EXPLOITING LSTM STRUCTURE IN DEEP NEURAL NETWORKS FOR SPEECH RECOGNITION

Tianxing He[*†] *Jasha Droppo*[†]

[*]Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
SpeechLab, Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
[†]Microsoft Research, Redmond

ABSTRACT

The CD-DNN-HMM system has became the state-of-art system for large vocabulary continuous speech recognition (LVCSR) tasks, in which deep neural networks (DNN) plays a key role. However, DNN training suffers from the vanishing gradient problem, limiting training of deep models. In this work, we address this problem by incorporating the successful long-short term memory (LSTM) structure, which has been proposed to help recurrent neural network (RNN) to remember long term dependencies, into DNN. Also, we propose a generalized formulation of the LSTM block, which we name general LSTM(GLSTM). In our experiments, it is shown that our proposed (G)LSTM-DNN scales well with more layers, and achieves 8.2% relative word error rate reduction on the 2000-hour Switchboard data set.

Index Terms— speech recognition, DNN, LSTM, acoustic model

1. INTRODUCTION

Recently, there has been great interest to use recurrent neural networks (RNN), especially with the long-short term memory (LSTM) structure, as an acoustic model in the automatic speech recognition (ASR) community[1, 2, 3, 4]. The LSTM-RNN has been shown to be able to preserve long-term temporal information in various tasks[5, 6]. By stacking multiple layers of LSTM layers, LSTM-RNN shows better performance than both the RNN model and the state-of-art DNN model.

The LSTM structure addresses the "vanishing gradient" problem[7] suffered by RNN training: the gradient flow will decay sharply through a non-linear operation. LSTM[8] alleviates this problem by introducing a "memory cell" structure which allows the gradient to travel without being squashed by a non-linear operation. Also, it has a set of gates which enable the model to decide whether to memorize, forget, or output information.

The "vanishing gradient" problem also exists for DNN[9], which is usually alleviated by layer-wise pre-training[10, 11]. In this work, we apply the LSTM structure to a DNN model. The "memory cell" structure in the LSTM structure can help train a deeper DNN, and the "gates" have potential to grant DNN more modeling power.

The idea of using the LSTM-like memory structure along depth instead of recurrence has been investigated by a few recent works[12, 13, 14]. [12, 14] proposed a unified way of using LSTM for multiple dimensions, and [13] introduces a simple memory structure into DNN. However, none of these works conducted experiments on LVCSR tasks.

The contributions of this work are as follows. Firstly, we incorporate the LSTM structure into DNN for LVCSR tasks and show the proposed model has better modeling power than the baseline DNN. We also report a way to prevent the model from over-fitting. Secondly, by spotting a symmetry of the cell layer and the output layer in the LSTM structure, we propose a general formulation of LSTM which provides a new view of that successful structure. Finally, we show that the proposed LSTM-DNN achieves significant WER improvement when applied to a large-scale LVCSR task.

2. MODEL FORMULATION

In this section, we describe in detail the formulation of the LSTM-DNN structure.

2.1. The formulation of LSTM-DNN

Denoting the output of each non-linear layer in DNN as h^l, we first formulate the building block of a normal DNN with L hidden layers as follows:

$$\begin{aligned} h^l &= \phi(W^l h^{l-1} + b^l) \\ y &= softmax(W^{L+1} h^L + b^{L+1}) \end{aligned} \tag{1}$$

where W^l is the transformation matrix and b^l is the bias vector, ϕ can be a chosen non-linear operation such as $Sigmoid$, $Tanh$, or $RELU$. The input feature vector x is denoted as h^0, the output y, which is the state posterior $P(s|x)$, is obtained after a soft-max normalization operation.

The proposed LSTM-DNN is obtained by stacking the LSTM-DNN blocks instead of the original normal DNN blocks. The LSTM-DNN network structure is depicted in

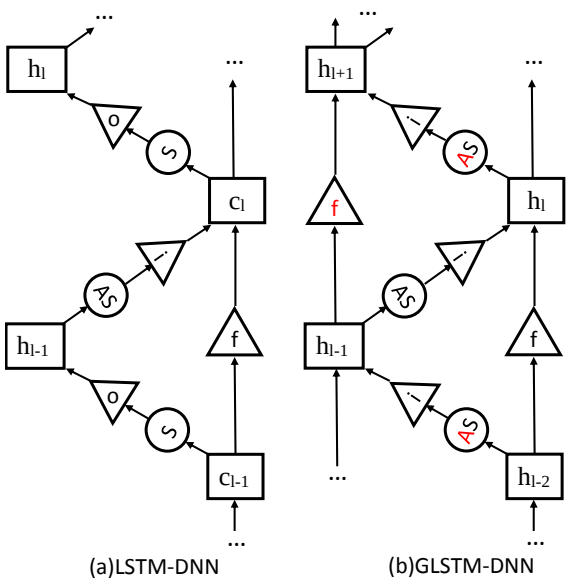

Fig. 1. The structure of LSTM-DNN and GLSTM-DNN block, each square represents a layer output vector, each triangle refers to a gating operation, an circle means an propagating operation("A" means an affine operation and "S" means a non-linear operation)

Figure 1(a) and layer l is formulated as follows: [1]

$$
\begin{aligned}
i^l &= Sigmoid(W^l_{hi}h^{l-1} + W^l_{ci}c^{l-1} + b^l_i) \\
f^l &= Sigmoid(W^l_{hf}h^{l-1} + W^l_{cf}c^{l-1} + b^l_f) \\
c^l &= i^l \odot \phi(W^l_{hc}h^{l-1} + b^l_c) + f^l \odot c^{l-1} \\
o^l &= Sigmoid(W^l_{ho}h^{l-1} + W^l_{co}c^l + b^l_o) \\
h^l &= o^l \odot \phi(c^l)
\end{aligned} \tag{2}
$$

where vector i^l, f^l, o^l are the input gate, forget gate, output gate, respectively, ϕ is a non-linear function such as $Tanh$. Note that a different set of transformation matrices are used for each layer block, which introduces significantly more parameters that a normal DNN. In this work, diagonal matrices are used for $W^*_{ci}, W^*_{cf}, W^*_{co}$ in the LSTM-DNN block as in [2]. In order to save parameters, the same set of W_{ci}, W_{cf}, W_{co} can be used across layers.

2.2. The formulation of GLSTM-DNN

The formulation of the LSTM-DNN block (2) is much more complicated than the normal DNN block (1) because it introduces an additional layer of cell memory, which is usually regarded as an inner structure. By regarding the cell memory layer c^l and output layer h^l both as normal layers, we propose a new structure called the general LSTM (GLSTM) block, which is depicted in Figure 1(b) and formulated as follows:

$$
\begin{aligned}
i^l &= Sigmoid(W^l_{h1i}h^{l-1} + W^l_{h2i}h^{l-2} + b^l_i) \\
f^l &= Sigmoid(W^l_{h1f}h^{l-1} + W^l_{h2f}h^{l-2} + b^l_f) \\
h^l &= i^l \odot \phi(W^l_{hh}h^{l-1} + b^l_c) + f^l \odot h^{l-2}
\end{aligned} \tag{3}
$$

where we follow the same notation convention as in Equation 2. Here no diagonal matrices are used.

Compared to the LSTM-DNN block, the GLSTM-DNN doesn't lose any modeling power. By stacking two GLSTM-DNN blocks, we get a LSTM-DNN block with additional operations. Furthermore, every layer in a GLSTM-DNN has a "memory" feature for the gradient flow to bypass the non-linearity, which doesn't exist for the h layers in the LSTM-DNN. Finally, the GLSTM-DNN block provides a new, clearer view of the successful LSTM structure, bringing a few natural extensions such as connecting memory from more than one previous layer.

3. EXPERIMENTS

We first analyze the modeling power and limitations of the proposed LSTM-DNN and GLSTM-DNN on a normal 80-hour LVCSR task AMI, and then apply it to the 2000-hour Switchboard and Fisher data set.

3.1. Experiments on AMI

First, we test the modeling power of different models on the public AMI LVCSR dataset[2]. The AMI corpus defines several partitions of the data for different purposes, and we choose to use the "full-corpus-ASR partition of meetings."[3] It contains 81 hours of training data, 9.7 hours of development data, and 9.1 hours of evaluation data. We use the close-talking IHM (individual headset microphone) signals exclusively in our experiments.

The base acoustic features are 40-dimensional mel-frequency filterbank coefficients(MFCC), processed through a 10th root filter in place of the more common logarithmic compression. These features are then processed using utterance level cepstral mean normalization. Following the CD-DNN-HMM framework, context dependent tri-phone HMM is used as phonetic model. They share a total of 5000 unique context dependent states, which corresponds to the output dimension of the DNN acoustic model. A context window of 27 is used(input dimension of DNN is 1080). The labels for training data are created by forced alignment with a classic GMM-HMM acoustic model trained on that data. The GMM-HMM has 24 gaussian components per state and 5000 shared states.

Cross entropy is used as the training criterion for DNN. In our experiments, the mini-batch size is fixed to 256, and a L2 regularization of 1e-6 is used. The learning rate is intially

[1] Readers maybe confused that no x appear in our formula, which is common in the LSTM-RNN literature[5]. In our preliminary experiments, we added links from the input vector to every hidden layer but got worse performance, so they are not kept.

[2] http://groups.inf.ed.ac.uk/ami/download/

[3] http://groups.inf.ed.ac.uk/ami/corpus/datasets.shtml

set to a large value, and is halved if the model performance on the development data degrades. No momentum is used. 24 hours of data will be fetched for every new epoch, and it takes around 50 epochs for the training to complete.

For the Sigmoid-DNN and highway-DNN (we follow the exact formulation as in [13]), we tried different numbers of layers and report the model with the best performance on cross-validation data. Note that for fair comparison with the highway-DNN, a normal DNN block is used for the first hidden layer in our (G)LSTM-DNN structure, followed by (G)LSTM-DNN blocks. As shown in Table 1, the highway network, which can be regarded as a simplified version of GLSTM-DNN, enables deeper DNN to learn but can not achieve better performance in this task. On the other hand, severe over-fitting is observed for RELU-DNN and LSTM-DNN. This indicates these models have great modeling power, but makes comparison based on training with the cross-validation difficult.

Model	Scale	CE(CV)	CE(TR)	WER
Sigmoid-DNN	2048L6	2.04	1.46	31.4
highway-DNN	2048L10	2.04	1.4	31.8
RELU-DNN	2048L6	2.49	1.25	-
LSTM-DNN	2048L3	2.34	1.28	-

Table 1. Results of training with cross-validation for different models, "2048L6" is to be interpreted as 6 hidden layers of 2048 hidden units. Cross entropy(CE) per sample is reported for the final model on the training and cross-validation data, WER(%) is reported for the models that did not over-fit.

To present a direct comparison of the modeling power of the proposed LSTM-DNN and the RELU-DNN model, we optimize the models directly on the training data without regard to the over-fitting issue or development set performance. Two critical question are to be discussed: how do the models scale with more layers, and how do they scale with total parameter count.

In Figure 2, performance of the baseline RELU-DNN is compared with the proposed (G)LSTM-DNN with varying model depth. The introduced memory path and gates in the proposed (G)LSTM-DNN model are working as expected to enable deeper (G)LSTM-DNN to get better performance, while RELU-DNN can not. We conclude that our proposed model has potential to out-perform the baseline DNN when applied to a larger LVCSR data-set.

To test the scaling of model power with model size, a large range of model scales and a few natural variants of the proposed (G)LSTM-DNN is tried, and the results is shown in Figure 3. Parameter number of each model is calculated and shown on the x-axis. The convex-hull of the baseline RELU-DNN model performance is drawn on the figure for easier comparison. It can be observed that although (G)LSTM-DNN achieves better performance compared to RELU-DNN of the same scale (hidden layer width and layer size), it does

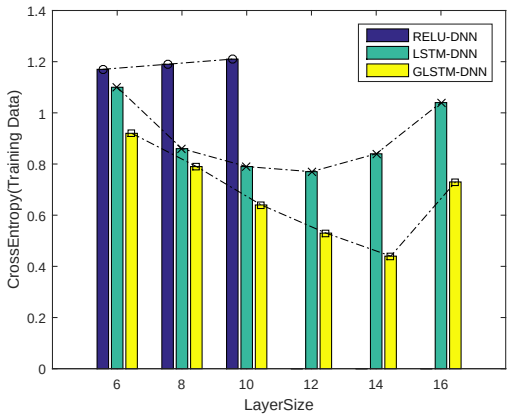

Fig. 2. Performance of RELU-DNN, LSTM-DNN and GLSTM-DNN of different layer size with 1024 hidden units

not achieve a good trade-off in terms of parameter number. LSTM-RELU-DNN has the same of number of parameters as the LSTM-DNN of the same scale and achieves better performance. It can be also observed that tying the gates helps GLSTM-RELU-DNN to get a better trade-off, which is reasonable because GLSTM-DNN introduces considerably more parameters. Overall, our proposed (G)LSTM-DNN does not get a significantly better trade-off than RELU-DNN in terms of parameter number, which is a limitation of the proposed model: the powerful model comes at a price of large number of parameters.

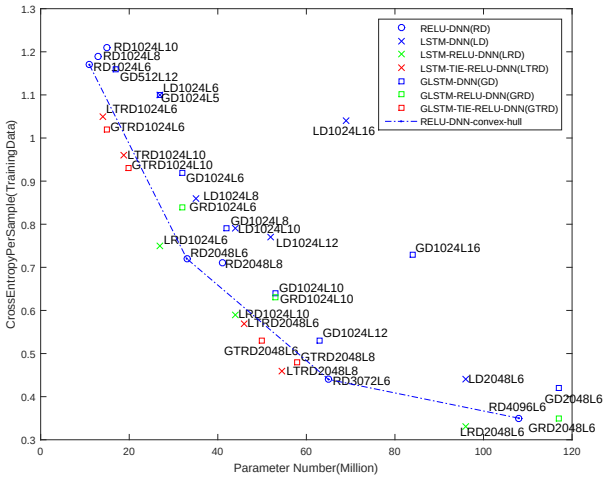

Fig. 3. Results for training data optimization. "(G)LSTM-RELU-DNN" means (G)LSTM-DNN with RELU as the non-linear function(ϕ in formula 2)."(G)LSTM-TIE-RELU-DNN" means (G)LSTM-RELU-DNN with the W parameters in the gating functions tied, as discussed in section 2.1.

Finally, L2 regularization and dropout[15] is tried for RELU-DNN and (G)LSTM-DNN to avoid over-fitting. A fixed dropout rate of 0.3 is applied for the first 20 epochs and reduced to zero afterwards, and a large L2 regularization of $1e-5$ is tried. Tests are conducted for a 2048L6 RELU-DNN

and (G)LSTM-DNN of different scales. As shown in Table 2, dropout prevents over-fitting of these models and gives reasonable decoding performance. However, achieving the best decoding result on AMI is beyond the scope of this work.

DNN Model	Scale	no dropout		with dropout		
		TR	CV	TR	CV	WER
RELU	2048L6	1.1	2.36	1.4	2.05	30.8
GLSTM	1024L8	1.1	2.15	1.3	1.96	30.3
LSTM	2048L6	1.04	2.09	1.28	1.99	31.11

Table 2. Performance of RELU-DNN, (G)LSTM-DNN with or without dropout. Cross-entropy on the training(TR) and cross-validation (CV) data and WER(%) on the test data are reported.

3.2. Experiments on Switchboard

To avoid over-fitting observed on the AMI data set, we move to the large-scale Switchboard 2000 hour data set. The training data consists of two publicly available Switchboard CTS corpora: 309 hours of SWBD1 and 1700 hours of Fisher. For testing, we use the publicly available Hub5'00 "sw"set (1831 utterances). The decoding language model is an interpolation of a tri-gram trained on the Fisher transcripts, and one trained on written background text. The same feature extraction procedure as on AMI data is followed. The labels for training DNN was created by forced alignment with a classic GMM-HMM acoustic model trained on 3850 hours of conversational speech, including Switchboard, Fisher, and lecture recordings. The GMM-HMM system has 24 gaussian components per state and 9000 shared states, which is also the output dimension of DNN. For DNN input a context window size of 23 is used.

During DNN training, 24 hours of data is fetched to form each epoch. A fixed learning rate and mini-batch size, and a momentum of 0.9 is used. In our baseline DNN experiment, only 1 GPU is used for training. However, to speed up the RELU and (G)LSTM-DNN experiments, we switch to 4-GPU parallel training with the one-bit SGD[16] technique. All models are trained until no improvement is observed on the cross-entropy of the training data.

DNN with 7 layers of 2048 hidden units are trained for the baseline Sigmoid and RELU DNN. We also tried 2048L9 and 3072L7 Sigmoid-DNN but observe no performance gain. In Table 3, it is shown that LSTM-DNN[4] has 8.2% relative performance gain compared to the baseline DNN, which means the introduced memory and gating structure is helping DNN to get better performance. However, the significant performance gain comes at a price of vast parameters: the LSTM-DNN has 3 times the parameters of baseline DNN. By tying the gating parameters, LSTM-TIE-DNN gets 4% relative performance gain with comparable parameter count with the baseline DNN model. Unfortunately, the 2048L11 GLSTM-DNN has really large amounts of parameters, but does not have significant decoding performance gain.

[4]Due to time and resource limitation, we only tried (G)LSTM-DNN with $Tanh$.

Model	Scale	PN	WER(%)
Sigmoid-DNN	2048L7	45M	15.69
RELU-DNN	2048L7	45M	16.29
LSTM-DNN	2048L7	121M	**14.39**
LSTM-TIE-DNN	2048L7	58M	15.05
GLSTM-DNN	2048L11	230M	15.02

Table 3. Decoding results on Switchboard for baseline models and our proposed models, parameter number is also shown. All models are randomly initialized. Parameter number is calculated for each model.

In Figure 2 it is shown that our proposed model scales well with more layers on AMI. To investigate whether this scaling behavior extends to a large data set like 2000-hour Switchboard , small-scale GLSTM-DNN models of different layer numbers are tried, and their learning curves are shown in Figure 4. It is shown that the GLSTM-DNN can also achieve better performance with more layers on Switchboard.

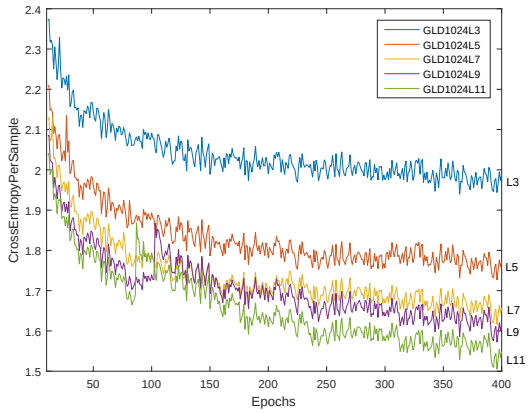

Fig. 4. Learning curves of GLSTM-DNN of different layers of 1024 hidden units

4. CONCLUSION AND FUTURE WORK

In this work, the modeling power of our proposed (G)LSTM-DNN is analyzed in various aspects. It is shown that our model is capable of utilizing deeper layers and get significant decoding performance gain on the Switchboard 2000 hour data set. We also proposed an generalized formulation, namely GLSTM. One concern, as discussed in section 3.1, is that the performance of the proposed (G)LSTM-DNN comes at a price of large parameter number, it would be worthwhile to find a configuration(e.g. using diagonal matrices) that saves parameter while retaining its modeling power.

Finally, an exciting next step would be to incorporate the LSTM-DNN back into the LSTM-RNN AM so that it would have LSTM structure both in depth and recurrence.

5. REFERENCES

[1] Jen-Tzung Chien and Tsai-Wei Lu, "Deep recurrent regularization neural network for speech recognition," in *Proc. ICASSP*, 2015.

[2] Hasim Sak, Andrew W. Senior, and Françoise Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," in *Proc. Interspeech*, 2014.

[3] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, 2013, pp. 273–278.

[4] Kai Chen, Zhi-Jie Yan, and Qiang Huo, "Training deep bidirectional lstm acoustic model for lvcsr by a context-sensitive-chunk bptt approach," in *Proc. Interspeech*, 2015.

[5] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "Lstm neural networks for language modeling," in *Interspeech*, Portland, OR, USA, Sept. 2012, pp. 194–197.

[6] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, Montreal, CA, 2014.

[7] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jrgen Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[8] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[9] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks.," in *AISTATS*, Yee Whye Teh and D. Mike Titterington, Eds. 2010, vol. 9 of *JMLR Proceedings*, pp. 249–256, JMLR.org.

[10] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, July 2006.

[11] Frank Seide, Gang Li, Xie Chen, and Dong Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *ASRU 2011*. December 2011, IEEE.

[12] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves, "Grid long short-term memory," *CoRR*, vol. abs/1507.01526, 2015.

[13] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber, "Highway networks," *CoRR*, vol. abs/1505.00387, 2015.

[14] Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer, "Depth-gated LSTM," *CoRR*, vol. abs/1508.03790, 2015.

[15] George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proc. ICASSP*, 2013.

[16] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu, "1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns," in *Proc. Interspeech*, 2014.