Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks

Rahul Dey*, Fathi M. Salem†
Circuits, Systems, and Neural Networks (CSANN) LAB
Department of Electrical and Computer Engineering
Michigan State University
East Lansing, MI 48824-1226, USA
*Email: *deyrahul@msu.edu, †salemf@msu.edu*

***Abstract*—The paper evaluates three variants of the Gated Recurrent Unit (GRU) in recurrent neural networks (RNNs) by retaining the structure and systematically reducing parameters in the update and reset gates. We evaluate the three variant GRU models on MNIST and IMDB datasets and show that these GRU-RNN variant models perform as well as the original GRU RNN model while reducing the computational expense. In this comparative study, we simply refer to the three variants as, respectively, GRU1, GRU2, and GRU3 RNNs.**

1. Introduction

Gated Recurrent Neural Networks (Gated RNNs) have shown success in several applications involving sequential or temporal data [1**?**]. For example, they have been applied extensively in speech recognition, music synthesis, natural language processing, machine translation, etc. [1, 2]. Long Short-Term Memory (LSTM) RNNs and the recently introduced Gated Recurrent Unit (GRU) RNNs have been successfully shown to perform well with long sequence applications [1–8].

Gated RNNs' success is primarily due to the gating network signaling that control how the present input and previous memory are used to update the current activation and produce the current state. These gates have their own sets of weights that are adaptively updated in the learning phase (i.e., the training and evaluation process). While these models empower successful learning in RNNs, they introduce an increase in parameterization through their gate networks. Consequently, there is an added computational expense vis-a-vis the simple RNN model [1, 4]. It is noted that the LSTM RNN employs 3 distinct gate networks while the GRU RNN reduces the gate networks to two.

In this paper, we focus on the GRU RNN and explore three new gate-variants with reduced parameterization. We comparatively evaluate the performance of the original and the variant GRU RNNs on two public datasets. Using the MNIST dataset [12], one generates the pixel-wise sequence [1, 4]. One sequence is obtained from each 28×28 image sample as pixel-wise long sequence of length $28 \times 28 = 784$ (basically, scanning from the upper left to the bottom right of the image). The other sequence type employs the IMDB movie review dataset [5] where one defines the length of the sequence in order to achieve high performance sentiment classification from a given review paragraph.

2. Background: RNN, LSTM and GRU Models

In principle, RNNs are more suitable for capturing relationships among sequential data types. The so-called simple RNN has a recurrent hidden state as in

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

where x_t is the (external) $m-$dimensional input vector at time t, h_t the $n-$dimensional hidden state, g is the (element-wise) activation function, such as the logistic function, the hyperbolic tangent function, or the rectified Linear Unit (ReLU) [1, 12], and $W,\ U\ and\ b$ are the appropriately sized parameters (two weights and a bias). Specifically, in this case, W is an $n \times m$ matrix, U is an $n \times n$ matrix, and b is an $n \times 1$ matrix (or vector).

Bengio et al. [9] showed that it is difficult to capture long-term dependencies using such simple RNNs because the (stochastic) gradients tend to either vanish or explode with long sequences. Two particular models, the Long Short-Term Memory (LSTM) RNNs [3, 4] and Gated Recurrent Unit (GRU) RNNs [1] have been proposed to solve the vanishing or exploding gradient problems. We will present these two models in sufficient details for our purposes below.

2.1. Long Short-Term Memory (LSTM) RNNs

The LSTM RNN architecture uses the computation of the simple RNN of equation (1) as an intermediate candidate for the internal memory cell (state), say $\tilde{c}_t$, and add it in a (element-wise) weighted-sum to the previous value of the internal memory state, say c_{t-1}, to produce the current value of the memory cell (state) c_t. This is expressed succinctly in the following discrete dynamic equations:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2}$$

$$\tilde{c}_t = g(W_c x_t + U_c h_{t-1} + b_c) \tag{3}$$

$$h_t = o_t \odot g(c_t) \tag{4}$$

978-1-5090-6389-5/17/$31.00 ©2017 IEEE

In Eqns (3) and (4), the activation nonlinearity g is typically the hyperbolic tangent function but more recently may be implemented as a rectified Linear Unit (reLU). The weighted sum is implemented in Eqn (2) via element-wise (Hadamard) multiplication denoted by $\odot$ to gating signals. The gating (control) signals $i_t,\ f_t\ and\ o_t$ denote, respectively, the *input*, *forget*, and *output* gating signals at time t. These control gating signals are in fact replica of the basic equation (3), with their own parameters, and replacing g by the logistic function. The logistic function limits the gating signals to within 0 and 1. The specific mathematical form of the gating signals are thus expressed as the vector equations:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{6}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{7}$$

where σ is the logistic nonlinearity and the parameters for each gate consist of two matrices and a bias vector. Thus, the total number of parameters (represented as matrices and bias vectors) for the 3 gates and the memory cell structure are, respectively, $W_i,\ U_i,\ b_i,\ W_f,\ U_f,\ b_f,\ W_o,\ U_o,\ b_o,\ W_c,\ U_c$ and b_c. These parameters are all updated at each training step and stored. It is immediately noted that the number of parameters in the LSTM model is increased 4-folds from the simple RNN model in Eqn (1). (Note that the activation and all the gates have the same dimensions). Assume that the cell state is n-dimensional. Assume also that the input signal is m-dimensional. Then, the total parameters in the LSTM RNN is equal to $4 \times (n^2 + nm + n)$.

2.2. Gated Recurrent Unit (GRU) RNNs

The GRU RNN reduces the gating signals to two from the LSTM RNN model. The two gates are called an update gate z_t and a reset gate r_t. The GRU RNN model is presented in the form:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{8}$$
$$\tilde{h}_t = g(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{9}$$

with the two gates presented as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{10}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{11}$$

One observes that the GRU RNN [Eqns (8)-(9)] is similar to the LSTM RNN [Eqns (2)-(3)], however with less external gating signal in the interpolation Eqn (8). This saves one gating signal and the associated parameters. We defer further information to reference [1], and the references therein. In essence, the GRU RNN has a 3-fold increase in parameters in comparison to the simple RNN of Eqn (1). Specifically, the total number of parameters in the GRU RNN equals $3 \times (n^2 + nm + n)$. In various studies, e.g., in [1] and the references therein, it has been noted that GRU RNN is comparable to, or even outperforms, the LSTM in most cases. Moreover, there are other reduced gated RNNs, e.g. the Minimal Gated Unit (MGU) RNN, where only one gate equation is used and it is reported that this (MGU) RNN performance is comparable to the GRU RNN, and by inference, to the LSTM RNN, see [1, 10].

In this paper, we focus on the GRU RNN model and evaluate new variants. Specifically, we retain the architecture of Eqns (8)-(9) unchanged, and focus on variations in the construct of the gating signals in Eqns (10) and (11). We apply the variations identically to the two gates for uniformity and simplicity.

3. The Variant GRU Architectures

The gating mechanism in the GRU (and LSTM) RNNs is a replica of the simple RNN in terms of parametrization. The weights corresponding to these gates are also updated using the backpropagation through time (BTT) stochastic gradient descent as it seeks to minimize a loss/cost function [3, 4]. Thus, each parameter update will involve information pertaining to the state of the overall network. Thus, all information regarding the current input and the previous hidden states are reflected in the latest state variable. There is a redundancy in the signals driving the gating signals. The key driving signal should be the internal state of the network. Moreover, the adaptive parameter updates all involved components of the internal state of the system [10, 11]. In this study, we consider three distinct variants of the gating equations applied uniformly to both gates:

3.1. Variant 1: GRU1

In this variant, each gate is computed using only the previous hidden state and the bias, thus reducing the total number of parameters, in comparison to the GRU RNN, by $2 \times nm$.

$$z_t = \sigma(U_z h_{t-1} + b_z) \tag{12}$$
$$r_t = \sigma(U_r h_{t-1} + b_r) \tag{13}$$

3.2. Variant 2: GRU2

In this variant, each gate is computed using only the previous hidden state, thus reducing the total number of parameters, in comparison to the GRU RNN, by $2 \times (nm + n)$.

$$z_t = \sigma(U_z h_{t-1}) \tag{14}$$
$$r_t = \sigma(U_r h_{t-1}) \tag{15}$$

3.3. Variant 3: GRU3

In this variant, each gate is computed using only the bias, thus reducing the total number of parameters, in comparison to the GRU RNN, by $2 \times (nm + n^2)$.

$$z_t = \sigma(b_z) \tag{16}$$
$$r_t = \sigma(b_r) \tag{17}$$

We have performed an empirical study of the performance of each of these variants as compared to the base

TABLE 0. NETWORK MODEL CHARACTERISTICS

Model	MNIST Pixel-wise	MNIST Row-wise	IMDB
Hidden Units	100	100	128
Gate Activation	Sigmoid	Sigmoid	Sigmoid
Activation	ReLU/tanh	ReLU/tanh	ReLU/tanh
Cost	Categorical Cross-entropy	Categorical Cross-entropy	Binary Cross-entropy
# Epochs	100	100	100
Optimizer	RMSProp	RMSProp	RMSProp
Dropout	20%	20%	20%
Batch Size	32	32	32

GRU RNN on, first, sequences generated from the MNIST dataset and then on the IMDB movie review dataset. From here on, we will refer to the base GRU RNN model as GRU0 and the three variants as GRU1, GRU2, and GRU3 respectively.

Our architecture consists of a single layer of one of the variants of GRU units driven by the input sequence and the activation function g set as $ReLU$ or $tanh$. For the MNIST dataset, we generate the pixel-wise and the row-wise sequences as in [**?**]. The networks have been generated in Python using the Keras library with Theano as the backend library. We modifed the Keras version of the GRU class to create classes for GRU1, GRU2, and GRU3. We trained and tested our models using the $tanh$, and also separately the $ReLU$, activation function on all these classes. The layer of units is followed by a softmax layer in the case of the MNIST dataset or a traditional logistic activation layer in the case of the IMDB dataset to predict the output class. Root Mean Square Propagation (RMSprop) is used as the choice of optimizer to adapt the learning rate for each of the parameters. To speed up training, we also decay the learning rate exponentially as a function of the cost in each epoch as

$$eta(n) = eta \times e^{cost(n-1)} \tag{18}$$

where eta represents the base constant learning rate, n is the current epoch number, $cost(n-1)$ is the cost computed in the previous epoch, and $eta(n)$ is the current epoch learning rate. We trained our models for 100 epochs in each case. The details of our models are delineated in Table 0.

4. Results and Discussion

4.1. Application to MNIST Dataset (Pixel-wise)

The MNIST dataset [12] consists of 60000 training images and 10000 test images, each of size 28×28 of handwritten digits. We evaluated our three variants against the original GRU model on the MNIST dataset by generating the sequential input in one case (pixel-wise, one pixel at a time) and in the second case (row-wise, one row at a time). The pixel-wise sequence generated from each image are 1-element signal of length 784, while the 28-element row-wise produces a sequence of length 28. For the pixel-wise case, we performed different iterations by varying the constant base learning rate eta. The results of our experiments are depicted below in the comparative summary in Table 1.

TABLE 1. MNIST PIXEL-WISE SEQUENCES, RELU ACTIVATION: PERCENTAGE ACCURACY OF DIFFERENT ARCHITECTURES USING 2 CONSTANT BASE LEARNING RATES eta IN 100 EPOCHS.

eta	10^{-3}		10^{-4}		# Params
Variant	Train	Test	Train	Test	
GRU0	99.2	98.6	95.3	95.3	30600
GRU1	98.9	98.4	95.3	95.4	30400
GRU2	98.9	98.1	92.9	92.2	30200
GRU3	10.4	10.3	63.1	63.3	10400

4.2. Application to the IMDB Dataset (Text Sequence)

The IMDB dataset is a natural-sequences dataset. It is composed of 25000 training data and 25000 test data consisting of movie reviews and their binary sentiment classification. Each review is represented by a maximum of 80 (most frequently occurring) words in a vocabulary of 20000 words [5]. We trained the dataset on all 4 GRU variants using three constant base learning rates of 1e-3, 1e-4 and 1e-5 over 100 epochs. In the training, we employ 128-dimensional GRU RNN variants and have adopted a batch size of 32. We shall illustrate the performance in a typical case as depicted in Fig. 1, using a base learning rate of 1e-3, we have observed that performance fluctuates visibly and testing results deteriorate. However, for a lower learning rate, the training and testing accuracy uniformly progresses over incrementally increasing accuracy profile-curves when the base learning rate equals 1e-4. Further decreasing the base learning rate to 1e-5 shows different accrucay curve-profiles: they match exactly for GRU0 and GRU2; however, GRU1 and GRU3 fail to learn when the learning rate is at this level or smaller. Tables 2 and 3 summarize the results of accuracy performance which show comparable performance among GRU0, GRU1, GRU2, and GRU3 with both $tanh$ and $ReLU$ activation functions. The number of parameters in each case is also listed. It is noted that the parameters are reduced in GRU1 and GRU2 by one third and for GRU3 the parameters are dramtically reduced by two thirds. Yet, comparable performances can be achieved.

TABLE 2. IMDB DATASET, RELU ACTIVATION: PERFORMANCE SUMMARY OF DIFFERENT ARCHITECTURES USING TWO BASE LEARNING RATES eta OVER 100 EPOCHS.

eta	10^{-3}		10^{-4}		10^{-5}		# Params
Variant	Train	Test	Train	Test	Train	Test	
GRU0	95.3	83.7	87.4	84.8	89.8	84.8	98688
GRU1	94.5	84.1	87.0	84.8	0.5	0.5	65920
GRU2	94.5	84.2	86.9	84.6	89.5	84.5	65664
GRU3	92.3	83.2	86.8	84.5	0.5	0.5	33152

The IMDB data experiments provide the most striking results. It can be clearly seen that all the three GRU variants perform comparably to the original GRU RNN while using less number of parameters. The learning pace of GRU3 was

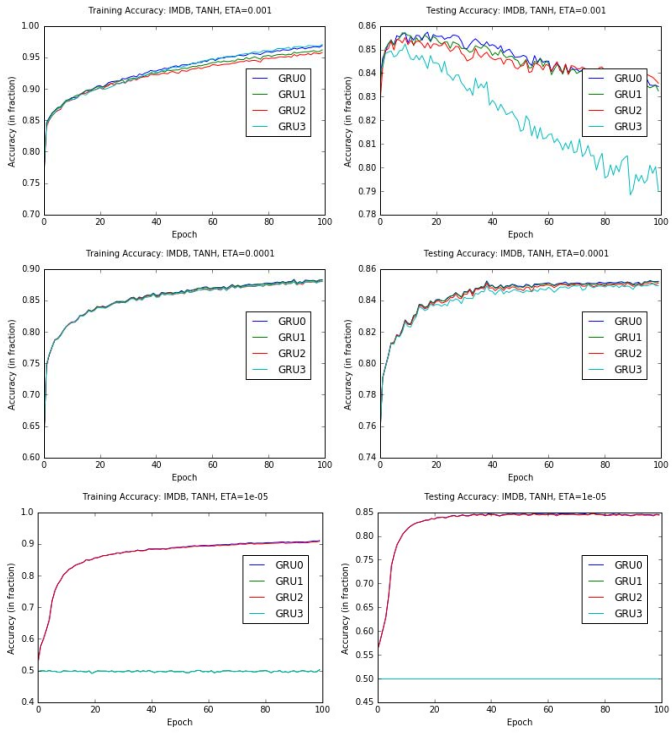

Figure 1. Training and Testing accuracy on IMDB dataset using TANH activation function and base learning rates of $eta = 1e-3$, $eta = 1e-4$ and $eta = 1e-5$

TABLE 3. IMDB DATASET, TANH ACTIVATION: PERFORMANCE SUMMARY OF DIFFERENT ARCHITECTURES USING TWO BASE LEARNING RATES eta OVER 100 EPOCHS.

eta	10^{-3}		10^{-4}		10^{-5}		# Params
Variant	Train	Test	Train	Test	Train	Test	
GRU0	96.9	83.4	88.3	85.2	91.0	84.5	98688
GRU1	96.2	83.2	88.2	85.2	0.5	0.5	65920
GRU2	95.8	83.6	88.0	85.1	90.8	84.4	65664
GRU3	97	79	88	84.9	0.5	0.5	33152

also similar to those of the other variants at the constant base learning rate of 1e-4. From Tables 2 and 3, it is noted that more saving in computational load is achieved by all the variants of GRU RNN as the input is represented as a large 128-dimensional vector.

5. Conclusion

The experiments on the variants GRU1, GRU2, and GRU3 vis-a-vis the GRU RNN have demonstrated that their test accuracy performance is comparable on three example sequence lengths. Two sequences generated from the MNIST dataset and the IMDB dataset. The main driving signal of the gates appear to be the (recurrent) state as it contains essential information about other signals [10]. Moreover, the use of the stochastic gradient descent implicitly carries information about the network state [10, 11]. This may explain the relative success in using the bias alone in the gate signals as its adaptive update carries information about the state of the network. The GRU variants reduce this redundancy and thus their performance has been comparable to the original GRU RNN. While GRU1 and GRU2 have indistinguishable performance from the GRU RNN, GRU3 frequently lags in performance, especially for relatively long sequences and may require more execution time to achieve comparable performance.

Acknowledgment

This work is partially supported by the NSF grant No. ECCS-1549517, which is gratefully acknowledged.

References

[1] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[2] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.

[3] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

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

[5] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 142–150.

[6] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato, "Learning longer memory in recurrent neural networks," *arXiv preprint arXiv:1412.7753*, 2014.

[7] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[8] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[9] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[10] F. M. Salem, "Reduced parameterization in gated recurrent neural networks," 2016.

[11] ——, "A basic recurrent neural network model," *arXiv preprint arXiv:1612.09022*, 2016.

[12] F. Chollet. Keras github. [Online]. Available: https://github.com/fchollet/keras/blob/master/examples/mnist_hierarchical_rnn.py