.

A. Training

After the model architecture has been taken care of for our network we now start with the training process. In our case we use the LSTM model from before and use the fit function to train the network. The first two parameter here are the inputs sequences and their respective outputs. We also have used the validation sets during the training process to get the better output and using the validation set, we can prevent the overfitting problem.

We have used a batch size of 128 samples and the training of the network is done for 100 epochs. Model checkpoints is a function which can be used which saves the weights of the network notes after every epoch so that we can stop our training at any point and still not lose all the hours of training the we have done before. This way we can stop the running of the network once we get a satisfying loss.

VI. Music Generation

We use the same model as before that we had used for training and the weights which were saved in the training session is used here.

A. Notes generation:

For the starting point of our note a random number is picked as the starting index from the list of note sequences which we have obtained from before. For different notes we just have to run it over and over again without having to change the starting note manually. To decode the output from numerical data(integers) to categorical data(notes) a mapping function is used. In our project we are generating around 500 notes to give good enough time so that melody is formed without it ending too soon.We know that every note is a sequence of some network. When a note is generated first, for the next sequence of notes following it we use the sequence of notes from the previous one and remove the first note and insert a new output from the iteration at the sequence end.

B. MIDI files generation:

The outputs from the previous function are in the form of an encoded array of Notes and chords and now we need to decode them into Notes and chord objects. If the output we are trying to decode is a Note, the string representation of the pitch in the pattern is used to create a Note object. If its a Chord, it has to be split into an array of notes. We loop through the string representation of each of these notes which is then converted into a Chord object which contains each one of these notes. Here to avoid the notes from stacking we increase the offset by 0.5 and the Note/Chord object which is created is then appended to a list. The MIDI files are then created to contain the music generated by the network.

VII. Results

We have trained the networks for 40 and 100 epochs respectively and we can compare the output of both the models performance. To see the evaluation of a model, we can compare the training and validation accuracy as well as the loss of the both the model and to see how the models has performed in the training. From the graph plotted which is mentioned below, we can see that both the training and the validation accuracy has a significant increase over the 20 epochs and after that, the validation accuracy have remained constant But the training accuracy have increased to more than 90 percentage after the 50 epochs and we have achieved maximum of 97.44% of training accuracy. When it comes to the training and the validation loss, training loss have showed a significant decrease and remained constant without increasing over the course of the 100 epochs but the validation loss have increases little bit when we compare to the training loss.

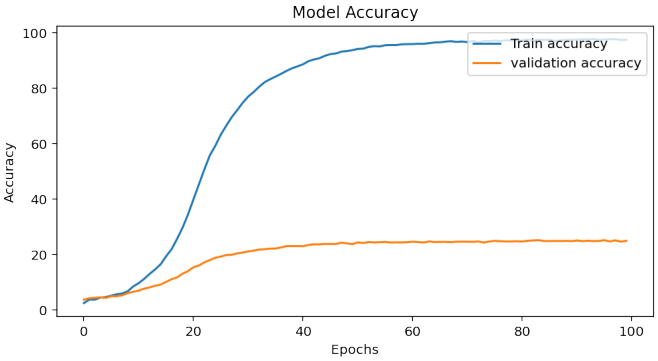

Fig. 8. Accuracy for 100 epochs

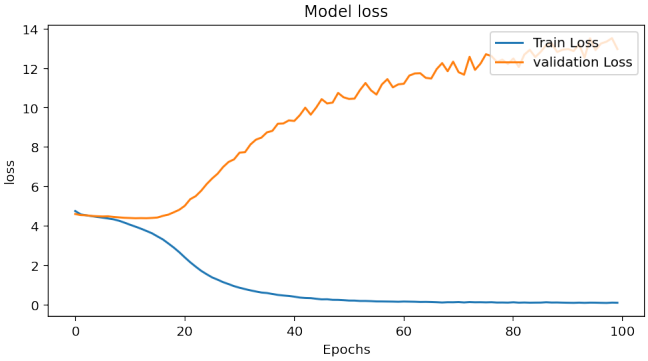

Fig. 9. Loss for 100 epochs

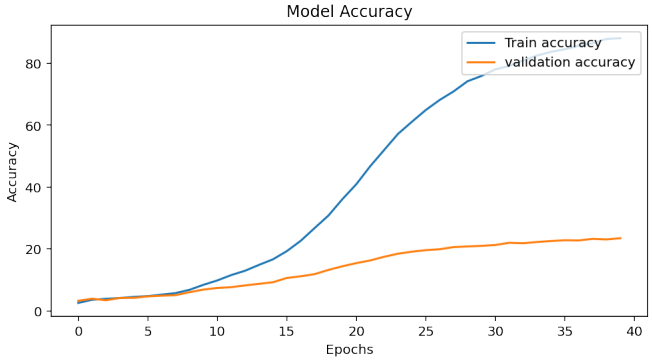

Fig. 10. Accuracy for 40 epochs

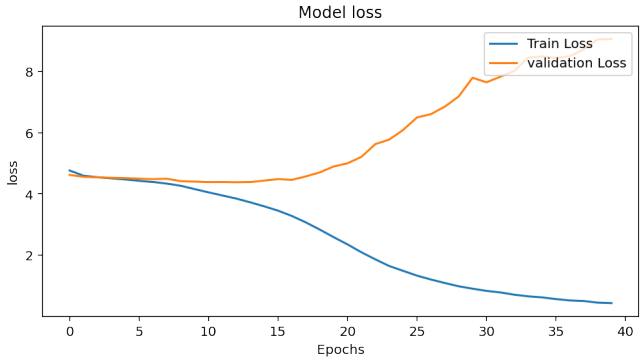

Fig. 11. Loss for 40 epochs

This is not only the best way to tell how good our music generated it and therefore it can be evaluated or validated by making a few people listen to it and take their feedback of generated music. Further more, we thought it would be good if we compare the number of unique pitches in the generated song with the original generated songs.

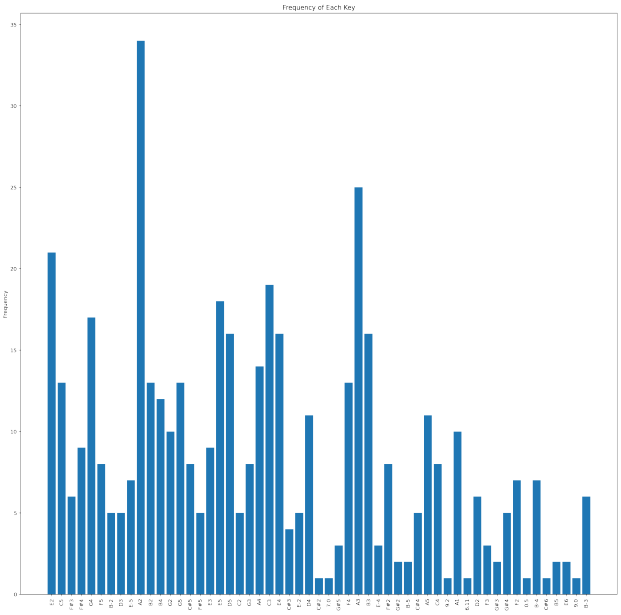

Fig. 12. Notes frequency for 100 epochs

The figure 13 & 14 we can see that both the pitch distribution seems to be almost same and has the pitch around 40-80 and average of 60. When we compare it with the input song pitch figure 5 it has the average pitch of 50-60 pitches. This could be a good evaluation criteria for comparing the song generated by the LSTM networks and the original songs.

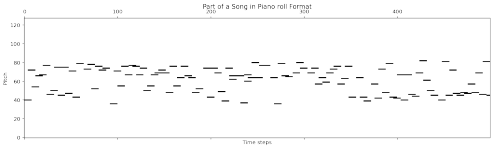

Fig. 13. Pitch distribution for song 100 epochs

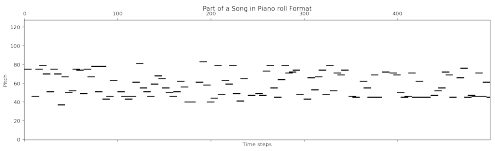

Fig. 14. Pitch distribution for song 40 epochs

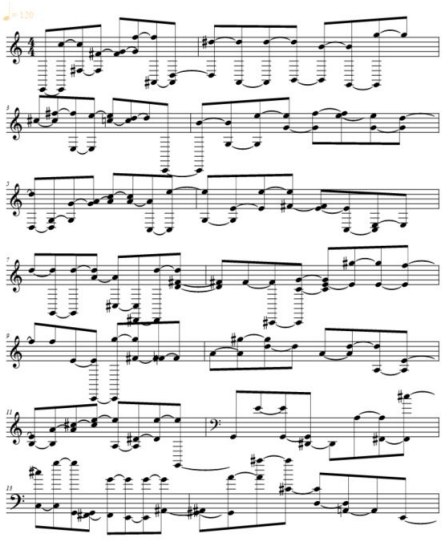

Fig. 15. MIDI output songs for 100 epochs

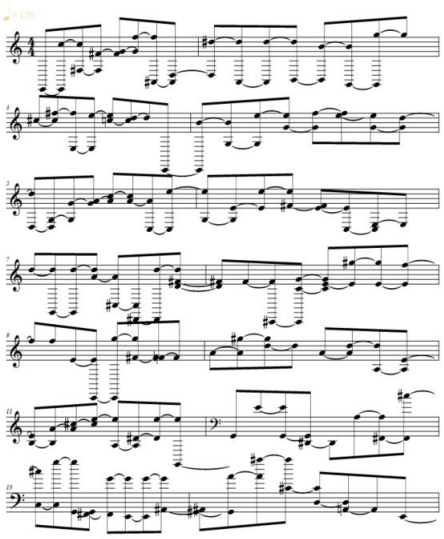

Fig. 16. MIDI output songs for 40 epochs

Figure 15 and 16 contains music sheet representation of music which is generated by our LSTM network. When we see it, it was almost same structure of the some original midi songs and people who is well known about this music sheet and who can read and understand the music sheet can tell how the machine generated songs looks like. Also when comparing the music sheets of 100 and 40 epochs songs, we can see both almost look the same but there are some minor differences in their structure. But when we listen to the songs, we can clearly decide that song generated by 100 epochs sounds good.

VIII. Conclusion

From this project, we have learnt how to produce good music by deep machine learning model like RNN based LSTM by considering the no of songs that we have used for the training and the validation part. However, the results may not be perfect or it contains some offsets from the original songs. The results are impressive and good and the accuracy of 89% has been reached while training. We can further improve it by using a deep LSTM model with more training songs and epochs to produce epic music with many instruments.

IX. Future Work

From all of the above discussion, we can see that the song which is generated by the LSTM network is pretty good and we can hear that good enough even though it does not have the best melody or well structured piano. However, these problems can be improved. The current song does not have variation in the notes and chords but it can be improved when we have more songs in the input and adding different variety of songs, not just piano. Also, if we want to add more classes in the songs, we need to increase the depth of the LSTM model and train it for more epochs but it requires better computational power for training and also take more time. Finally, if we add more instruments to our whole dataset, we can find a interesting song which involves various instruments and various classes.

References

[1] LSTM Based Music Generation System https://arxiv.org/pdf/1908.01080 (2019)

[2] Neural Networks For Music: A Journey Through Its History https://towardsdatascience.com/neural-networks-for-music-a-journey-through-its-history-91f93c3459fb (2018)

[3] Automatic-Music-Generation-System https://github.com/codepradosh/Automatic-Music-Generation-System (2019)

[4] An Improved RNN-LSTM based Novel Approach for Sheet Music Generation https://www.sciencedirect.com/science/article/pii/S1877050920310152 (2020)

[5] Music-generation-using-rnn https://github.com/shubham3121/music-generation-using-rnn (2017)

[6] From scratch-An LSTM model to predict commodity prices https://medium.com/@vinayarun/from-scratch-an-lstm-model-to-predict-commodity-prices-179e12445c5a (2018)

[7] Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks https://arxiv.org/abs/1909.09586v1 (2019)

[8] Generating Original Classical Music with an LSTM Neural Network and Attention https://medium.com/@alexissa122/generating-original-classical-music-with-an-lstm-neural-network-and-attention-abf03f9ddcb4 (2019)