Music Generation using Recurrent Neural Networks

A Comparative Study using SRNs, LSTMs and GRUs character level predictors

Chetan Chawla
Electronics and Communication Department
Bharati Vidyapeeth's College Of Engg.
New Delhi, India
chchawla@yahoo.co.in

***Abstract*- Music is rightfully defined as the god's own language. With advancements in technology, it is imperial that music must also be revolutionized. But the beauty of music lies in its essence of uniqueness. In this project, we find ways to incorporate Recurrent Neural Networks with Long Short Term Memory Cells to form a generative neural network model working on character level predictions to compose a sequence of music. We show that even after the degree of unpredictability, the magical working of RNNs [1] gets testing accuracies as high as 54% and works to generate musically sound patterns of notes. We present a comparative study using various generations of evolving indifferent types of Neural Networks-SRNs(Simple Recurrent Networks), LSTMs and GRUs(Gated Recurrent Units). We use ABC text notations for music representation input to our generative model.**

Keywords—LSTMs,SRNs,GRUs,Generative Neural Networks, character level predictors

I. INTRODUCTION

Piano is a very versatile instrument in the fact that we can create infinite number of sequences based on different timings-pauses and lengths, polyphony, and combinations of the same 7 notes repeated at different frequencies and pitches to produce something blissful. However, the 10 fingers of our hands can only experience the extent of composition which is allowed by our physical and physiological constraints. This thought as well as the inertia of human behavior is the key factor in the origination of computerized compositions which could expand dimensionality of music, keeping intact its melody, or even enhancing it to new heights.

The first works of music generation were done in the 18th century when Musikalisches Würfelspiel (Musical Dice game) was played using a dice which used to probabilistically give the musician next random notes to play, thus generating random sequences. This however relied only on random guesses. Since Leonard Bernstein's Lecture on music and language in 1973 at Harvard, the community has progressed towards a perception of musical grammar as a set of language attributes governing the music. Conventional methods use human perception, emotional behavior and state of mind as the attributes towards making the base of what is to be produced as music. These systems are very strict natured and doesn't show any variation whatsoever outside the scope of this grammar model of generation. The other conventional type of music generation is the knowledge base music generation approach which, in new terms, makes use of basic Artificial Neural Networks to predict each step's prediction based on the data. However, this model is not able to compose a variation of what has already been described in its dataset. Evolutionary method makes use of Elman Neural Networks or SRNs[2] which learn the music but on a short-short term memory basis.

In this report, we try to analyze the learning based approach for making the computer learn how to compose music by studying, analyzing and adjusting the weights (parameters) of the neural network in order to produce music in a fashion similar to the ABC notation input text to it, changing the hyper parameters to analyze the fashion in which the model changes the way it works. For doing this, we use two different datasets, both in ABC notation and three different types of architecture components (SRNs, LSTMs and GRUs), each with different arrangements of hidden layers and stacking of these components. We do not put any constraint on the pitch or rhythms.

The rest of the report is organized as follows- Related Work talks about various implementations and advancements in the field, Model, dataset and metrics are internally divided in architecture and discuss each aspect correspondingly. The results section includes the comparative analysis of each produced music composition followed by conclusion.

II. Related Work

The oldest works of deep learning with music generation was done by Chen et al [3] in his paper which explained a way to generate sheet music using evolution based Simple Recurrent Networks. It took into consideration the concept of musical grammar and attempted to simulate Bela Bartok's musical grammar by constrained generation of music according to the rules. The network had only one hidden layer of RNN. It used a notation of its own rather than a standard notation and thus used relative pitch and duration. They used a genetic algorithm to guide their neural network, called SANE (Symbiotic Adaptive Neuro-Evolution). The main disadvantages of the outcomes were usage of only 5 rhythmic notations, a shortened 3 octave pitch and no use of dotted notes, chords and rests.

To generate polyphonic music, approaches such as RTRBM (Recurrent Temporal Restricted Boltzmann Machine) are used to learn both harmony and melody in an unconstrained manner represent the state of art results. These models allow representation of complex data distribution over each time step rather than single data points at each interval. This allows very wide use of polyphony as explained in [4]. In 2002, Eck et al[5] used two different LSTMs in parallel, one for sequence and chord generation and the other controlling the divergence of chord and melody from each other.

We do a comparative analysis to the research paper by Allen [6] which uses a 2-layered LSTM Recurrent Neural Network architecture and use it to make a character level predictor which generates the sequence and thus the next note in the composition. The data is taken in Piano Roll Format in this paper which is a time signature reference of different notes in a music sheet. They create an embedding matrix for every token, thus forming a concatenated embedding matrix list for the time sequences which are used as inputs to the LSTM. The output of LSTM is fed into softmax and the loss is cross entropy. In this way, they generate new music in piano roll format. They do so for the chords as well and show that they attained comparable sophisticated results as that of RTRBM.

III. Architecture

We compare several architectures. The observations of all such architectures are compiled in the results section. In this section, the best result bearing architecture is shown as a 2 layered LSTM model.

A. Data

We use two different datasets open sourced over the net, one being the popular Nottingham music database* while the other one being 4 times as complex and larger as the Nottingham database**. The Nottingham database consists of 340 folk songs encoded in ABC format notations having total 125 thousand characters. The letters A through G in the ABC format represents different notes. Quotation marks with a character represent the need of an accompanying chord with the note. The output of the Model is also in ABC notation, hence a python backed open source software, EasyABC, was used to convert the generated notation to midi files. Figure 1 shows a glimpse of how the data in ABC notation looks like.

```
X: 3
T:The American Dwarf
% Nottingham Music Database
S:FTB, via EF
M:6/8
K:D
P:A
A|"D" def fed|"G" BdB AFD| "D"DFA "G"B2 A|"Em" cee "A7" e2 A|
"D" def fed|"G" BdB "D"AFD|"D" DFA "G"B2 A|"A7" Add "D" d2:|
"B"e|"D"fga agf|"G" gab "A7"bag|"D"fga "D"agf|"Em" gfg "A7"e2 g|
"D"fga agf|"G"gab "A7"bag|"D" fga "A7"efg|"D" fdd d2 :|
```

Fig1: ABC notation example from Nottingham Database

B. Model

We build a two-layered LSTM model. The input layer provides the input character to the first LSTM. This LSTM has 128 (or 256) hidden layers and thus unrolls into these many linear layers having a cell state and each sequence is returned and fed into the next layer of LSTM after a dropout of 0.1, which further has 128 (or 256) hidden layers to learn the timed musical sequences. The outputs from this are again dropped at 0.1 probability and fed to the output layer which is a Dense Fully Connected Layer having N number(character vocabulary in the database) of neurons present in it. Further, softmax activation is applied to all the outputs which gives us the probability of each note that might appear next. Categorical Cross Entropy is used as the determining factor for the learning optimizer Adam. Figure 2 explains the architecture of the model we choose.

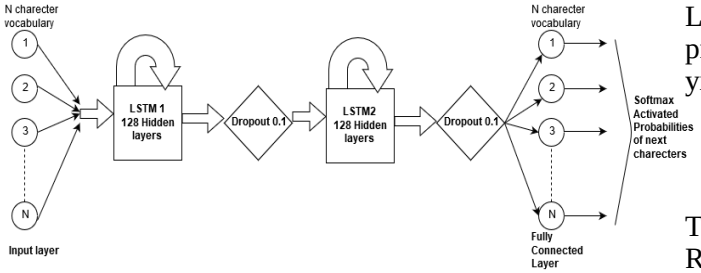

Fig 2: Model Architecture representation.

The model is built and run in Keras library of python using TensorFlow in the backend and is run on an i5-6th generation CPU, which is considerably under performing as compared to a GPU. The model for Nottingham database can be summarized as follows

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, None, 128)	111104
dropout_1 (Dropout)	(None, None, 128)	0
lstm_2 (LSTM)	(None, 128)	131584
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 88)	11352
activation_1 (Activation)	(None, 88)	0

Total params: 254,040
Trainable params: 254,040
Non-trainable params: 0

C. Working

We use this Long Short Term Memory network to train our parameters. The model is character based prediction model. Each character from the ABC notation of a song is first converted into an indexed form where each index represents a distinct character. These indices are then converted into one hot vectors which represent the subsequent characters. One character is fed from the notation to the LSTM network at time T. The model generates an Output matrix of dimensions of vocabulary of characters. This output as T+1 time is converted to a one hot representation and is checked for quality with the next character in the sequence, which acts as the target for the network. The weights are back-propagated to minimize categorical cross entropy. This run for the whole database defines the training procedure.

For generation, an initial character, known as seed, is given to the network at time T. The network produces an output character based on what it has learned at time T+1. This output is again fed back to the input of the LSTM network, thus forming a cycle. And hence, this procedure is repeated for M (900) number of times to yield a sequence of desired length.

IV. Results

We studied the results over a variety of architectures. The two major factors changed were the stacking of RNNs and the types of RNNs. We note that Non-stacked RNNs tended to perform poorer in complexity of time sequences as compared to stacked RNNs. This is due to the fact that different layers in the architecture correspond to a deeper model and more time stamps of learning in the model. Also, Simple Recurrent Networks gave relatively simpler tones while GRU and LSTMs were almost comparative in every aspect. As per the validation accuracies, LSTMs outperformed GRUs.

A. Simple Neural Networks

Hidden layers= 128
Database=Nottingham
Drop out=0.1
Batch size=50
Optimizer = Adam
loss=categorical_crossentropy
metrics=accuracy
final activation=softmax
Internal activations = tanh
Used 2 simpleRNNs and a fully connected layer
Prime length=25 and generation length=900

Metrics-

Epochs	100	150
Training Accuracy	0.9314	0.9312
Validation Accuracy	0.532	0.4962
Comments	Unreadable data	Readable and melodious data. Non Complex

B. Gated Recurrent Units

The GRUs far outperformed SRNs in complexity, but they required more time as compared to SRNs They showed readable data in 100 epochs but the output was not particularly melodious. The output began getting melodious only after 200 epochs. However it still required minor bug fixes to make the audio playable. The maximum training accuracy achieved was 0.9902 and validation accuracy was 0.54.

C. Long Short Time Memory cells

1) With single layer-

The LSTM with single layer performed well in way of generating music but it had erratic data and

very negligible time complexity. We used 256 hidden layers in one LSTM to compare it to 128 hidden layered and two layered LSTM. Maximum training accuracy was 0.9970 and validation accuracy was 0.52. It outperformed others in case of small number of epochs. But as the complexity of produced music increases, it produces non melodic music.

2) With two layers

LSTMs with internal *tan-h* activation used two layers stacking with 128 hidden layers as well as 256 hidden layers. While in 256 hidden layered network, the complexity was far greater than the 128 one, but the musical tenancy of the 256 hidden layer was worse off. The notes and timings produced were very good but too mechanical to be pleasing to the ear. Best results were found on the second database which was far more complex using 128 hidden layers, 2-layer stacked LSTM model, ironically, with the least training and validation accuracy (0.9788 and 0.429) as compared to its counter parts.

The complete set of results could be found and studied at ****.

CONCLUSION

The use of LSTM networks in a stacked manner with ABC notations of music sheets provided very good generative results which are musically sound and unconstrained in pitch and produced eight octaves of music. It produced polyphonic music with only one LSTM model. The results were comparable to any model that has been developed on the basis of musical appetite. Future work would include working on the far simpler Piano roll format and would work on improving the musical appetite further to create songs by multiple intruments.

REFERENCES

[1] Andrej Karpathy, "The Unreasonable effectiveness of Recurrent Neural Networks", http://karpathy.github.io/2015/05/21/rnn-effectiveness/ May 2015

[2] T Mikolov, M Karafiát, L Burget, J Cerno, "Recurrent neural network based language model", Interspeech, 2010 - fit.vutbr.cz

[3] Chun-Chi J. Chen and Risto Miikkulainen, "Creating melodies with evolving recurrent neural networks", Proceedings of the 2001 International Joint Conference on Neural Networks, 2001.

[4] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription", Proceedings of the 29th International Conference on Machine Learning, (29), 2012.

[5] Douglas Eck and Jurgen Schmidhuber, "A first look at music composition using lstm recurrent neural networks", Technical Report No. IDSIA-07-02, 2002.

[6] A Huang, R Wu, "Deep learning for music", arXiv preprintarXiv:1606.04930,2016

[7] * The ABC Music project - The Nottingham Music Database : http://abc.sourceforge.net/NMD/jigs.txt

[8] **https://github.com/saketsharmabmb/Music-Generation-Character-level-RNN/tree/master/data

[9] ***http://abcnotation.com/software

[10] ****https://docs.google.com/document/d/1bgggdsoTreNfru06Wz3tIgXXh7_edNPt1-o-fE_WSuc/edit?usp=sharing