Reading Selectively via Binary Input Gated Recurrent Unit

Zhe Li[1*], **Peisong Wang**[1,2], **Hanqing Lu**[1] and **Jian Cheng**[1,2]
[1]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[2]Center for Excellence in Brain Science and Intelligence Technology
lizhe2016@ia.ac.cn, peisong.wang@nlpr.ia.ac.cn, hanqing.lu@ia.ac.cn, jcheng@nlpr.ia.ac.cn

Abstract

Recurrent Neural Networks (RNNs) have shown great promise in sequence modeling tasks. Gated Recurrent Unit (GRU) is one of the most used recurrent structures, which makes a good trade-off between performance and time spent. However, its practical implementation based on soft gates only partially achieves the goal to control information flow. We can hardly explain what the network has learnt internally. Inspired by human reading, we introduce binary input gated recurrent unit (BIGRU), a GRU based model using a binary input gate instead of the reset gate in GRU. By doing so, our model can read selectively during interference. In our experiments, we show that BIGRU mainly ignores the conjunctions, adverbs and articles that do not make a big difference to the document understanding, which is meaningful for us to further understand how the network works. In addition, due to reduced interference from redundant information, our model achieves better performances than baseline GRU in all the testing tasks.

1 Introduction

Recurrent Neural Networks (RNNs) have become the most popular approach to address machine learning tasks involving sequential data, such as machine translation [Bahdanau *et al.*, 2015], document classification [Le and Mikolov, 2014], sentiment analysis [Socher *et al.*, 2011], language recognition [Graves *et al.*, 2013], and image generation [Villegas *et al.*, 2017].

However, the vanilla RNN suffers from the gradient exploding/vanishing problem during training. To address this problem, long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997b] was proposed by introducing gate functions to control the information flow within a unit. LSTM has shown impressive results in several applications, but it has much higher computational complexity, which means longer inference time and more power consumption [Cheng *et al.*, 2018a; Cheng *et al.*, 2018b; Wang and Cheng, 2016; He and Cheng, 2018]. Therefore, K. Cho et al. proposed Gated Recurrent Unit (GRU) [Cho *et al.*, 2014]. Compared with LSTM that has three gate functions, GRU has only two gates: a reset gate function to determine how much previous information can be ignored, and an update gate function to select whether the hidden state is to be updated with the current information. Thus GRU can better balance the relationship between performance and computational complexity, which makes it very widely used. For ease of optimization, GRU uses sigmoid as the gate function, whose outputs are soft values between 0 and 1. However, the sigmoid gate function is easy to overfit [Zaremba *et al.*, 2014], and the information flow behind this gate function is not graspable to human. No matter LSTM or GRU, the model reads all the text available to them. But inspired by the human reading, we know that not all inputs are equally important. Therefore, the fact that texts are usually written with redundancy makes us think about the possibility of reading selectively. If the network can distinguish the important words from the decorated or connection words in a document, we can better understand how the network works internally.

In this paper, we want to explore a new GRU based RNN structure that can read only the important information and ignore the redundancies. The reset gate function in GRU is replaced by our proposed binary input gate function. With the hard binary gate, the model learns which word should be skimmed. One benefit of our model is that the binary input gate can clearly show the flow of information. As shown in our experiments, only significant information flows into the network. Another advantage is that while a model lies in a flat region of loss surface, it is likely to generalize well [Hochreiter and Schmidhuber, 1997a], because any small perturbation to the model makes little fluctuation to the loss. As training a binary gate means pushing the value to 0 and 1 residing in the flat region of the sigmoid function, small disturbances of the output value of sigmoid functions are less likely to change the binary gate output. The model ought to be more robust.

However, training binary gate function is also very challenging. The binary gate is obviously not differentiable, a common method for optimizing functions involving discrete variables is REINFORCE algorithm [Williams, 1992], but it uses the reward signal, increasing floating point operations [Bengio *et al.*, 2013]. Another way is to use the estimator [Bengio *et al.*, 2013; Courbariaux *et al.*, 2016; Hu *et al.*, 2018] which has been successfully applied in differ-

*Contact Author

ent works. By using the estimator, all the model parameters can be trained to minimize the target loss function with standard backpropagation and without defining any additional supervision or reward signal. We conduct document classification task and language modeling task on 6 different datasets to verify our model and our model achieves better performance. In summary, this paper makes the following contributions:

- We propose a novel interpretable RNN structure based on GRU: We replace the reset gate functions of GRU by the binary input gate functions, and retain the update gate functions.
- Our model can read the input sequences selectively: In our model, we can find more clearly whether the current information is passed into the network or not. In the experimental analysis, we show the gates in our learned model are meaningful and intuitively interpretable.
- The proposed method achieves better results compared to the baseline model: In the experiments, with the same amount of parameters and computational complexity, our model outperforms the baseline algorithms on all tasks.

The reminder of the paper is organized as follows. Section 2 introduces other works related to our research. Section 3 describes the model of binary input gated recurrent unit and how to train it. The experiments and analysis are shown in section 4. Finally, we conclude our model and discuss the future work in the last section.

2 Related Work

As all texts have emphasis and redundancy, making neural networks identify the importance of words has drown much attention recently. LSTM-Jump [Yu *et al.*, 2017] augments an LSTM cell with a classification layer that decides how many words to jump after reading a fixed number of words. Moreover, the maximum jump distance and the number of jumps allowed all need to be chosen in advance. The model is trained with the REINFORCE algorithm, where the reward is defined based on whether the model predicts correctly or not. These hyperparameters define a reduced set of subsequences that the model can sample, instead of allowing the network to learn any arbitrary sampling scheme. Skim-RNN [Seo *et al.*, 2018] contains two RNNs, a "small" RNN and a "big" RNN. At every time step the model determines to use the small RNN or the big RNN, which is based on the current input and the previous states. When the word is considered unimportant, the small model will be chosen. Then only a few dimensions in word vector will be updated and the other parts are retained. Instead of the REINFORCE algorithm, this network uses Gumble softmax to handle the non-differentiable problem. Skip RNN [Campos Camunez *et al.*, 2018] is a RNN with a binary state update gate, selecting whether the state of RNN will be updated or copied from the previous time step. This can be considered as completely skipping a word. This network use the straight-through estimator to solve the discrete problem, which guarantees that the amount of calculation does not increase. Structural-Jump-LSTM [Hansen *et al.*, 2018] introduces a new RNN structure with a skip agent and a jump agent. In every time step, the skip agent determines whether to skip the current word, and the jump agent selects a jump to the next word, or the next sub-sentence separator sympol (,;), or the next end of sentence sympol (.!?), or to the end of the text (which is also an instance of end of sentence). G2-LSTM [Li *et al.*, 2018] proposes a new way for LSTM training. It uses the developed Gumbel-Softmax trick to push the output of sigmoid function towards 0 and 1, and the gate functions are still continuous. Although this network reads all the text, the word is considered less important if the value of its input gate function is closer to 0.

Overall, all the above state-of-art models can identify the importance of words and read the text selectively. However, except G2-LSTM which reads all the text, other networks that read selectively have shown promising results only on specific temporal tasks: they targeted merely on the document classification task or the question answering task on small datasets. These tasks are easier than language modeling task and require less understanding of every word. And so far, there are no such reading selectively models that can match the performance of vanilla RNN on the word-level language modeling task. In this paper, we present the gated recurrent unit with binary input gate functions which has superior performances to the baseline algorithm on the documents classification task and the word-level language modeling task as well.

3 Model Description

In this section, we present a new structure of GRU, which we call it Binary Input Gated Recurrent Unit (BIGRU), containing binary input gate functions and update gate functions. The binary input gate functions are in charge of controlling the information input and the update gate functions are responsible for the prediction. Therefore, BIGRU has the ability to read selectively on both documents classification task and word-level language modeling task.

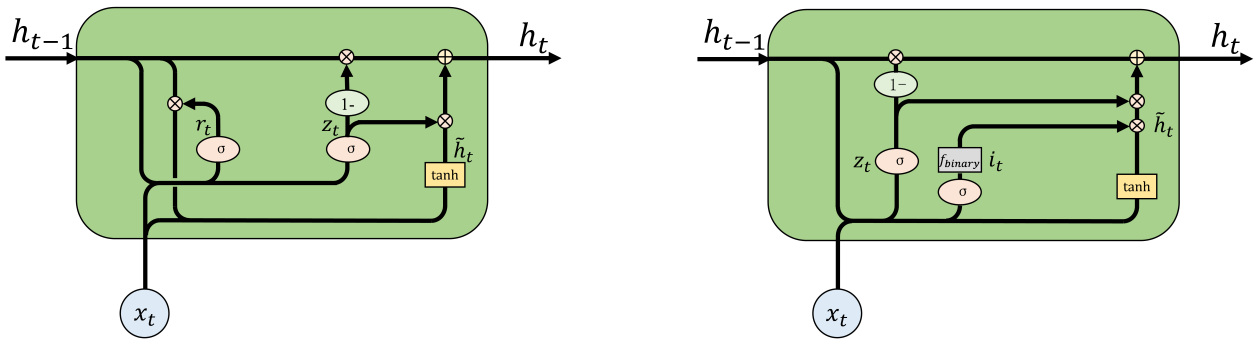
3.1 Gated Recurrent Unit

Recurrent Neural Network (RNN) takes an input sequence $\{x_1, x_2, \cdots, x_T\}$ and generates a hidden state sequence $\{h_1, h_2, \cdots, h_T\}$. In recurrent neural networks, the hidden states in layer k are used as inputs to layer $k+1$. The hidden states in last layer are used for prediction and making decision. The hidden state sequence is generated by iteratively applying a parametric state transition model S from time $t = 1$ to T:

$$h_t = S(h_{t-1}, x_t) = f(W_h h_{t-1} + W_x x_t + b)$$

where W_h, W_x are weights, b is bias and f is the activation function. In general, we use hyperbolic tangent (tanh) as the activation function.

Despite the remarkable success of RNNs in processing variable-length sequences, traditional RNNs suffer from the exploding and vanishing gradient problem that occurs when learning long-term dependencies, which limits the wide application of RNN. To alleviate this issue, Gated Recurrent Unit (GRU) [Cho *et al.*, 2014] and Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997b] were introduced. In general, LSTM has the state-of-art performance,



(a) GRU (b) BIGRU

Figure 1: Model architecture of GRU and BIGRU. (**a**) Complete architecture of a GRU cell, where r_t and z_t are reset gate and update gate, respectively. (**b**) Complete architecture of a BIGRU cell, where i_t is the binary input gate and z_t is the update gate.

but it needs longer inference time and more energy consumption. While GRU is a carefully designed recurrent structure which makes a good trade-off between performance and speed. Therefore, GRU has been prevalently used in both academia and industry. The recurrent transition of GRU is obtained by:

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \tag{1}$$
$$r_t = \sigma(W_r[h_{t-1}, x_t]) \tag{2}$$
$$\tilde{h_t} = \tanh(W[r_t \odot h_{t-1}, x_t]) \tag{3}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t} \tag{4}$$

where $\{W_z, W_r, W\}$ denote the recurrent weights and $h_t, \tilde{h_t}$ are hidden states. The logistic sigmoid function and Hadamard product are denoted as σ and $\odot$, respectively. The update gate z_t selects whether the hidden state h_t is to be updated with a new hidden state $\tilde{h_t}$. Small value of the update gates causes the cell to remember most of its previous values and use little current information. The reset gate r_t decides whether the former hidden state can be ignored. The smaller the value of r_t, the more previous information is forgotten.

3.2 Reading Selectively by Binary Input Gated Recurrent Unit

In Figure 1(a) we can see the structure of GRU, the reset gate is responsible for the current information input and the memory of previous information. However, the update gate also decides how much previous information to ignore, which means the two gates have duplicate functions. If we could separate the function of these two gates, the cell may have a better performance. Therefore, we design the binary input gate function which only decides whether the current information should flow into the network or not. It is similar to the input gate functions of LSTM. Thus, $\tilde{h_t}$ can be considered as the current input. As a result, the binary input gate function and update gate function in our model are responsible for current information input and previous information forgetting, respectively.

Our model can be seen in Figure 1(b), the binary input gate only decides whether the current information flows into the network or not and we retain the update gate to obtain hidden states. Thus the division of labor between two gates is more clear, and the task can be better accomplished without increasing the computational complexity. The proposed model, Binary Input Gated Recurrent Unit (BIGRU), works as follows during training:

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \tag{5}$$
$$i_t = f_{binary}(\sigma(W_r[h_{t-1}, x_t])) \tag{6}$$
$$\tilde{h_t} = i_t \odot \tanh(W[h_{t-1}, x_t]) \tag{7}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t} \tag{8}$$

where Eqn.(5) is the update gate whose function is similar to the one of GRU, Eqn.(6) is the binary input gate whose output value is 0 or 1. If the current information is not important to the task, the value of i_t will be 0 and the current input information will not be passed into the network. As the input value of the binary function is between 0 and 1, we use the stochastic binary method. The probability of getting 0 or 1 for the binary function can be computed by:

$$P(f_{binary}(x) = 1) = x \tag{9}$$
$$P(f_{binary}(x) = 0) = 1 - P(f_{binary}(x) = 1) \tag{10}$$

Afterwards, we stochastically sample from the Bernoulli distribution to obtain the binary value. However, the binary input gate functions create a tricky problem. Obviously, the whole model is differentiable except for f_{binary} and the discrete value cannot directly calculate the derivative. To solve this problem, we generally have two solutions. The first method for optimizing the discrete output value functions is REINFORCE algorithms [Williams, 1992], but it has some disadvantages. The reward is difficult to design and it will increase the computational complexity. In recent years, several estimators have been proposed for the particular case of neurons with binary outputs [Bengio *et al.*, 2013], and they have been successfully applied in different works [Courbariaux *et al.*, 2016]. Since we want to keep the same inference time as GRU, we choose to use the second method. We use the straight-through estimator, which consists of approximating the step function by the identity when computing gradients

Dataset	answer type	answer type	Number of examples	Avg. Len	vocab size
SST	Sentiment Analysis	Pos/Neg	6,920 / 872 / 1821	19	13,750
IMDb	Sentiment Analysis	Pos/Neg	21,143 / 3,857 / 25,000	282	61,046
AGNews	News Classification	4 categories	101,851 / 18,149 / 7,600	43	60,088
DBPedia	Topic Classification	14 categories	475,999 / 84,000 / 69,999	47	840,843

Table 1: Statistics of the classification datasets that BIGRU is evaluated on, where SST refers to Stanford Sentiment Treebank.

during the backward pass:

$$\frac{\partial f_{binary}(x)}{\partial x} = 1$$

By using the straight-through estimator as the backward pass for f_{binary}, all the model parameters can be trained to minimize the target loss function with standard backpropagation and without defining any additional supervision or reward signal. So we can ensure that the computational complexity does not increase.

There are several advantages in using the binary input gate functions. First, compared with vanilla GRU, BIGRU can decide to pay how much attention to different words like human reading. The binary input gate functions only select the important information to read and the redundant information will be discarded, leading to faster convergence in some tasks involving long sequences. Second, unlike some other reading selectively models [Campos Camunez *et al.*, 2018; Hansen *et al.*, 2018; Yu *et al.*, 2017], whose update gate functions do not work when the word is skipped, the update gate functions of BIGRU still work even if the output value of the input gate functions is 0. Therefore, the hidden state is updated at every time step, which is significant in the language modeling tasks. Besides, BIGRU can decide how much information can be forgotten at every time step. As a result, our model performs better than baseline GRU in all our test tasks. In the next section, we will present in detail the model performance in the document classification task and language modeling task.

4 Experiments

We evaluate the effectiveness of BIGRU in both document classification task and word-level language modeling task on six different datasets. We compare our model with the baseline GRU.

4.1 Document Classification

In the language classification task, the input is a sequence of words and the output is the vector of categorical probabilities. The evaluation criteria are generally the accuracy (Acc). We choose four different datasets, including Stanford Sentiment Treebank (SST), IMDb, AGNews and DBPedia. The detail of the datasets can be seen in Table 1.

For both GRU and BIGRU, we use a stacked three-layer RNN. Each word is embedded into a 100-dimensional vector. We make a linear transformation on the last hidden state of the network and then apply softmax function to obtain the classification probabilities. All models are trained with Adam, with the initial learning rate of 0.0001. We set gradient clip to 2.0. We use batch size of 32 for SST and 128 for the remaining. For both models, we set an early stop if the validation accuracy does not increase for 1000 global steps.

4.2 Language Modeling

For further evaluating the impact of our model, we perform word-level language modeling over a preprocessed version of the Penn Treebank (PTB) and WikiText-2 dataset. The Penn Treebank dataset has long been a central dataset for experimenting with language modeling. The dataset is heavily preprocessed and does not contain capital letters, numbers, or punctuation. The vocabulary is also capped at 10,000 unique words. Compared to the preprocessed version of PTB, WikiText-2 is over 2 times larger. The WikiText-2 dataset also features a far larger vocabulary and retains the original case, punctuation and numbers. As it is composed of full articles, the dataset is well suited for models that can take advantage of long term dependencies. The word-level language modeling task is to predict every next word in condition on the previous words. A model is evaluated by the prediction perplexity: smaller the perplexity, better the performance.

We follow the experimental settings in [Merity *et al.*, 2017] for GRU and our model. For each dataset we use a small model (one-layer network) and a big model (a stacked three-layer network with drop-connect on recurrent weights) for testing. For training the model, we choose averaged stochastic gradient descent (ASGD) [Polyak and Juditsky, 1992] for optimization. We use an initial learning rate of 10 for all experiments and carry out gradient clipping with maximum norm 0.25. We use a batch size of 80 for WikiText-2 and 40 for PTB. We train 1000 epochs for the small model and 2000 epochs for the large model. For all the models, all other experimental settings, including random BPTT length, the word embedding size and dropout probability, are the same as [Merity *et al.*, 2017].

4.3 Experimental Results and Analysis

Document Classification

Table 2 shows the accuracy of our model compared with the baseline GRU. The models are evaluated by accuracy (Acc) and reading rate (Rer). The reading rate is computed as follows. First, we get the output vectors of binary input gate functions in the first layer of the model. The output vector are called word input vector. We calculate the proportion of 1 in all word input vectors as the reading rate. The reading rate is considered as the ratio of reading information to all information. Absolutely, standard GRU uses the reset gate and reads the whole input sequence, so its reading rate can be seen as 1. In all four datasets, our model has a better performance than the baseline, about two percentage points higher,

Model	SST		IMDb		AGNews		DBPedia	
	Acc	Rer	Acc	Rer	Acc	Rer	Acc	Rer
GRU	0.771	1.00	0.803	1.00	0.815	1.00	0.876	1.00
BIGRU	**0.793**	0.35	**0.823**	0.37	**0.834**	0.43	**0.897**	0.41
GRU [1]	0.768	0.89	0.781	0.86	0.809	0.89	0.863	0.88

Table 2: Document Classification results on SST, IMDb, AGNews and DBPedia. Evaluation metrics are accuracy (Acc) and reading rate (Rer) compared to standard GRU.

Positive	**For** all **that** has **been** said about **the** subject matter, and **the** controversy that surrounded **it**, please do not overlook what I feel **to** be the most important part of **the** film: the salient struggles **of** everyone to keep their pride **through** their trials. Whether dealing **with** self-imposed male braggadocio, a sexual reawakening, **or** even life **itself**, everybody is human.
Negative	**An** obscure horror show filmed in the **Everglades**. Two couples stay **overnight** in a cabin after being made a little uneasy by **the** unfriendliness of the locals. Who, **or what**, are **the** Blood Stalkers? After **awhile** they find out. Watch for the character **of** the village idiot who clucks like a chicken, he certainly is weird.

Table 3: Examples of reading selectively on IMDb. Positive and Negative are two categories in IMDb. The skimmed words are shown in bold.

although it reads significantly less information. In these four datasets, our model reads no more than 43% information of all input sequence, but it has at least 1.9% promotion in terms of accuracy. For further comparison, we remove the stopwords (a, the, that, this, which, what, why, when, where, while, who, whom) in the datasets, and use the pre-processed dataset to train GRU. Compared with the dataset without being pre-processed, the experimental performance has a certain decline. This is because redundant words are difficult to know from the priori, and pre-defined stopwords are not always redundant. The same word has different meanings in different contexts, e.g. "that" is meaningless in some cases but in some other cases "that" refers to the subject and we cannot ignore it. Our model can learn which words are redundant based on context information, so BIGRU has a better performance. In order to find out whether this is the case, we make further analysis.

For each word, we specify that the word is skimmed if no less than 90% dimensions of its word input vector are 0. As the embedding size is 100, the skimmed word input vector has no more than 10 nonzero dimensions, which is a fairly strict criterion. Then we choose two comments in the IMDb test set, calculating the word input vector of every word. Table 3 shows the result: the skimmed words are shown in bold. Almost all the skimmed words are conjunctions, adverbs and articles, which have no special effect on the understanding of the whole sentence. These words have no contribution to our judgment of the category of this comment. This also illustrates that our model is more interpretable than GRU. We know what is input into the network, and our model has a better performance.

[1]Use the pre-processed dataset.

Model	Validation	Test
GRU (small)	85.97	82.96
BIGRU (small)	**80.65**	**77.79**
GRU (large)	74.50	72.13
BIGRU (large)	**70.28**	**68.04**

Table 4: Language Modeling results on PTB.

Model	Validation	Test
GRU (small)	102.09	97.08
BIGRU (small)	**95.93**	**91.18**
GRU (large)	87.82	83.30
BIGRU (large)	**80.78**	**77.92**

Table 5: Language Modeling results on WikiText-2.

Language Modeling

The results of two models on PTB and WikiText-2 are shown in Table 4 and Table 5, respectively. On PTB dataset, the small and large BIGRU outperform the baseline algorithm by 5.17 and 4.09 points in terms of test perplexity, respectively. On the more difficult dataset WikiText-2, the perplexity upgrade can even reach 5.90/5.38 points in the test set. It is easy to find that the improvement is more significant in language modeling tasks than in text classification tasks, so we want to explore why the task is more difficult, the more obvious the promotion, and we do further analysis.

Language modeling task is more sensitive and difficult than classification task. We can judge the category of the article by a few words, but when we want to predict every next word, we need the context information. This is why the former reading

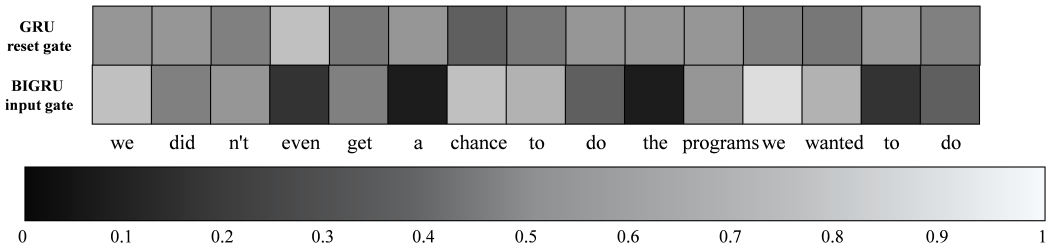
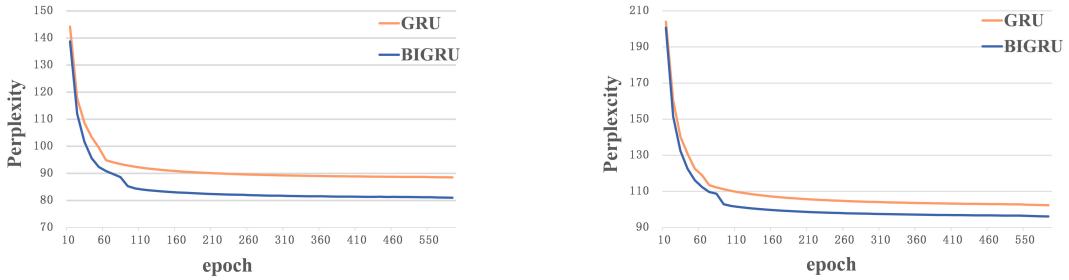


Figure 2: Visualization of the average gate value in GRU and BIGRU.



(a) PTB (b) WikiText-2

Figure 3: Perplexity of GRU and BIGRU on the PTB and WikiText-2 validation set.

selectively model performed poorly on word-level language modeling task. However, there must be important words and meaningless words in a document, so it is also important to distinguish them in order to avoid distractions. Similar to human intensive reading and skimming, BIGRU can focus on the more important words, while other unimportant words information will not be input completely to the network. In order to prove this, we do the visualization of words. We calculate the average value of the word input vectors in BIGRU. As a comparison, we also calculate the average output value of the reset gate functions in GRU. We plot the heatmap of an example sentence on the PTB test dataset in Figure 2.

We can see that our model pays less attention to the articles and adverbs, besides almost all of the subjects and verbs are completely read into the network. In contrast, the output of the reset gate functions in GRU has no obvious propensity for all words. In other word, the GRU is more like a black box to human. It's hard to explain how it works internally. On the contrary, our model is more interpretable. BIGRU learns the importance of each word like a human, which means it has a stronger learning ability, so its promotion in hard tasks is more obvious. We then plot the perplexity of validation dataset during training in Figure 3. We can see that on both PTB and WikiText-2 datasets, our model converges faster at the beginning of training. Therefore, if we want to get a model with the same effect, our model needs less training time than standard GRU.

5 Conclusion

In this paper, we present BIGRU, a GRU based RNN model that can read selectively. BIGRU is inspired by human reading, and can distinguish the important part from redundant part to avoid the unnecessary interference. In BIGRU we use binary input gate functions instead of the reset gate functions of the standard GRU and retain the update gate functions. Thus, the two gate functions in our model are responsible for current information input and previous information forgetting, respectively. As a result, our model can learn what is the important part of the input sequence based on the task, and ignore dynamically the redundancy. Through extensive experimental analysis, we demonstrate that our model ignores mostly the meaningless adverbs, conjunctions and articles and reads the main information of the text. In other word, BIGRU reads document like human, paying different attention to different words. After removing the interference of unnecessary information, BIGRU has a better performance. In both the document classification task and the word-level language modeling task, our model performs better than standard GRU on six different datasets. As the previous reading selectively RNN are all based on LSTM and they are not effective on word-level language modeling task, we contribute the first reading selectively GRU which has better results without increasing the amount of calculation.

In future work, we are going to investigate similar structure in LSTM, making LSTM read selectively and spend less time during inference. We also want to apply our model to hardware such as FPGA to further increase the inference speed while maintaining the performance.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (No.61876182, 61872364), the Strategic Priority Research Program of Chinese Academy of Science(No.XDB32050200).

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[Bengio *et al.*, 2013] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[Campos Camunez *et al.*, 2018] Victor Campos Camunez, Brendan Jou, Xavier Giró Nieto, Jordi Torres Viñals, and Shih-Fu Chang. Skip rnn: learning to skip state updates in recurrent neural networks. In *Sixth International Conference on Learning Representations: Monday April 30-Thursday May 03, 2018, Vancouver Convention Center, Vancouver:[proceedings]*, pages 1–17, 2018.

[Cheng *et al.*, 2018a] Jian Cheng, Pei-song Wang, Gang Li, Qing-hao Hu, and Han-qing Lu. Recent advances in efficient computation of deep convolutional neural networks. *Frontiers of Information Technology & Electronic Engineering*, 19(1):64–77, 2018.

[Cheng *et al.*, 2018b] Jian Cheng, Jiaxiang Wu, Cong Leng, Yuhang Wang, and Qinghao Hu. Quantized CNN: A unified approach to accelerate and compress convolutional networks. *IEEE Trans. Neural Netw. Learning Syst.*, 29(10):4730–4743, 2018.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[Courbariaux *et al.*, 2016] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[Graves *et al.*, 2013] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[Hansen *et al.*, 2018] Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. Neural speed reading with structural-jump-lstm. *ICLR*, 2018.

[He and Cheng, 2018] Xiangyu He and Jian Cheng. Learning compression from limited unlabeled data. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, pages 778–795, 2018.

[Hochreiter and Schmidhuber, 1997a] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[Hochreiter and Schmidhuber, 1997b] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2018] Qinghao Hu, Peisong Wang, and Jian Cheng. From hashing to cnns: Training binary weight networks via hashing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3247–3254, 2018.

[Le and Mikolov, 2014] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[Li *et al.*, 2018] Zhuohan Li, Di He, Fei Tian, Wei Chen, Tao Qin, Liwei Wang, and Tie-Yan Liu. Towards binary-valued gates for robust lstm training. *arXiv preprint arXiv:1806.02988*, 2018.

[Merity *et al.*, 2017] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

[Polyak and Juditsky, 1992] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[Seo *et al.*, 2018] Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. Neural speed reading via skim-rnn. *ICLR*, 2018.

[Socher *et al.*, 2011] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics, 2011.

[Villegas *et al.*, 2017] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. *arXiv preprint arXiv:1704.05831*, 2017.

[Wang and Cheng, 2016] Peisong Wang and Jian Cheng. Accelerating convolutional neural networks for mobile applications. In *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*, pages 541–545, 2016.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[Yu *et al.*, 2017] Adams Wei Yu, Hongrae Lee, and Quoc Le. Learning to skim text. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1880–1890, 2017.

[Zaremba *et al.*, 2014] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.