EXPLORING LAYER TRAJECTORY LSTM WITH DEPTH PROCESSING UNITS AND ATTENTION

Jinyu Li, Liang Lu, Changliang Liu, Yifan Gong

Microsoft AI and Research

ABSTRACT

Traditional LSTM model and its variants normally work in a frame-by-frame and layer-by-layer fashion, which deals with the temporal modeling and target classification problems at the same time. In this paper, we extend our recently proposed layer trajectory LSTM (ltLSTM) and present a generalized framework, which is equipped with a depth processing block that scans the hidden states of each time-LSTM layer, and uses the summarized layer trajectory information for final senone classification. We explore different modeling units used in the depth processing block to have a good tradeoff between accuracy and runtime cost. Furthermore, we integrate an attention module into this framework to explore wide context information, which is especially beneficial for uni-directional LSTMs. Trained with 30 thousand hours of EN-US Microsoft internal data and cross entropy criterion, the proposed generalized ltLSTM performed significantly better than the standard multi-layer time-LSTM, with up to 12.8% relative word error rate (WER) reduction across different tasks. With attention modeling, the relative WER reduction can be up to 17.9%. We observed similar gain when the models were trained with sequence discriminative training criterion.

Index Terms— LSTM, depth processing block, attention

1. INTRODUCTION

There has been a significant progress in automatic speech recognition (ASR) since the transition from the deep feedforward neural networks (DNNs) [1, 2, 3, 4, 5] to recurrent neural networks (RNNs) with long short-term memory (LSTM) units [6]. LSTMs alleviate the gradient vanishing or exploding issues in standard RNNs by using input, output and forget gates, thus improve the capacity of the network to capture long temporal context information in audio sequences. These LSTM-RNNs [7, 8, 9, 10, 11, 12] and their variants such as convolutional LSTM DNN (CLDNN) [13] and two-dimensional LSTM-RNNs [14, 15, 16, 17] have been shown to outperform DNNs on a variety of ASR tasks [18].

It is a standard practice to stack multiple LSTM layers to obtain greater modeling power [8], especially when a large amount of training data is available. However, deeper LSTMs are not guaranteed to achieve higher accuracy due to the training difficulty [19, 20]. This issue can be partially solved by adding skip connections like in residual LSTMs [21, 22] or gating functions between layers such as highway LSTMs [19]. However, the gain in terms of recognition accuracy is very limited as shown in [23, 24]. Grid LSTM [25] is a more general LSTM architecture, which arranges the LSTM memory cells into a multidimensional grid along both time and depth axes. It was extended in [20, 26] 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 to the LSTM of the next layer at the same time step. The output of the final LSTM layer is used for senone (tied triphone states) classification. However, this design may not be optimal for the LSTM outputs to serve the purposes of both recurrence modeling along time axis and senone classification along the depth axis.

In [24], we proposed a layer trajectory LSTM (ltLSTM) which is equipped with a depth-LSTM that scans the hidden states of time-LSTMs for senone classification. This architecture decouples the tasks of time recurrence modeling and senone classification for LSTMs. Furthermore, the depth-LSTM creates auxiliary connections for gradient flow, thereby making it easier to train deeper LSTMs. Although we obtained significant gain in terms of accuracy over standard time-LSTM, the computational cost is much higher, making it challenging to deploy this type of models for online speech service that has strict computational cost requirements. In addition, the models in our previous work were trained with cross-entropy criterion, which raises the question if the gain can still hold after sequence discriminative training [9, 27, 28, 29].

In this paper, we extend our previous work on ltLSTM in several directions. First, we propose the generalized ltLSTM architecture (gltLSTM) by introducing the concept of depth processing block. We investigate low cost depth processing units for gltLSTMs that balance the accuracy with computational cost. To this end, we present gated feedforward units as well as max-pooling feedforward units designed for the depth processing block with low computational cost. Second, we incorporate the attention scheme into the gltLSTM architecture, inspired by our previous study on attention model [30] for connectionist temporal classification (CTC) model[31]. The motivation is to explore the wide context information and relax the hard alignment issue of hybrid models. Third, we look at singular value decomposition (SVD) compression to further reduce the computational cost and memory footprint following [32]. Finally, we evaluate the model with sequence discriminative training by the maximum mutual information (MMI) criterion with F-smoothing [29], and show that the accuracy improvements are from strong baseline systems. Our experiments were performed using around 30 thousand hours of anonymized EN-US data, which is a collection of Microsoft Cortana and Conversation data with mixed close-talk and far-field audios. Our results show that the gltLSTM can significantly outperform the standard multi-layer LSTM and residual LSTM with marginal increase in computational cost. The attention mechanism can achieve further reduction in terms of error rate for the gltLSTM model.

The rest of the paper is organized as follows. In Section 2, we briefly overview the standard multi-layer LSTM and Residual LSTM. In Section 3, we describe the proposed gltLSTM and its three implementations with LSTM, gated DNN, and maxout units serving in the layer process block respectively. We evaluate the proposed models in Section 4, and conclude our study in Section 5.

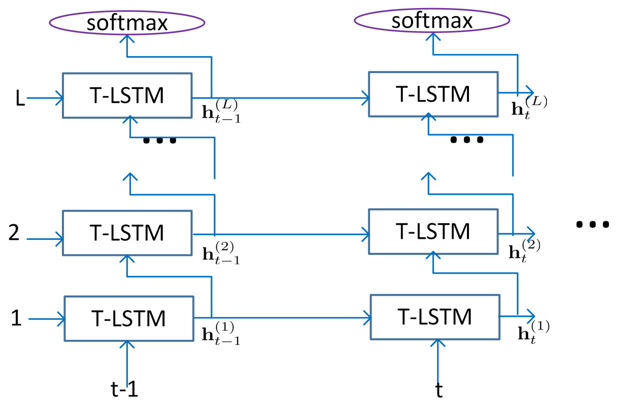

Fig. 1. Flowchart of multi-layer time-LSTM (T-LSTM). The output of a T-LSTM is used as the input of the T-LSTM at the same time step in the next layer and the recurrent input of the T-LSTM at the next time step in the same layer.

2. LSTM

In this section, we review the standard multi-layer LSTM and residual LSTM (ResLSTM) that will be used to compared our ltLSTM models in the experimental section.

2.1. LSTM

We refer to the standard LSTM as time-LSTM since it does temporal modeling via time recurrence by taking the output of previous time step as the input of the current time step. To increase the 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 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. $l = 1...L$, where L is the total number of hidden layers. 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.

2.2. Residual LSTM

Similar to Residual convolutional neural network (CNN) [33] 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 alleviated.

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

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.

3. GENERALIZED LAYER TRAJECTORY LSTM

In this section, we formulate our proposed generalized layer trajectory LSTM (gltLSTM) by using a depth processing block for target classification. Then, we present three realizations of gltLSTM with different units in the depth processing block.

3.1. Generalized layer trajectory LSTM

We proposed ltLSTM in [24] with the motivation of decoupling time recurrence modeling and senone classification by separated LSTM units, i.e., time-LSTM and depth-LSTM. In this paper, we extend this framework by introducing the concept of depth processing block, and propose the generalized ltLSTM architecture (gltLSTM), as shown in Figure 2. In gltLSTM, the time-LSTM is used for the purpose of temporal modeling via time recurrence, while the depth processing block scans the outputs from multiple time-LSTM layers and uses the summarized layer trajectory information for final senone classification. The idea is that the lower layer hidden states from time-LSTM may also carry valuable information for classification, and it may not be optimal to use the hidden state of the last time-LSTM for classification as in the conventional practice. Furthermore, the depth processing block also creates auxiliary connections for time-LSTM, which may facilitate the gradient flow to deal with training problem of deep LSTMs.

More precisely, the time-LSTM in gltLSTM is standard as Eqs. (1) – (5), while in the depth processing block, there is no time recurrence. The l-th layer output of the depth processing block can be expressed as

$$\mathbf{g}_t^l = F(\mathbf{g}_t^{l-1}, \mathbf{h}_t^l | \theta_l) \quad (8)$$

where $\mathbf{h}_t^l$ is the hidden state from the l-th layer of time-LSTM at the time step t, calculated from Eq. (5); $\mathbf{g}_t^{l-1}$ denotes the output of the previous layer in the depth processing block, and we set $\mathbf{g}_t^0 = \mathbf{s}_t$. $F(\cdot|\theta_l)$ denotes the function to process the l-th layer in the depth processing block which is parameterized by θ_l. In gltLSTM, it is flexible to choose different functions for $F(\cdot|\theta_l)$, which may be implemented by different neural network modules. In this study, we explore three realizations of $F(\cdot|\theta_l)$ with LSTM, gated DNN, and maxout units, respectively.

It is worth noting that the output from the depth processing block is not fed into the time-LSTM, which is the key difference from grid LSTM [25] or variants such as prioritized grid LSTM [20]. The benefit is that the depth processing block is not involved in time recurrence modeling, which may be more adaptable, and easier to integrate different kinds of features into the network. In addition, this architecture makes it straightforward to parallelize the computation in time-LSTM and depth processing block, because the forward-propagation of the time-LSTM at next time step is independent from the computation of the depth processing block at current time step. Thus, the forward-propagation of time-LSTM and depth processing

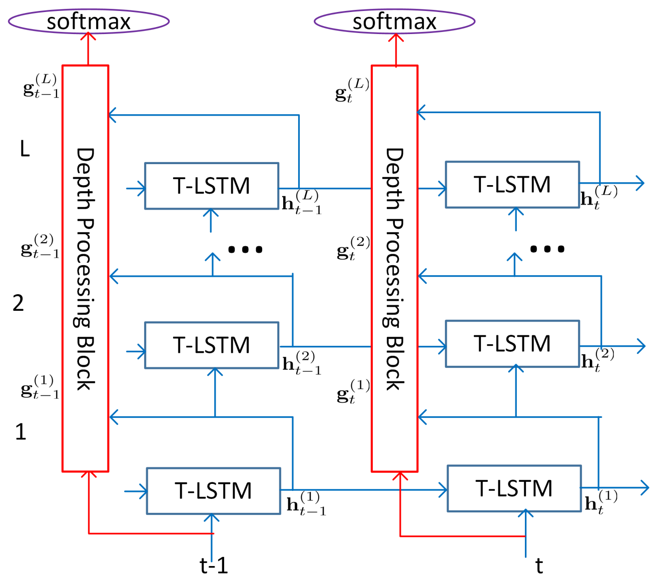

Fig. 2. Flowchart of generalized layer trajectory LSTM (gltLSTM). depth processing block is used to scan the outputs of time-LSTM (T-LSTM) across all layers at the current time step to get summarized layer trajectory information for senone classification. There is no time recurrence for depth processing blocks. Time recurrence only exists between T-LSTMs at different time steps.

block can be computed by two separate threads in parallel. As long as the computation cost of the depth processing block is not higher than that of time-LSTM, the network inference time can be the same as the standard time-LSTM [24].

3.2. Depth processing block by LSTM

As mentioned before, the depth processing block has the flexibility to be implemented by various kinds of neural network units. One realization is to use LSTM units as in our previous work [24], which is referred to ltLSTM-L in this paper. In this model, the lth layer of the depth processing block can be expressed as:

$$\mathbf{j}_t^l = \sigma(\mathbf{U}_{jh}^l \mathbf{h}_t^l + \mathbf{U}_{jg}^l \mathbf{g}_t^{l-1} + \mathbf{q}_j^l \odot \mathbf{m}_t^{l-1} + \mathbf{d}_j^l) \tag{9}$$
$$\mathbf{e}_t^l = \sigma(\mathbf{U}_{eh}^l \mathbf{h}_t^l + \mathbf{U}_{eg}^l \mathbf{g}_t^{l-1} + \mathbf{q}_e^l \odot \mathbf{m}_t^{l-1} + \mathbf{d}_e^l) \tag{10}$$
$$\mathbf{m}_t^l = \mathbf{e}_t^l \odot \mathbf{m}_t^{l-1} + \mathbf{j}_t^l \odot \phi(\mathbf{U}_{sh}^l \mathbf{h}_t^l + \mathbf{U}_{sg}^l \mathbf{g}_t^{l-1} + \mathbf{d}_s^l) \tag{11}$$
$$\mathbf{v}_t^l = \sigma(\mathbf{U}_{vh}^l \mathbf{h}_t^l + \mathbf{U}_{vg}^l \mathbf{g}_t^{l-1} + \mathbf{q}_v^l \odot \mathbf{m}_t^l + \mathbf{d}_v^l) \tag{12}$$
$$\mathbf{g}_t^l = \mathbf{v}_t^l \odot \phi(\mathbf{m}_t^l) \tag{13}$$

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 depth-LSTM, respectively. $\mathbf{g}_t^l$ is the output of the depth-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. (9) – (13), we can see the biggest difference is the recurrence now happens across the layers (weights are not shared) with $\mathbf{g}_t^{l-1}$ in depth-LSTM, compared to the time recurrence with $\mathbf{h}_{t-1}^l$ in time-LSTM. Depth-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. The total computational cost of the ltLSTM-L is almost doubled compared to that of the time-LSTM. However, the computation of time-LSTM and depth-LSTM can be done in two parallel threads because the inference of time-LSTM does not depend on the inference of depth-LSTM, the computational time of gltLSTM with depth-LSTM units is the same as that of time-LSTM from the inference latency perspective.

3.3. Depth processing block by gated DNN

Using LSTM units for the depth processing block can significantly increase the computational cost. The number of model parameters will almost be doubled if we use the same number of hidden states for both depth- and time-LSTMs. In order to deploy the model for runtime-cost constrained applications, we can cut down the computational cost by using cheaper units for the depth processing block. One approach we have explored is to use the gated feedforward units, which can be written as:

$$\mathbf{g}_t^l = \phi\left(\sigma(\mathbf{O}_h^l \mathbf{h}_t^l) \odot \mathbf{U}_h^l \mathbf{h}_t^l + \sigma(\mathbf{O}_g^l \mathbf{g}_t^{l-1}) \odot \mathbf{U}_g^l \mathbf{g}_t^{l-1}\right). \tag{14}$$

Again, $\mathbf{g}_t^l$ and $\mathbf{h}_t^l$ are hidden states of the l-th layer of the depth processing block and time-LSTM at the time step t respectively. σ is the Sigmoid function that computes the soft gates for $\mathbf{h}_t^l$ and $\mathbf{g}_t^{l-1}$. ϕ denotes the hyperbolic tangent nonlinearity. $\mathbf{O}$ and $\mathbf{U}$ are weight matrices. The Sigmoid gate functions control the contributions from each time-LSTM and depth processing block layer during forward and backward computation. Without the gate functions, we observed that the model training can diverge easily – an phenomenon of gradient explosion. The two gate functions can mitigate the problem well in our experiments. Comparing to the depth-LSTM unit, this function has much lower computational cost. We refer to this model as ltLSTM-G in the experimental section.

3.4. Depth processing block by maxout DNN

In Eq (14), we use Sigmoid functions parameterized by two weight matrices at each layer to compute the soft gates. To further reduce the computational cost, we can use maxout units [34, 35, 36], which are hard [0, 1] gates without trainable model parameters. This can be represented as:

$$\mathbf{g}_t^l = \phi\left(\max(\mathbf{U}_h^l \mathbf{h}_t^l, \mathbf{U}_g^l \mathbf{g}_t^{l-1})\right), \tag{15}$$

where we use element-wise $\max$ operation after linear transformations of $\mathbf{h}_t^l$ and $\mathbf{g}_t^{l-1}$. From our experiments, the $\max$ operation is helpful to mitigate the gradient explosion problem. This model is referred to as ltLSTM-M in the experiment.

3.5. Attention module

Similar to the conventional LSTM acoustic model, gltLSTM uses the single $\mathbf{g}_t^L$ at the time step t for frame-by-frame classification. As mentioned before, the target of time step t may not be accurate due to the alignment error. Furthermore, the wide context information may be valuable for frame-level classification, which is evidenced by the strong gain by bi-directional LSTM over its unidirectional counterpart. We propose to explore the wide context information from the neighboring frames at each time step by attention mechanism for gltLSTM. Figure 3 shows an example of applying attention mechanism into gltLSTM with LSTM based depth processing unit. To

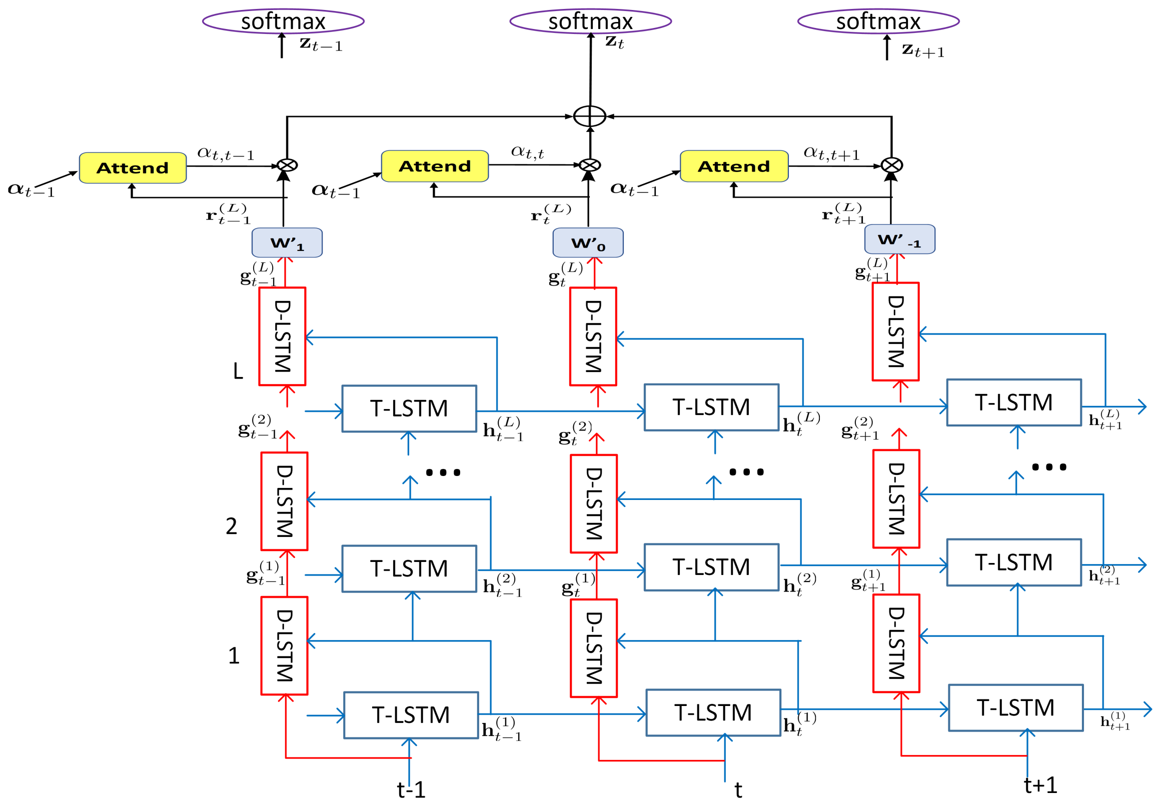

Fig. 3. Flowchart of attention-based gltLSTM with depth-LSTM units. T-LSTM and D-LSTM denote time-LSTM and depth-LSTM, respectively. Attention is applied to the output of the top layer depth-LSTM.

incorporate the information from contextual frames, we first transform $\mathbf{g}_\delta^L$ in a context window for each $\delta \in [t-\tau,\ t+\tau]$ as

$$\mathbf{r}_\delta = \mathbf{W}'_{t-\delta}\mathbf{g}_\delta^L \tag{16}$$

Here, $\mathbf{r}_\delta$ represents the transformed signal at time δ and $\mathbf{W}'_{[-\tau,\ \tau]}$ are trainable parameters. We can average $\mathbf{r}_\delta$ in the context window into the context vector $\mathbf{z}_t$ for final senone classification.

$$\mathbf{z}_t = \sum_{\delta=t-\tau}^{t+\tau} \mathbf{r}_\delta^L \tag{17}$$

$$= \gamma \sum_{\delta=t-\tau}^{t+\tau} \alpha_{t,\delta}\mathbf{r}_\delta^L. \tag{18}$$

$\mathbf{z}_t$ represents a special case context vector with *uniform* attention weights $\alpha_{t,\delta} = \frac{1}{\gamma}$, where $\gamma = 2\tau + 1$. However, higher accuracy may be achieved with the non-uniform attention as in the attention-based encoder-decoder network [30]. We first define the energy signal for attention as

$$\mathbf{e}_{t,\delta} = \tanh(\mathbf{W}\mathbf{r}_\delta + \mathbf{V}\mathbf{f}_{t-1} + \mathbf{b}) \tag{19}$$

where $\mathbf{f}_{t-1} = \mathbf{F} * \boldsymbol{\alpha}_{t-1}$. $\mathbf{W}$, $\mathbf{V}$, and $\mathbf{F}$ are trainable matrices, $\mathbf{b}$ is a bias vector, and the operation $*$ denotes convolution. $\boldsymbol{\alpha}_t$ denotes the combination coefficients for filtered vectors $\mathbf{r}_\delta$ and its components will be defined in Eq. (20).

Instead of content-based attention, we only use location-based attention with $\boldsymbol{\alpha}_{t-1}$ in which the location information is encoded. The reason is that we do not have the decoder state as query as in the attention-based encoder-decoder framework [30]. While logits may be used to replace the decoder states as queries for the content-based attention as in [31], we do not follow this approach due to the high computational overhead. For the benefit of accuracy [31], Eq. (19) generates an energy vector for every δ, different from the standard attention mechanism which produces a scalar value by multiplying it with a vector.

Now, we have column energy vectors $[\mathbf{e}_{t,t-\tau}, \cdots, \mathbf{e}_{t,t+\tau}]$ where each $\mathbf{e}_{t,\delta} \in (-1,1)^n$ (n is the vector dimension). Let $e_{t,\delta,j} \in (-1,1)$ be the jth component of the vector $\mathbf{e}_{t,\delta}$. To compute $\alpha_{t,\delta,j}$ from $e_{t,\delta,j}$, we normalize $e_{t,\delta,j}$ across δ keeping j fixed. Thus, $\alpha_{t,\delta,j}$ is computed as

$$\alpha_{t,\delta,j} = \frac{\exp(e_{t,\delta,j})}{\sum_{\delta'=t-\tau}^{t+\tau}\exp(e_{t,\delta',j})}, \quad j = 1, \cdots, n. \tag{20}$$

Here, $\alpha_{t,\delta,j}$ can be interpreted as the amount of contribution from $r_\delta(j)$ in computing $z_t(j)$. Now, the context vector $\mathbf{z}_t$ can be computed using

$$\mathbf{z}_t = \gamma \sum_{\delta=t-\tau}^{t+\tau} \boldsymbol{\alpha}_{t,\delta} \odot \mathbf{r}_\delta, \tag{21}$$

where $\odot$ is the Hadamard product. Hence, our proposed method is a dimension-wise location-based attention. In Eq. (21) $\boldsymbol{\alpha}_{t,\delta}$ is a vector, different from the scalar $\alpha_{t,\delta}$ in Eq. (18).

Table 1. WERs of LSTM, ResLSTM, and gltLSTM cross entropy models on Cortana, Conversation, and DMA test sets. All test sets are mixed with close-talk and far-field utterances.

	Cortana	Conversation	DMA
4-layer LSTM	10.37	19.41	20.66
6-layer LSTM	9.85	19.20	20.19
10-layer LSTM	10.58	19.92	21.62
6-layer ResLSTM	9.99	18.85	19.49
10-layer ResLSTM	9.68	18.15	18.62
12-layer ResLSTM	9.59	18.19	18.78
6-layer ltLSTM-L	9.28	17.47	17.61
6-layer ltLSTM-G	9.63	17.87	18.63
6-layer ltLSTM-M	9.77	18.26	18.90
6-layer ltLSTM-L attention	**8.96**	**17.22**	**16.57**
6-layer ltLSTM-G attention	9.25	17.68	17.76

4. EXPERIMENTS

In this section, we report experimental results comparing the standard multi-layer LSTM and ResLSTM to our proposed gltLSTM models. In our experiments, they are all uni-directional, and were trained with 30 thousand hours of anonymized and transcribed Microsoft production data, including Cortana and Conversation data, recorded in both close-talk and far-field conditions. All LSTM models use 1024 hidden units and the output of each LSTM layer is reduced to 512 using a linear projection layer. The softmax layer has 9404 nodes to model the senone labels. The target senone label is delayed by 5 frames as in [8]. The input feature is 80-dimension log Mel filter bank. We applied frame skipping [12] to reduce the runtime cost. The language model is a 5-gram with around 100 million (M) ngrams.

4.1. Results by cross entropy training

We evaluate all cross entropy (CE) trained models with Microsoft 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 from conversations. We also evaluate the models on the third test set named as DMA with 29k words, which is not in Cortana or Conversation domain. The DMA domain was unseen during the model training, and is served to evaluate the generalization capacity of the model. As shown in Table 1, the 4-layer LSTM model obtained 10.37%, 19.41%, and 20.66% WER on these 3 test sets, respectively. By increasing the number of layers from 4 to 6, the multi-layer LSTM was improved across all tasks. However, when increasing the number of layers to 10, we observed considerable accuracy degradation, which is consistent with the observations in literature [19, 20].

The 6-layer ResLSTM is close to the 6-layer LSTM in terms of WERs, with some improvements on Conversation and DMA test sets, but slight degradation on the Cortana test set. However, in contrast to the behavior of the 10-layer versus 6-layer LSTM models, we observed consistent improvements by increasing to 10 layers for ResLSTM model, which achieved 9.68%, 18.15%, and 18.62% WERs on Cortana, Conversation, and DMA test sets, respectively. This clearly demonstrates the effectiveness of skipping connections for reducing the gradient vanishing issue. However, further increasing to 12 layers doesn't improve the overall accuracy of ResLSTM.

Table 2. WERs of LSTM and gltLSTM sequence-trained models on Cortana, Conversation, and DMA test sets.

	Cortana	Conversation	DMA
6-layer SE-SVD-LSTM	7.93	19.17	17.45
6-layer SE-SVD-ltLSTM-L	**7.29**	**17.24**	15.80
6-layer SE-SVD-ltLSTM-G	7.48	17.77	**15.52**

We then evaluated the gltLSTM model with different depth processing units as described above. As shown by Table 1, with 6 hidden layers, the gltLSTM models consistently outperform the vanilla LSTM and ResLSTM across all the three evaluation sets. Among these three models, the 6-layer ltLSTM-L performed the best, obtaining 9.28%, 17.47% ad 17.61% WERs on Cortana, Conversation, and DMA test sets, respectively. This represents 5.8%, 9.0%, and 12.8% relative WER reduction from the 6-layer LSTM on those three test sets. The 6-layer ltLSTM-G is slightly left behind compared to ltLSTM-L in terms of accuracy, while the 6-layer ltLSTM-M is even worse. However, they all outperform the baseline 6-layer LSTM model.

Finally, on top of the 6-layer ltLSTM-L and ltLSTM-G models, we applied the attention mechanism described in Section 3.5, where we set τ to be 4. Larger τ may be beneficial to accuracy, but it will also introduce larger latency, so we did not tune this hyperparameter. As shown in the last two rows of Table 1, attention module can further reduce the WERs of both ltLSTM-L and ltLSTM-G models. In particular, ltLSTM-L with attention can achieve 8.96%, 17.22%, and 16.57% WER on Cortana, Conversation, and DMA test sets, respectively, improving the WER reductions to relative 9.1%, 10.3%, and 17.9% compared to the 6-layer LSTM baseline system on the evaluation datasets. It is interesting that the largest relative improvement is on the DMA test set which is unseen during model training.

4.2. Results by sequence discriminative training

We then compare the models after sequence discriminative training (SE). For the sake of runtime deployment [37], we first performed SVD compression [32] to all the weight matrices before sequence training using the MMI criterion with F-smoothing [29]. In table 2, we show the WERs of the 6-layer LSTM, ltLSTM-L, and ltLSTM-G models, and demonstrate that the gains from the CE training stages are still held after SE training. The 6-layer SE-SVD-ltLSTM-L improves the baseline 6-layer SE-SVD-LSTM with respectively 8.1%, 10.1%, and 9.5% relative WER reduction on Cortana, Conversation, and DMA test sets, while the 6-layer SE-SVD-ltLSTM-G obtained respectively 5.6%, 7.3%, and 11.1% relative improvement. Note that, the gap between ltLSTM-G and ltLSTM-L systems becomes smaller after SE training, which is encouraging as the former is much smaller and is more suitable for runtime deployment. In the future, we shall also study SE training of these two models with attention module.

4.3. Runtime comparison

In Table 3, we examine the total computational costs of all models. Both 6-layer LSTM and ResLSTM have 26M operations for hidden time-LSTM evaluation and 5M operations for softmax evaluation per frame, resulting in totally 31M operations per frame.

The 6-layer ltLSTM-L almost doubles the total computational cost of the 6-layer LSTM, with 57M operations per frame in total, which is the same as the computational cost of the 12-layer ResLSTM. The 6-layer ltLSTM-G significantly reduces the total

Table 3. Total computational costs of LSTM, ResLSTM, and gltLSTM models in terms of million (M) operations per frame.

	Total (M)
4-layer LSTM	22
6-layer LSTM	31
10-layer LSTM	49
6-layer ResLSTM	31
10-layer ResLSTM	49
12-layer ResLSTM	57
6-layer ltLSTM-L	57
6-layer ltLSTM-G	37
6-layer ltLSTM-M	33
6-layer ltLSTM-L attention	59
6-layer ltLSTM-G attention	39
6-layer SVD-LSTM	16
6-layer SVD-ltLSTM-L	29
6-layer SVD-ltLSTM-G	19

computational cost from the 6-layer ltLSTM-L, resulting in totally 37M operations per frame. The 6-layer ltLSTM-M has even smaller overall computational cost. Furthermore. the attention module only slightly increases the computational cost from their counterparts, while SVD compression significantly reduces the model size and computation. Finally, given the complexity of LSTM models, we didn't reduce the model size heavily as what we have done for DNNs [32]. The SVD version models half the computational cost of their full-size counterparts.

5. CONCLUSIONS

In this paper, we extended our previous work on ltLSTM and presented a generalized framework – gltLSTM, which scans the hidden states of the multi-layer time-LSTM with a depth process block for final senone classification. The depth processing block has the flexibility to be implemented by different kinds of neural network components, and we have shown three examples using LSTM, gated DNN, and maxout units, which have different computation and accuracy tradeoffs. We further investigated the dimension-wise location-based attention modeling to explore the contextual information for gltLSTM models.

From our experiments with around 30k hours of Microsoft internal speech training data, the 6-layer CE-trained gltLSTM significantly outperformed the baseline CE-trained LSTM and ResLSTM, with relative 5.8%, 9.0%, and 12.8% WER reductions from the 6-layer LSTM on Cortana, Conversation, and DMA sets, respectively. The attention modeling further pushed the relative WER reductions to 9.1%, 10.3%, and 17.9% on these three evaluation sets. We observed similar gain when the models were trained with sequence (SE) discriminative training criterion. The SE-trained 6-layer SVD ltLSTM-G obtained very similar WER as the SE-trained 6-layer SVD ltLSTM-L, but with much smaller computational cost. This makes it very desirable for product deployment.

6. ACKNOWLEDGEMENT

The first author would like to thank Wei-Ning Hsu at MIT for sharing lots of insights of his prioritized grid LSTM work.

7. REFERENCES

[1] Dong Yu, Li Deng, and George E. Dahl, "Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition," in *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[2] T. Sainath, B. Kingsbury, B. Ramabhadran, et al., "Making deep belief networks effective for large vocabulary continuous speech recognition," in *Proc. ASRU*, 2011, pp. 30–35.

[3] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proc. INTERSPEECH*, 2012.

[4] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara 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.

[5] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al., "Recent advances in deep learning for speech research at microsoft," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*, 2013, pp. 8604–8608.

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

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

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

[9] Haşim Sak, Oriol Vinyals, Georg Heigold, Andrew Senior, Erik McDermott, Rajat Monga, and Mark Mao, "Sequence discriminative distributed training of long short-term memory recurrent neural networks," in *Fifteenth annual conference of the international speech communication association*, 2014.

[10] Xiangang Li and Xihong 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.

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

[12] 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.

[13] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4580–4584.

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

[15] J. Li, A. Mohamed, G. Zweig, and Y. Gong, "Exploring multidimensional LSTMs for large vocabulary ASR," in *ICASSP*, 2016.

[16] Tara N Sainath and Bo Li, "Modeling time-frequency patterns with LSTM vs. convolutional architectures for LVCSR tasks," in *Proc. Interspeech*, 2016.

[17] Bo Li and Tara N Sainath, "Reducing the computational complexity of two-dimensional LSTMs," in *Proc. Interspeech*, 2017.

[18] 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, July 2017.

[19] Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yao, Sanjeev Khudanpur, and James Glass, "Highway long short-term memory rnns for distant speech recognition," *ICASSP*, 2016.

[20] Wei-Ning Hsu, Yu Zhang, and James 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.

[21] Yuanyuan Zhao, Shuang Xu, and Bo Xu, "Multidimensional residual learning based on recurrent neural networks for acoustic modeling," in *Proc. Interspeech*, 2016, pp. 3419–3423.

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

[23] Golan Pundak and Tara N Sainath, "Highway-LSTM and recurrent highway networks for speech recognition," in *Proc. of Interspeech*, 2017.

[24] Jinyu Li, , Changliang Liu, and Yifan Gong, "Layer trajectory LSTM," in *Proc. Interspeech*, 2018.

[25] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves, "Grid long short-term memory," *arXiv preprint arXiv:1507.01526*, 2015.

[26] Maryam Najafian, Wei-Ning Hsu, Ahmed Ali, and James 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.

[27] Brian Kingsbury, Tara N Sainath, and Hagen Soltau, "Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[28] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks.," in *Interspeech*, 2013, pp. 2345–2349.

[29] Hang Su, Gang Li, Dong Yu, and Frank 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.

[30] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

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

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

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[34] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[35] Yajie Miao, Florian Metze, and Shourabh Rawat, "Deep maxout networks for low-resource speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 398–403.

[36] Pawel Swietojanski, Jinyu Li, and Jui-Ting Huang, "Investigation of maxout networks for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 7649–7653.

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