Music Generation With Melody Recurrent Neural Networks And Comparison of Basic, Lookback And Attention RNN Models

Amita Yadav
Dept of Computer Science and Engineering
Maharaja Surajmal Institute of Technology New Delhi, India

Abstract- *A Recurrent Neural Network is a type of network which combines different units to form a directed cycle. RNNs can use their internal memory to process sequence of inputs. This makes their application in areas such as handwriting recognition and speech recognition. In this paper, we are generating music with the help of Melody RNN model which generates a new stream of music every time from existing tones using machine learning. We are experimenting with different types of configurations of Melody RNN model such as basic RNN, attention RNN and look back RNN on the basis of various factors such as perplexity, accuracy, input sequence, loss etc to check the efficiency of each type in Tensor board.*

Keywords- Recurrent Neural Networks, Melody RNN, Tensor board, perplexity.

I. INTRODUCTION

A Recurrent neural network is a network that captures the previous state of information to process new step. It makes use of sequential information. RNNs perform same task for every element in the sequence so they are known as recurrent.

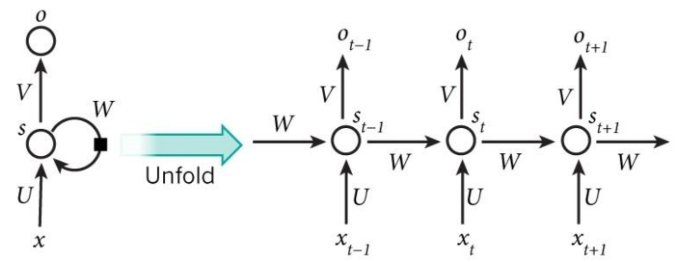


Figure1. A Recurrent Neural Network

Figure 1 depicts a recurrent neural network unfolding into a full network.

Recurrent networks are distinguished from feedforward networks by that feedback loop, ingesting their own outputs moment after moment as input. Recurrent networks are said to have memory, which serves a specific purpose: There is information in the sequence itself, and recurrent networks use it to perform tasks that feedforward networks can't[1,5]. That sequential information is preserved in the recurrent network's hidden state, which manages to create many steps as it cascades forward to affect the processing of each new example.

Over the years researchers have developed more sophisticated types of RNNs:

Bidirectional RNNs are based on the idea that the output at time t may not only depend on the previous elements in the sequence, but also future elements. For example, to predict a missing word in a sequence you want to look at both the left and the right context. Bidirectional RNNs are quite simple. They are just two RNNs stacked on top of each other[2]. The output is then computed based on the hidden state of both RNNs.

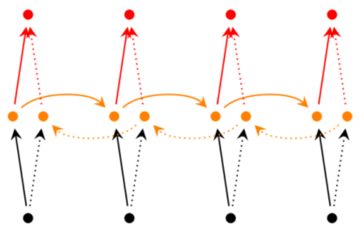


Figure 2. A BiDirectional Recurrent Neural Network

Figure 2 depicts a bidirectional recurrent neural network unfolding into a full network.

Deep (Bidirectional) RNNs are similar to Bidirectional RNNs, only that we now have multiple layers per time step. In practice this gives us a higher learning capacity (but we also need a lot of training data).

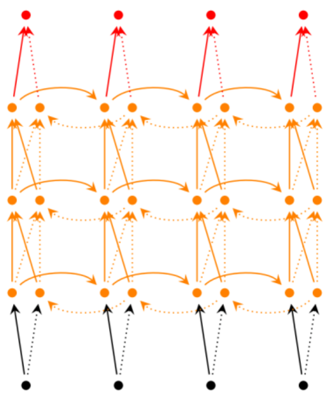


Figure 3. A Deep BiDirectional Recurrent Neural Network
Figure 3 depicts a deep bidirectional recurrent neural network unfolding into a full network.

LSTM networks are quite popular these days. LSTMs don't have a fundamentally different architecture from RNNs, but they use a different function to compute the hidden state. The memory in LSTMs are called cells and you can think of them as black boxes that take as input the previous state h_{t-1} and current input x_t. Internally these cells decide what to keep in (and what to erase from) memory. They then combine the previous state, the current memory, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies.

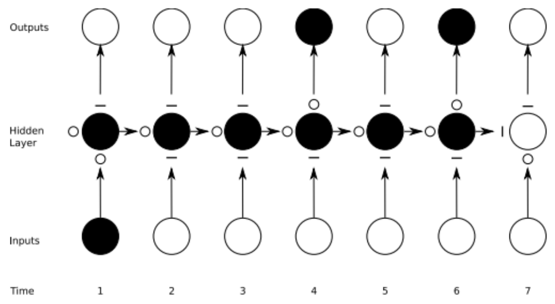


Figure 4. A Recurrent Neural Network with hidden layer displaying processing.

Figure 3 depicts that while feedforward networks map one input to one output, recurrent nets can map one to many, as above (one image to many words in a caption), many to many (translation), or many to one (classifying a voice).

II. RESEARCH OBJECTIVE

In this paper, we are generating music with the help of Melody RNN model which generates a new stream of music every time from existing tones using machine learning[7]. We are experimenting with different types of configurations of Melody RNN model such as basic RNN, attention RNN and lookback RNN on the basis of various factors such as perplexity,accuracy,input sequence, loss etc to check the efficiency of each type in Tensor board.

III. MELODY RNN MODEL

Melody RNN model applies language modeling to melody generation using an LSTM. It has three configurations which are used to generate melody:

- **BasicRNN** – This configuration acts as a baseline for melody generation with an LSTM model. It uses basic one-hot encoding to represent extracted melodies as input to the LSTM.
- **Attention RNN**-Attention allows the model to more easily access past information without having to store that information in the RNN cell's state[6]. This allows the model to more easily learn longer term dependencies, and results in melodies that have longer arching themes.At this point, Magenta can only generate a single stream of notes. Efforts have been made to combine the generated melodies with drums and guitars – but based on human input, as of yet.
- **LookbackRNN -**Lookback RNN introduces custom inputs and labels.The custom inputs allow the model to more easily recognize patterns that occur across 1 and 2 bars. They also help the model recognize patterns related to an events position within the measure[3].The custom labels reduce the amount of information that the RNN's cell state has to remember by allowing the model to more easily repeat events from 1 and 2 bars ago. This results in melodies that wander less and have a more musical structure.

IV. GENERATING A MELODY

A melody is generated by training the melody RNN model with the help of MIDI (Musical Instrument Digital Interface) files. Generating a melody is done using following steps:

1. Convert a collection of MIDI files into Notesequences.
2. Sequence examples are fed to the model after or during training.
3. Training of the model will be done using sequence examples and number of steps.
4. After training, evaluation is done.

5. Both training and evaluation parameters are seen on tensorboard.

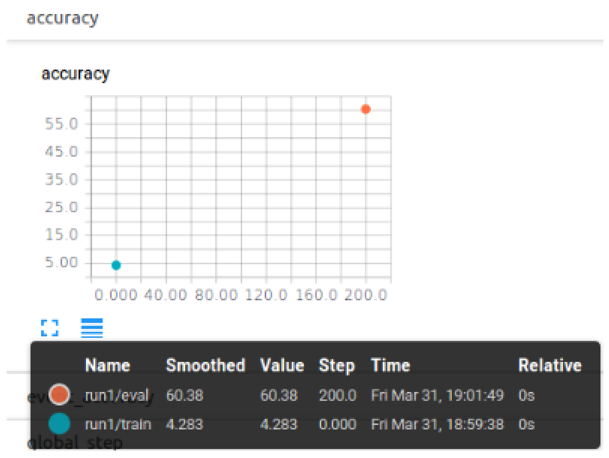
Our first step will be to convert a collection of MIDI files into Note Sequences. Note Sequences are protocol buffers, which is a fast and efficient data format, and easier to work with than MIDI files. Sequence Examples are fed into the model during training and evaluation. Each Sequence Example will contain a sequence of inputs and a sequence of labels that represent a melody[3,8]. Two collections of Sequence Examples will be generated, one for training, and one for evaluation, where the fraction of Sequence Examples in the evaluation set is determined by --eval_ratio. Tensorboard is being run to view the training and evaluation data[4]. Melodies can be generated during or after training.

V. SIMULATION AND RESULTS

The results are obtained by using melody RNN model's configurations on magenta(Environement). The results are compared on the basis of following parameters :

A. ACCURACY

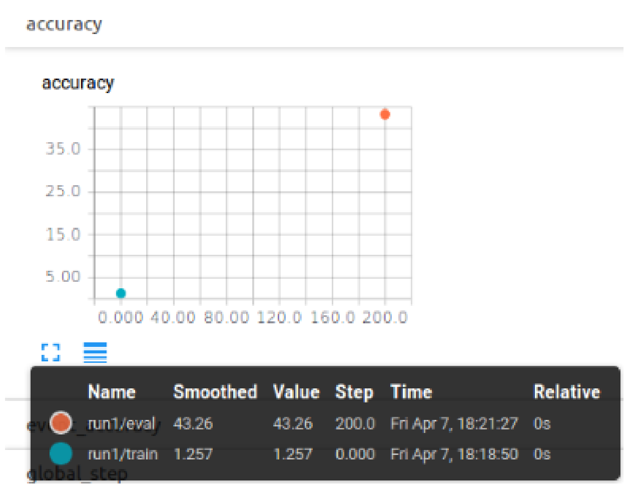
CASE 1. Basic RNN



Name	Smoothed	Value	Step	Time	Relative
run1/eval	60.38	60.38	200.0	Fri Mar 31, 19:01:49	0s
run1/train	4.283	4.283	0.000	Fri Mar 31, 18:59:38	0s

Figure5. accuracy of Basic RNN model

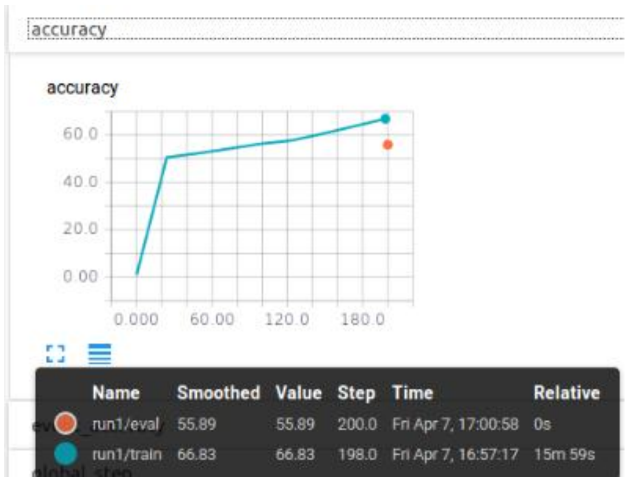
CASE 2. LookBackRNN



Name	Smoothed	Value	Step	Time	Relative
run1/eval	43.26	43.26	200.0	Fri Apr 7, 18:21:27	0s
run1/train	1.257	1.257	0.000	Fri Apr 7, 18:18:50	0s

Figure 6. accuracy of LookBackRNN model

CASE 3. Attention RNN



Name	Smoothed	Value	Step	Time	Relative
run1/eval	55.89	55.89	200.0	Fri Apr 7, 17:00:58	0s
run1/train	66.83	66.83	198.0	Fri Apr 7, 16:57:17	15m 59s

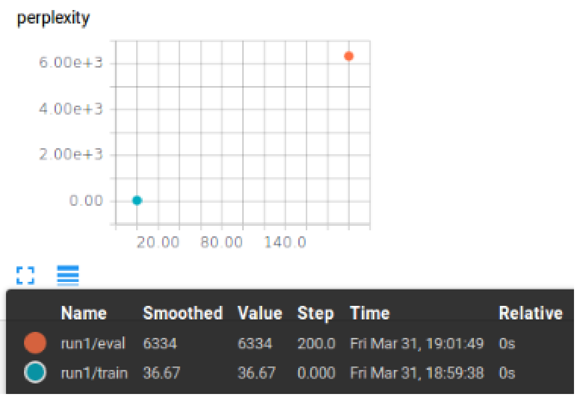
Figure7. accuracy of Attention RNN model

In the figure5, Basic RNN model's accuracy is shown as 60.38, whereas in figure 6, LookBackRNN model's accuracy is calculated as 43.26 and in figure 7, Attention RNN model's accuracy has come as 55.89.

B. PERPLEXITY

Perplexity means the measure of entangled state. The more the perplexity, more is the complexity of the model.

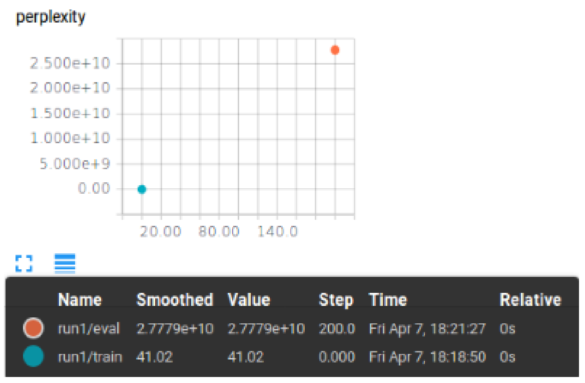
CASE 1.Basic RNN



	Name	Smoothed	Value	Step	Time	Relative
●	run1/eval	6334	6334	200.0	Fri Mar 31, 19:01:49	0s
●	run1/train	36.67	36.67	0.000	Fri Mar 31, 18:59:38	0s

Figure 8. perplexity of Basic RNN model

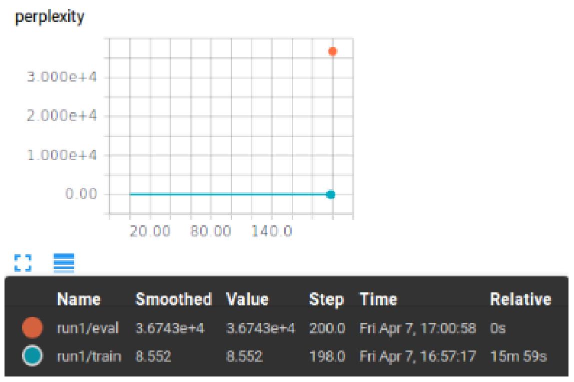
CASE 2.LookBack RNN



	Name	Smoothed	Value	Step	Time	Relative
●	run1/eval	2.7779e+10	2.7779e+10	200.0	Fri Apr 7, 18:21:27	0s
●	run1/train	41.02	41.02	0.000	Fri Apr 7, 18:18:50	0s

Figure 9. perplexity of LookBackRNN model

CASE 3.Attention RNN



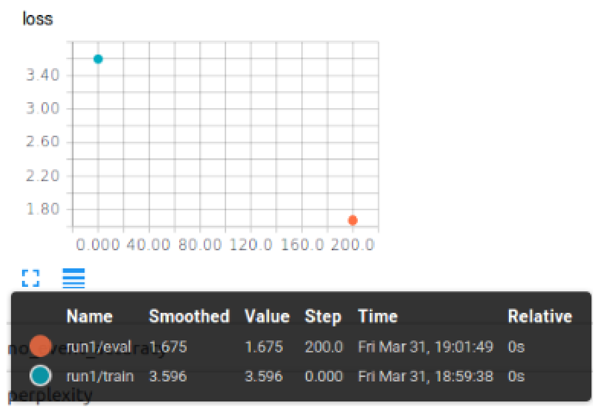
	Name	Smoothed	Value	Step	Time	Relative
●	run1/eval	3.6743e+4	3.6743e+4	200.0	Fri Apr 7, 17:00:58	0s
●	run1/train	8.552	8.552	198.0	Fri Apr 7, 16:57:17	15m 59s

Figure 10. perplexity of Basic RNN model

In the figure8, Basic RNN model's perplexity is shown as 6334, whereas in figure 9, LookBackRNN model's perplexity is calculated as 2.7776e+10 and in figure 10, Attention RNN model's perplexity has come as 3.674e+4. Here LookBack model has the highest perplexity which means it is more complex than the other two models.

C. LOSS

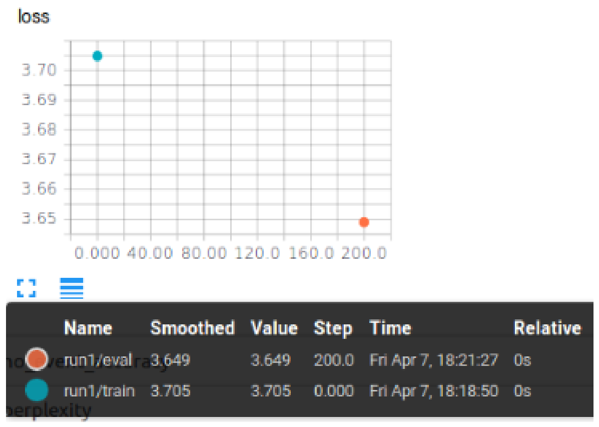
CASE1. Basic RNN



	Name	Smoothed	Value	Step	Time	Relative
●	run1/eval	1.675	1.675	200.0	Fri Mar 31, 19:01:49	0s
●	run1/train	3.596	3.596	0.000	Fri Mar 31, 18:59:38	0s

Figure 11. loss in Basic RNN model

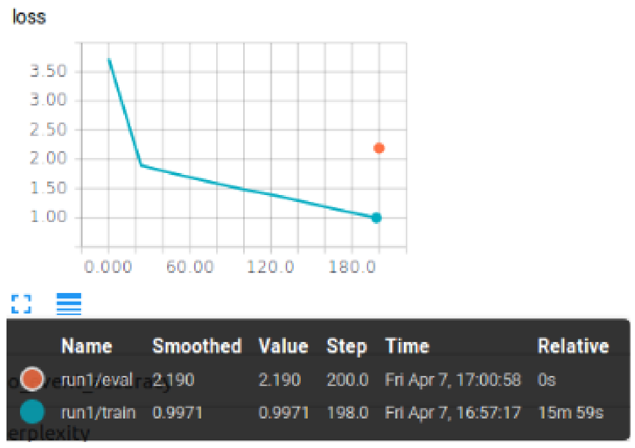
CASE2. LookBackRNN



	Name	Smoothed	Value	Step	Time	Relative
●	run1/eval	3.649	3.649	200.0	Fri Apr 7, 18:21:27	0s
●	run1/train	3.705	3.705	0.000	Fri Apr 7, 18:18:50	0s

Figure 12.loss in LookBackRNN model

CASE3.Attention RNN



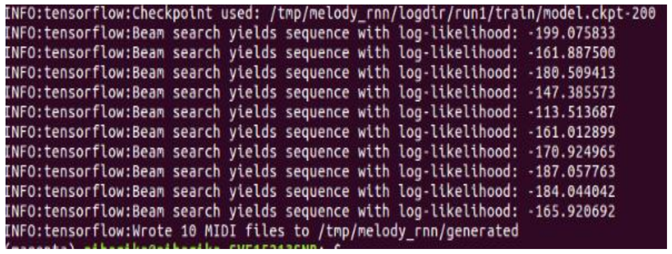
	Name	Smoothed	Value	Step	Time	Relative
●	run1/eval	2.190	2.190	200.0	Fri Apr 7, 17:00:58	0s
●	run1/train	0.9971	0.9971	198.0	Fri Apr 7, 16:57:17	15m 59s

Figure 13.loss in Attention RNN model

In the figure11, Basic RNN model's loss is shown as 1.675, whereas in figure 12, LookBackRNN model's loss is calculated as 3.649 and in figure 13, Attention RNN model's loss has come as 2.190. Here LookBack model has the highest loss which means it is less feasible than the other two models.

D. LIKELIHOOD

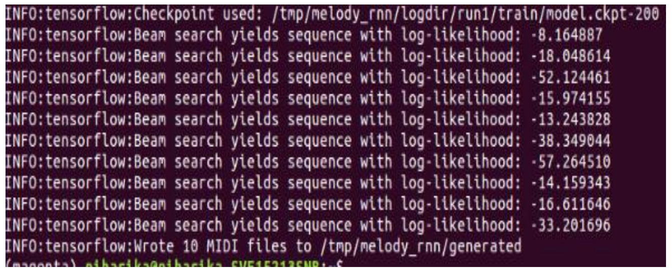
CASE1. Basic RNN



```
INFO:tensorflow:Checkpoint used: /tmp/melody_rnn/logdir/run1/train/model.ckpt-200
INFO:tensorflow:Beam search yields sequence with log-likelihood: -199.075833
INFO:tensorflow:Beam search yields sequence with log-likelihood: -161.887500
INFO:tensorflow:Beam search yields sequence with log-likelihood: -180.509413
INFO:tensorflow:Beam search yields sequence with log-likelihood: -147.385573
INFO:tensorflow:Beam search yields sequence with log-likelihood: -113.513687
INFO:tensorflow:Beam search yields sequence with log-likelihood: -161.012899
INFO:tensorflow:Beam search yields sequence with log-likelihood: -170.924965
INFO:tensorflow:Beam search yields sequence with log-likelihood: -187.057763
INFO:tensorflow:Beam search yields sequence with log-likelihood: -184.044042
INFO:tensorflow:Beam search yields sequence with log-likelihood: -165.920692
INFO:tensorflow:Wrote 10 MIDI files to /tmp/melody_rnn/generated
```

Figure 14. likelihood in Basic RNN model

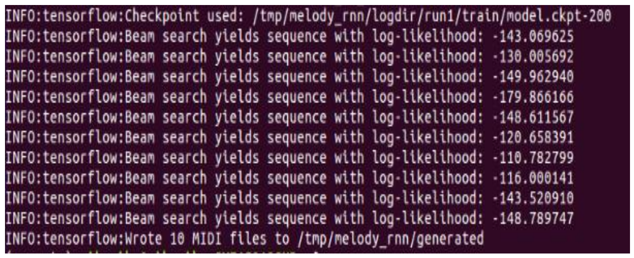
CASE2. LookBackRNN



```
INFO:tensorflow:Checkpoint used: /tmp/melody_rnn/logdir/run1/train/model.ckpt-200
INFO:tensorflow:Beam search yields sequence with log-likelihood: -8.164887
INFO:tensorflow:Beam search yields sequence with log-likelihood: -18.048614
INFO:tensorflow:Beam search yields sequence with log-likelihood: -52.124461
INFO:tensorflow:Beam search yields sequence with log-likelihood: -15.974155
INFO:tensorflow:Beam search yields sequence with log-likelihood: -13.243828
INFO:tensorflow:Beam search yields sequence with log-likelihood: -38.349044
INFO:tensorflow:Beam search yields sequence with log-likelihood: -57.264510
INFO:tensorflow:Beam search yields sequence with log-likelihood: -14.159343
INFO:tensorflow:Beam search yields sequence with log-likelihood: -16.611646
INFO:tensorflow:Beam search yields sequence with log-likelihood: -33.201696
INFO:tensorflow:Wrote 10 MIDI files to /tmp/melody_rnn/generated
```

Figure 15. likelihood in lookbackRNN model

CASE3.Attention RNN



```
INFO:tensorflow:Checkpoint used: /tmp/melody_rnn/logdir/run1/train/model.ckpt-200
INFO:tensorflow:Beam search yields sequence with log-likelihood: -143.069625
INFO:tensorflow:Beam search yields sequence with log-likelihood: -130.005692
INFO:tensorflow:Beam search yields sequence with log-likelihood: -149.962940
INFO:tensorflow:Beam search yields sequence with log-likelihood: -179.866166
INFO:tensorflow:Beam search yields sequence with log-likelihood: -148.611567
INFO:tensorflow:Beam search yields sequence with log-likelihood: -120.658391
INFO:tensorflow:Beam search yields sequence with log-likelihood: -110.782799
INFO:tensorflow:Beam search yields sequence with log-likelihood: -116.000141
INFO:tensorflow:Beam search yields sequence with log-likelihood: -143.520910
INFO:tensorflow:Beam search yields sequence with log-likelihood: -148.789747
INFO:tensorflow:Wrote 10 MIDI files to /tmp/melody_rnn/generated
```

Figure 16. likelihood in Attention RNN model

In the figure14, Basic RNN model's likelihood is measured as 165.54, whereas in figure 15, LookBackRNN model's likelihood is calculated as 33.64 and in figure 16, Attention RNN model's likelihood has come as 143.56. Here LookBack model has the lowest loss which means it produces less alike tones of melody than others.

VI. CONCLUSION

In this paper, we implemented Melody RNN model and its configurations in magenta environment. By comparing all the three configurations- basic, lookback and attention RNN model, we come to a conclusion that attention produces the most accurate melody from existing MIDI files whereas lookback model produces the minimum likelihood factor which means it gives the new melody every time. The third configuration-basic model gives least perplexity which means it is the least complex model. By implementing all the configurations on the basis of parameters like accuracy, loss, perplexity and likelihood, we have generated new melodies trained on existing Melody Recurrent Neural Networks.

VII. FUTURE SCOPE

In this paper, we have implemented all the three configurations of melody RNN model and made a comparison on the basis of parameters like loss, accuracy, perplexity and likelihood. In future, we can implement other models of melody like Improve RNN, Drum RNN and Polyphonic RNN and try to compare these with existing types of melody models using Recurrent Neural Networks.

REFERENCES

[1] Douglas Eck, Natasha Jaques, ShixiangGu, Richard and E. Turner" Tuning Recurrent Neural Networks With Reinforcement Learning" , Under review as a conference paper at ICLR 2017, https://arxiv.org/pdf/1611.02796v4

[2] Cheng-Zhi Anna Huang,TimCooijmans, Adam Roberts , Aaron Courville,"Counterpoint by Convolution",Under review as a conference paper at ICLR 2017, https://openreview.net/pdf?id=r1Usiwcex

[3] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, Use of Kernel Deep Convex Networks and End-To-End Learning for Spoken Language Understanding, in Proc. of SLT, 2012.

[4] G. Tur and L. Deng. Intent Determination and Spoken Utterance Classification, in Chapter 4, Spoken Language Understanding: Systems for Extracting Semantic Information from Speech, pp. 81-104, Wiley, 2011.

[5] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on. IEEE, 2013, pp. 273–278.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[7] T. Mikolov, M. Karafiat, L. Burget, J. ´ Cernock ˇ y, and S. Khudan- ´ pur, "Recurrent neural network based language model," in Proceedings of INTERSPEECH, vol. 2010, no. 9. International Speech Communication Association, 2010, pp. 1045–1048.

[8] F. A. Gers and J. Schmidhuber, "LSTM recurrent networks learn simple context free and context sensitive languages," IEEE Transactions on Neural Networks, vol. 12, no. 6, pp. 1333–1340, 2001