Gate-Variant of Long Short Term Memory (LSTM) Neural Networks for Music Generation

Jonathan Dowdall and Fathi M. Salem
College of Engineering
Michigan State University
East Lansing, Michigan 48825
Email: *dowdallj@msu.edu* || *salemf@msu.edu*

***Abstract*—This paper evaluates a variant of Long Short Term Memory (LSTM) cells by reducing the number of parameters in the input, forget, and update gates. The LSTM variant demonstrates its ability to compose novel polyphonic music similar to the original LSTM model with significantly fewer parameters.**

I. Introduction

Recurrent neural networks (RNN) have fundamentally changed the role of artificial neural networks. Through recurrent connections, neural networks are capable of learning with temporal understanding. The gap between artificial and human intelligence continues to shrink as recurrent neural networks conquer tasks from language modeling [1] to stock price pattern recognition [2]. Furthermore, the implementation of memory gates in the LSTM as described by Hochreiter et al [3] made it practical for RNNs to capture longer term dependencies, effectively maintaining memory not too unlike human intelligence. LSTM networks have demonstrated novel applications such as drawing [4] and composing music [5].

LSTM cells use gating network signals to control the retention of memory and the influence of new input. The three gates- input, forget, and output- each contain as many parameters as the memory cell they regulate. Dey and Salem [6] have shown that gated recurrent neural networks can acheive similar performance at a fraction of the computational expense by reducing the number of parameters in the memory gates. This was demonstrated on Gated Recurrent Units, a gated RNN similar to the LSTM except with only two memory gates. The proposed GRU variants were able to match and even outperform the original GRU on the MNIST dataset of handwritten digits.

This paper aims to demonstrate that LSTM networks can also benefit from the parameter reduction method shown in [6]. The LSTM and its variant are implemented in otherwise identical Biaxial LSTM (BALSTM) networks. Introduced by Johnson [7], the BALSTM has demonstrated a very impressive ability to recognize patterns in timing and melodic structure from midi compositions. Replacing the LSTM with the LSTM variant results in similar convergence with a shorter training time.

II. Background

A. LSTM

The simple RNN model computes hidden activation as

$$h_t = g(Wx_t + Uh_{t-1} + b) \tag{1}$$

where x_t is the *m-dimensional* input vector at time t, h_t is the *n-dimensional* hidden state, g is the activation function commonly the logistic (sigmoid) or hyperbolic tangent function. W, U, and b are the parameters which are respectively sized $n \times m$, $n \times n$, and $n \times 1$.

In order to accommodate for exploding and vanishing gradients, the LSTM RNN architecture introduces three memory gates: input i, forget f, and ouput o, which regulate the final output (state) of the memory cell as follows:

$$\tilde{c} = g(W_c x_t + U_c h_{t-1} + b_c) \tag{2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c} \tag{3}$$

$$h_t = o_t \odot c_t \tag{4}$$

The intermediate candidate for the cell reflects Eqn (1). In Eqn (2), we compute the internal hidden state c as a weighted average of $\tilde{c}$ and the previous internal memory state c_{t-1}. The forget gate f_t and the input gate i_t give weight to the old and new hidden states respectively, via element-wise multiplication. Finally, the output gate o is multiplied by the internal hidden state of the cell to produce a final activation output h_t.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{6}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{7}$$

Each gate contains its own set of parameters W and U which are adaptively updated at each training step. This means the LSTM model has 4 times the amount of parameters as the simple RNN model shown in Eqn (1). This is a considerable increase in computation.

B. Gate Variant

All of the gates' parameters are updated with the same information pertaining to the state of the overall network. This leads to a redundancy in the signals driving the gating signals [6]. The proposed variant modifies the LSTM so that each gate is computed using only the previous hidden state.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{8}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{9}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{10}$$

This modified architecture should be much less computationally expensive without significant decrease in performance.

C. Experiment

In order to generate music, the network takes midi input, a binary representation of notes and timings. There are 88 keys on a piano, so the network contains 4 layers of 88 stacked LSTMs, forming a BALSTM. The details of this implementation can be found in [7]. The final layer is activated by the logistic function to indicate whether or not to play a note or hold a note if it is already being articulated. The goal is to produce interesting music, so there is no accuracy measure. However, log-likelihood is used to calculate loss at each step for adaptive parameter updates. Two models were trained: one with normal LSTM cells, and one with the LSTM variant.

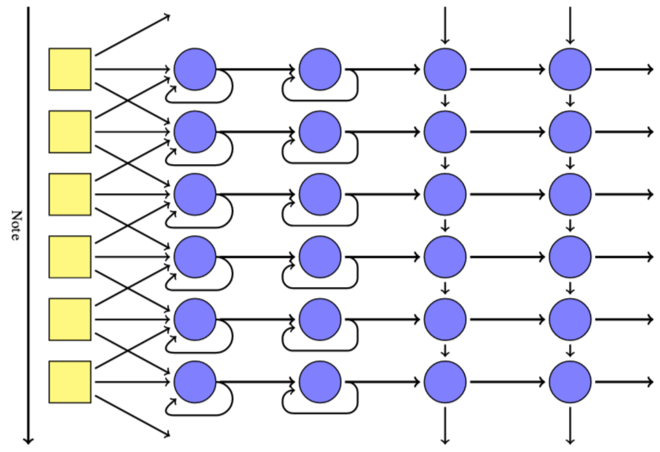

Fig. 1. Network architecture

III. Results and Discussion

The LSTM variant took more iterations before the loss started to descend. However, it was able to reach convergence just like the original LSTM model. Furthermore, the LSTM variant was able to converge at a lower loss than the original LSTM. As expected, the LSTM variant required less compute time per iteration, and finished 10,000 iterations almost an hour sooner than the original LSTM. Overall, the LSTM variant performed exceptionally well over the original LSTM model and satisfied the hypothesis.

TABLE I. Experimental Values

Machine	AWS EC2 G2.2
Operating System	Ubuntu 16.04
Software	Theano
Iterations	10,000
Optimizer	Adagrad
Layer 1 size	300
Layer 2 size	300
Layer 3 size	150
Layer 4 size	50
Dropout	0.7
Batch Size	20
Sequence Length	8

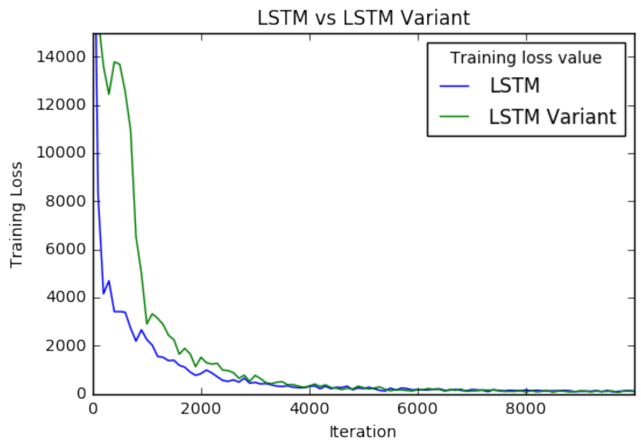

Fig. 2. Loss Comparison

TABLE II. Experimental Results

	LSTM	LSTM Variant
Seconds/iteration	3.47	3.22
Total runtime	9h39m07s	8h57m18s
Loss	105.05	90.80

The music produced from the model can be found in the relevant links section of this paper. The purpose of this project was to compare the LSTM and the variant model. In the future, it would be interesting to experiment more with the architecture of the BALSTM to produce even more interesting music and reach a lower loss. It would also be interesting to design some accuracy measure for music generation.

Appendix
Relevant Links

Github

Generated Music Demo

BALSTM Blog

References

[1] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," 2013.

[2] K.-i. Kamijo and T. Tanigawa, "Stock price pattern recognition-a recurrent neural network approach," in *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*. IEEE, 1990, pp. 215–221.

[3] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra, “DRAW: A recurrent neural network for image generation,” *CoRR*, vol. abs/1502.04623, 2015. [Online]. Available: http://arxiv.org/abs/1502.04623

[5] D. Eck, “A first look at music composition using lstm recurrent neural networks.”

[6] R. Dey and F. M. Salem, “Gate-variants of gated recurrent unit (GRU) neural networks,” *CoRR*, vol. abs/1701.05923, 2017. [Online]. Available: http://arxiv.org/abs/1701.05923

[7] D. D. Johnson, “Generating polyphonic music using tied parallel networks,” in *International Conference on Evolutionary and Biologically Inspired Music and Art*. Springer, 2017, pp. 128–143.