An Improved Residual LSTM Architecture for Acoustic Modeling

Lu Huang[1], *Jiasong Sun*[1], *Ji Xu*[2], *Yi Yang*[1]

[1]Department of Electronic Engineering, Tsinghua University, Beijing, China
[2]Key Laboratory of Speech Acoustics and Content Understanding,
Chinese Academy of Sciences, Beijing, China

huanglu.th@gmail.com, xuji@hccl.ioa.ac.cn, {sunjiasong,yangyy}@tsinghua.edu.cn

Abstract

Long Short-Term Memory (LSTM) is the primary recurrent neural networks architecture for acoustic modeling in automatic speech recognition systems. Residual learning is an efficient method to help neural networks converge easier and faster. In this paper, we propose several types of residual LSTM methods for our acoustic modeling. Our experiments indicate that, compared with classic LSTM, our architecture has more than 8% relative reduction in Word Error Rate (WER) on TIMIT tasks. At the same time, our residual fast LSTM approach shows 4% relative reduction in WER on the same task. Besides, we find that all these methods could have good results on THCHS-30 corpus.

Index Terms: long short-term memory, residual learning, acoustic modeling, speech recognition

1. Introduction

GRU

$$z_t = \sigma(U_z * x_t + W_z * s_{t-1}) \quad (1)$$

$$r_t = \sigma(U_r * x_t + W_r * s_{t-1}) \quad (2)$$

$$h_t = \tanh(U_h * x_t + W_h * (s_{t-1} \odot r_t)) \quad (3)$$

$$s_t = (1 - z_t) \odot h_t + z_t \odot s_{t-1} \quad (4)$$

$$y_t = W_p * s_t \quad (5)$$

$$s_t = y_t(1 : n_r) \quad (6)$$

Artificial Neural Networks (ANNs) had been widely researched in many Automatic Speech Recognition (ASR) systems for a long time. Since the year of 2011, Microsoft proposed its first deep neural networks which extremely improved the performance of speech recognition system [1]. After that, Deep Neural Networks (DNNs), Convolution Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been the most important research and development ways and tools. But there are still some problems on training deeper network for the existence of vanishing gradients and exploding gradients [2]. The standard LSTM-RNNs [3] has been designed to address these problems [4].

At the same time, the residual networks have been applied in image recognition [5] and speech recognition [**?**, **?**, **?**, **?**, **?**]. Especially, residual LSTM is proposed to improve the performance of speech recognition systems [**?**, **?**]. Meanwhile, fast LSTM [**?**, **?**] is proposed to reduce the training time without sacrificing performance, sometimes even with improved performance.

In this paper, inspired by Residual LSTM in [**?**], we propose an improved residual LSTM to replace vector's addition by splicing vector with various shortcut connection locations. They have good performance as the acoustic modeling on TIMIT [**?**] and THCHS-30 [**?**] tasks.

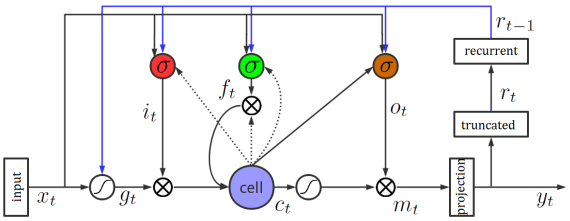

Figure 1: The projected LSTM in Kaldi.

We start by describing some fundamental LSTM architectures in Section 2, including standard LSTM, projected LSTM [4], fast LSTM and residual LSTM [**?**]. Then the improved residual LSTM is proposed in Section 3. After that, we provide some experiments and results on TIMIT and THCHS-30 corpora in Section 4, which is followed by conclusions in Section 5.

2. Fundamental LSTM-related Architectures

Since LSTM was proposed in [3], it has achieved significant performance in sequence labelling and prediction [**?**], especially as acoustic modeling and language modeling in ASR systems.

In this section, some LSTM-related works are provided, included projected LSTM, fast LSTM, and residual LSTM [**?**].

2.1. LSTM

In the year of 2014, the projected LSTM is proposed by Google for speech recognition [4], which is called LSTM projected (LSTMP) architecture in some other literatures. When compared with the standard LSTM, projected LSTM add a projection layer to the output, and also the recurrent vector is truncated from the output vector in Kaldi [**?**], as illustrated in Fig.1 and shown by the following equations.

$$i_t = \sigma(W_{ix}x_t + W_{ir}r_{t-1} + W_{ic}c_{t-1} + b_i) \quad (7)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}r_{t-1} + W_{fc}c_{t-1} + b_f) \quad (8)$$

$$g_t = \tanh(W_{cx}x_t + W_{cr}r_{t-1} + b_c) \quad (9)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (10)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}r_{t-1} + W_{oc}c_t + b_o) \quad (11)$$

$$m_t = o_t \odot \tanh(c_t) \quad (12)$$

$$y_t = W_{rp}m_t \quad (13)$$

$$r_t = y_t(1 : n_r) \tag{14}$$

Where the W represents weight matrix and the b is bias vector. For example, W_{ic} is the matrix of weights from cell activation vectors to input gate, b_i is the bias vector of input gate. i_t, f_t and o_t are the input gate, forget gate and output gate respectively. c_t and m_t are cell activation vector and cell output activation vector. All of i_t, f_t, o_t, c_t and m_t are the same size. Besides, g_t is the processed input, r_t is for recurrence, y_t is the output, W_{rp} is the projection matrix. $\odot$ stands for element-wise multiplication, $y_t(1 : n_r)$ means that r_t is the first n_r elements of y_t.

2.2. Fast LSTM

LSTM-RNN has more complexity than general RNN, which leads to a slower training speed. To accelerate its training, several algorithms have been done in previous work [**?**, **?**, **?**], in which the fast LSTM [**?**, **?**] can cut down half training time with incidentally improvement of performance. Compared to LSTM, fast LSTM doesn't use c_t or c_{t-1} to compute i_t, f_t and o_t, as shown in the following equations.

$$i_t = \sigma(W_{ix}x_t + W_{ir}r_{t-1} + b_i) \tag{15}$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}r_{t-1} + b_f) \tag{16}$$

$$o_t = \sigma(W_{ox}x_t + W_{or}r_{t-1} + b_o) \tag{17}$$

Obviously, equation (9), (10), (11) and (3) have the same inner operation, except that the active function is different for equation (3). Since these four equations have the same inner operation and inputs, a larger matrix is adopted in Kaldi to convert them into one operation with a large output, which contains four parts to represent the inputs of active functions: i_t, f_t, o_t and g_t. This kind of large matrix is more suitable for Graphics Processing Unit (GPU) computing than small matrices when they have the same amount of parameters.

2.3. Residual LSTM

As mentioned above, the DNN's training will become harder with the increase of depth. The reason is attributed to vanishing gradients and exploding gradients. Residual learning has been proposed to solve these problems in image recognition [5], and recently in speech recognition [**?**, **?**].

The output of each layer is the sum of network's input and network's output in [**?**], i.e., there is one shortcut connection between network's input and output, which has the same form as on most image recognition tasks. However, the shortcut connection is ended at one network's interior node, instead of network's output in [**?**].

3. The Improved Residual LSTM Architectures

Inspired by the principle in [**?**], our LSTM architectures have a short connection in LSTM on three different locations. Two of them are inner nodes and one of them is output, which is shown in Fig.2. Besides, we replace vector's addition by splicing vector in each location, and then project the spliced vector to the original dimension in location 1 and 3 to prevent the exploding dimension of vectors. In location 2, we reuse the projection matrix W_{rp} by increasing its input dimension.

When the junction is located at position 1, as shown in Fig.2, we name our improved Residual LSTM as LSTM Res-1, and the same is true for LSTM Res-2 and LSTM Res-3. So

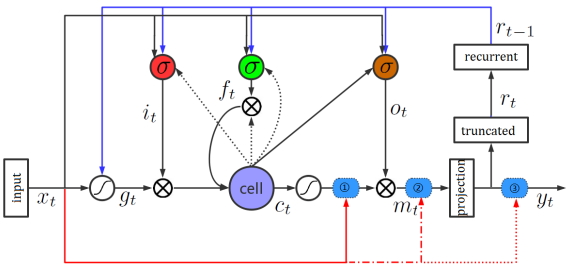

Figure 2: The improved residual LSTM with a shortcut connection from input to the later node and there are three locations.

compared to the original LSTM, our first method LSTM Res-1 has different equations from equation (6) as below.

$$h_t = (\tanh(c_t), x_t) \tag{18}$$

$$m_t = o_t \odot (W_{res1}h_t) \tag{19}$$

Where h_t is the spliced vector, W_{res1} is the projection matrix. Our second method LSTM Res-2 has different equations from equation (7) as below.

$$h_t = (m_t, x_t) \tag{20}$$

$$y_t = W_{res2}h_t \tag{21}$$

Where W_{res2} is the projection matrix for replacing W_{rp} in equation (7). Similarly, the third one LSTM Res-3 changes equation (7) and (8) into following equations.

$$z_t = W_{rp}m_t \tag{22}$$

$$r_t = z_t(1 : n_r) \tag{23}$$

$$h_t = (z_t, x_t) \tag{24}$$

$$y_t = W_{res3}h_t \tag{25}$$

Where z_t is an intermediate vector, W_{res3} is the projection matrix.

Since fast LSTM is faster to train, we also examine these thoughts into fast LSTM and test their performance on TIMIT and THCHS-30 tasks.

4. Experiments

We use nnet3 recipes in Kaldi, a popular toolkit for speech recognition, to implement our LSTM architectures. The input of neural networks is a 300 dimensional vector, consisting of 40 dimensional MFCC (Mel Frequency Cepstral Coefficient) features without cepstral truncation of five frames, which are spliced across in ± 2 frames of context, and 100 dimensional i-vectors to perform speaker adaptation [**?**].

Here all the implemented LSTM architectures have 2, 3 or 4 LSTM layers followed by a fully connected layer and a softmax layer. They are trained by cross-entropy (CE) criterion [**?**]. We use dual Nvidia GPUs to accelerate the training based on the parallel algorithm proposed in [**?**].

In order to evaluate our recipes quickly, speed perturbation [**?**] is not considered in our system. Speed perturbation is adopted to generate more training data for audio augmentation in Kaldi by default, but its performance improvement is very limited.

Since the training of LSTM is very slow, LSTM is just operated on TIMIT task. And these thoughts with fast LSTM are evaluated on THCHS-30 task.

Table 1: Results of TIMIT Task (WER%)

AM Types	#D	#P	core	complete
LSTM	2	9.6M	21.0	20.3
	3	14.3M	21.2	**20.0**
	4	19.0M	**20.8**	20.1
Residual LSTM [**?**]	2	9.8M	21.4	20.4
	3	14.6M	20.8	20.1
	4	19.3M	**20.7**	**19.7**
LSTM Res-1	2	12.5M	20.0	19.1
	3	18.8M	20.0	18.7
	4	25.1M	**19.3**	**18.4**
LSTM Res-2	2	10.0M	21.2	19.8
	3	15.0M	**19.8**	19.3
	4	20.0M	20.2	**19.1**
LSTM Res-3	2	10.5M	20.1	19.1
	3	15.8M	**19.9**	18.8
	4	21.0M	20.2	**18.6**
Fast LSTM	2	9.6M	20.3	19.3
	3	14.3M	20.1	18.7
	4	19.0M	**20.0**	**18.7**
Fast LSTM Res-1	2	12.5M	19.5	18.5
	3	18.8M	**19.2**	**18.1**
	4	25.1M	19.5	**18.1**
Fast LSTM Res-2	2	10.0M	19.9	18.6
	3	15.0M	**19.3**	18.2
	4	20.0M	19.6	**17.9**
Fast LSTM Res-3	2	10.5M	19.6	18.7
	3	15.8M	**19.2**	18.2
	4	21.0M	19.3	**18.0**

4.1. On TIMIT Task

TIMIT Speech Corpus [1] is widely used in acoustic-phonetic studies. in our experiments, 3.14 hours of data was selected as training data, with 0.16 hours as core test set and 0.81 hours as a complete test set, which was suggested by official [**?**]. The core test set is the subset of the complete test set.

The baseline LSTM implemented on TIMIT is the projected LSTM provided by Kaldi and Residual LSTM [**?**]. All LSTM, fast LSTM and their residual architectures have 1024-dimensional cell, with 512-dimensional recurrent projection and zero-dimensional no-recurrent projection, i.e., all elements of y_t is used for recurrence. Residual LSTM is the architecture proposed in [**?**]. All neural networks are trained for 15 epochs for the amount of training data is very small. The results are illustrated in Table 1[2].

According to the matrix W_{ic}, W_{fc} and W_{oc} in equation (1), (2) and (5), which are all diagonal matrices and have very small amount of parameters when compared them with other general matrices such as W_{ix} and W_{ir}, the number of parameters of LSTM is approximately equal to fast LSTM.

As shown in Table 1, Residual LSTM proposed in [**?**] has a very small improvement in performance; the residual methods we proposed outperform both baseline LSTM and Residual LSTM, with a relative reduction in WER about 8% on complete test set for LSTM Res-1. Besides, fast LSTM is not only faster than standard LSTM, but also outperform standard LSTM.

[1]There are more details on website: `https://catalog.ldc.upenn.edu/ldc93s1`.

[2]The depth here is the number of LSTM layers. #D means depth of LSTM, and #P stands for the amount of parameters.

Table 2: Results of THCHS-30 Task (WER%/PER%)

AM Types	#D	#P	word	phone
Fast LSTM	2	8.2M	22.94	9.53
	3	11.9M	22.74	9.15
	4	15.6M	**22.62**	**9.01**
Fast LSTM Res-1	2	11.2M	23.29	9.13
	3	16.4M	22.97	8.87
	4	21.7M	**22.83**	**8.74**
Fast LSTM Res-2	2	8.6M	22.64	9.31
	3	12.6M	22.36	8.75
	4	16.5M	**22.15**	**8.38**
Fast LSTM Res-3	2	9.2M	22.92	8.88
	3	13.4M	22.57	8.72
	4	17.6M	**22.40**	**8.31**

When applied to fast LSTM, our residual recipes also gain performance improvement, with a relative reduction in WER about 4% on both core test set and complete test set. More importantly, our fast LSTM Res recipes have the best performance. So on the following THCHS-30 task, we just apply these three ideas to fast LSTM.

4.2. On THCHS-30 Task

THCHS-30 (Tsinghua Chinese 30-hour database) is a free and open-source Chinese speech database[3]. On this task, we use about 27 hours of data for training and about 6 hours of data for testing [**?**].

Different with TIMIT task, the baseline fast LSTM and the three residual fast LSTM recipes here are implemented with 1024-dimensional cell, 256-dimensional recurrent projection and 256-dimensional no-recurrent projection. Besides, we train all neural networks for 8 epochs. When decoding, we perform both phone decoding and word decoding, which convert the speech signal into phone sequences and word sequences. The results are shown in Table 2[4].

Obviously, the recipes we proposed outperform the standard fast LSTM in most cases, especially on phone task. And in several cases, there are about 7% relative improvement of performance on phone task and about 2% relative improvement of performance on word task.

Meanwhile, the best result provided by CSLT@THU is 23.18% on word task and 10.01% on phone task [**?**]. And our best result is 22.15% on word task and 8.31% on phone task, with 4.4% relative reduction in WER and 16.9% relative reduction in PER.

5. Conclusions

In this paper, we propose our improved residual LSTM architectures for acoustic modeling. The experiments on TIMIT corpus show that our recipes outperform the standard LSTM and the Residual LSTM proposed in [**?**], with about 8% reduction in WER on complete test set. Also, we apply our thoughts to fast LSTM and achieve our best performance in both TIMIT corpus and THCHS-30 corpus. Besides, the best performance was shown by using our fast LSTM structure on THCHS-30 task.

[3]There are more details on website: `http://data.cslt.org/thchs30/README.html`.

[4]PER means phone error rate. WER is for word decoding, and PER is for phonetic decoding.

Next, we are interested in bidirectional LSTM [**?**] and deeper LSTM. We would also like improving performance on other corpora by using our improved residual LSTM architectures.

6. Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (Nos. 11590770-4).

7. References

[1] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[2] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

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

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

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.