Interspeech 2018
2-6 September 2018, Hyderabad

Investigation on the combination of batch normalization and dropout in BLSTM-based acoustic modeling for ASR

Wenjie Li[1,3], *Gaofeng Cheng*[1,3], *Fengpei Ge*[1,3], *Pengyuan Zhang*[1,3], *Yonghong Yan*[1,2,3]

[1] Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics,
[2]Xinjiang Laboratory of Minority Speech and Language Information Processing, Xinjiang Technical Institute of Physics and Chemistry, Chinese Academy of Sciences, China,
[3]University of Chinese Academy of Sciences

liwenjie@hccl.ioa.ac.cn

Abstract

The Long Short-Term Memory (LSTM) architecture is a very special kind of recurrent neural network for modeling sequential data like speech. It has been widely used in the large scale acoustic model estimation recently and performs better than many other neural networks. Batch normalization(BN) is a good way to accelerate network training and improve the generalization performance of neural networks. However, applying batch normalization in the LSTM model is more complicated and challenging than in the feed-forward network. In this paper, we explored some novel approaches to add batch normalization to the LSTM model in bidirectional mode. Then we investigated some ways to combine the BN-BLSTM model with dropout, which is a traditional method to alleviate the overfitting problem in neural network training. We evaluated the proposed methods on several speech recognition tasks. Experiments show that the best performance on Switchboard task achieves 9.8% relative reduction on word error rate compared to the baseline, using the total Hub5'2000 evaluation dataset. Additionally, it is easy to implement and brings little extra computation.

Index Terms: speech recognition, batch normalization, dropout, BLSTM

1. Introduction

Recurrent neural network like LSTM [1] and Gated Recurrent Unit (GRU)[2] have recently been widely applied to a variety of sequential tasks, such as machine translation [3], speech recognition [4] and language models [5]. Because they can abstract the high-level understanding of sequence data and have been found to get remarkable performance. As a result, plenty of new techniques have been proposed to enhance the performance of LSTM, both on speed and on generalization.

Batch normalization [6] was proposed to overcome the problem called internal covariate shift in the training procedure of feed forward back propagation neural network, which means the continuous change of input distribution to a model.

Batch normalization addresses the problem by normalizing the input of layers and merges the normalization into model architecture. This method enforces the means and variates of network layers to be constant. It is proven to accelerate the training for faster convergence and promote the performance of many neural networks, like feed-forward network and convolution neural network (CNN). It leads to large improvement in object recognition [7] and many other tasks. However, it is proven to be difficult to apply in recurrent architecture [8] like LSTM, for the complicated interaction between LSTM model, especially the recurrent operation. People hold different views on how to use batch normalization on the recurrent neural network [9] [10]. In [8], adding batch normalization on the input-to-hidden transition led to faster convergence and better generalization, while adding it to the hidden-to-hidden transition didn't help. However, [10] showed a beneficial approach to apply batch normalization in the hidden-to-hidden transition of the recurrent models. Thus, we explore some different approaches to apply the batch normalization in the LSTM network to promote the accuracy of speech recognition.

Overfitting is another problem in deep learning, also in LSTM network. Dropout is a simple way to prevent neural network from overfitting [11], which becomes very widespread in many deep learning architectures. More and more variants of dropout designed for the different neural networks can be seen in the recent studies. A mechanism called per-frame dropout was proposed in [12]. It works quite well in LSTM architecture, which randomly drops out some frames in a chunk rather than throwing the nodes away as the traditional dropout method. It brings some variation into the fixed chunk in LSTM training, thus relieving the overfitting somehow. In this paper, we find a way to combine the per-frame dropout and batch normalization in the LSTM network to improve the performance in the large-scale speech recognition. The network is trained with the state-of-art lattice-free MMI (maximum mutual information) [13] criterion. Using these methods, we were able to achieve reductions in word error rate (WER) across multiple corpora with the largest gain being 11.6% relative improvement.

The rest of this paper is organized as follows. Section 2 introduces the projected LSTM architecture. Section 3 describes the approaches we use in detail. Section 4 is our experimental setup. The results and discussion are presented in Section 5. Finally, Section 6 is the conclusion.

2. Prerequisite: Projected LSTMs

Our baseline LSTM model uses the projected LSTM(LSTMP) [4] with little change. This architecture has a separate linear projection layer after LSTM layer, and the recurrent connection is produced by this projection layer. In implementing, a dim-range-node was used to select half hidden units to assign the recurrent connections as shown in figure 1. The formulation of the LSTMP is presented below:

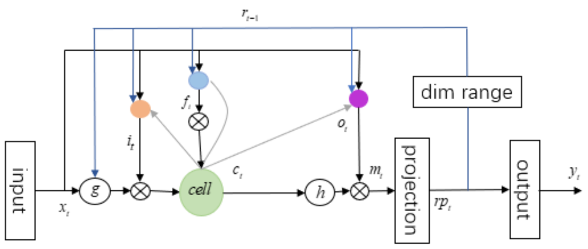

Figure 1: *Architecture of LSTMP*

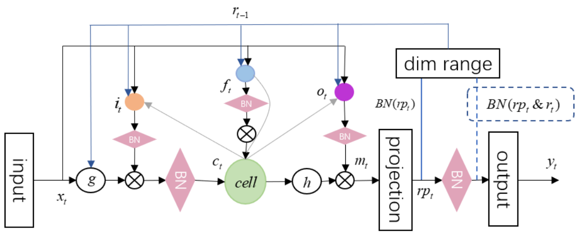

Figure 2: *Adding batch normalization in PLSTM, the $BN(rp_t\&r_t)$ connection is the first way we mention, and the $BN(rp_t)$ connection is the second way we describe.*

$$\begin{aligned}
i_t &= \sigma(W_{ix}x_t + W_{ir}r_{t-1} + W_{ic}c_{t-1} + b_i)\\
f_t &= \sigma(W_{fx}x_t + W_{fr}r_{t-1} + W_{fc}c_{t-1} + b_f)\\
c_t &= f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cr}r_{t-1} + b_c)\\
o_t &= \sigma(W_{ox}x_t + W_{or}r_{t-1} + W_{oc}c_t + b_o)\\
m_t &= o_t \odot h(c_t)\\
rp_t &= W_{pm}m_t\\
r_t &= \delta(rp_t)\\
y_t &= rp_t
\end{aligned} \tag{1}$$

Here σ means sigmoid nonlinearity function, $\odot$ suggesting element-wise multiplication. h means the tanh function and δ stands for the dim range operation, which chooses the first half part of the projection layer to produce the recurrent units. The dimension of cell and gates are 1024, and projection (rp_t) dimension is 512, recurrent (r_t) dim is 256.

3. Proposed method

3.1. Batch normalization in LSTMP

The phenomenon of covariate shift is a problem in deep learning, which reduces the training efficiency. With the continuous change on the statistics of training data, the layers of the model must constantly adapt to the changing distribution, which affects the training process. Batch normalization is a method to solve this problem properly.

3.1.1. Batch normalization on cell state and three gates

Batch normalization is implemented by standardizing the mean and variance of network activations, using the statistics in the current minibatch. The transform is as follows:

$$BN(h; \gamma, \beta) = \beta + \gamma \odot \frac{h - E[h]}{\sqrt{Var[h] + \epsilon}} \tag{2}$$

Where $E[h]$ and $Var[h]$ are estimated by the sample mean and variance of a minibatch. β and γ are two trainable parameters determining the mean and variance of the batch normalization output. We also take the β and γ as shifting and scaling, which is implemented by an affine layer in the LSTM architecture.

We apply the batch normalization to the bi-direction LSTMP model after cell state, called c_t BN as follows:

$$\begin{aligned}
o_t &= \sigma(W_{ox}x_t + W_{or}r_{t-1} + W_{oc}BN(c_t) + b_o)\\
m_t &= o_t \odot h(BN(c_t))
\end{aligned} \tag{3}$$

We use batch normalization after three gates as well, which is represented by $i_t\&f_t\&o_t$ BN in the following paper.

3.1.2. Batch normalization on the projection layer in two different ways

We also add the batch normalization on the projection layer. Since the recurrent layer comes from the projection layer, there might be some different details when adding BN to the projection:

Firstly, we directly apply the batch normalization after rp_t and using the $BN(rp_t)$ to generate r_t , as shown in the formulation (4). In such case, the batch normalization is added both on projection layer and recurrent operation, shown as $rp_t\&r_t$ BN.

$$\begin{aligned}
r_t &= \delta(BN(rp_t))\\
y_t &= BN(rp_t)
\end{aligned} \tag{4}$$

The second way is adding the BN only on the rp_t which is also named y_t sometimes. It means we are going to use the activation of rp_t before batch normalization to produce the r_t , called rp_t BN as follows:

$$\begin{aligned}
r_t &= \delta(rp_t)\\
y_t &= BN(rp_t)
\end{aligned} \tag{5}$$

We can observe the difference between the two ways described above in figure 2.

3.2. Combination with Per-frame dropout in LSTM

Per-frame dropout [12] was proposed to improve generalization in DNN training. It might randomly drop out some frames in a chunk, which brings some variation into the fixed chunk width during LSTM training. Unlike the traditional per-element dropout in which the dropout elements are chosen independently, in the per-frame dropout, the mask of a whole frame vector is all set to either zero or one. Different dropout rates can be used to determine the portion of the dropout frames. Moreover, it can be gradually modified by a schedule in training.

In this paper we carried out per-frame dropout on three different locations: 3 gates, cell state, the output . While dropout was used in a batch normalization neural network, we explored some combination patterns of these two technologies. When we add both two methods on the same place, the batch normalization will be used first followed by the per-frame dropout.

Table 1: *Batch normalization on different locations for Switchboard task*

model	location(BN)	swbd	callhm	total
BLSTMP	-	9.5	19.0	14.3
BLSTMP-BN	$i_t\&f_t\&o_t$	9.6	18.9	14.3
BLSTMP-BN	c_t	9.7	18.4	14.1
BLSTMP-BN	$rp_t\&r_t$	9.7	19.0	14.4
BLSTMP-BN	rp_t	**9.3**	**18.0**	**13.7**
BLSTMP-BN	r_t	10.0	18.9	14.5
BLSTMP-BN	$rp_t\&c_t$	**9.2**	**17.7**	**13.5**

4. Experiment setup

We evaluated our method on several LVCSR tasks, mainly on the 300-hour Switchboard task, using Kaldi speech recognition toolkit [14]. We also tested the stability and generalization of our model on AMI dataset, and 2300-hour Fisher+Switchboard task. The lattice-free MMI [13] criterion was used to train the acoustic model, as well as a cross entropy(CE) and L2 norm as the regularization terms. Our baseline model was 3layers bi-direction LSTMP (BLSTMP) model with 1024 units each layer and recurrent dim of 256, training for 4 epochs. The feature we used was 40-dimensional Mel-frequency cepstral coefficients (MFCCs) and 100 dimensional i-vectors [15]. We spliced the input frames of context ± 2 . In all the three tasks above, a speed perturbation method [16] was applied to augment the data to 3-fold.

For Switchboard, we report results on Hub5'2000 evaluation set. It contains Switchboard and CallHome subsets, which are indicated in the results as "swbd"and "callhm"respectively, and "total"means the results on the full Hub5'2000 test set. The results comparisons are mainly based on the "total"set. For convenience, we present the results on "swbd"and "callhm"subsets as well.

For AMI, we evaluated our model on individual headset microphone (IHM) and single distant microphone (SDM) tasks, and the alignments for SDM system was generated by the IHM model.

For Fisher+Switchboard task, the training data includes 2000 hours fisher data and 300 hours Switchboard data, around 2300 hours totally. The test set is the same as what we use in Switchboard task.

5. Experiment

This section presents our experiments of different aspects of the batch normalization in BLSTMP model. We investigated the effect of batch normalizing location in LSTM architecture. Then we explored the proper ways to combine the batch normalization with per-frame dropout. In all ASR experiments, we only report the word error rate (WER) results.

5.1. Batch normalization on different locations in LSTM

In the first set of experiments, we added the batch normalization in the model as we described in 3.1 section. We batch normalized the activations of different places in BLSTM model, then using an affine layer to operate the scaling and shifting: β and γ in the formulation.

The resulting word error rates are reported in Table 1. It can be seen that by adding batch-normalization only on the rp_t but not on r_t , we could achieve the best result 13.7%, which is

Table 2: *Different dropout locations (with dropout rate of 0.1) with batch normalization on $rp_t\&c_t$ for Switchboard task*

model	location(dro.)	swbd	callhm	total
BLSTMP	-	9.5	19.0	14.3
BLSTMP-BN	-	9.2	17.7	13.5
BLSTMP-BN-dro.	$i_t\&f_t\&o_t$	9.1	18.0	13.6
BLSTMP-BN-dro.	c_t	9.4	17.9	13.7
BLSTMP-BN-dro.	rp_t	**9.3**	**17.4**	**13.4**
BLSTMP-BN-dro.	$rp_t\&c_t$	9.4	18.0	13.7

Table 3: *Changing epochs for batch normalized ($rp_t\&c_t$) model with dropout on rp_t for Switchboard task*

model	epoch	rate	swbd	callhm	total
BLSTMP	4	-	9.5	19.0	14.3
BLSTMP	5	-	9.6	19.6	14.6
BLSTMP-BN	4	-	9.2	17.7	13.5
BLSTMP-BN-dro.	4	0.1	9.3	17.4	13.4
BLSTMP-BN-dro.	5	0.1	9.1	17.2	13.1
BLSTMP-BN-dro.	6	0.1	**9.0**	**16.8**	**12.9**
BLSTMP-BN-dro.	7	0.1	8.9	17.1	13.0

4.2% relative reduction than baseline on WER. Adding batch-normalization to the cell state also benefits the performance. We also find that batch-normalizing the r_t degrades the results. Furthermore, the results on $rp_t\&r_t$ is worse than only on rp_t. From these, we can infer that using batch normalization on r_t in LSTM is not a good choice for speech recognition.

Table 1 shows that adding batch normalization to c_t and rp_t can both make improvement on results. Thus, we apply batch normalization on both locations. The results are on the last line of table 1. We obtain a little more gain by using batch normalization on those two places simultaneously. It reduces the WER of 5.6% relatively.

5.2. Cooperation with per-frame dropout

By observing the training loss plot, we found that overfitting was still an issue in the BN-LSTM. Therefore, we decided to use the per-frame dropout to relieve the overfitting. As we described in section 3.2, we 've tried various of combinations of these two methods. In the following experiments, we used the batch normalization on both rp_t and c_t , since it led to the best results so far.

Table 2 shows results of using dropout on different locations on BN-LSTM. In this Table, the dropout rate we used is 0.1. As we can see, adding dropout on rp_t makes little improvement on the performance. But we don 't get better results by other means.

Then we tried to use more training epochs and change the dropout rate to see the impact. In all the former experiments, we used 4 training epochs. Here we applied the $rp_t\&c_t$ batch normalization and rp_t per-frame dropout with rate 0.1.

Results are presented in Table 3. Simply by adding the epoch to 5 will degrade the performance on the baseline. However, on the model with batch normalization and dropout, when we increase the training epoch to 5 and 6, we get further improvement in recognition results. The best of our results is through using 6 epochs, which is 1.4% absolute and 9.8% relative reduction on total Hub5'2000 test set. On CallHome subset,

Table 6: *Batch normalization on rp_t&c_t with dropout on rp_t (rate is 0.1) for AMI data set*

Model	epoch	AMI IHM		AMI SDM			
		Dev	Eval	Dev4	Dev1	Eval4	Eval1
BLSTMP	4	21.9	22.0	41.3	31.8	45.0	34.3
BLSTMP	5	22.3	22.5	41.7	32.4	45.4	34.6
BLSTMP-BN	4	21.7	21.4	41.0	31.2	44.5	33.5
BLSTMP-BN-dro.	4	21.6	21.2	40.4	30.5	44.3	33.4
BLSTMP-BN-dro.	5	**21.3**	**20.8**	**40.3**	**30.3**	**43.7**	**32.8**
BLSTMP-BN-dro.	6	21.5	21.0	40.5	30.8	44.1	33.2

Table 4: *Modifying the dropout rate on Switchboard task*

model	epoch	rate	swbd	callhm	total
BLSTMP	4	-	9.5	19.0	14.3
BLSTMP-BN-dro.	6	0.1	**9.0**	**16.8**	**12.9**
BLSTMP-BN-dro.	4	0.08	9.2	17.6	13.4
BLSTMP-BN-dro.	5	0.08	9.0	17.2	13.1
BLSTMP-BN-dro.	6	0.08	9.0	17.1	13.1
BLSTMP-BN-dro.	4	0.15	8.9	17.6	13.3
BLSTMP-BN-dro.	5	0.15	8.9	17.5	13.2
BLSTMP-BN-dro.	6	0.15	8.9	17.6	13.3

Table 5: *Batch normalization on rp_t&c_t with dropout on rp_t (rate is 0.1) for Fisher+Switchboard task*

model	epoch	swbd	callhm	total
BLSTMP	4	8.5	15.0	11.9
BLSTMP-BN-dro.	4	**8.1**	**14.4**	**11.4**

we can see nearly 12% relative improvement.

Table 4 shows word error rates for different dropout rates. With the dropout rates of 0.15 and 0.08, we also include results for different epochs here.

5.3. Experiments on Fisher+Switchboard dataset

Then we evaluated our method on 2300 hours Fisher+Switchboard dataset using the rp_t&c_t batch normalization and per-frame dropout on rp_t , and the dropout rate was 0.1. We used the same structure of BLSTM, training with LF-MMI criterion and 3-fold speed perturbation.

As shown in Table 5, the proposed method leads to 0.5% absolute and 4.2% relative reduction on total Hub5'2000 test set with 4 training epochs.We can see the generalization of the proposed method on a large dataset.

5.4. Experiments on AMI dataset

Based on the results so far, we decided to test our batch normalization with dropout on AMI dataset. To accomplish that, we were using the consistent configurations as in Switchboard experiments. We tested the network on IHM and SDM sets. When scoring the test sets of SDM, we used two kinds of overlap speakers limitation, 1 and 4, shown in the table as Eval1 and Eval4 respectively.

Finally, Table 6 shows results of adding batch normalization after both output and cell state on AMI data. Then we combined the BN-BLSTM with dropout on rp_t , which resulted in extra gain as well. And it yielded benefits similar with the Switchboard task. As the results show, cooperating with dropout, the BN-BLSTM gets the best results when training for 5 epochs.

6. Conclusions

In this paper, we explored some different ways of adding batch normalization on LSTM architecture for ASR. We have demonstrated that batch-normalizing the output of hidden layers and the cell states in LSTM can both improve the performance on speech recognition. Moreover, normalizing those two locations simultaneously gets the highest promotion. The batch normalized LSTM can cooperate with per-frame dropout appropriately, which yields large reduction comparing with the baseline model. On Switchboard task, the proposed method achieves nearly 10% relative reduction on WER for Hub5'2000 total test set. Further experiments on Fisher+Switchboard and AMI datasets prove the reliability and generalization of this method. As a result, we demonstrate that proper batch normalization with per frame dropout in the LSTM architecture can make the network generalize better and benefit the results on speech recognition.

7. Acknowledgements

This work is partially supported by the National Key Research and Development Plan (Nos.2016YFB0801203, 2016YFB0801200), the National Natural Science Foundation of China (Nos.11590770-4, U1536117,11504406,11461141004), the Key Science and Technology Project of the Xinjiang Uygur Autonomous Region (No.2016A03007-1), the Pre-research Project for Equipment of General Information System (No.JZX2017-0994/Y306). We'd like to thank the cooperation project with the Sony (China) Limited.

8. References

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

[2] L. Jing, C. Gulcehre, J. Peurifoy, Y. Shen, M. Tegmark, M. Soljačić, and Y. Bengio, "Gated orthogonal recurrent units: On learning to forget," *arXiv preprint arXiv:1706.02761*, 2017.

[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] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual Conference of the International Speech Communication Association*, 2014.

[5] T. Mikolov, "Statistical language models based on neural networks," *Presentation at Google, Mountain View, 2nd April*, 2012.

[6] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[8] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, “Batch normalized recurrent neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 2657–2661.

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

[10] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville, “Recurrent batch normalization,” *arXiv preprint arXiv:1603.09025*, 2016.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[12] G. Cheng, V. Peddinti, D. Povey, V. Manohar, S. Khudanpur, and Y. Yan, “An exploration of dropout with lstms,” in *Proceedings of Interspeech*, 2017.

[13] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, “Purely sequence-trained neural networks for asr based on lattice-free mmi.” in *Interspeech*, 2016, pp. 2751–2755.

[14] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, “The kaldi speech recognition toolkit,” in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[15] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, “Speaker adaptation of neural network acoustic models using i-vectors.” in *ASRU*, 2013, pp. 55–59.

[16] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, “Audio augmentation for speech recognition,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.