UNSUPERVISED ADAPTATION FOR DEEP NEURAL NETWORKS USING ALTERNATING DIRECTION METHOD OF MULTIPLIERS

Roger Hsiao, Tim Ng and Man-Hung Siu

Raytheon BBN Technologies
10 Moulton Street, Cambridge, MA 02138, USA
{whsiao,tng,msiu}@bbn.com

ABSTRACT

In this paper, we continue our work on linear least squares based adaptation (LLS) for deep neural networks. We show that our previously proposed algorithm is a special case of an optimization algorithm called Alternating Direction Method of Multipliers (ADMM). We demonstrate that the adaptation algorithm can improve the performance on various deep neural networks including the bidirectional long short term memory (BLSTM). On the Switchboard subset of the Hub5 2000 evaluation set, we show that LLS adaptation can achieve 6 to 9% relative word error rate (WER) reduction, and improve our two-pass system to 7.5% WER. In this paper, we also analyze the factors that could contribute to the success of an adaptation algorithm. This helps us to understand under what circumstances, adaptation could improve the system performance.

Index Terms— Speech recognition, unsupervised adaptation, Switchboard

1. INTRODUCTION

The goal of adaptation is to improve system performance by using a small amount of target data. Back in the days when Gaussian Mixture Models (GMM) was the mainstream in acoustic modeling, adaptation algorithms like maximum likelihood linear regression (MLLR) [1], constrained MLLR (CMLLR) [2] and maximum-a-posteriori (MAP) [3] have shown to be effective for adaptation and consistently improved system performance on different domains, languages and conditions. Since deep neural networks (DNN) became the prevalent acoustic model, many new adaptation algorithms have been developed, but the task remains challenging. One of the reasons is the sheer amount of parameters in a DNN. With limited amount of adaptation data, estimating the parameters reliably is difficult.

In this paper, we focus on model space adaptation approaches. Existing model space approaches like fine-tuning [4, 5, 6] aim to adjust the parameters with careful tuning and regularization methods. While these techniques are successful for supervised adaptation, results on unsupervised adaptation are mixed. Learning hidden unit contributions(LHUC) is a novel adaptation algorithm [7, 8]. Instead of estimating the DNN parameters directly, LHUC learns a scaling factor for each hidden unit so the activation could be adjusted based on the adaptation data. LHUC shows successful results on unsupervised adaptation, however, the performance varies on different test sets. For instance, [8] shows that LHUC is effective on smaller tasks like AMI and TED talks, it fails to improve bigger tasks like Switchboard. Generally speaking, the model space approach may not work consistently across different domains or conditions, and it is often unclear about the underlying reasons that contribute to the success or failure of an algorithm.

In this paper, we continue our work on linear least squares based adaptation (LLS) [9]. We show that our approach would work on the state-of-the-art large scale speech recognition systems. We also recast our algorithm as a special case of an optimization algorithm called Alternating Direction Method of Multipliers (ADMM) [10]. We analyze the factors that contribute to the success of an algorithm, and resulting in a set of conditions that are required by our adaptation algorithm. We believe the same analysis method can be applied to study other adaptation algorithms.

This paper is organized as follows: In section 2, we describe our adaptation algorithm and how it connects to ADMM. In section 3, we describe our experiments on Switchboard. Section 4 is an analysis of our results and we study the behavior of our proposed algorithm. Finally, we conclude our work in section 5.

2. LINEAR LEAST SQUARES ADAPTATION AND ADMM

ADMM is a Lagrange multipliers based optimization algorithm [10]. Instead of using back-propagation to train a deep model, it transforms the optimization problem into a series of LLS problems where each of them has a simple and tractable solution. ADMM can be used to optimize most deep models but for simplification, we focus on describing how ADMM can be used to optimize a DNN with fully connected layers. However, the idea can be applied to more complicated models like recurrent neural networks.

We define a DNN using the following notations: a DNN may contain $N + 2$ layers where the first layer (indexed with 0) is the input layer, and the last layer (indexed with $N + 1$) is the output layer. Layer 1 to N are the hidden layers and each layer contains K_i units. Each unit has an activation function. Mathematically, each layer of the DNN can be evaluated by the following equations,

$$z_t^i = f_i(y_t^i) \quad (1)$$

$$y_t^i = A_i x_t^i + b_i \quad (2)$$

$$x_t^i = z_t^{i-1} \quad \text{if } i > 0 \quad (3)$$

where z_t^i is the output of the i-th layer; f_i is the activation function of the i-th layer, say a sigmoid function for the hidden layers and a softmax function for the output layer; A_i and b_i are the weights of the i-th layer; x_t^i is the input to the i-th layer and z_t^{i-1} is the output of the $(i-1)$-th layer which serves as an input to the i-th layer.

For simplicity, assuming that we optimize i-th layer of a DNN with an objective function, E, our goal is to estimate A_i and b_i such that E is minimized. Conventional back-propagation algorithm

would use gradient descent, but for ADMM, it introduces a slack variable y_t^{i*} such that the optimization problem becomes

$$\begin{aligned} \underset{A_i,b_i,y_t^{i*}}{\text{argmin}} \quad & E(A_i,b_i,y_t^{i*}) \\ \text{s.t.} \quad & z_t^i = f_i(y_t^{i*}) \\ & y_t^{i*} = y_t^i = A_i x_t + b_i \end{aligned} \tag{4}$$

The purpose of introducing this slack variable, y_t^{i*}, is to separate the optimization problem into two parts. The first part tries to find y_t^{i*}, which represents an ideal internal activation that minimizes E. Then, A_i and b_i are optimized to approximate y_t^{i*}. The first part can be done using gradient descent and the second part can be formulated as a simple LLS problem. Mathematically, this entire process can be described using augmented Lagrangian [11],

$$L = E + \lambda_t'(y_t^{i*} - y_t^i) + \frac{\alpha}{2}||y_t^{i*} - y_t^i||_2^2 \tag{5}$$

where λ is a vector of Lagrange multipliers; α is a term to control the weight of the augmentation term, which serves for the regularization purpose.

The method of Lagrange multipliers can be considered as a game played by two players. The first player controls the primal variables including y_t^{i*}, A_i and b_i and the goal is to minimize the Lagrangian. The second player controls the dual variables, that is λ_t, and the goal is to maximize the Lagrangian. With the Lagrangian, we need to derive the strategy of the first player. If it is classical Lagrangian method, we would optimize the slack variable y_t^{i*} and the DNN parameters A_i and b_i jointly. However, the problem would become intractable in our case. Instead, ADMM first optimizes y_t^{i*} with gradient descent,

$$\frac{\partial L}{\partial y_t^{i*}} = \frac{\partial E}{\partial y_t^{i*}} + \lambda_t + \alpha y_t^{i*} \tag{6}$$

Then, we fix y_t^{i*} and optimize A_i and b_i. It is important to note that once y_t^{i*} is fixed, A_i, b_i no longer depend on the objective function E. We only need to solve the LLS problem so that y_t^i could match y_t^{i*} as close as possible. The LLS problem is defined as,

$$\underset{A_i,b_i}{\text{argmin}} \quad (1+\tfrac{\alpha}{2})||y_t^{i*} - (A_i x_t + b_i)||_2^2 + \lambda_t'(y_t^{i*} - (A_i x_t + b_i)) \tag{7}$$

Finally, we derive the strategy of the second player assuming the primal variables are fixed.

$$\frac{\partial L}{\partial \lambda_t} = y_t^{i*} - y_t^i \tag{8}$$

In summary, each ADMM iteration has three steps: First, optimize the slack variables. Second, solve the LLS problem to derive the DNN parameters. Third, we update the Lagrange multipliers. This process iterates until it converges.

As an optimization algorithm, the advantage of ADMM is easy parallelization (solving the LLS problem in parallel). However, the disadvantage is that ADMM exhibits slow convergence in many problems [10]. Although it is a disadvantage for model training, we argue that it is an advantage for adaptation, since we often deal with a small amount of unreliable data. Hence, ADMM's slow convergence actually prevents the overfitting issue. Our previous work in [9] can be considered as a special case of ADMM where we do not update the Lagrange multipliers. In practice, we see little difference in performance, especially we only run 1 or 2 iterations for adaptation. However, by pointing out the relationship between our adaptation algorithm with ADMM, it helps us to understand the algorithm better from a theoretical standpoint, including its behavior and limits.

3. EXPERIMENTAL RESULTS

3.1. Data description

In this work, we use the same training data as described in [12] for both acoustic and language model training. The acoustic model training set consists of the Switchboard I and II, CallHome, and Cellular corpora. There are 2300 hours of audio data in the acoustic training. The text corpora used for language modeling include 27M words of the transcripts from the acoustic data, 260.3M words of Broadcast News transcripts, 115.9M words of CNN transcripts, and 525M words of Web data from the University of Washington. Based on the text corpora, we built a 4-gram language model with 75K vocabulary size. The language model has around 132M 4-grams.

We evaluated our baseline systems and our adaptation algorithms on Hub5 2000 evaluation set. This evaluation set consists of Switchboard test set and CallHome test set. The entire evaluation set has around 3.6 hours of audio data and 80 speakers.

3.2. Baseline systems

Our baseline DNN system is based on the recipe for the hybrid DNN-HMM system described in [13]. The input features to the bottle-neck (BN) MLP for feature extraction are 32 log Mel filter band energies with 3 Kaldi pitch features which consist of normalized F0 across sliding window, probability of voicing and delta. Mean subtraction is applied on the input features for each conversation side. Hamming window followed by Discrete Cosine Transform (DCT) are applied on the 11-frame concatenation of the input features to reduce the dimension to 210. There are four hidden layers with 1500 neurons except the BN layer which is the third hidden layer. There are 40 neurons with linear activation in the BN layer. The final DNN is trained on the BN features after CMLLR transform with a splicing of $[-10, -5, 0, 5, 10]$. There are 6 hidden layers in the DNN, and each hidden layer consists of 2048 neurons with the sigmoid activation. The hidden layers are initialized using stacked Restricted Boltzmann Machines (RBMs). To reduce the training time, only 10% of the training data is used in the RBM training. Finally, the DNN is retrained using cross-entropy followed by sequence-discriminative training as described in [14].

In addition to the baseline DNN system, a Time Delay Neural Network (TDNN) is also developed as described in [15]. There are 40 Mel-frequency cepstral coefficients and 100-dimensional I-vectors in the features for each time step input to the TDNN. The context specification as proposed in [15] is used while three hidden layers are appended after the contextual layers. There are 7 hidden layers in total in the TDNN. Every hidden layer is followed by a p-norm nonlinearity layer to reduce the dimension from 3000 to 1500, and a normalization layer for stability [16]. The data augmentation technique in [17] is also applied in the training. Two augmentation copies of the training data corresponding to speed perturbations of 0.9 and 1.1 are pooled with the original copy in the training for the TDNN.

We also investigate recurrent neural networks based on long short-term memory (LSTM) [18] and its bidirectional variant (BLSTM) [19]. Our LSTM and BLSTM are built on the same CMLLR transformed BN features used by the DNN model. Our LSTM model consists of two fully connected hidden layers with 2048 sigmoid units, followed by two LSTM layers. Each LSTM layer consists of 2048 memory cells with a recurrent projection layer that projects the output to 1024 dimension. Our BLSTM model is

similar to the LSTM model that has two fully connected layers with 2048 sigmoid units followed by two BLSTM layers. Each direction in a BLSTM layer consists of 1024 memory cells and a recurrent projection layer that projects the output to 300 dimension. The fully connected layers in our LSTM/BLSTM models are initialized with RBM, which we simply reuse the initialization from the DNN model. The LSTM/BLSTM layers are randomly initialized but the bias of the forget gate is initialized as one according to [20].

expt	UDEC	ADEC	SWB	CH
1	GMM	-	19.4%	29.2%
2	GMM	DNN	9.5%	17.0%
3	TDNN	-	9.1%	16.3%
4	TDNN2	-	8.8%	15.5%
5	GMM	LSTM	8.7%	15.3%
6	GMM	BLSTM	8.0%	13.9%

Table 1. WER using the baseline systems for Switchboard (SWB) and CallHome (CH)

The models used in our experiments are built using Sage [21], which is a speech processing platform that integrates multiple sources including Kaldi [22] and CNTK [23]. Table 1 shows the performance of the DNN, TDNN, LSTM and BLSTM models. In the table, UDEC is the unadapted decoding pass, while ADEC is the adapted decoding pass. Since DNN, LSTM and BLSTM are using CMLLR transformed features, they need to decode with a GMM to estimate CMLLR transforms. In contrast, the TDNN uses MFCC features and I-vectors, so it only needs one pass. TDNN2 is a TDNN retrained with the alignment computed by the BLSTM. We found that it could improve the performance of TDNN. Our baseline systems are competitive compared to various work using Hub5 2000 as an evaluation set [24, 25, 26].

3.3. Adaptation results

We evaluate our LLS adaptation with CMLLR transformed features. CMLLR is a popular feature based adaptation technique for deep models and LLS is only meaningful if it can provide additional gain. For our experiments, the setting of our LLS adaptation follows our work in [9], except that we perform adaptation on all fully connected layers instead of only the output layer.

Table 2 and 3 contains the results of adaptation using CMLLR only and CMLLR+LLS adaptation respectively. The results in table 2 shows that CMLLR is insensitive to the choice of the first pass model. Although both TDNN2 and LSTM reduce the word error rate(WER) by half compared to the GMM, it does not affect the final result at all. However, with LLS adaptation, we can observe WER reduction if we use certain first pass models. In which, using TDNN2 as the first pass model and BLSTM as the second pass gives 6.3% relative improvement in WER and reaches 7.5% WER on Switchboard subset (comparing to experiment 6 in table 1). Also, while TDNN2 and LSTM have similar performance, combining them could give 9.2% relative improvement as shown in experiment 4 in table 3 (comparing to experiment 5 in table 1).

These results support that LLS adaptation is an effective adaptation algorithm and it works even on a very strong baseline. However, results in table 3 also show that the improvement depends on the choice of the first pass model. For instance, using DNN or LSTM in the first pass would not help BLSTM as much as TDNN2. In the next section, we study the behavior of this LLS adaptation algorithm and investigate the factors that would contribute to the success.

expt	ADEC1	ADEC2	SWB (Improv.)	CH (Improv.)
1	GMM*	BLSTM	8.0% (-)	13.9% (-)
2	TDNN2*	BLSTM	8.0% (0.0%)	13.8% (0.7%)
3	LSTM	BLSTM	8.0% (0.0%)	13.8% (0.7%)
4	DNN	BLSTM	8.0% (0.0%)	13.7% (1.4%)

Table 2. WER and relative improvement on Switchboard(SWB) and CallHome(CH) using CMLLR adaptation only (*: For GMM and TDNN2, it is UDEC instead of ADEC1)

expt	ADEC1	ADEC2	SWB (Improv.)	CH (Improv.)
1	TDNN2*	BLSTM	7.5% (6.3%)	13.3% (4.3%)
2	LSTM	BLSTM	7.7% (3.8%)	14.2% (-2.2%)
3	DNN	BLSTM	7.9% (1.3%)	14.6% (-5.0%)
4	TDNN2*	LSTM	7.9% (9.2%)	14.1% (7.8%)

Table 3. WER and relative improvement on Switchboard(SWB) and CallHome(CH) using CMLLR+LLS adaptation (*: For TDNN2, it is UDEC instead of ADEC1)

4. ANALYSIS

For discussion purpose, we call the model for generating the first hypotheses as AM1, and the model chosen to perform LLS adaptation as AM2. In our analysis, we examine how different conditions may affect WER reduction provided by LLS. For example, we may study given the WER difference between AM1 and AM2 is within certain range, what is the WER reduction if we use AM1 to generate the hypotheses, followed by performing LLS on AM2 and decode with the adapted AM2 (compared to using AM2 without LLS). By doing so, we can find out under which conditions, we should perform LLS.

To gather the statistics, we perform LLS on different combination of models and record the WER reduction under different conditions. Based on the results, we notice that in general, the gain from adaptation increases as the accuracy of AM1 increases. This may imply that the performance of AM1 is important. Figure 1 shows the correlation between the WER difference between AM1 and AM2 versus WER reduction. Each bar in the figure represents the average WER reduction given the variable of interest is in the range of a bin. The range of each bin is decided dynamically so each bin has roughly the same number of speakers. On each bar, there is an error bar that represents the standard deviation (+/- 1 standard deviation).

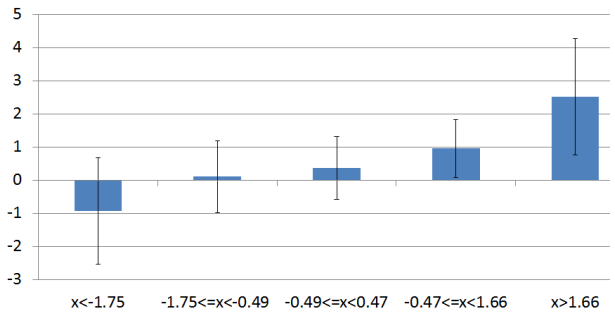

Fig. 1. WER difference vs WER reduction (x=[AM2's WER - AM1's WER]; larger x means AM1 is more accurate than AM2)

The correlation in figure 1 shows the importance of AM1. In practice, we do not know the WER of an unseen test set, so this figure

cannot tell us when we should apply LLS adaptation. However, we know that confidence measure often correlates with WER fairly well. Figure 2 shows the correlation of the difference in AM1 and AM2 confidence scores versus WER reduction. The confidence scores are computed using the geometric mean of the posterior scores of the first best hypotheses in the lattices. This figure shows a similar trend that if AM1 is more confident, it is more likely that LLS adaptation would improve performance. The benefit of using confidence measure is that we could use that as a feature to decide whether we perform LLS adaptation or not.

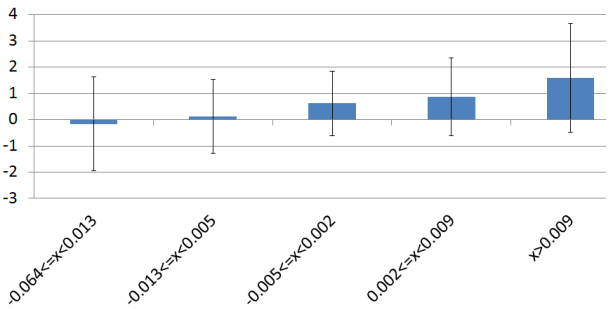

Fig. 2. Difference in confidence scores vs WER reduction(x=[AM1's confidence - AM2's confidence]; larger x means AM1 is more confident than AM2)

Although figure 2 gives us some indication of when to apply LLS, it could not explain two phenomena. First, there is a high variance in WER reduction. This is true even for the cases where AM1 is clearly better than AM2 (the rightmost bar in figure 1). Second, although LSTM and TDNN2 perform at similar performance, the gain from LLS varies a lot when their hypotheses are used to adapt the BLSTM (experiment 1 and 2 in table 3).

Figure 3 shows the relationship between the amount of adaptation and WER reduction. This figure shows that when there is less than 120 seconds of adaptation data, LLS adaptation does not improve performance and it also introduces large variance in performance. Although the performance does not further improve as more adaptation data becomes available, the variance reduces. This result shows that LLS adaptation may need more data than the conventional adaptation algorithms for GMM like MLLR. It is understandable as there are much more parameters in a deep model compared to a GMM. As a result, LLS adaptation may require more adaptation data.

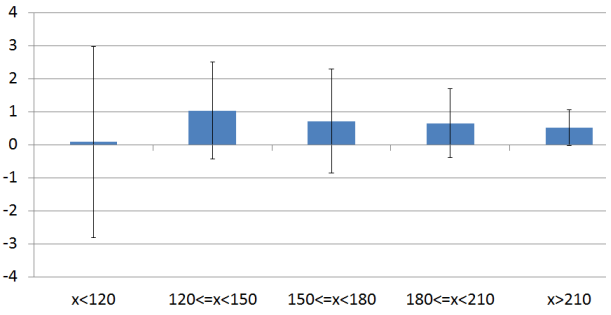

Fig. 3. Amount of adaptation data vs WER reduction (x= amount of data in seconds; larger x means there is more adaptation data)

To understand the performance difference in experiment 1 and 2 in table 3, we adopt the symmetrical KL divergence.

$$D = \frac{1}{T}(D_{KL}(P||Q) + D_{KL}(Q||P)) \quad (9)$$

where P and Q are two probability distributions; T is the number of frames. In our case, P and Q are the outputs of AM1 and AM2, and we compute the symmetrical KL divergence between two acoustic models and normalize it with the number of frames. Our hypothesis is LLS would work better if AM1 and AM2 are more different, so this divergence measure can quantify the differences between two models. Both AM1 and AM2 are not adapted with LLS when we measure the divergence, so in principle, this analysis could tell us in advance whether we should perform LLS adaptation.

Figure 4 is a plot between symmetrical KL divergence versus WER reduction given that there are more than 120 seconds of adaptation data, and AM1's confidence is higher than AM2's. This figure shows that under these conditions, WER reduction increases as the divergence increases. This explains why using TDNN2 for adaptation is better than LSTM even though their performance is similar. It is because from the perspective of the BLSTM, TDNN2 provides more diversity than the LSTM: the symmetric KL divergence of TDNN2 and BLSTM ranges from 2.8 to 4.2 (corresponding to the three bars on the right in figure 4) while the divergence between the LSTM and BLSTM only ranges from 0.4 to 0.8 (corresponding to the two bars on the left in figure 4).

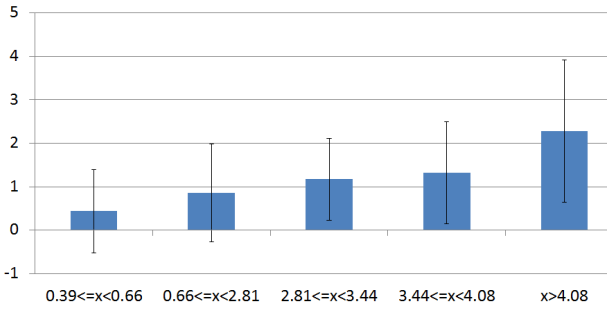

Fig. 4. Symmetrical KL divergence vs WER reduction given more than 120s of adaptation data and AM1's confidence is higher than AM2 (larger x means AM1 and AM2 are more different)

In summary, we learn that for LLS adaptation to work, there are three conditions. First, there should be enough adaptation data (over 120 seconds). Second, the model used for adaptation has to perform well and third, good diversity of the models. This finding matches our conventional wisdom and we believe this analysis provides a systematic method to understand an adaptation algorithm.

5. CONCLUSIONS

In this paper, we continue to investigate LLS adaptation. We explain that our adaptation algorithm is a special case of ADMM, and as an optimization algorithm, ADMM has some properties that are suitable for adaptation. We evaluate our algorithm on Hub5 2000 evaluation against a strong baseline and we show that LLS adaptation can improve system performance and achieve 7.5% WER on the Switchboard test set. We also analyze the algorithm and show that for LLS adaptation to succeed, there are three conditions: enough adaptation data, competitive first pass performance and diversity of the models. This analysis not only provides a clearer picture about LLS adaptation, but we believe the same method can be applied to studying other adaptation algorithms.

6. REFERENCES

[1] C.J. Leggetter and P.C. Woodland, "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models," *Computer Speech and Language*, vol. 9, pp. 171–185, 1995.

[2] M. J. F. Gales, "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

[3] J.L. Gauvain and C.H. Lee, "Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.

[4] 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 *IEEE Spoken Language Technology Workshop*, 2012, pp. 366–369.

[5] H. Liao, "Speaker Adaptation of Context Dependent Deep Neural Networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 7947 – 7951.

[6] 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 *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[7] Pawel Swietojanski and Steve Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proceedings of the IEEE Spoken Language Technology Workshop*, 2014.

[8] Pawel Swietojanski and Steve Renals, "SAT-LHUC: Speaker Adaptive Training for Learning Hidden Unit Contributions," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2016.

[9] R. Hsiao, S. Tsakalidis T. Ng, L. Nguyen, and R. Schwartz, "Unsupervised Adaptation for Deep Neural Network using Linear Least Square Method," in *Proceedings of the INTERSPEECH*, 2015.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[11] M. R. Hestenes, "Multiplier and Gradient Methods," *Journal of Optimization Theory and Applications*, vol. 4, pp. 303–320, 1969.

[12] R. Prasad, S. Matsoukas, CL Kao, J.Z. Ma, DX Xu, T. Colthurst, O. Kimball, R. Schwartz, J.L. Gauvain, L. Lamel, and et al., "The 2004 bbn/limsi 20xrt english conversational telephone speech recognition system," in *Proceedings of the INTERSPEECH*, 2005.

[13] M. Karafiát, K. Veselý, I. Szoke, L. Burget, F. Grézl, M. Hannemann, and J. Cernocký, "BUT ASR system for BABEL Surprise evaluation 2014," in *Proceedings of the IEEE Spoken Language Technology Workshop*, 2014.

[14] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative Training of Deep Neural Networks," in *Proceedings of the INTERSPEECH*, 2013.

[15] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proceedings of the INTERSPEECH*, 2015.

[16] Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2014.

[17] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proceedings of the INTERSPEECH*, 2015.

[18] Hasim Sak, Andrew Senior, and Franoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proceedings of the INTERSPEECH*, 2014.

[19] Alex Graves, Navdeep Jaitly, and Abdel rahman Mohamed, "Hybrid Speech Recognition with Deep Bidirectional LSTM," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[20] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," in *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, 1999, vol. 2, pp. 850–855 vol.2.

[21] R. Hsiao, R. Meermeier, T. Ng, Z. Huang, M. Jordan, E. Kan, T. Alumäe, J. Silovsky, W Hartmann, F. Keith, O. Lang, M. Siu, and O. Kimball, "Sage: The new BBN speech processing platform," in *Interspeech*, 2016.

[22] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukás Burget, O. Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý, "The Kaldi Speech Recognition Toolkit," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

[23] Amit Agarwal, Eldar Akchurin, Chris Basoglu, Guoguo Chen, Scott Cyphers, Jasha Droppo, Adam Eversole, Brian Guenter, Mark Hillebrand, Ryan Hoens, Xuedong Huang, Zhiheng Huang, Vladimir Ivanov, Alexey Kamenev, Philipp Kranen, Oleksii Kuchaiev, Wolfgang Manousek, Avner May, Bhaskar Mitra, Olivier Nano, Gaizka Navarro, Alexey Orlov, Marko Padmilac, Hari Parthasarathi, Baolin Peng, Alexey Reznichenko, Frank Seide, Michael L. Seltzer, Malcolm Slaney, Andreas Stolcke, Yongqiang Wang, Huaming Wang, Kaisheng Yao, Dong Yu, Yu Zhang, , and Geoffrey Zweig, "An Introduction to Computational Networks and the Computational Network Toolkit," Tech. Rep. MSR-TR-2014-112, Microsoft, 2014.

[24] A. Mohamed, F. Seide, D. Yu, J. Droppo, A. Stolcke, G. Zweig, and G. Penn, "Deep bi-directional recurrent networks over spectral windows," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2015.

[25] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," in *Proceedings of the INTERSPEECH*, 2016.

[26] G. Saon, T. Sercu, S. Rennie, and H.-K. J. Kuo, "The IBM 2016 English Conversational Telephone Speech Recognition System," in *Proceedings of the INTERSPEECH*, 2016.