Musical Notes Generation

A.N. Vaishnavi, Sree Vidya, G Keerti, Deeksha Nayab and Sai Sree Nithya
UnderGraduates at IIIT SriCity

***Abstract*—A unique way of musical notes generation based on neural networks is put forward in this paper.It works by taking existing notes or music as a raw data on which required procedure is performed in order to generate a soothing and pleasant music. This paper proposes the procedure that can be applied to generate music efficiently using modern methods of stylizing the notes. The architecture to the above is illustrated in the subsequent paper.**

***Index Terms*—rnn, lstm, midi, abc notation,**

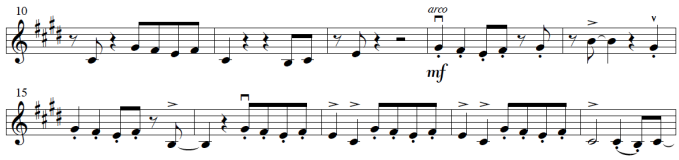

Fig. 1. Sheet Music

I. Introduction

For many artistic purposes skilled musicians use traditional media or modern computer tools to create a variety of expressive styles that are very appealing but problem occurs when he has reached his bottleneck. This paper aims to aid those who intend to produce high quality music which is pleasing to the ears without putting in much effort. Fine details is particularly important feature of music generation, yet it is extremely time consuming. The existing literature includes some algorithmic techniques that are able to create fine music.

It focuses on generating music automatically using Recurrent Neural Network(RNN).

It is not necessary to be a music expert in order to generate music. Even a non expert can generate a decent quality music using RNN. Listening to interesting music and if there is some way to generate music automatically, particularly decent quality music then it's a big leap in the world of music industry.

Task: The main motto is to take some existing music data then train a model using this existing data. The model learns the patterns in music that one enjoys. Once it learns this, the model generates new music for an individual according to his/her interest. Unlike brute method where simply copy-pasting works here it understands the patterns of music to generate new music.

The intuition is to generate a decent music but firstly what is music? In short music is nothing but a sequence of musical notes.The input to the model is a sequence of musical events/notes/raw music file. The output will be new sequence of musical events/notes/music. In this case-study it is limited to a single instrument music. In future, extending this to multiple instrument music looks quite promising and appealing as it will be a breakthrough in the music industry.

II. Problem Statement

Pertaining to the facts that many artists till date use traditional methods to generate a decent music. Although the output is of great quality but the main problem lies in the amount of time it takes to generate it. Moreover if an artist starts off with an amazing notes and reaches his bottleneck in due course. It can be extremely tiring and frustrating.

The main motivation behind this paper is to aid those who are challenged by such problems. With the use of present day modern techniques the quality of the music can be shaped. Key task is to generate music as a sequence of events using RNN which takes sequence as an input. Music is represented by a sequence of musical notes. Each musical note is separated by a space. This can be used to represent both single instrument and multi instrument music.

The challenges can be to train the models. In the ever changing world there is always a possibility of new evolution. Hence, for such a model generating the music from scratch seems troublesome. Existing models have been trained over almost all existing tunes but one never knows when a new tune or music can shoot out. The solution proposed is to take in existing notes/music as an input and train them along with the new ones in such a way that it adapts to the existing ones.

Ones interests can vary hugely when it comes to music. Some may like pop music while others may like soothing music which have calming effects. Taking the raw music as an input(be it midi or mp3) and generating a new music of the same genre.

Data-set that comes into picture is the Nottingham Data-set which utilises the ABC notation for music generation.

NMD is thus a personal collection of an English folk music enthusiast and computer scientist with decades of experience in playing and dancing to this kind of music. Much of the collection is focused on dance music (jigs, reels, hornpipes, waltzes, slip jigs, Morris, Playford's), but some of it is specialised (Miscellaneous, Christmas).NMD is his collection of rough transcriptions of tunes that should never be performed as written, but when performed well should make "folks want to dance".

As observed in Fig 2 it has two parts:

Part-1 represents the meta data. Lines in the Part-1 of the tune notation, beginning with a letter followed by a colon, indicate various aspects of the tune such as the index, when there are more than one tune in a file (X:), the title (T:), the

```
X: 1
T: The Kesh
R: jig
M: 6/8
L: 1/8
K: Gmaj
|:G3 GAB| A3 ABd|edd gdd|edB dBA|
GAG GAB|ABA ABd|edd gdd|BAF G3:|
B2B d2d|ege dBA|B2B dBG|ABA AGA|
BAB d^cd|ege dBd|gfg aga| bgg g3:|
```

Fig. 2. ABC Notation

time signature (M:), the default note length (L:), the type of tune (R:) and the key (K:).

Part-2 represents the tune, which is a sequence of characters where each character represents some musical note.

MIDI(Musical Instruments Digital Interface):

To think that MIDI itself makes sound is baseless, it is just a series of messages like "note on," "note off," "note/pitch," "pitch-bend," and many more. These messages are interpreted by a MIDI instrument to produce sound. A MIDI instrument can be a piece of hardware (electronic keyboard, synthesizer) or part of a software environment (ableton, garageband, digital performer, logic and so on).

III. Proposed Methodology

A. Maintaining the Integrity of the architecture

Data Handling:

Feeding the data into batches. Next feed the batch of sequences at once into the RNN model being used. First we have to construct our batches The set contains the following parameters:

Batch Size = 16

Sequence Length = 64

There are total of 154399 characters in our data. Total number of unique characters are 87. Next a numerical index is assigned to each unique character giving a dictionary where key belongs to a character and its value is it's index.Similarly the opposite of it, where key belongs to index and its value is it's character is also created.

Char-RNN Model:

Music is a sequence of characters so the best and obvious choice will be RNN or variations of RNN like LSTMs or GRUs which can process sequence information accurately by understanding the patterns in the input.

A special type of RNN called char RNN is feeded with the sequence of notes character by character. The output will be the next character in the sequence. Therefore, the number of output will be equal to the number of inputs when fed to a single neuron. Since this is the case Many-to-Many RNN is used, where number of output is equal to the number of input.

	Batch-1	Batch-2	...	Batch-150	Batch-151
0	0...63	64...127	...	9536...9599	9600...9663
1	9701...9764	9765...9829	...	19237...19300	19301...19364
.	...	...	...	...	...
.	...	...	...	...	...
.	...	...	...	...	...
14	135814...135877	135878...135941	...	145350...145413	145414...145477
15	145515...145578	145579...145642	...	155051...155114	155115...155178

Fig. 3. Batch construction: The dataset contains 87 unique characters which are assigned to indices.

In Fig 4 the image has many-to-many RNN with equal number of inputs and outputs are shown on right-side which is the bottom layer. Here, middle is RNN unit is a repeating structure.

In Fig 5, we consider a single character at time 't' as an input to RNN unit. The output of the previous time 't-1' character is given as an input to RNN at time 't'. It then works towards generating output 'ht'. This output 'ht' is again feeded as input to RNN as 'Ot-1' in the next time step.

This output 'ht' will be a next character in sequence. Our music is represented as [a, b, c, a, d, f, e,...]. Now, giving first character 'a' as an input the RNN is expects to generate 'b' as an output. Then in next time-step when given 'b' as an input and 'c' as an output is expected. Then in next time-step 'c' is given as an input and 'a' as an output is expected and so on. In such a way the RNN is trained so that it should output generate the next character in the sequence. This is how it will learn whole sequence and generate new sequence on its own.

This keeps going until all of the inputs are feeded. Since, the music is a combination of many characters and the output is one of those characters it can be thought of as multi-class classification problem. Here, "Categorical Cross-Entropy" is used as a loss function which is also known as "Multi-Class log-loss". The last layer contains "Softmax" activations. The number of "Softmax" activation units in last layer will be equal to the number of all unique characters in all of the music in train data. Each RNN can be a LSTM which contains 'tanh' activation unit at — input-gate — which is a differentiable function. Therefore, this RNN structure can be trained using back-propagation and on iterating it using "Adam" optimizer it will converge. At the end, the RNN will be able to learn sequence and patterns of all the musical notes that are given to it as input during training.

After all this is done the thing that is left is how to generate new music using the present char RNN. Once the char RNN model is trained, it will give any one random character — from the set of unique characters that is feeded to the char RNN during training time — to the trained char RNN, it will then generate characters automatically which will be based on the sequences and patterns that it has learnt during training phase.

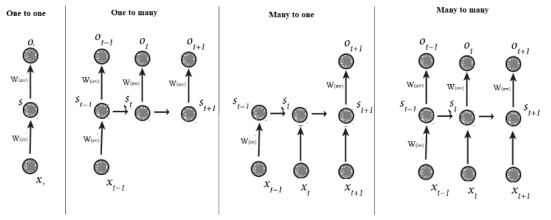

Fig. 4. Variance of fully connected RNN

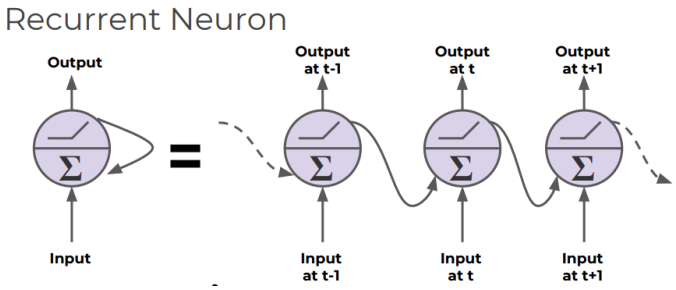

Fig. 5. Recurrent Neuron

In Fig 5 we have Many-to-Many RNN where 'Ci' is a first character input that is given to RNN unit. For first RNN we provide a zero input which is nothing nothing but a dummy input because RNN always takes current input and previous output as input. Since, previous output isn't available the input for the first iteration zero. Now, the requirement for RNN is to produce next character as output. So, the output of 'Ci' will be 'Ci+1' which is nothing but a next character in sequence. Now, our next input itself is a next character, and at the same time the output of previous input is also next character so we feed both of them to next RNN unit again and produces 'Ci+2' as output which is next character in the sequence. This is how Many-to-Many RNN works. Note, the above image shows time-unwrapping of RNN. It is only one RNN unit which repeats itself at every time-step.

Fig 6 shows 'n' RNN units in single RNN layer. The requirment is a 256 LSTM-RNN Units in one layer of RNN. At each time step all of the RNN units generate output which

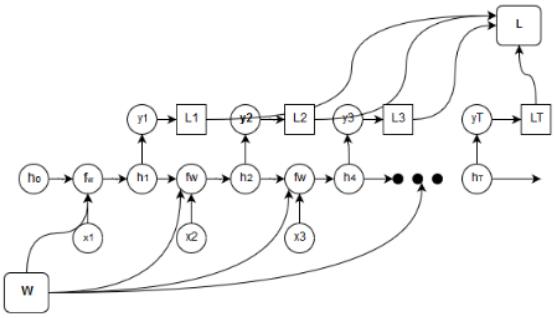

Fig. 6. Many to Many RNN single layer with n RNN units

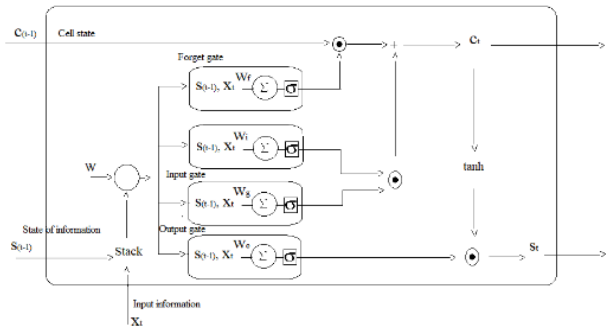

Fig. 7. LSTM neuron core

will be an input to the next layer and also the same output will again be any input to the same RNN unit. Here, in the paper each RNN unit is considered as an LSTM unit. In Keras library, in LSTM there is a parameter called 'returnsequences' which is False by default. But if it is True, then each RNN unit will generate output for each character means at each time step. This is what is expected here. What is required is to generate output as the next character when given input as previous character in the sequence.Stacked up many LSTM units here with each unit learning different aspect of the sequence and creating a more robust model overall is the ultimate goal.

After three such layers of RNN, it is required to apply 'TimeDistributed' dense layers with "Softmax" activations in it. This wrapper applies a layer to every temporal slice of an input. Since the shape of each output after third LSTM layer is (16*64*256).87 unique characters are there in the dataset and the output at each time-stamp will be a next character in the sequence which is one of the 87 characters. So, time-distributed dense layer contains 87 "Softmax" activations and it creates a dense connection at each time-stamp. Finally, it will generate 87 dimensional output at each time-stamp which will be equivalent to 87 probability values. It helps to maintain Many-to-Many relationship.

There is one more parameter in LSTM which is known as "stateful"."stateful = True" is a must here. If True, then the last state for each sample at index 'i' in a batch will be used as initial state for the sample of index 'i' in the batch. This is used because all of the batches contains rows in continuation. So, if the last state of a batch is feeded as initial state in the next batch then our model will learn more longer sequences.

IV. Results and Analysis

The trained model and weights are used for making predictions. In order to make prediction, any of the 87 unique characters is given as input to the model and finally it will generate 87 probability values through softmax layer. From these returned 87 probability values, it is required to choose the next character probabilistically and not deterministically and finally the chosen character is again feeded back to the model and so on. Repetetive concatenation of the output character

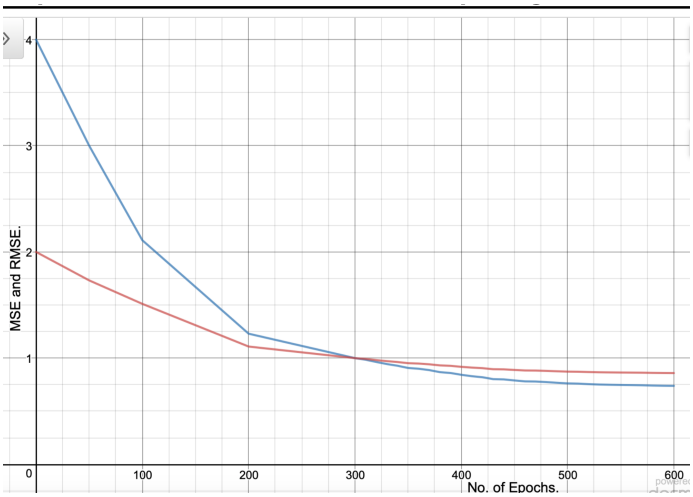

Fig. 8. Graph for testing data

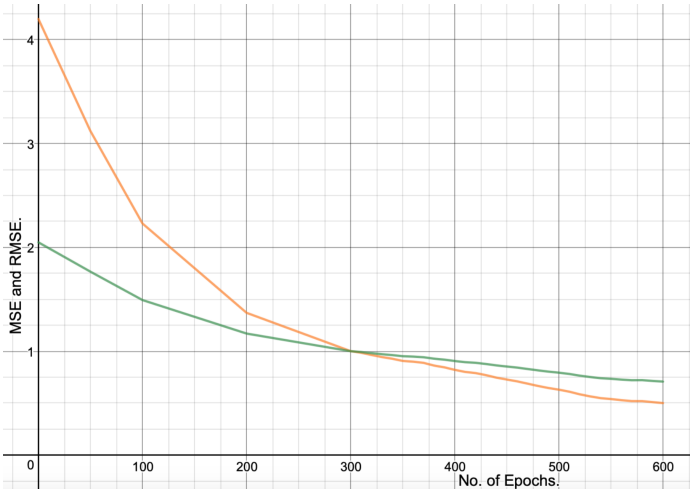

Fig. 9. Graph for training data

generates music of some length. This is how music will be generated.

V. CONCLUSION, FUTURE DIRECTION AND DISCUSSION

We have generated a good quality music. We trained in such a way that starting and ending music can be added in every new generated tune to give a tune a better start and better ending. Doing so, our generated music becomes melodious. We trained the model with more than 1000 tunes. By training the model with more musical tunes, our model not only is exposed to more variety of music but the number of classes also increased. By this more melodious and at the same time more variety of music is generated through the model.

Scope of the paper can be extended by training the model with multi-instrument tunes. It would be interesting to listen what music the model will produce if it is trained on multi-instrument music. Finally, a method can be added into the model which can handle unknown notes in the music. By filtering unknown notes and replacing them with known notes, model can generate more robust quality music. In the course of this work, our goal was to provide adequate solutions to each of the constituent sub-problems. In no case would we claim that these solutions are optimal; there are considerable opportunities to push the work further, improving the quality of the result, the ease of use, or reducing the resources required.

REFERENCES

[1] https://ieeexplore.ieee.org/document/8398657
[2] https://towardsdatascience.com/generate-piano-instrumental-music-by-using-deep-learning-80ac35cdbd2e
[3] https://ieeexplore.ieee.org/document/5584069
[4] https://www.tandfonline.com/doi/full/10.1080/25765299.2019.1649972
[5] https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5
[6] https://cs224d.stanford.edu/reports/allenh.pdf
[7] https://www.twilio.com/blog/generate-music-python-neural-networks-magenta-tensorflow