Simplified Minimal Gated Unit Recurrent Neural Network Exploration

Deliang Yang
Department of Electrical and Computer Engineering
Michigan State University
East Lansing, Michigan 48824
yangdeli@msu.edu

***Abstract*—Recently recurrent neural networks (RNN) has been successful in processing and extracting sequence data, and been extensively studied by many researcher. However, comprehending RNN and finding the best suitable one for given tasks is not easy, one of the reason is that it involves complex hidden units such as Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). To simplify and understand the working mechanism behind the recurrent unit, Minimal Gated Unit (MGU) is proposed by scholars recently.**

In this work, the detailed design of MGU is demonstrated. Two of its variations that reduce parameters are also proposed. Extensive evaluation and parameters tuning are conducted to explore and compare the capability and limitation of each of the proposed model. Our results indicate that the parameter-reduced MGU has faster converge rate as well as similar performance to complex model like LSTM in simple application, which makes it a potentially good demo or sketching model.

I. Introduction

Recurrent Neural Network has been studied widely and it is proven to be naturally suitable for sequence data processing applications, such as language translation [1], [2], [3], speech recognition [4], image captioning [5], [6], [7], video captioning [8], [9] or short term precipitation prediction [10].

Many different types of architecture have been proposed. Some of the most popular ones are LSTM, GRU [1] and MGU [11]. LSTM is the most complicated one among the three, since it employs three gates to control the data flow, while GRU has two and MGU has only one. In the recurrent unit, each gate takes previous state, current input and the cell memory as input and compute the output with activation on the fusion input signal result. Thus in the original model, each gate has its own weights, including 2 matrices and 1 bias vector. The actual parameter size of the whole model is highly dependent on the number of gates of each recurrent units. Based on this, researchers are trying hard to not only reduce the number of gate in each unit, like aforementioned GRU and MGU, but also the input signals at each gate, while preserving the functionality and performance when handling sequential data.

In this paper, we propose two new variants from the MGU model, one is achieved by limiting the input signals at the input gate, the other one replaces the matrices in the network to vectors, and matrices multiplication becomes point-wise product of each element in two vectors.

Evaluation and comparison on LSTM, standard MGU (MGU-Std), MGU variant 2 (MGU-2) and MGU variant 4 (MGU-4) are conducted on different data set, such as MNIST [12], IMDB and NIST [13]. Evaluation results indicates that MGU-2 has similar performance to stardard MGU model in handwritten digit recognition. In another scenario which is textual input sequence analysis, MGU-2 is slightly worse than MGU-4 but has faster converge rate. Our time benchmark indicates MGU model and its variants have advantage on time consumption reduction over LSTM.

The rest of this paper is organized as follows. Section II will introducing some related works, Section III will demonstrate the detailed design and architecture of the MGU-2 and MGU-4. Section IV evaluates and compares the mentioned models above. Some discussions on the results and models are proposed in Section V. Finally we conclude the paper.

II. Related Works

Firstly we introduce simple RNN, which is the base for all the other modification. The hidden state in a recurrent unit is compute as

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

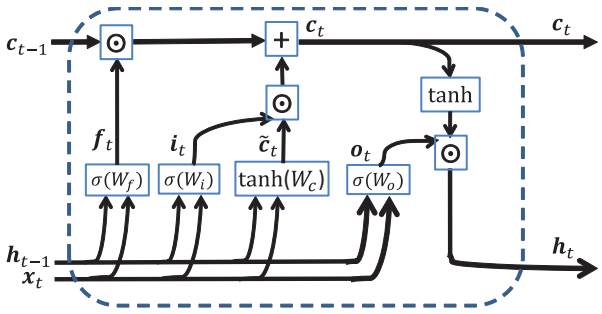

Fig. 1. LSTM model.

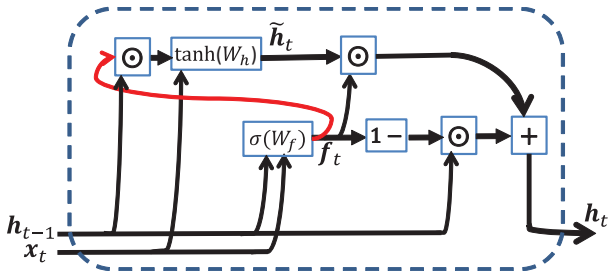

Fig. 2. MGU model.

where x_t is the external input vector at time t, h_t is the new hidden state at time t, σ is the point-wise activation function. W, U and b are the 3 matrices we need to train based on our input and output mapping. Simple RNN can not remember the long term dependencies within the sequence. To conquer this, LSTM has been proposed.

Fig. 1 shows the LSTM unit structure. Compared to simple RNN, LSTM has a memory cell in it. The cell is controlled by a combination of input gate and memory control gates, formulated by equations below.

$$\tilde{c} = \sigma(W_c x_t + U_c h_{t-1} + b_c) \quad (2)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (3)$$
$$h_t = o_t \odot \sigma(c_t) \quad (4)$$

In Equation 3, the weighted sum of the gate control signal is implemented by element-wise multiplication denoted by "$\odot$". As we can see, LSTM adds two more gates to control the output and previous inputs, thus has the ability to capture the long term dependencies in the sequence.

After LSTM's success in the areas mentioned in the Section I, researchers try to keep the function of different gate but also reduce the number of parameters so that the model is simplified.

MGU is one of the examples. The architecture of MGU is illustrated as Fig. 2. In MGU model, there is only one gate left. The forget and output gates are substituted by further computation on the output of the only gate. The MGU model can be formulated as follows.

$$i_t = \sigma(U_i c_{t-1} + W_i x_t + b_i) \quad (5)$$
$$c_t = (1 - i_t) \odot c_{t-1} + i_t \odot \sigma(U h_{t-1} + W x_t + b) \quad (6)$$
$$h_t = i_t \odot c_t \quad (7)$$

As we can see, the memory of previous state as well as current state input are kept, controlled by only one gated signal.

Guo-Bing Zhou et al. [11] compare the performance of both GRU and MGU on adding problem, IMDB dataset, MNIST dataset, Penn Treebank. The results show that the proposed MGU not only reduces the training time due to reduced parameterization but also outperforms GRU in some applications.

III. System Design and Architecture

In this work, we primarily design two variants from the MGU model. After background introduction, it is clear that reduction on parameters helps to lower the training time while achieve comparable accuracy as the original version. Yet intuitively, removing too many parameters would oversimplify the model and may encounter negative impact on the performance, such as lower accuracy.

Guided by the above idea, we propose two variants based on MGU. The first one is called MGU-2 (Variant 2) (Here I use name that is consistent with project guideline, for publishable draft, I would change it to normal number order.), where we remove the input signal and bias in the gate. This means that the gate output will primarily dependent on the previous hidden state. By the removal of the two signals, we do not need to train the W and b matrices anymore. In this case, the formulas for this variant are

$$i_t = \sigma(U_i c_{t-1}) \quad (8)$$
$$c_t = (1 - i_t) \odot c_{t-1} + i_t \odot \sigma(U h_{t-1} + W x_t + b) \quad (9)$$
$$h_t = i_t \odot c_t \quad (10)$$

It is not hard to calculate that by this means we reduce the parameter size by $nm + n^2$, where n is cell state dimension and m is the input signal dimension.

The second variant is called MGU-4 (Variant 4), whose formulas are similar to the Variant 2. The difference is that the matrix U_i in the model becomes a vector, denoted as u_i. Since the hidden state h_{t-1} has the same dimension as u_i, we compute the matrices multiplication

in previous variant as inner product, i.e. pixel-wise multiplication. In this way, we further reduced the number of parameters by $n(m-1)$. The corresponding formulas are

$$i_t = \sigma(u_i \cdot c_{t-1}) \tag{11}$$

$$c_t = (1 - i_t) \odot c_{t-1} + i_t \odot \sigma(Uh_{t-1} + Wx_t + b) \tag{12}$$

$$h_t = i_t \odot c_t \tag{13}$$

The proposed two variants will be evaluated and discussed in the following sections.

IV. Evaluation

In this section, we will evaluate the effectiveness of MGU using three datasets. For comparison, 4 models will be evaluated, which are standard LSTM, standard MGU, MGU-2 and MGU-4. Section IV-A uses MNIST dataset for basic image classification benchmark. Section IV-B evaluates the models' ability of handling sequential sentiment data classification. After that, we will evaluate the model limit with complicated image classification problem with NIST dataset in Section IV-C. Details about the datasets will be provided later.

Since the goal is to compare the models under controlled conditions, we simply use two layers in the recurrent neural network. The first one is our core recurrent layer, the next one is a fully connected layer that extract the class label. Theoretically we certainly can improve the accuracy by adding convolutional layer in image processing problem, however, this behavior introduces another bunch of parameter that is going to be trained, which will weaken the persuasion of the drawn conclusion. Because simplified model like MGU-4 can somewhat be compensated by the additional layer. As a result, we use only two layers for the benchmark.

Another issue here is about parameter tuning. Once a certain parameter is updated, we need to apply it to all the models, such that consistency is preserved. It is ture that we are able to use optimized parameter set for simplified model to achieve similar performance as complex model like LSTM, but that is not our goal. After all, this is not a parameter-tuning or optimum-seeking game. We just need to focus on the difference of each model.

Some of the hyper parameters we use are listed below, which will be applied to all the experiments:

- Batch size: 128
- Epochs: 100
- RNN layer activation function: tanh
- Hidden unit size: 100

For different dataset, different learning rates and optimizers are employed, which are shown as Table I.

TABLE I
Optimizer, learning rate and activation function used for different datasets

Dataset	Optimizer	Learning rate	Dense layer units	Activation function at dense layer
MNIST	SGD	0.03	10	Softmax
IMDB	Adam	0.001	1	Sigmoid
NIST	RMSProp	0.002	62	Softmax

The software package we use is `Keras`, all the tasks are running on the personal computer, which may takes longer time for each epoch than running on High Performance Computing Center (HPCC) in Michigan State University. However, since the PC also has GPU acceleration, the actual difference is not large.

A. MNIST

The MNIST dataset by [12] contains images of handwritten digits ('0'-'9'). All the images are of size 28×28. There are 60,000 images in the training set, and 10,000 in the test set. The images are preprocessed such that the center of mass of these digits are at the central position of the 28×28 image.

MNIST has been a very popular testbed for deep neural network classification algorithms, but is not widely used in evaluating RNN models yet. During evaluation, we treat each row (28 pixels) as a single input in the input sequence. Hence, an image is a sequence with length 28, corresponding to the 28 image rows (from top to bottom).

Accuracies and losses of the 4 models on both the training and test sets in this task are shown in Fig. 3, where the x-axis is the number of epoch. The losses are the logarithm values of the original ones.

As we can notice, the LSTM model takes fewer epochs to converge to the same loss as other models. The MGU-4 model actually has 2nd fast converge rate, however, if we take longer term into consideration, MGU-4 doesn't minimize the loss too much from standard MGU and MGU-2. Besides converge rate, all the model have comparable train and test accuracy. Yet we still need other results to make a conclusion.

B. IMDB

The second problem we study is sentiment classification in the IMDB movie reviews, whose task is to separate the reviews into positive and negative ones. This

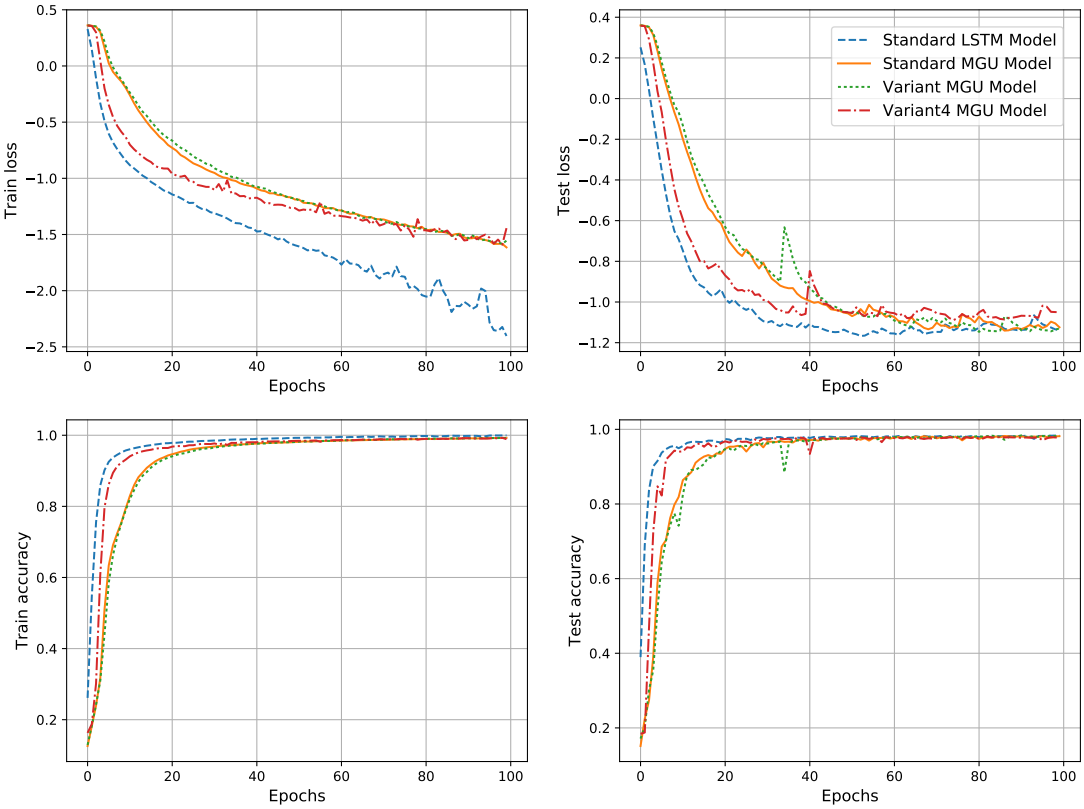

Fig. 3. Evaluation results for the MNIST dataset.

dataset was generated in [14]. There are 25,000 movie reviews in the training set, another 25,000 for testing. We use the provided bag-of-words format as our sequence input. The maximum sequence length is 128. Since this is a binary classification problem, we use only one unit at dense layer and use sigmoid function to extract the results.

The losses and accuracies are plotted in Fig. 4. There are many results that we can observe. First, one can tell from the graph that all the four models are suffering from overfitting issue. Because all the models have an increasing trend in test loss. Second, another interesting observation is that, all the train loss fluctuates at the beginning of the training process, however, when the training accuracy reaches 1.00, the train losses become steady. The model seems get stuck at some local optimum. What's more, it is also surprised to notice that all the model has highest test accuracy at the first epoch, which indicates that the sequential data in this problem is relatively simple, it is not necessary to train more than 2 epochs in common practice. Next, comparing the long term test accuracy, LSTM is still the best model among them. Although MGU-2 has fastest converge rate, it suffers from low test accuracy after 100 epochs. Finally, in the train loss graph, the MGU-4 model actually stops learning after it reaches 100% accuracy. This suggests that MGU-4 may have a quite different local optimum case than the other three models.

This problem to some degree differentiates the difference between the four models. We still need to conduct one more experiment to conclude.

C. NIST SD-19

NIST Special Database 19 (SD-19) contains NIST's entire corpus of training materials for handprinted document and character recognition. It publishes Handprinted Sample Forms from 3600 writers, 810,000 character images isolated from their forms, ground truth classifications for those images, reference forms for further data collection, and software utilities for image management and handling. There are 62 classes in total, which are "A-Z", "a-z" and "0-9". One can tell that NIST is the superset of MNIST we evaluate above. The images are 128×128, with white background and black written

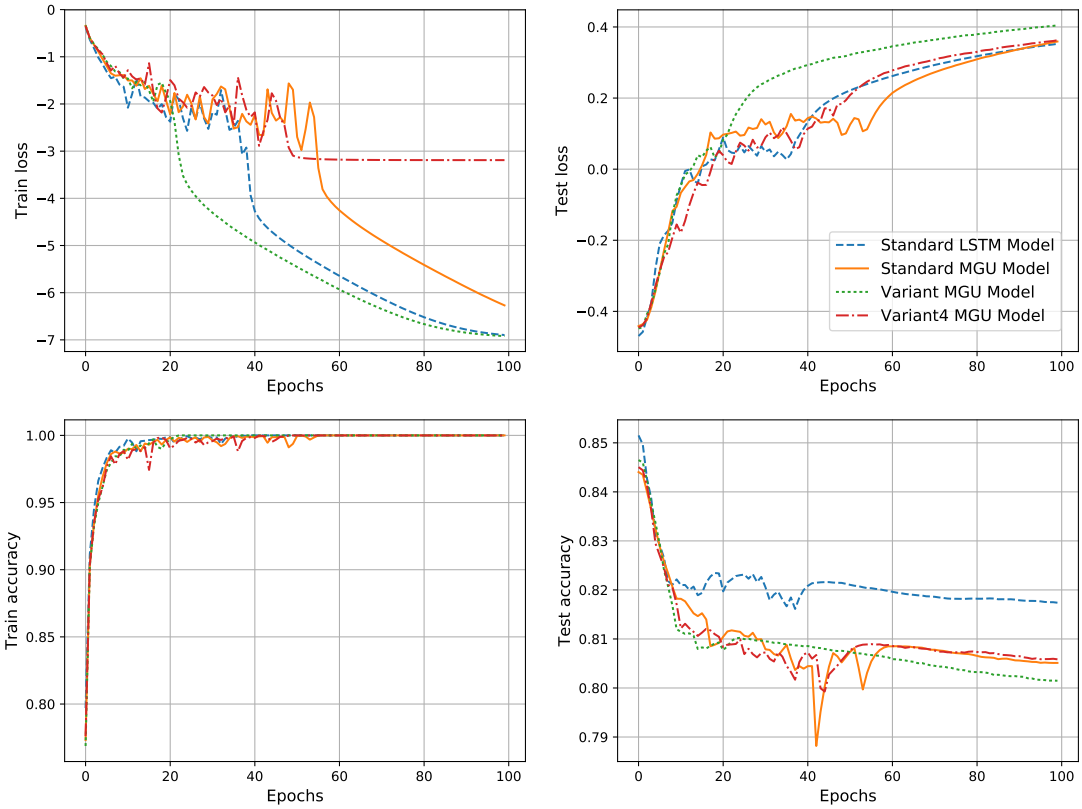

Fig. 4. Evaluation results for the IMDB dataset.

characters. It is not necessary to train all the image in the dataset for model comparison purpose, so we extract a subset from the whole database. There are 1,024 training samples and 256 test samples from each one of the 62 classes. So the total training size is 63,488 and the testing size is 15,872.

The losses and accuracy for the four models are shown in Fig. 5. This problem is actually the one that really differentiates the capability of the models. It is easy to tell that the performance ranking is: standard LSTM, MGU2, standard MGU, and finally MGU-4. Here we would like to clear one point, that for MGU-4 we use Adadelta optimizer, otherwise it would not converge at all, which means the model can not learn any feature from the given images. One interesting thing we can notice is that, during the training process, the MGU-2 is always steady, compared to the existence of noticeable fluctuations in the standard MGU and LSTM.

V. Discussion

When we look at the big picture, combining the results from all the dataset evaluations, we can make interesting discussion based on them.

A. Time Consumption

Let's first look at Table II for more information. The losses and accuracies for training and testing are collected from the final epochs. It may not be the best or worst sample batch during training, but it is the results from the steady states around 100th epoch. The third column in the table shows the total training time for the all 100 epochs in each independent experiment. Roughly, the time consumption ratio for standard LSTM, standard MGU and MGU-2 is 1.85:1.12:1.0, however, the MGU4 is not consistent. In MNIST case, it consumes 2nd more time in the three model but is the least in IMDB case. Yet in the NIST case, the time is close to the standard MGU. So one can not draw a conclusion on the time consumption ratio to other models.

B. Variant Models

As we know, MGU-2 and MGU-4 are two variants from MGU model. The performance for these two are quite different. MGU-2 usually can achieve good results

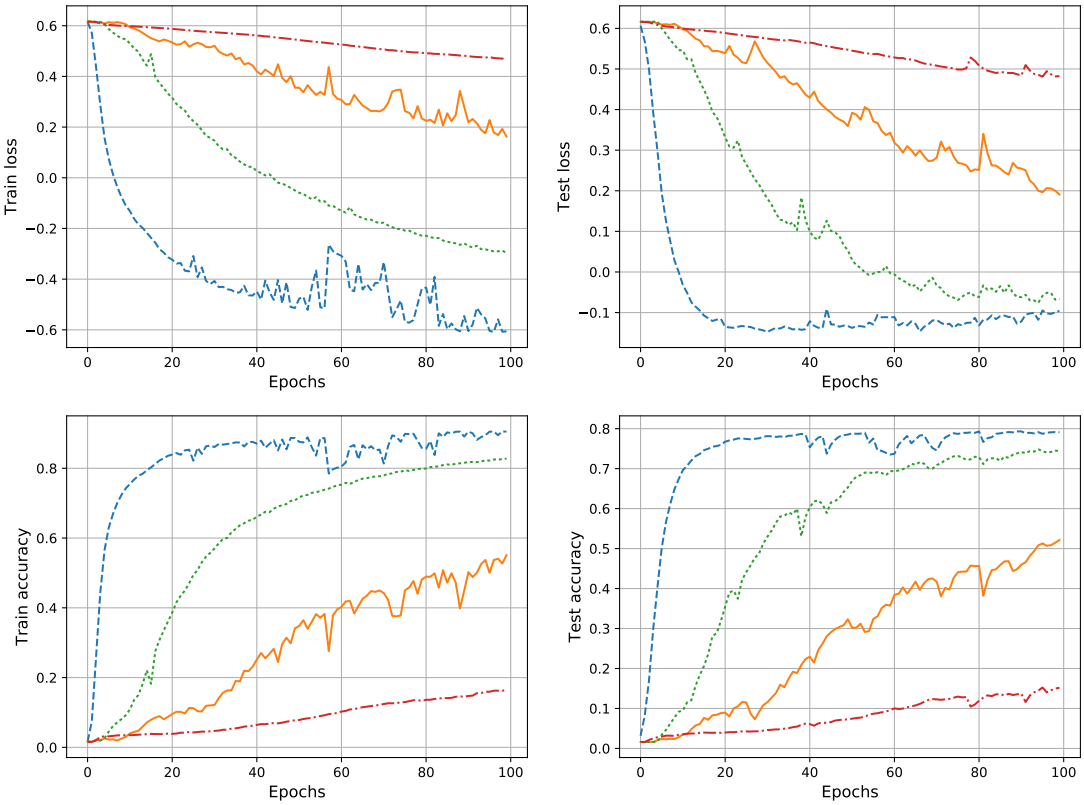

Fig. 5. Evaluation results for the NIST dataset.

across all the datasets, while MGU-4 either has lower test accuracy or fails to learn the features from the given input. Thus, in comparison between the two variants, we can conclude that the MGU-2 is better.

C. Variant and Standard Models

After the comparison between the two variants, we take a look at MGU-2 and the standard MGU. In MNIST case, MGU-2 is actually identical to the standard MGU; in the IMDB case, although MGU-2 fails in competition in test accuracy with standard MGU by 0.5%, it takes much less time and epochs (about 40 epochs) to achieve the same loss value; in the NIST dataset case, there is no doubt that MGU-2 performs better than standard MGU. So if we take everything into account, the MGU-2 variant is better than standard MGU, which is somewhat a surprising conclusion.

D. Dominance of LSTM

As one may notice, LSTM, which has three gates to control the flow, has the best performance across all the dataset evaluations. However, LSTM consumes more time to train. Based on the No-Free-Lunch theorem, every advantage comes with some cost. Using LSTM or the simplified MGU essentially depends on the application scenario.

VI. Conclusion

In this paper, we briefly introduce the history of recurrent neural layer and some of the successful applications based on it. And then the main contribution of this paper, two Minimal Gated Unit variant models are proposed, which are MGU-2 and MGU-4. Extensive experiments on three different datasets, MNIST, IMDB and NIST are conducted to evaluate the performance and limitation of the variant models. Along with the experiment, we use standard LSTM and standard MGU as the performance reference.

Evaluation results show that MGU-2 has great advantages over both MGU-4 and standard MGU, which makes it a promising alternative in simple demo or application for LSTM. Because removing gates and reduced parameterization can lower the time consumption and enable fast product or model shipping.

TABLE II
TRAINING RESULT COMPARISON ACROSS THE MODEL

Dataset	Model	Training Time (s) (100 epochs)	Train Loss	Train Accuracy (%)	Test Loss	Test Accuracy (%)
MNIST	Std LSTM	1618.9	0.0039	99.96	0.0716	98.36
	Std MGU	965.8	0.0242	99.27	0.07	98.18
	MGU-2	904	0.0279	99.2	0.0759	98.14
	MGU-4	1002.6	0.036	98.9	0.0891	97.83
IMDB	Std LSTM	3991.1	1.266 e-7	100	2.2649	81.72
	Std MGU	2411.8	5.385 e-7	100	2.3033	80.51
	MGU-2	2040.7	1.207 e-7	100	2.5497	80.14
	MGU-4	1835.3	6.450 e-4	100	2.3226	80.57
NIST	Std LSTM	3991.7	0.2477	90.5	0.8127	79.01
	Std MGU	2466.2	1.4538	55.08	1.5014	53.13
	MGU-2	2195	0.5062	82.78	0.8653	74.38
	MGU-4	2458.5	2.9426	16.58	3.0412	15.45

Future work includes benchmarking the variant model in a more complicated network structure or more comprehensive application. We can also try to come up with some of the similar variant models that are suitable for different application scenarios.

ACKNOWLEDGMENT

The authors would like to thank Dr. Salem for providing the basic knowledge about RNN and CNN, as well as the important comment and feedback that help to improve this work.

REFERENCES

[1] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[4] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[5] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

[6] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3128–3137.

[7] R. Lebret, P. O. Pinheiro, and R. Collobert, "Phrase-based image captioning," *arXiv preprint arXiv:1502.03671*, 2015.

[8] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[9] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International Conference on Machine Learning*, 2015, pp. 843–852.

[10] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, 2015, pp. 802–810.

[11] G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou, "Minimal gated unit for recurrent neural networks," *International Journal of Automation and Computing*, vol. 13, no. 3, pp. 226–234, 2016.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] P. J. Grother, "Nist special database 19 handprinted forms and characters database," *National Institute of Standards and Technology*, 1995.

[14] 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.