Layer Trajectory LSTM

Jinyu Li, Changliang Liu, Yifan Gong

Microsoft AI and Research

{jinyli, chanliu, ygong}@microsoft.com

Abstract

It is popular to stack LSTM layers to get better modeling power, especially when large amount of training data is available. However, an LSTM-RNN with too many vanilla LSTM layers is very hard to train and there still exists the gradient vanishing issue if the network goes too deep. This issue can be partially solved by adding skip connections between layers, such as residual LSTM. In this paper, we propose a layer trajectory LSTM (LT-LSTM) which builds a layer-LSTM using all the layer outputs from a standard multi-layer time-LSTM. This layer-LSTM scans the outputs from time-LSTMs, and uses the summarized layer trajectory information for final senone classification. The forward-propagation of time-LSTM and layer-LSTM can be handled in two separate threads in parallel so that the network computation time is the same as the standard time-LSTM. With a layer-LSTM running through layers, a gated path is provided from the output layer to the bottom layer, alleviating the gradient vanishing issue. Trained with 30 thousand hours of EN-US Microsoft internal data, the proposed LT-LSTM performed significantly better than the standard multi-layer LSTM and residual LSTM, with up to 9.0% relative word error rate reduction across different tasks.

Index Terms: speech recognition, LSTM, layer trajectory, factorized gate

1. Introduction

Recently, significant progress has been made in automatic speech recognition (ASR) when switching from the deep neural networks (DNNs) [1] to recurrent neural networks (RNNs) with long short-term memory (LSTM) units [2], which solve the gradient vanishing or exploding issues in standard RNNs by using input, output and forget gates to achieve a network that can maintain state and propagate gradients in a stable fashion over long spans of time. These LSTM-RNNs [3, 4, 5, 6, 7] and their variants such as two-dimensional LSTM-RNNs [8, 9, 10] have been shown to outperform DNNs on a variety of ASR tasks.

It is popular to stack multiple LSTM layers to get better modeling power [4], especially when large amount of training data is available. However, an LSTM-RNN with too many vanilla LSTM layers is very hard to train and there still exists the gradient vanishing issue if the network goes too deep [11, 12]. This issue can be partially solved by adding skip connections or gating functions between layers.

Residual LSTM [13, 14] uses shortcut connections between LSTM layers, and hence provides a way to alleviate the gradient vanishing problem. In the highway LSTM [11], memory cells of adjacent layers are connected by gated direct links which provide a path for information to flow between layers more directly without decay. Therefore, it alleviates the gradient vanishing issue and enables the training of much deeper LSTM-RNN networks. In [15], highway LSTM was investigated with large scale of training data, but only very limited gain was obtained over the standard multi-layer LSTM. Grid LSTM [16] is a more general LSTM which arranges the LSTM memory cells into a multidimensional grid along both time and layer axis. It was extended in [12, 17] as prioritized grid LSTM which was shown to outperform highway LSTM on several ASR tasks.

All the aforementioned models work in a layer-by-layer and step-by-step fashion. The output of a LSTM unit (either the standard time LSTM or grid LSTM) is used as the input of the LSTM at the same time step in the next layer and the recurrent input of the LSTM at the next time step in the same layer. The output of the highest layer LSTM is used for final senone (tied triphone states) classification. However, it may not be optimal that the LSTM outputs serve the purpose of both recurrence along time axis (for temporal modeling) and senone classification along the layer axis (for target discrimination).

In this paper, we decouple the purposes of time recurrence and senone classification by proposing a layer trajectory LSTM (LT-LSTM) which builds a layer-LSTM using the outputs from all the layers of a standard multi-layer time-LSTM. The time-LSTM is used for temporal modeling with time recurrence, while the layer-LSTM scans the outputs from multiple time-LSTM layers and uses the summarized layer trajectory information for final senone classification. Hence, the forward-propagation of the time-LSTM in next frame is independent of the calculation of the layer-LSTM in current frame, therefore the evaluation of time-LSTM and layer-LSTM can be handled in two separate threads in parallel and the network computational time can be kept the same as the standard time-LSTM. With a layer-LSTM running through layers, a gated path is provided from the output layer to the bottom layer, reducing the gradient vanishing issue. We evaluate the proposed method by training various models with 30 thousand (k) hours of EN-US data which pools from Microsoft Cortana, Conversation, and xBox data with mixed close-talk and far-field utterances. The proposed LT-LSTM is significantly better than the standard multi-layer LSTM and residual LSTM.

The rest of the paper is organized as follows. In Section 2, we explore different LSTM structures: standard multi-layer LSTM, Residual LSTM, and the proposed LT-LSTM. We also propose a new way to reduce the computational cost of LSTM by factorizing the gates calculation. We evaluate the proposed models in Section 3, and conclude our study in Section 4.

2. Exploring LSTM structures

In this section, we first introduce the standard multi-layer LSTM and residual LSTM (ResLSTM). Then, we describe our proposed layer trajectory LSTM. Finally, a factorized gate LSTM is proposed to reduce the computational cost of LSTM units.

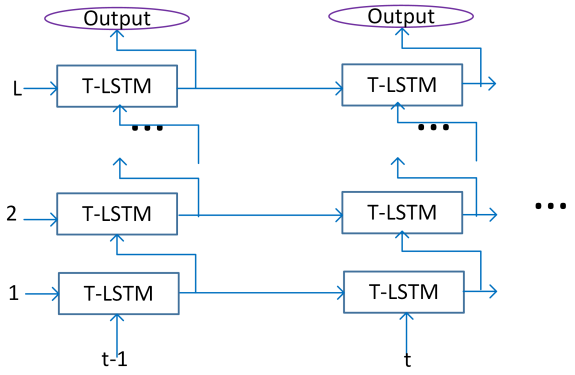

Figure 1: *Flowchart of multi-layer time-LSTM (T-LSTM). The output of a T-LSTM is used as the input of the T-LSTM at the same time step in the next layer and the recurrent input of the T-LSTM at the next time step in the same layer.*

2.1. LSTM

The standard LSTM is a time-LSTM which does temporal modeling via time recurrence by taking the output of time-LSTM at previous time step as the input of the time-LSTM at current time step. To increase modeling power, multiple layers of LSTM units are stacked together to form a multi-layer LSTM which is shown in Figure 1. At time step t, the vector formulas of the computation of the l-th layer LSTM units can be described as:

$$\mathbf{i}_t^l = \sigma(\mathbf{W}_{ix}^l \mathbf{x}_t^l + \mathbf{W}_{ih}^l \mathbf{h}_{t-1}^l + \mathbf{p}_i^l \odot \mathbf{c}_{t-1}^l + \mathbf{b}_i^l) \quad (1)$$

$$\mathbf{f}_t^l = \sigma(\mathbf{W}_{fx}^l \mathbf{x}_t^l + \mathbf{W}_{fh}^l \mathbf{h}_{t-1}^l + \mathbf{p}_f^l \odot \mathbf{c}_{t-1}^l + \mathbf{b}_f^l) \quad (2)$$

$$\mathbf{c}_t^l = \mathbf{f}_t^l \odot \mathbf{c}_{t-1}^l + \mathbf{i}_t^l \odot \phi(\mathbf{W}_{cx}^l \mathbf{x}_t^l + \mathbf{W}_{ch}^l \mathbf{h}_{t-1}^l + \mathbf{b}_c^l) \quad (3)$$

$$\mathbf{o}_t^l = \sigma(\mathbf{W}_{ox}^l \mathbf{x}_t^l + \mathbf{W}_{oh}^l \mathbf{h}_{t-1}^l + \mathbf{p}_o^l \odot \mathbf{c}_t^l + \mathbf{b}_o^l) \quad (4)$$

$$\mathbf{h}_t^l = \mathbf{o}_t^l \odot \phi(\mathbf{c}_t^l) \quad (5)$$

where $\mathbf{x}_t^l$ is the input vector for the l-th layer with

$$\mathbf{x}_t^l = \begin{cases} \mathbf{h}_t^{l-1}, & \text{if } l > 1 \\ \mathbf{s}_t, & \text{if } l = 1 \end{cases} \quad (6)$$

$\mathbf{s}_t$ is the speech spectrum input at time step t. The vectors $\mathbf{i}_t^l$, $\mathbf{o}_t^l$, $\mathbf{f}_t^l$, $\mathbf{c}_t^l$ are the activation of the input, output, forget gates, and memory cells, respectively. $\mathbf{h}_t^l$ is the output of the time-LSTM. $\mathbf{W}_{.x}^l$ and $\mathbf{W}_{.h}^l$ are the weight matrices for the inputs $\mathbf{x}_t^l$ and the recurrent inputs $\mathbf{h}_{t-1}^l$, respectively. $\mathbf{b}_.^l$ are bias vectors. $\mathbf{p}_i^l$, $\mathbf{p}_o^l$, $\mathbf{p}_f^l$ are parameter vectors associated with peephole connections. The functions σ and ϕ are the logistic sigmoid and hyperbolic tangent nonlinearity, respectively. The operation $\odot$ represents element-wise multiplication of vectors.

From Figure 1, we can see that the output of a time-LSTM is used as the input of the time-LSTM at the same time step in the next layer and the recurrent input of the time-LSTM at the next time step in the same layer. The last hidden layer's output is used to predict senone labels for senone classification. Therefore, the same output is used for the purpose of temporal model along time axis and the purpose of target discrimination along the layer axis. However, these two purposes are indeed very different. Hence, the standard time-LSTM modeling may not be optimal.

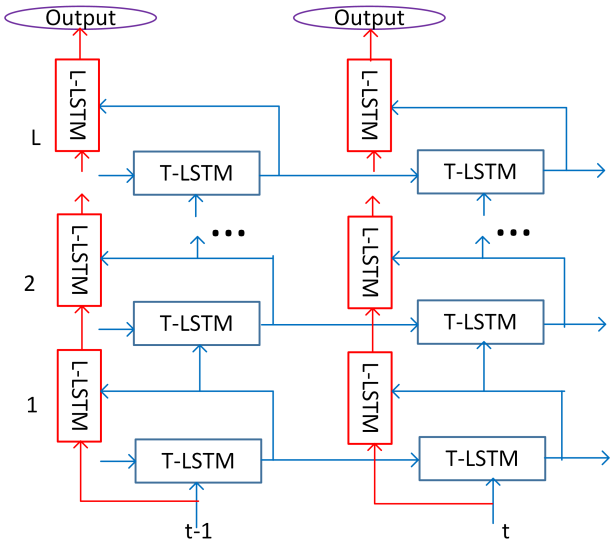

Figure 2: *Flowchart of layer trajectory LSTM (LT-LSTM). Layer-LSTM (L-LSTM) is used to scan the outputs of time-LSTM (T-LSTM) across all layers at the current time step to get summarized layer trajectory information for senone classification. There is no time recurrence for L-LSTM. Time recurrence only exists between T-LSTMs at different time steps.*

2.2. Residual LSTM

Similar to Residual CNN [18] which recently achieves great success in the image classification task, residual LSTM (ResLSTM) is very straightforward with the direct shortcut path across layers by changing Eq. (6) to Eq. (7) so that gradient vanishing issue can be partially solved.

$$\mathbf{x}_t^l = \begin{cases} \mathbf{x}_t^{l-1} + \mathbf{h}_t^{l-1}, & \text{if } l > 1 \\ \mathbf{s}_t, & \text{if } l = 1 \end{cases} \quad (7)$$

We will use ResLSTM as a baseline model with skip connection in Section 3.

Although ResLSTM can partially solve the gradient vanishing issue, it still has the same challenges as the standard time-LSTM – the output vector works for two very different purposes: temporal modeling and senone classification.

2.3. Layer trajectory LSTM

As discussed above, it may not be optimal that the output of time-LSTM serves both the purposes of temporal modeling and senone classification. In this study, we decouple these two purposes by proposing a layer trajectory LSTM (LT-LSTM) which builds a layer-LSTM using the outputs from all the time-LSTM layers, shown in Figure 2. The weights are not shared between layers because sharing doesn't bring any computational benefit. The time-LSTM is used for the purpose of temporal modeling via time recurrence, while the layer-LSTM scans the outputs from multiple time-LSTM layers and uses the summarized layer trajectory information for final senone classification. With a layer-LSTM running through layers, a gated path is provided from the output layer to the bottom layer, reducing the gradient vanishing issue.

In LT-LSTM, the formulation of time-LSTM is still the standard LSTM formulation, with Eqs. (1) – (5). As shown

in Figure 2, there is no time recurrence between layer-LSTMs across different time steps. Hence, the formulation of layer-LSTM only has the recurrence across layers as:

$$\mathbf{j}_t^l = \sigma(\mathbf{U}_{jh}^l \mathbf{h}_t^l + \mathbf{U}_{jg}^l \mathbf{g}_t^{l-1} + \mathbf{q}_j^l \odot \mathbf{m}_t^{l-1} + \mathbf{d}_j^l) \quad (8)$$

$$\mathbf{e}_t^l = \sigma(\mathbf{U}_{eh}^l \mathbf{h}_t^l + \mathbf{U}_{eg}^l \mathbf{g}_t^{l-1} + \mathbf{q}_e^l \odot \mathbf{m}_t^{l-1} + \mathbf{d}_e^l) \quad (9)$$

$$\mathbf{m}_t^l = \mathbf{e}_t^l \odot \mathbf{m}_t^{l-1} + \mathbf{j}_t^l \odot \phi(\mathbf{U}_{sh}^l \mathbf{h}_t^l + \mathbf{U}_{sg}^l \mathbf{g}_t^{l-1} + \mathbf{d}_s^l) \quad (10)$$

$$\mathbf{v}_t^l = \sigma(\mathbf{U}_{vh}^l \mathbf{h}_t^l + \mathbf{U}_{vg}^l \mathbf{g}_t^{l-1} + \mathbf{q}_v^l \odot \mathbf{m}_t^l + \mathbf{d}_v^l) \quad (11)$$

$$\mathbf{g}_t^l = \mathbf{v}_t^l \odot \phi(\mathbf{m}_t^l) \quad (12)$$

The vectors $\mathbf{j}_t^l$, $\mathbf{v}_t^l$, $\mathbf{e}_t^l$, $\mathbf{m}_t^l$ are the activation of the input, output, forget gates, and memory cell of the layer-LSTM, respectively. $\mathbf{g}_t^l$ is the output of the layer-LSTM. The matrices $\mathbf{U}_{.h}^l$ and $\mathbf{U}_{.g}^l$ terms are the weight matrices for the inputs $\mathbf{h}_t^l$ and the recurrent inputs $\mathbf{g}_t^{l-1}$, respectively. The $\mathbf{d}_.^l$ are bias vectors. The $\mathbf{q}_j^l$, $\mathbf{q}_v^l$, $\mathbf{q}_e^l$ are parameter vectors associated with peephole connections.

Comparing Eqs. (1) – (5) with Eqs. (8) – (12), we can see the biggest difference is the recurrence now happens across the layers with $\mathbf{g}_t^{l-1}$ in layer-LSTM, compared to the time recurrence with $\mathbf{h}_{t-1}^l$ in time-LSTM. Layer-LSTM uses the output of time-LSTM at current layer, $\mathbf{h}_t^l$, as the input, compared to the $\mathbf{x}_t^l$ in time-LSTM.

It is a common practice to deploy a complicated model by reducing parallel computational time, e.g., [19]. Because the forward-propagation of the time-LSTM at next time step is independent of the calculation of the layer-LSTM at current time step, the forward-propagation of time-LSTM and layer-LSTM can be handled in two separate threads in parallel and the network computational time is the same as the standard time-LSTM which operates in a layer-by-layer and frame-by-frame fashion. Another advantage of decoupling the time and layer operation in LT-LSTM is that layer-LSTM can be evaluated with batching [20] which was proposed to improve the runtime of feed-forward DNNs by evaluating the network scores from several time frames at the same time. However, batching cannot be applied to standard time-LSTM because the input of current frame is from the output of previous frame. Since there is no time recurrence between layer-LSTM across different frames, batching can be applied to evaluate layer-LSTM once the time-LSTM vectors have been calculated in multiple frames.

2.4. Comparison with grid LSTMs

Grid LSTM (gLSTM) [16] and prioritized grid LSTM (pgLSTM) [12, 17] can be considered as a multidimensional LSTM which arranges the LSTM memory cells along both time and layer axis. They modify the LSTM units with multi-dimensional formulation and still process the speech input in a step-by-step and layer-by-layer fashion. The operation along time and layer dimensions is mixed together. In contrast, LT-LSTM decouples the jobs of temporal modeling with time-LSTM and target classification with layer-LSTM. The evaluation of time-LSTM doesn't rely on the value of layer-LSTM in previous time step or lower layer. Hence, LT-LSTM enjoys clear computational advantage as discussed in the previous section. Because of the function decoupling, it is not necessary to use LSTM units for modeling layer dependency. We can just replace layer-LSTM units with layer-DNN units or any other units, which gives more modeling flexibility [21].

2.5. Factorized gate LSTM

The computational cost of LSTM is always a concern. There are lots of attempts [22] to reduce the computational cost, such as getting low-rank matrices with singular value decomposition (SVD) [23, 24], model compression via teacher-student (T/S) learning [25] or knowledge distillation [26], scalar quantization [20], and vector quantization [27] etc. The computational cost can also be reduced by exploring different model structures [7, 28] or using lower frame rate strategies [7, 29].

In this section, we focus on reducing the size of weight matrices used to calculate input, output, and forget gates in the LSTM unit. Those matrices usually are of large size, resulting the major computational cost in the network evaluation. For example, some typical LSTM-RNN systems usually have around 1024 memory cells [4, 30] in the LSTM unit, which means the dimension of gate vectors is 1024. Usually a linear projection layer is applied to the LSTM output vector to reduce its dimension, for example to 512. Hence the two weight matrices , $\mathbf{W}_{ix}^l$ and $\mathbf{W}_{ih}^l$, used to calculate the input gate vector in Eq. (1) are of dimension 1024x512.

In Eq. (15), we factorize the input gate vector calculation as the square root of the product of two vectors, $\acute{\mathbf{i}}_t^l$ and $\{\grave{\mathbf{i}}_t^l\}^T$, which are calculated by Eqs. (13) and (14). $vec(.)$ is the operation that squashes a kxk matrix into a m-dimension vector, where $m = k$xk. Peephole connections are not used in Eqs. (13) and (14) because of dimension mismatch between the state vector (m dimension) and factorized gate vector (k dimension).

$$\acute{\mathbf{i}}_t^l = \sigma(\acute{\mathbf{W}}_{ix}^l \mathbf{x}_t^l + \acute{\mathbf{W}}_{ih}^l \mathbf{h}_{t-1}^l + \acute{\mathbf{b}}_i^l) \quad (13)$$

$$\grave{\mathbf{i}}_t^l = \sigma(\grave{\mathbf{W}}_{ix}^l \mathbf{x}_t^l + \grave{\mathbf{W}}_{ih}^l \mathbf{h}_{t-1}^l + \grave{\mathbf{b}}_i^l) \quad (14)$$

$$\hat{\mathbf{i}}_t^l = vec(sqrt(\acute{\mathbf{i}}_t^l \ \{\grave{\mathbf{i}}_t^l\}^T)) \quad (15)$$

For the above example, instead of having two 1024x512 matrices for the input gate calculation in Eq. (15), only four 32x512 matrices ($1024 = 32$x32) are involved in Eqs. (13) and (14). The computation cost is reduced to only 1/16 for the input gate vector calculation.

Similar formulations can be applied to the forget and output gates of the time-LSTM in Eqs. (2) and (4), and the input, forget, and output gates of layer-LSTM in Eqs. (8), (9) and (11).

3. Experiments

We compare standard multi-layer LSTM, ResLSTM, and the proposed LT-LSTM in this section. All these models use standard LSTM units as basic building blocks, different from grid LSTM which modifies the LSTM units with multi-dimensional formulation. All models were trained with 30 thousand (k) hours of anonymized and transcribed Microsoft production data, including Cortana [30], xBox [31], and Conversation data, which is a mixture of close-talk and far-field utterances from a variety of devices. The first model was built as a 4-layer LSTM-RNN with projection layer as what we usually did [30]. Each LSTM layer has 1024 hidden units and the output size of each LSTM layer is reduced to 512 using a linear projection layer. The output layer has 9404 nodes, modeling the senone labels. The target senone label is delayed by 5 frames as in [4]. The input feature is 80-dimension log Mel filter bank. We applied frame skipping [7] to reduce the runtime cost. Note that in this study, we only compare the baseline full-rank cross-entropy models. If we want to deploy models into production, we will further apply SVD training [23] and sequence discriminative

Table 1: *WERs of LSTM, ResLSTM, and LT-LSTM models on Cortana and Conversation test sets. Both test sets are mixed with close-talk and far-field utterances.*

	Cortana	Conversation
4-layer LSTM	10.37	19.41
6-layer LSTM	9.85	19.20
10-layer LSTM	10.58	19.92
6-layer ResLSTM	9.99	18.85
10-layer ResLSTM	9.68	18.15
6-layer LT-LSTM	**9.28**	**17.47**

Table 2: *Total and parallel per-thread computational costs of LSTM, ResLSTM, and LT-LSTM models in terms of million (M) operations per frame.*

	Total (M)	Parallel per thread (M)
4-layer LSTM	22	22
6-layer LSTM	31	31
10-layer LSTM	49	49
6-layer ResLSTM	31	31
10-layer ResLSTM	49	49
6-layer LT-LSTM	**57**	**31**

training using the maximum mutual information (MMI) criterion with F-smoothing [32], as the systems described in [30]. The LM is a 5-gram with around 100 million (M) ngrams.

We evaluate all models with Cortana and Conversation test sets. Both sets contain mixed close-talk and far-field utterances, with 439k and 111k words, respectively. The Cortana test set has shorter utterances related to voice search and commands, while the Conversation test set has longer utterances for conversation. As shown in Table 1, the 4-layer LSTM model obtained 10.37% and 19.41% WER on these 2 test sets, respectively.

Then, we simply increased the number of LSTM layers to 6 and 10. Increasing from 4 layers to 6 layers, the multi-layer LSTM got improvement across all tasks, with 9.85% and 19.20% WERs on Cortana and Conversation test sets, respectively. However, when increasing to 10 layers, the multi-layer LSTM got lots of degradation, consistent with the observations in literature [11, 12].

The 6-layer ResLSTM obtained very similar WERs as the 6-layer LSTM, with improvement on Conversation test sets, but slight degradation on Cortana test sets. However, different from the behavior of the 10-layer LSTM, consistent improvement was obtained with the 10-layer ResLSTM which got 9.68% and 18.15% WERs on Cortana and Conversation test sets, respectively. This clearly demonstrates the effectiveness of skipping connection for reducing the gradient vanishing issue.

Finally, the 6-layer LT-LSTM got significant improvement over all models, obtaining 9.28% and 17.47% WERs on Cortana and Conversation test sets, respectively. This represents 5.8% and 9.0% relative WER reduction from the 6-layer LSTM, or 4.1% and 3.7% relative WER reduction from the 10-layer ResLSTM on Cortana and Conversation test sets, respectively.

In Table 2, we examine the total and parallel computational costs of all LSTM, ResLSTM, and LT-LSTM models. Both LSTM and ResLSTM operate in a frame-by-frame and layer-by-layer fashion, therefore the total and parallel computational costs are same. As described in Section 2.3, the layer-LSTM and time-LSTM inside LT-LSTM can be evaluated in parallel as there is no time recurrence between layer-LSTMs at different time steps. As a result, the parallel computational cost is about 31 M per frame, which is the same as that of the 6-layer LSTM.

Table 3: *WERs of the 6-layer LT-LSTM and its factorized gate versions on Cortana and Conversation test sets. Both test sets are mixed with close-talk and far-field utterances.*

	Cortana	Conversation
full	**9.28**	**17.47**
factorized input	9.62	18.31
factorized output	9.50	18.11
factorized forget	9.57	17.97

We applied factorized gate LSTM described in Section 2.5 to the 6-layer LT-LSTM and evaluated the method in Table 3. We factorized input gates in both time and layer LSTMs by reducing the calculation of a 1024-dimension gate vector into the calculation of two 32-dimension gate vectors as in Eqs. (13), (14), and (15). We also applied similar operation to factorize output and forget gates in both time and layer LSTMs. All the factorized gate operation increased WER. Clearly, no magic happens even with the factorization in Eq.(15) because two 32-dimension gate vectors carry much less information than what a 1024-dimension gate vector can carry. The impact of factorizing forget gate is the smallest, with relative 2.3% and 3.7% WER increase from the full version of LT-LSTM without any factorization, although it is still better than all the LSTM and ResLSTM models. Factorizing input gates has the biggest degradation. Given the loss, we didn't evaluate the setup which factorizes all the gates together. With single gate factorization, the parallel computational cost is reduced to 25 M operation per frame which is even lower than that of the 6-layer LSTM or ResLSTM while the WER of factorized gate LT-LSTM is clearly better than that of the 6-layer LSTM or ResLSTM.

4. Conclusions and Future Works

In this paper, we proposed a novel model called LT-LSTM which scans the outputs of the multi-layer time-LSTM with a layer-LSTM to learn layer trajectory information which is used for classification. This model decouples the tasks of temporal modeling and target classification by using time-LSTM and layer-LSTM, respectively. It brings the benefits of both accuracy and runtime. Trained with 30k hours of speech data, the 6-layer LT-LSTM improves the baseline 6-layer LSTM with relative 5.8% and 9.0% WER reduction on Cortana and Conversation test sets respectively and reduces the WERs of the 10-layer ResLSTM by 4.1% and 3.7% relative. With parallel computation, the model evaluation time of the 6-layer LT-LSTM is kept the same as that of the 6-layer LSTM. Furthermore, we proposed to factorize gates inside LSTM units to reduce the runtime cost. Applied to the 6-layer LT-LSTM, the model has smaller parallel computational cost and better accuracy than that of the 6-layer LSTM or ResLSTM.

Recently, we blended attention mechanism [33] into CTC modeling and achieved very good accuracy improvement [34, 35]. We are now using similar idea to further improve LT-LSTM. As noted in Section 2.4, it is not necessary to use LSTM units for modeling layer dependency. We are working on a generalized LT-LSTM which can employ any units for modeling layer dependency. All these works will be reported in [21].

5. References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

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

[3] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, 2013, pp. 6645–6649.

[4] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." in *INTERSPEECH*, 2014, pp. 338–342.

[5] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*. IEEE, 2015, pp. 4520–4524.

[6] Y. Miao and F. Metze, "On speaker adaptation of long short-term memory recurrent neural networks." in *INTERSPEECH*, 2015, pp. 1101–1105.

[7] Y. Miao, J. Li, Y. Wang, S. Zhang, and Y. Gong, "Simplifying long short-term memory acoustic models for fast training and decoding," in *ICASSP*, 2016.

[8] J. Li, A. Mohamed, G. Zweig, and Y. Gong, "LSTM time and frequency recurrence for automatic speech recognition," in *ASRU*, 2015.

[9] ——, "Exploring multidimensional LSTMs for large vocabulary ASR," in *ICASSP*, 2016.

[10] T. N. Sainath and B. Li, "Modeling time-frequency patterns with LSTM vs. convolutional architectures for LVCSR tasks," in *INTERSPEECH*, 2016.

[11] Y. Zhang, G. Chen, D. Yu, K. Yao, S. Khudanpur, and J. Glass, "Highway long short-term memory rnns for distant speech recognition," *ICASSP*, 2016.

[12] W.-N. Hsu, Y. Zhang, and J. Glass, "A prioritized grid long short-term memory RNN for speech recognition," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 467–473.

[13] Y. Zhao, S. Xu, and B. Xu, "Multidimensional residual learning based on recurrent neural networks for acoustic modeling," in *INTERSPEECH*, 2016, pp. 3419–3423.

[14] J. Kim, M. El-Khamy, and J. Lee, "Residual LSTM: Design of a deep recurrent architecture for distant speech recognition," *arXiv preprint arXiv:1701.03360*, 2017.

[15] G. Pundak and T. N. Sainath, "Highway-LSTM and recurrent highway networks for speech recognition," in *Proc. of INTERSPEECH*, 2017.

[16] N. Kalchbrenner, I. Danihelka, and A. Graves, "Grid long short-term memory," *arXiv preprint arXiv:1507.01526*, 2015.

[17] M. Najafian, W.-N. Hsu, A. Ali, and J. Glass, "Automatic speech recognition of Arabic multi-genre broadcast media," in *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*. IEEE, 2017, pp. 353–359.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[19] B. Li and T. N. Sainath, "Reducing the computational complexity of two-dimensional LSTMs," in *Proc. Interspeech*, 2017.

[20] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, 2011.

[21] J. Li, L. Lu, C. Liu, and Y. Gong, "Generalized layer trajectory LSTM with attention," in *submitted to SLT*, 2018.

[22] D. Yu and J. Li, "Recent Progresses in Deep Learning Based Acoustic Models," *IEEE/CAA J. of Autom. Sinica.*, vol. 4, no. 3, pp. 399–412, Jul. 2017.

[23] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *INTERSPEECH*, 2013, pp. 2365–2369.

[24] R. Prabhavalkar, O. Alsharif, A. Bruguier, and L. McGraw, "On the compression of recurrent neural networks with an application to LVCSR acoustic modeling for embedded speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*. IEEE, 2016, pp. 5970–5974.

[25] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria." in *INTERSPEECH*, 2014, pp. 1910–1914.

[26] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[27] Y. Wang, J. Li, and Y. Gong, "Small-footprint high-performance deep neural network-based speech recognition using split-vq," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*. IEEE, 2015, pp. 4984–4988.

[28] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[29] G. Pundak and T. N. Sainath, "Lower frame rate neural network acoustic models," in *INTERSPEECH*, 2016, pp. 22–26.

[30] J. Li, R. Zhao, Z. Chen *et al.*, "Developing far-field speaker system via teacher-student learning," in *Proc. ICASSP*, 2018.

[31] C. Liu, J. Li, and Y. Gong, "SVD-based universal DNN modeling for multiple scenarios," in *Proc. of INTERSPEECH*, 2015.

[32] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6664–6668.

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

[34] A. Das, J. Li, R. Zhao, and Y. Gong, "Advancing connectionist temporal classification with attention modeling," in *Proc. ICASSP*, 2018.

[35] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, "Advancing Acoustic-to-Word CTC Model," in *Proc. ICASSP*, 2018.