INVESTIGATIONS ON SPEAKER ADAPTATION OF LSTM RNN MODELS FOR SPEECH RECOGNITION

Chaojun Liu, Yongqiang Wang, Kshitiz Kumar, Yifan Gong

Microsoft Corporation, Redmond WA, USA
{chaojunl, erw, kskumar, ygong}@microsoft.com

ABSTRACT

Recently Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNN) acoustic models have demonstrated superior performance over deep neural networks (DNN) models in speech recognition and many other tasks. Although a lot of work have been reported on DNN model adaptation, very little has been done on LSTM model adaptation. In this paper we present our extensive studies of speaker adaptation of LSTM-RNN models for speech recognition. We investigated different adaptation methods combined with KL-divergence based regularization, where and which network component to adapt, supervised versus unsupervised adaptation and asymptotic analysis. We made a few distinct and important observations. In a large vocabulary speech recognition task, by adapting only 2.5% of the LSTM model parameters using 50 utterances per speaker, we obtained 12.6% WERR on the dev set and 9.1% WERR on the evaluation set over a strong LSTM baseline model.

Index Terms— Long Short-Term memory (LSTM), Recurrent Neural Networks, Speaker Adaptation, Acoustic Modeling

1. INTRODUCTION

The application of deep neural networks (DNN) to speech recognition has achieved tremendous success due to its superior performance over the traditional GMM/HMM acoustic models ([1], [2]). It has become the dominant acoustic modeling approach for speech recognition, especially for large vocabulary tasks. While it has strong modeling power through multiple layers of nonlinear processing, it is still not immune to many known problems for GMM/HMM based systems such as mismatch of training and test data. When tested in unseen conditions or unseen test speakers, it still has performance degradation. To address this problem, many adaptation techniques have been proposed. There are several categories of speaker adaptation approaches. First, speaker-independent (SI) DNN model, or certain layer(s) of the model are updated on adaptation data ([3] [4]). To avoid over-fitting, regularization such as [3] is applied. Second, inserting and adapting speaker-dependent linear layer into the network to transform either input feature (fDLR [5]), top-hidden-layer output ([6]), or hidden layer activations ([7] [8] [9])). Third, using speaker adaptive features ([5] [10]), or augumenting input features with speaker information ([11] [12] [13] [14] [15]). Fourth, subspace method such as [16] [17] [18].

Another problem with DNN is that it only provides limited temporal modeling by operating on a fixed-size sliding windows of acoustic frames. As a contrast, recurrent neural networks (RNN) can model long-term dependency due to its recurrent (loop) structure. However standard RNN training suffers from the gradient vanishing and exploding problem ([19]), long short-term memory (LSTM) RNN has been proposed [20] to address the issue and achieved great success in many tasks. Recently it has been applied to speech recognition ([21] [22] [23] [24] [25]) and demonstrated superior performance. A natural question is that once we switch to LSTM-RNN models whether we can we still do effective speaker adaptation as for DNN and what are the issues specific to LSTM-RNN structure. Although a lot of studies have been done for DNN speaker adaptation, there is very little research on LSTM-RNN model adaptation. The only one we are aware of is [26] where only a modest gain is demonstrated on Switchboard task.

In this paper we present an extensive study on speaker adaptation of LSTM-RNN models. We investigated different adaptation methods, in combination with KL-divergence based regularization, which is shown to be crucial. We explored every network component to discover the most effective one to adapt. We had a few distinct and important observations, different from [26]. We also compared supervised vs. unsupervised adaptation, LSTM adaptation vs. DNN adaptation, and conducted an asymptotic analysis. In a large vocabulary speech recognition task, by adapting only 2.5% of the model parameters using 50 utterances per speaker, we obtained 12.6% WERR on the dev set and 9.1% WERR on the evaluation set, over a strong LSTM baseline model.

The remainder of this paper is organized as follows: Section 2 reviews LSTM RNN model; Section 3 presents LSTM adaptation methodology; Section 4 presents the experimental results; We present our conclusions and thoughts for future work in Section 5.

2. LSTM RNNS

Different from feedforward DNNs, RNNs are able to model the temporal dynamics by using some form of memory. Given an input sequence $\mathbf{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T)$, a RNN layer computes the hidden states $\mathbf{H} = (\boldsymbol{h}_1, \cdots, \boldsymbol{h}_T)$, *iteratively* via $\boldsymbol{h}_t = \mathcal{H}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}; \mathcal{M})$ Here, $\mathcal{M}$ is the parameter of the RNN and $\mathcal{H}$ is a nonlinear function. In the LSTM architecture, the nonlinear $\mathcal{H}$ function is calculated using the following equations:

$$\boldsymbol{i}_t = \sigma(\mathbf{W}_{\mathtt{ix}}\boldsymbol{x}_t + \mathbf{W}_{\mathtt{ih}}\boldsymbol{h}_{t-1} + \mathbf{W}_{\mathtt{ic}}\boldsymbol{c}_{t-1} + \boldsymbol{b}_{\mathtt{i}}) \quad (1)$$

$$\boldsymbol{f}_t = \sigma(\mathbf{W}_{\mathtt{fx}}\boldsymbol{x}_t + \mathbf{W}_{\mathtt{fh}}\boldsymbol{h}_{t-1} + \mathbf{W}_{\mathtt{fc}}\boldsymbol{c}_{t-1} + \boldsymbol{b}_{\mathtt{f}}) \quad (2)$$

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

$$\boldsymbol{o}_t = \sigma(\mathbf{W}_{\mathtt{ox}}\boldsymbol{x}_t + \mathbf{W}_{\mathtt{oh}}\boldsymbol{h}_{t-1} + \mathbf{W}_{\mathtt{oc}}\boldsymbol{c}_t + \boldsymbol{b}_{\mathtt{o}}) \quad (4)$$

$$\boldsymbol{h}_t = \phi(\boldsymbol{c}_t) \odot \boldsymbol{o}_t \quad (5)$$

Here $\odot$ denotes element-wise multiplication, ϕ is a nonlinear function which squashes the input to $[-1, 1]$; $\boldsymbol{c}_t$ denotes the states the memory cells; $\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t$ are the input gates, forget gates and output gates of the memory cells respectively; $\mathbf{W}_{\mathtt{ic}}, \mathbf{W}_{\mathtt{fc}}$ and $\mathbf{W}_{\mathtt{oc}}$ are the *peephole* connection matrices, which are usually diagonal. Moreover, it is proposed in [23] that a recurrent projection matrix $\mathbf{P}$ can be used to transform the output to a low-dimensional space, i.e, the

Eq. (5) is replaced by the following equation:

$$\boldsymbol{h}_t = \mathbf{P}(\phi(\boldsymbol{c}_t) \odot \boldsymbol{o}_t) \tag{6}$$

Given the above LSTM structure, a deep LSTM can be built by stacking multiple layers. The hidden state of the last LSTM layer, $\boldsymbol{h}_t^{(L)}$, is used to predict the senone posterior $\boldsymbol{y}_t$ using the softmax operation, i.e., $\boldsymbol{y}_t = \text{softmax}(\mathbf{W}\boldsymbol{h}_t^{(L)} + \boldsymbol{b})$. Figure 1 illustrates a L-layer deep LSTM architecture with optional affine transform, $\mathcal{A}^{(l)}$, between layers, which will be discussed in the next section.

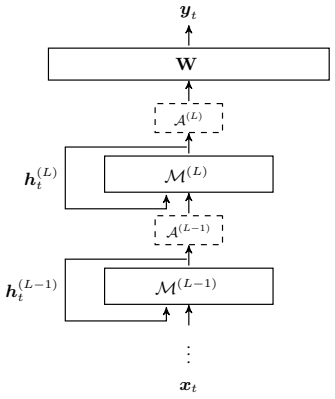

Fig. 1. A deep LSTM neural network, with optional affine transforms between layers, is used to convert the input $\boldsymbol{x}_t$ to the senone posteriors $\boldsymbol{y}_t$.

3. ADAPTATION OF LSTM RNN MODEL

A practical constraint for a large scale speech recognition system is that the system needs to serve many users; therefore the user specific parameters should be kept small. The main goal of this investigation is to develop methods to effectively adapt the speaker-independent (SI) model using a minimal number of speaker-specific parameters. Two approaches are studied in this work: adapting existing neural network components and adapting inserted affine transformation between layers.

3.1. Adapt Existing Network Components

Various network components can be modified to a particular speaker to reduce the mismatch introduced by speaker differences. There are three kinds of matrices in a LSTM layer :

- Matrices connecting static input ($\boldsymbol{x}_t$) to gates and cells, i.e., $\mathbf{W}_{\mathtt{ix}}, \mathbf{W}_{\mathtt{fx}}, \mathbf{W}_{\mathtt{ox}}$ and $\mathbf{W}_{\mathtt{cx}}$;
- Matrices connecting recurrent input ($\boldsymbol{h}_{t-1}$) to gates and cells, i.e., $\mathbf{W}_{\mathtt{ih}}, \mathbf{W}_{\mathtt{fh}}, \mathbf{W}_{\mathtt{oh}}$ and $\mathbf{W}_{\mathtt{ch}}$;
- The $\mathbf{P}$ matrix which transforms the cell output to a low dimension input for the next frame.

Since the second and third kinds of matrices are inside the recurrent loop, they are also impacted by frames in history. One issue mentioned in [26] is that adapting weight matrices inside the recurrent loop using limited data may not be robust. However, we believe that with a proper regularization, a reliable adaptation can be achieved. On the other hand, we argue that adapting matrices inside the loop can potentially be more powerful than the non-recurrent counterpart as some speaker characteristics exhibits in the temporal pattern (e.g. speaking style). This will be demonstrated in the Section 4.

3.2. Affine Transformation Based Adaptation

Inspired by the success of SVD-based DNN adaptation [7], we propose to insert an affine transform on top of each LSTM layer, as shown in the dashed box in Figure 1. The transformation matrix is square and has the same size as the LSTM output $\boldsymbol{h}_t$. Due to the projection matrix inside LSTM layer, its output dimension is smaller than the cell dimension; therefore the inserted affine transform matrix is smaller than those internal matrices inside LSTM. Note that the transform matrix is inserted after the recurrent connection ($\boldsymbol{h}_t$ to $\boldsymbol{h}_{t-1}$); hence it does not affect the recurrent hidden activations.

In the above two approaches, either some model parameters are updated for each speaker or a speaker specific transformation is used. For convenience, we denote these speaker-specific parameters for the s-th speaker as $\mathbf{M}_s$.

3.3. KL-Divergence Regularized Adaptation

To estimate $\mathbf{M}_s$, a conventional approach is to estimate them on the adaptation data using the cross entropy criterion, i.e.,

$$\mathbf{M}_{\mathtt{s}} = \arg\min_{\mathbf{M}} \mathcal{E}\{\text{KL}(p^{\text{data}}(\boldsymbol{y}_t|\boldsymbol{x}_t)||p(\boldsymbol{y}_t|\boldsymbol{x}_t, \mathbf{M}))\} \tag{7}$$

where the expectation is taken on the s-th speaker's adaptation data; $p(\boldsymbol{y}_t|\boldsymbol{x}_t, \mathbf{M})$ is the posterior distribution predicted using speaker-specific parameters $\mathbf{M}$, and $p^{\text{data}}(\boldsymbol{y}_t|\boldsymbol{x}_t)$ is the ground-truth posterior distribution; if a hard alignment is used, $p(y|\boldsymbol{x}_t) = \delta(y = s_t)$ where δ is Kronecker delta and s_t is the label of the $\boldsymbol{x}_t$.

However, when there is limited adaptation data, directly optimizing this cross entropy criterion often leads to poor performance due to over-fitting. KL-Divergence regularization is proposed [3] to force the senone distribution estimated from the adapted model to be close to that from the baseline model, i.e, the following criterion is used:

$$\begin{aligned}\mathbf{M}_{\mathtt{s}} = \arg\min_{\mathbf{M}} \mathcal{E}\{&(1-\rho)\text{KL}(p^{\text{data}}(\boldsymbol{y}_t|\boldsymbol{x}_t)||p(\boldsymbol{y}_t|\boldsymbol{x}_t, \mathbf{M}) \\ &+\rho\text{KL}(p(\boldsymbol{y}_t|\boldsymbol{x}_t, \mathbf{M}^{\text{si}})||p(\boldsymbol{y}_t|\boldsymbol{x}_t, \mathbf{M}))\}\end{aligned} \tag{8}$$

where ρ is the regularization weight, and $\mathbf{M}^{\text{si}}$ is the unadapted speaker-independent parameters.

4. EXPERIMENTS

We conducted experiments on a Microsoft US-English mobile phone personal assistant (Cortana) task. Training and test data set contains mobile phone voice search (VS), short message dictation (SMD) and commands. Training set has 1200 hour live audio with human transcriptions. Two test sets are used. For evaluation of the baseline DNN and LSTM models, a large test set of 125K words (referred as cortana-test below) without speaker information is used. For speaker adaptation experiments, a data set of 50 speakers is used, each has 100 to 200 utterances, out of which 50 for testing (referred as spks50-test, in total 12K words), the remaining for adaptation. Except for the asymptotic analysis, only 50 utterances per speaker is used for adaptation, as this represents the most typical scenario in our system. There is no time overlap between adaptation set and test set. For development (e.g. tuning hyper-parameters) purpose and fast experiment turn-around, we use another data set of 10 speakers (also 50 utterances for adaptation, 50 for testing).

4.1. Baseline Systems

The front-end generates 29-dimensional log-mel filterbank features plus their 1st and 2nd order derivatives, resulting a 87-dimensional feature vector per frame. DNN uses a sliding context window of

11 frames, so network input dimension is 957. LSTM-RNN uses a single frame feature vector as input. A GMM-HMM system is first built with 5976 context-dependent tied HMM states (or senones). Training data is force-aligned with this model to generate frame level targets for DNN and LSTM-RNN training, so both DNN and LSTM-RNN have the same output layer size of 5976. Our best DNN model contains 5 hidden layers, each layer has 2048 nodes. Our best LSTM-RNN model has 4 stacked LSTM layers with projection, each layer has 1024 memory cells and 512 output units. Both DNN and LSTM-RNN were trained to minimize the frame-level cross-entropy criterion. LSTM-RNN training uses the truncated back propagation through time (BPTT) algorithm (back to 20 frames). To speed up training, we stack 20 utterances in parallel in a mini-batch. We delay the output HMM state label by 3 frames so as to use future frames to help LSTM make better predictions for the current frame. All experiments were conducted using the Computational Networks toolkit (CNTK) ([27]). Table 1 gives the comparison of DNN model and LSTM-RNN model. LSTM-RNN outperforms DNN with a strong 10.3% WERR on the large Cortana test set and 11.0% on the 50-speaker set.

Table 1. WERs (in %) of DNN and LSTM-RNN models.

model	#parameters (million)	Cortana-test	spks50-test
DNN	35.2	15.73	22.85
LSTM-RNN	20.2	14.11	20.34

4.2. Speaker Adaptation of LSTM RNN models

We first investigate the role of KLD regularization for LSTM adaptation. Table 2 shows the results of adapting the full LSTM network, on the 10-speaker dev set. Without regularization, only 5.6% WERR is obtained but with a regularization weight of 0.2, we get 13.2% WERR. This clearly shows the importance of regularization. Figure 2 shows WERR over SI LSTM model when adapting the recurrent projection matrix in the last layer ($\mathbf{P}^{(4)}$). Again we can see with a proper degree of regularization, significantly more improvement can be obtained even with adapting only a single matrix. Without regularization, 7.3% WERR is obtained, but with $\rho = 0.3$, 12.6% WERR is obtained on the 10-speaker dev set. We notice that with $\rho = 1.0$, which means the training target is completely determined by the baseline model and any new information from the adaptation data is ignored. This, to no surprise, causes regression. We use KLD-regularization in following experiments.

Table 2. Adapting full model with different regularization. Numbers in the brackets are the WERRs (%) compared with baseline.

model	WER%
SI-LSTM	24.9
$\rho = 0.0$	23.5 (5.6)
$\rho = 0.2$	21.6 (13.2)
$\rho = 0.5$	21.9 (12.0)

4.2.1. Supervised vs. Unsupervised Adaptation

Another question we want to answer is that as a sequential model in nature, whether LSTM-RNN is more sensitive to training target errors. Presumably a training target error not only affect the current frame, it will also propagate to following frames for RNN model training. We compared supervised adaptation (with human transcription) and unsupervised adaptation (with a real production recognizer

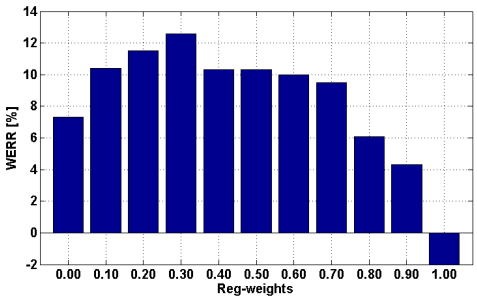

Fig. 2. WERR% of adapting 4th LSTM layer projection matrix with different KLD-regularization weights on the 10-speaker set.

Table 3. Supervised vs. unsupervised speaker adaptation. Numbers are WERRs(%) over the unadapted model, on spks50-test set.

mode / Model	LSTM	DNN
unsupervised	9.1	11.5
supervised	15.1	19.5

generated hypothesis as transcription, the recognizer used a higher quality DNN model and language model) of the same projection matrix ($\mathbf{P}^{(4)}$) as above, but on the 50-speaker set. Results are given in Table 3. We can clearly see the large gap between supervised and unsupervised adaptation (15.1% WERR vs. 9.1% WERR) for LSTM adaptation. Results of DNN model adaptation on the same speakers (though a different DNN model, same structure but trained on more data) were also given in the table (19.5% WERR vs 11.5% WERR). In both LSTM and DNN models, supervised adaptation delivers nearly 70% more gain over unsupervised adaptation. Although from these numbers it remains unclear whether LSTM adaptation is more sensitive to training target errors, it would be worthwhile to investigate methods (e.g. using confidence score, user feedback such as click-through [28]) to bridge the large gap between supervised and unsupervised adaptation. Comparing along the horizontal line, we can see DNN speaker adaptation gives more improvement than LSTM adaptation. This seems to suggest that LSTM models already capture and normalize some of the speaker characteristics, therefore less mismatch between training and testing speakers. As transcribing data for every user is not practical, we will focus on unsupervised speaker adaptation in following experiments.

4.2.2. Adapting Existing Network Components

We try to find and adapt the most effective component to match the test speaker. We begin with adapting the softmax layer and the projection weight matrix in each LSTM layer. Results (in Table 4) are on spks50-test set. We can see simply adapting the softmax layer matrix gives a descent gain of 7.7% WERR. However one problem with this approach is that only connections to the seen units in the adaptation data will be updated. Adapting layers below softmax usually leads to better generalization. A 9.1% WERR is obtained by adapting the top hidden layer projection matrix. One distinct observation is that adapting the projection matrix from within the top LSTM layer to within the bottom LSTM layer becomes progressively less effective. We consistently observe this pattern in our experiments. This is different from DNN, where both [29] and [30, 31] show a U-shape curve for adaptation at different DNN layer. Another remark is that the output from projection not only feeds to the next layer as input, it also feeds to the same hidden layer (recurrent) at

Table 4. Adapting softmax layer matrix and projection matrices in different layers. WERRs are improvement over baseline in Table 1.

Adapted Param.	$\mathbf{W}$	$\mathbf{P}^{(4)}$	$\mathbf{P}^{(3)}$	$\mathbf{P}^{(2)}$	$\mathbf{P}^{(1)}$
WER (%)	18.77	18.50	19.01	19.22	19.89
WERR (%)	7.7	9.1	6.6	5.5	2.2

Table 5. Adapting component matrices that controls the gates and the LSTM cell update in the 4-th layer. Results are on the spks50-test set.

Param.	$\mathbf{W}^{(4)}_{\mathtt{ch}}$	$\mathbf{W}^{(4)}_{\mathtt{oh}}$	$\mathbf{W}^{(4)}_{\mathtt{fh}}$	$\mathbf{W}^{(4)}_{\mathtt{ih}}$
WER(%)	18.59	18.69	19.01	19.32
WERR(%)	8.5	8.1	6.6	5.0
Param.	$\mathbf{W}^{(4)}_{\mathtt{cx}}$	$\mathbf{W}^{(4)}_{\mathtt{ox}}$	$\mathbf{W}^{(4)}_{\mathtt{fx}}$	$\mathbf{W}^{(4)}_{\mathtt{ix}}$
WER(%)	19.23	18.90	18.80	19.50
WERR(%)	5.4	7.1	7.6	4.1

next frame. Our observation is different from [26] where it is shown that if adaptation occurs inside the loop, it is much less effective.

Now we explore adaptation of each weight matrix inside the top LSTM layer. Results are given in Table 5. Note that all these matrices have the same size (1024x512). We can see that it is more effective to adapt the weight matrices from the hidden layer activations back to those gates than the input-to-gates matrices. Adapting how much past information back into the cell ($\mathbf{W}^{(4)}_{\mathtt{ch}}$) gives the most gain (8.5% WERR) while adapting current input to the cell is much less effective (5.4% WERR), suggesting that the sequential nature of the model indeed can capture temporal dynamics, which carries speaker characteristics to some degree (e.g. speaking rate and style). Among the three gates, adapting output gate is the most effective (7-8% WERR), while adapting input gate is the least effective (only 4-5% WERR).

Adapting combinations of multiple weight matrices (e.g. all matrices to each gate, respectively) on the dev set does not show extra gain over the best single matrix and therefore is not reported. We also tried adapting the matrices for the peephole connections (note that these matrices are diagonal). It does not show any gain by adapting them alone, or add extra gain when combining with other full weight matrices. Same for adapting the bias vectors. We also observe that on the dev set, adapting lower layer LSTM cell internal matrices gives less gain than adapting those in the top LSTM layer.

4.2.3. Adapting Inserted Linear Layer

We insert a speaker-specific linear transform on top of each LSTM layer to transform the hidden activations. Table 6 gives the results. We see a similar pattern that making speaker specific transformation in the higher part of the network is more beneficial. Adapting the transformation inserted at the top layer gives the best improvement (8.2% WERR), but there is very little gain by adapting the transformation inserted at the bottom layer. We also tried adapting the input features by inserting a speaker-specific linear transformation between the input and the first layer, it did not work. This is different from the observation in [26] where the input feature transform (IFT) method gave small (3-4% WERR) gain for speaker independent features. One reason could be that our network is deeper (4 LSTM layers instead of 2), the bottom layer suffers more from gradient vanishing. Comparing adapting the transformation inserted at the top layer with adapting $\mathbf{P}^{(4)}$ in Table 4, we can see that adaptation inside the recurrent loop gives slightly better performance (8.2% vs 9.1% WERR).

Table 6. Adapting affine transforms inserted at different layers. Results are on the spks50-test set.

Inserted at layer	4	3	2	1
WER (%)	18.68	18.96	19.06	19.89
WERR (%)	8.2	6.8	6.3	2.2

4.2.4. Asymptotic Analysis

In this study we vary the amount of adaptation data per speaker and compare WERR on the fixed test set. This is a different data set containing 13 speakers with 600 human transcribed utterances. We keep 50 utterances per speaker for testing. $\mathbf{P}^{(4)}$ matrix is adapted in supervised mode [1]. Results are shown in Figure 3. We can see that more data gives better performance, and it saturates at around 100-300 utterances. With only 10 utterances, adaptation gives very little gain, but with 400 utterances it cuts nearly half of the errors. It also suggests that for power users we should keep accumulating data until about 400 utterances to get the most benefit before we slide the data window to keep freshness of adaptation data. It is also interesting to see a sharp WERR increase from 50 utterances to 75 and 100.

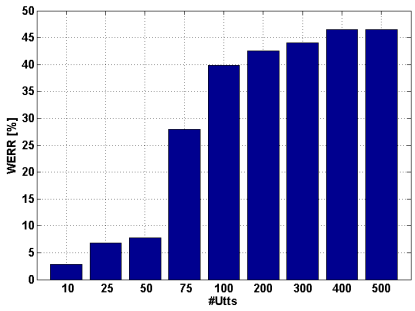

Fig. 3. Asymptotic behavior of supervised LSTM model adaptation: WERR(%) versus the number of adaptation utterances.

5. CONCLUSIONS AND FUTURE WORK

We summarize this work with a few conclusions: (a) KLD-based regularization is crucial to obtain good performance in speaker adaptation with little data; (b) Unlike DNN model where intermediate layer adaptation is more effective than other layers, for stacked LSTM-RNN model, adapting top hidden layers is clearly more effective than adapting lower layers. Adapting input feature transformation layer does not work for deeply-stacked LSTM model; (c) Both adapting existing network components and adapting inserted speaker-dependent layer are effective; (d) Adapting LSTM cell internal matrices are effective, among which adapting the projection weight matrix and hidden activations to cell internal memory are the most effective; adapting the top layer projection matrix gives a large improvement of 9.1% WERR with only 50 unsupervised utterances; (e) Asymptotic analysis suggests that we can start building a user specific model with as few as 25 utterances, and sliding the data window once the amount of data is over 400 utterances. For our future work, we will investigate ways to build more compact speaker-dependent components, and reduce the WERR gap between supervised and unsupervised adaptation.

[1] The recognizer-generated hypotheses for these data were not available at the time this study was performed.

6. REFERENCES

[1] G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, 2012.

[2] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, pp. 30–34, 2012.

[3] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *International Conference on Acoustic, Speech and Signal Processing*, 2013.

[4] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *International Conference on Acoustic, Speech and Signal Processing*, 2013.

[5] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

[6] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong, "Adaptation of context-dependent deep neural networks for automatic speech recognition," in *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2012.

[7] J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *International Conference on Acoustic, Speech and Signal Processing*, 2014.

[8] B. Li and K. C. Sim, "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid nn/hmm systems," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2010.

[9] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural networks acoustic models," in *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2015.

[10] S. P. Rath, D. Povey, K. Vesely, and J. Cernocky, "Improved feature processing for deep neural networks," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2013.

[11] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[12] A. Senior and I. Lopez-Moreno, "Improving dnn speaker independence with i-vector inputs," in *International Conference on Acoustic, Speech and Signal Processing*, 2014.

[13] Y. Miao, H. Zhang, and F. Metze, "Towards speaker adaptive training of deep neural network acoustic models," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2014.

[14] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid nn/hmm model for speech recognition based on discriminative learning of speaker code," in *International Conference on Acoustic, Speech and Signal Processing*, 2013.

[15] P. Karanasou, Y. Wang, M. Gales, and P. Woodland, "Adaptation of deep neural network acoustic models using factorised i-vectors," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2014.

[16] S. Dupont and L. Cheboub, "Fast speaker adaptation of artificial neural networks for automatic speech recognition," in *International Conference on Acoustic, Speech and Signal Processing*, 2000.

[17] C. Wu and M. Gales, "Multi-basis adaptive neural network for rapid adaptation in speech recognition," in *International Conference on Acoustic, Speech and Signal Processing*, 2015.

[18] T. Tan, Y. Qian, M. Yin, Y. Zhuang, and K. Yu, "Cluster adaptive training for deep neural networks," in *International Conference on Acoustic, Speech and Signal Processing*, 2015.

[19] Y. Bengio, P. Simard, and P. Frasconi, "Learning long term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, pp. 157–166, 1994.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–1780, 1997.

[21] A. Graves, A. Mohamed, and G. Hilton, "Speech recognition with deep recurrent neural networks," in *International Conference on Acoustic, Speech and Signal Processing*, 2013.

[22] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bi-directional LSTM," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

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

[24] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, "Sequence discriminative distributed training of long short-term memory recurrent neural networks," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2014.

[25] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *International Conference on Acoustic, Speech and Signal Processing*, 2015.

[26] Y. Miao and F. Metze, "On speaker adaptation of long short-term memory recurrent neural networks," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2015.

[27] D. Yu, A. Eversole, M. Seltzer, et al., "An introduction to computational networks and the computational network toolkit," *Microsoft Technical Report MSR-TR-2014-112*, 2014.

[28] Y. Zhao, J. Li, J. Xue, and Y. Gong, "Investigating online low-footprint speaker adaptation using generalized linear regression and click-through data," in *International Conference on Acoustic, Speech and Signal Processing*, 2015.

[29] T. Ochiai, S. Matsuda, X. Lu, C. Hori, and S. Katagiri, "Speaker adaptive training using deep neural networks," in *International Conference on Acoustic, Speech and Signal Processing*, 2014.

[30] K. Kumar, C. Liu, K. Yao, and Y. Gong, "Intermediate-layer dnn adaptation for offline and session-based iterative speaker adaptation," in *Annual Conference of the International Speech Communication Association (Interspeech)*, 2015.

[31] K. Kumar, C. Liu, and Y. Gong, "Non-negative intermediate-layer DNN adaptation for a 10-kb speaker adaptation profile," in *International Conference on Acoustic, Speech and Signal Processing*, 2016.