IMPROVING LATENCY-CONTROLLED BLSTM ACOUSTIC MODELS FOR ONLINE SPEECH RECOGNITION

Shaofei Xue, Zhijie Yan

Alibaba Inc., Beijing, China

{shaofei.xsf, zhijie.yzj}@alibaba-inc.com

ABSTRACT

Bidirectional long short-term memory (BLSTM) recurrent neural networks are powerful acoustic models in terms of recognition accuracy. When BLSTM acoustic models are used in decoding, the speech decoder needs to wait until the end of a whole sentence is reached, such that forward-propagation in the backward direction can then be performed. The nature of BLSTM acoustic models makes them inappropriate for real-time online speech recognition because of the latency issue. Recently, the context-sensitive-chunk BLSTM and latency-controlled BLSTM acoustic models have been proposed, both chop a whole sentence into several overlapping chunks. By appending several left and/or right contextual frames, forward-propagation of BLSTM can be down within a controlled time delay, while the recognition accuracy is maintained when comparing with conventional BLSTM models. In this paper, two improved versions of latency-controlled BLSTM acoustic models are presented. By using different types of neural network topology to initialize the BLSTM memory cell states, we aim at reducing the computational cost introduced by the contextual frames and enabling faster online recognition. Experimental results on a 320-hour Switchboard task have shown that the improved versions accelerate from 24% to 61% in decoding without significant loss in recognition accuracy.

***Index Terms*—** BLSTM, latency-controlled BLSTM, online speech recognition

1. INTRODUCTION

Recently, bidirectional long short-term memory (BLSTM) based acoustic models have greatly improved accuracy on many tasks in automatic speech recognition (ASR) [1, 2, 3, 4]. BLSTM is powerful because the past and future contextual information can be simultaneously utilized for each time instance, which leads to higher prediction accuracy. BLSTM often performs better on sequence classification tasks than LSTM [5, 6, 7] and feed-forward neural networks (FFNN) [8, 9]. However, the conventional BLSTM acoustic models are inappropriate for real-time online speech recognition because they need a whole sentence for decoding which introduces unbearable latency.

Several methods have been developed to enable online speech recognition with BLSTM acoustic models. A context-sensitive-chunk (CSC) BLSTM method is proposed in [10, 11]. In this method, a speech sentence is firstly split into non-overlapping chunks of fixed length N_c. Then N_l past and N_r future contextual frames are appended before and after each chunk (a typical setup could be $N_c = 60$, and $N_l = N_r = 30$). The appended frames are only used to initialize BLSTM memory cell states by providing contextual information. They do not generate error signals nor posterior probabilities during training and decoding. Online decoding becomes feasible using CSC-BLSTM because forward-propagation can be done chunk-by-chunk in a streaming manner. The cost is that $N_l + N_r$ more time steps are evaluated for each chunk than conventional BLSTM.

Later, latency-controlled (LC) BLSTMs are proposed in [12] which avoids the computation of left context. The memory cell states of the forward direction of the BLSTM are directly initialized by carrying over states from the previous chunk, while the memory cell states of the backward direction are still initialized using N_r future frames like CSC-BLSTM. In this way, the extra evaluation overhead is reduced to N_r time steps for each chunk.

The motivation of this work is to develop methods that further decrease the computation overhead of latency-controlled BLSTM, enable faster real-time online decoding, while maintaining the recognition accuracy. The N_r future contextual frames are mainly used for two purposes: 1) to initialize the memory cell states in the backward direction, and 2) to calculate the output activations which are essentially used to initialize the memory cell states for the upper layers. Therefore, only to fulfil the initialization purposes, a less accurate but more efficient feed-forward neural network could be adopted to replace the time-consuming BLSTM. In our experiments on a 320-hour Switchboard task, this method accelerates about 40% in decoding and maintains comparable word error rate (WER) with the baseline LC-BLSTM.

Another method that further modifies the LC-BLSTM topology is also studied. This method keeps the forward direction of BLSTM unchanged, and uses simple RNNs [13] with rectified linear units (ReLU) [14] to replace backward direction LSTM cells. Because the evaluation of simple RNNs is much faster than LSTMs, the backward computation can thus be reduced. Target delay needs to be involved in this case because it has been shown that target delay is important to simple RNNs. Experimental results show that this model topology speeds up about 61% in decoding with only 0.3% increase in WER. Further, if we increase the number of memory cells a little bit, no significant accuracy loss can be achieved with 27% acceleration compared to the baseline BLSTM.

The rest of this paper is organized as follows: In Section 2 we briefly review some previous works. In Section 3 two types of improved LC-BLSTM models are introduced. Section 4 shows our experimental setups and detailed results. Conclusions are drawn in Section 5.

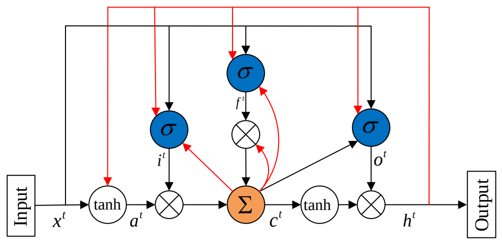

Fig. 1. The architecture of LSTM network with a memory cell.

2. PREVIOUS WORK

2.1. LSTM

As shown in Fig.1, the standard LSTM cell architecture contains one self-connected cell and three controlling gates (the red lines indicate time-delayed connections). In the memory cell, input gate and output gate manage information flow into and out of the memory cell. Meanwhile, the forget gate is used to provide a way for the cell to reset themselves. Furthermore, there are peephole weights connecting the gates to the cell, which are used for obtaining more accurate Constant Error Carousel (CEC) information [15] and improving the LSTM's ability to learn precise timing and counting of the internal states. A series of researches have demonstrated that LSTM alleviates the vanishing gradient and exploding gradient problems to some extent. The operation of the network follows the following equations:

$$i^t = \sigma(W_{xi}x^t + W_{hi}h^{t-1} + W_{ci}c^{t-1} + b_i) \tag{1}$$
$$f^t = \sigma(W_{xf}x^t + W_{hf}h^{t-1} + W_{cf}c^{t-1} + b_f) \tag{2}$$
$$a^t = \tanh(W_{xc}x^t + W_{hc}h^{t-1} + b_c) \tag{3}$$
$$c^t = f^t \odot c^{t-1} + i^t \odot a^t \tag{4}$$
$$o^t = \sigma(W_{xo}x^t + W_{ho}h^{t-1} + W_{co}c^t + b_o) \tag{5}$$
$$h^t = o^t \odot \tanh(c^t) \tag{6}$$

where x^t is the input, i, f, o and a are the input gate, forget gate, output gate and cell input activation vector, respectively, c is a self-connected state vector, and all of them are of the same size as the hidden vector h. W_{ci}, W_{cf} and W_{co} are peephole connection weights which are diagonal. σ denotes sigmoid activation function.

2.2. BLSTM

Unidirectional LSTMs only make use of past context. In contrast, bidirectional LSTMs [5] use both the past and future context by processing the input in both forward and backward directions with two separated hidden layers, which are then concatenated as the final output. BLSTM acoustic models often achieve better performance in speech recognition than LSTMs. It comes with a cost that significant latency needs to be introduced since decoding needs to wait until seeing a whole sentence. This makes BLSTM acoustic models inappropriate for real-time online speech recognition.

2.3. CSC-BLSTM and LC-BLSTM

To reduce decoding latency and also speed up the training process of BLSTMs, CSC-BLSTM [11] and LC-BLSTM [12] are proposed.

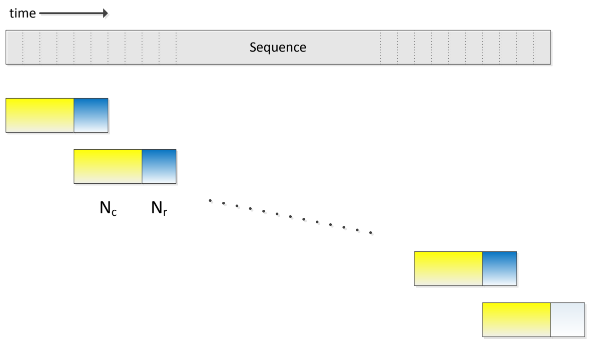

Fig. 2. Illustration of latency-controlled BLSTM.

LC-BLSTM is a more efficient version because it carries over the whole past history from chunk to chunk, while only using a truncated future context to initialize the memory cell states in backward direction. As shown in Fig.2, a sentence is firstly split into non-overlapping chunks of fixed length N_c, and then N_r future frames are appended as right context. During training process the appended frames generate no output, so no error signals will be generated from the contextual frames. Training LC-BLSTM is generally more than 20 times faster than traditional BLSTM since it is memory-efficient, so many utterances can be grouped in a "mini-batch" and processed in parallel.

In decoding, the initial memory cell states of the forward direction can be directly copied from the previous chunk. The memory cell states of the backward direction are initialized by evaluating the N_r contextual frames. Therefore, the latency is controlled by adjusting N_r in decoding. The significant decreasing in latency comes at cost of a slightly increasing of computational overhead. For each chunk, the decoding overhead using LC-BLSTM is increased by a factor of $\frac{N_r}{N_c+N_r}$ compared with conventional BLSTMs. It has been shown in previous experiments that if one wants to maintain recognition accuracy, N_r needs to be set large enough, e.g., $N_c = 22$ and $N_r = 21$ [12]. This overhead directly affects the real-time factor (RTF) in decoding and thus cannot be ignored.

3. IMPROVED LC-BLSTM ACOUSTIC MODELS

In this section, two improved LC-BLSTM acoustic models in terms of efficiency are introduced to enable faster real-time online speech recognition.

3.1. Forward approximation and backward DNN initialization

The evaluation steps of each BLSTM layer can be decomposed as follows: In forward direction, we first compute from frame $\#1$ to frame $\#N_c$, the output activations will be fed as the input to the next layer until the final output (a vector of posterior probabilities) is predicted. The memory cell states of frame $\#N_c$ will also serve as the initial states for the next chunk. Then from frame $\#(N_c+1)$ to $\#(N_c+N_r)$, although the output activations are fed to the next layer similarly, they are only used to initialize the backward memory cell states of the next layer, and do not contribute to final output (posterior probabilities) directly.

In backward direction, we firstly initialize the backward memory

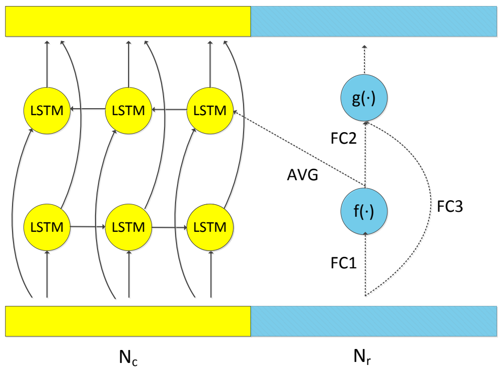

Fig. 3. Topology of LC-BLSTM-FABDI.

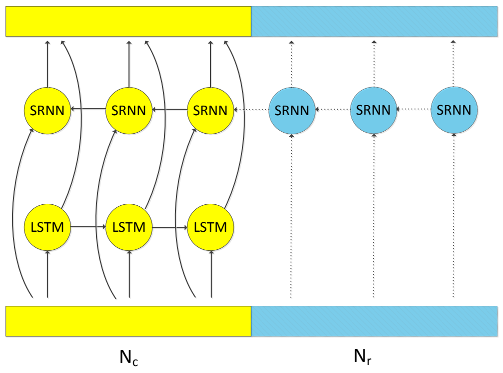

Fig. 4. Topology of LC-BLSTM-FABSR.

cell states to 0 and the computation starts from frame $\#(N_c + N_r)$ to $\#(N_c + 1)$. After this step, the backward memory cell states of the current layer at frame $\#N_c$ are initialized. Again the activations generated in this step only contribute to the initialization of the backward memory cell states of the next layer. Then, computation continues from frame $\#N_c$ to $\#1$ which generates activations output to the next layer and in-term contribute to the final posterior probabilities.

Note that the computation using the N_r future context frames is mainly to provide initialization to the memory cell states in the backward direction. So we first try to discard the forward computation from frame $\#(N_c + 1)$ to $\#(N_c + N_r)$ completely, which is named as forward approximation (FA). In this case, the forward part of the activations generated for the N_r contextual frames are approximately set to 0. After that, a feed-forward neural network is introduced in the backward direction which takes the N_r right contextual frames as input, generates activations fed to the next layer, and also initializes backward memory cell states for the current layer. This new topology is named "forward approximation and backward DNN initialization" (FABDI), and can be illustrated in Fig. 3.

The feed-forward structure may contain three full connections (marked as FC1, FC2 and FC3) and two nonlinear functions, or we can try different combinations of them (FC1+FC2 and FC1+FC3). An average operation (marked as AVG) is conducted to collapse N_r vectors into a single vector, which is used to initialize the backward memory cell states for frame $\#T_c$. We choose logistic sigmoid for $f(\cdot)$ and ReLU for $g(\cdot)$ in this study. The output activations generated by $g(\cdot)$ serve as the right-context frames N_r for the next layer.

The feed-forward structure along with the BLSTMs are jointly trained in the training process using back-propagation. Because a feed-forward structure is used instead of backward LSTM for initialization using N_r contextual frames, the training and evaluation efficiency can be significantly improved. The problem remains is that whether and how much this approximation affects speech recognition accuracy. This is examined in our experiments empirically.

3.2. Forward approximation and backward simple RNN

The second topology is named "forward approximation and backward simple RNN" (FABSR). As illustrated in Fig.4, we use forward LSTM and discard the computing for N_r contextual frames as in FABDI. The backward direction is completely replaced by using simple recurrent neural networks (SRNN) [13].

One of the key challenges for training SRNNs is that long-term dependency is difficult to handle because of the gradient-exploding problem. We apply more strict gradient-clipping in the training process of LC-BLSTM-FABSRs to improve convergence.

4. EXPERIMENTS

The two improved latency-controlled BLSTM topologies are evaluated on a 320-hr Switchboard task. The Switchboard training data consists of 309 hour Switchboard-I training set and 20 hour Call Home English training set (1540 speakers in total). The NIST 2000 Hub5e set (containing 1831 utterances from 40 speakers) is used as the evaluation set in this work. For feature extraction, waveform signal is analyzed using a 25-ms Hamming window with a 10-ms fixed frame rate. We use 39-dimensional MFCC features (static, first- and second-order derivatives) to train a standard tri-phone GMM-HMM model consisting of 8882 tied states based on the maximum likelihood estimation (MLE). The GMM-HMM models are used to obtain state-level labels by forced-alignment. 108-dimensional filterbank features (static, first- and second-order derivatives) are used for training all latency-controlled BLSTM models. A 4-gram language model (LM) is trained using 3M words of the training transcripts and 11M words of the Fisher English Part 1 transcripts.

The weights in all the networks are initialized to the range (-0.01, 0.01) with a uniform distribution. The initial learning rate is set to 1e^{-5}, the momentum is kept as 0.9. We use $N_c = 60$ and $N_r = 30$ in the training process and try with different setups in testing. All models are trained in a distributed manner using asynchronous stochastic gradient descent (ASGD) [16] optimization on 4 GPUs. A hybrid LC-BLSTM model stacked with 3 BLSTM layers (500 memory cells for each direction), 2 ReLU layers (2048 hidden nodes for each layer) and 1 Softmax output layer is trained as the baseline using frame-level cross entropy (CE) criterion.

4.1. Comparison of different configurations in decoding LC-BLSTM baseline

We first evaluate the performance in decoding with different configurations of N_c and N_r on the LC-BLSTM baseline. As shown in Table 1, full-latency BLSTM ($N_c = \infty$) provides the lower-bound WER (13.0%) as expected. As we gradually decrease the latency by varying N_c and N_r's, the WER increases accordingly. It can be seen from the results that longer context N_r is helpful to maintain

Table 1. *WER of decoding with different N_c and N_r on LC-BLSTM baseline.*

$N_c + N_r$	WER (%)
$\infty + /$	13.0
$60 + 60$	13.1
$60 + 30$	13.2
$30 + 30$	13.3
$30 + 15$	13.6
$20 + 20$	13.5
$20 + 10$	14.3

Table 2. *Performance of FABDI methods with different connection schemes.*

	WER (%)	speed-up
LC-BLSTM ($N_c = N_r = 30$)	13.3	-
FA only	13.2	1.24x
FABDI (FC1+FC2+FC3)	13.4	1.40x
FABDI (FC1+FC2)	13.6	1.44x
FABDI (FC1+FC3)	13.5	1.44x

accuracy. This is consistent with previous work [11]. We choose $N_c = N_r = 30$ as the LC-BLSTM baseline in all following experiments because it is a good trade-off between decoding latency and WER (only 0.3% absolute worse than the full-latency setup).

4.2. Forward approximation and backward DNN initialization

We present the WER along with the relative speed-up in decoding for each FABDI setup. A modified version of publicly available Kaldi decoder is used in all experiments on an Intel Xeon E5-2682 v4 CPU to measure the improvements in decoding speed.

We first try to use forward approximation only, and the results are given as "FA only" in Table 2. The forward approximation achieves 1.24x speed-up, and does not degrade recognition accuracy. Then we try different combinations on the feed-forward structure. The node size in backward DNN part is set to 250 in all comparisons. When FC1, FC2 and FC3 are all used, comparable WER is obtained, while a 1.40x speed-up can be achieved. We also try different setups like FC1+FC2 and FC1+FC3, both achieve a slightly better speed-up of 1.44x, and the WERs increase a little bit from 0.2% to 0.3% absolute, respectively.

The results suggest that the forward evaluation on the N_r contextual frames is not critical to prediction accuracy. Simply by discarding this, 24% acceleration can be achieved. Our FABDI method using all three feed-forward connections is a good trade-off. 40% acceleration can be achieved using this setup at a cost of only 0.1% absolute WER increase from the baseline.

4.3. Forward approximation and backward simple RNN

In training LC-BLSTM-FABSRs, gradient-clipping is set to 5 for LSTM and 1 for SRNN. Firstly, we use same number of nodes as the LC-BLSTM baseline (500 memory cells for each direction). In this case, the number of the model parameters is 3.3M, which is 15% less than that of the baseline (3.9M). As shown in Table 3, it seems that target delay [5] is useful to achieve better accuracy for SRNN. The WER increases to 13.9% (0.6% absolute loss) without target delay and 13.6% (0.3% absolute loss) with target delay of 5 time-steps. These setups both achieve a speed-up of 1.61x, which is appealing.

Table 3. *Performance of FABSR methods with different target delay and number of nodes.*

	WER (%)	speed-up
LC-BLSTM ($N_c = N_r = 30$)	13.3	-
FABSR(500)+ target delay 0	13.9	1.61x
FABSR(500)+ target delay 5	13.6	1.61x
FABSR(600)+ target delay 5	13.3	1.27x

Next, we increase the number of nodes for LSTM and SRNN to 600 such that the model now has similar amount of free parameters (3.8M) to the baseline. The results show that by using this setup, LC-BLSTM-FABSR is able to obtain a comparable WER as the baseline, while still gives a speed-up of 1.27x.

5. CONCLUSION

In this paper, we present two improved version of latency-controlled BLSTM acoustic models to speed-up the evaluation and enable faster real-time online speech recognition. Firstly, an FABDI method is introduced, which discards the computation on contextual frames in the forward direction, and use a feed-forward structure in backward direction to initialize the memory cell states. Secondly, an FABSR method is used, which uses simple RNNs with ReLU units to replace LSTM in the backward direction. Experimental results show that different setups of the two methods accelerate from about 24% to 61% in decoding, without significant loss in WER.

6. REFERENCES

[1] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber, "Bidirectional LSTM networks for improved phoneme classification and recognition," in *International Conference on Artificial Neural Networks*, 2005, pp. 799–804.

[2] Martin Wollmer, Florian Eyben, Bjorn Schuller, and Gerhard Rigoll, "A multi-stream ASR framework for BLSTM modeling of conversational speech," in *IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 4860–4863.

[3] Alex Graves, Jaitly Navdeep, and Mohamed Abdel-rahman, "Hybrid speech recognition with deep bidirectional LSTM," in *Automatic Speech Recognition and Understanding (ASRU)*, 2013.

[4] Zeyer Albert, Schluter Ralf, and Ney Hermann, "Towards online-Recognition with deep bidirectional LSTM acoustic models," in *INTERSPEECH*, 2016.

[5] Alex Graves and Jürgen Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[6] Felix Gers, Nicol Schraudolph, and Jürger Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, vol. 3, no. 1, pp. 115–143, 2010.

[7] Hasim Sak, Andrew Senior, and Francoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH*, 2014, pp. 338–342.

[8] George Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[9] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[10] Kai Chen, Zhijie Yan, and Qiang Huo, "A context-sensitive-chunk BPTT approach to training deep LSTM/BLSTM recurrent neural networks for offline handwriting recognition," in *ICDAR*, 2015, pp. 411–415.

[11] Kai Chen, Zhijie Yan, and Qiang Huo, "Training deep bidirectional LSTM acoustic model for LVCSR by a context-sensitive-chunk BPTT approach," in *INTERSPEECH*, 2015, pp. 3600–3604.

[12] Yu Zhang, Guoguo Chen, Dong Yu, and Kaisheng Yao, "Highway long short-term memory RNNs for distant speech recognition," in *IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*, 2016, pp. 5755–5759.

[13] Oriol Vinyals, Suman Ravuri, and Daniel Povey, "Revisiting recurrent neural networks for robust ASR," in *IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4085–4088.

[14] George Dahl, Tara Sainath, and Geoffrey Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *IEEE International Conference of Acoustics,Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8609–8613.

[15] Alex Graves, *Supervised sequence labelling with recurrent neural networks*, vol. 385, Springer, 2012.

[16] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, et al., "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.