INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Learning Speaker Aware Offsets for Speaker Adaptation of Neural Networks

Leda Sarı[1], *Samuel Thomas*[2], *Mark Hasegawa-Johnson*[1]

[1]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
[2]IBM Research AI

{lsari2,jhasegaw}@illinois.edu, sthomas@us.ibm.com

Abstract

In this work, we present an unsupervised long short-term memory (LSTM) layer normalization technique that we call adaptation by speaker aware offsets (ASAO). These offsets are learned using an auxiliary network attached to the main senone classifier. The auxiliary network takes main network LSTM activations as input and tries to reconstruct speaker, (speaker,phone) and (speaker,senone)-level averages of the activations by minimizing the mean-squared error. Once the auxiliary network is jointly trained with the main network, during test time we do not need additional information for the test data as the network will generate the offset itself. Unlike many speaker adaptation studies which only adapt fully connected layers, our method is applicable to LSTM layers in addition to fully-connected layers. In our experiments, we investigate the effect of ASAO of LSTM layers at different depths. We also show its performance when the inputs are already speaker adapted by feature space maximum likelihood linear regression (fMLLR). In addition, we compare ASAO with a speaker adversarial training framework. ASAO achieves higher senone classification accuracy and lower word error rate (WER) than both the unadapted models and the adversarial model on the HUB4 dataset, with an absolute WER reduction of up to 2%.

Index Terms: speaker adaptation, speech recognition, neural networks

1. Introduction

Although deep neural networks (DNNs) are successfully used in automatic speech recognition, their performance is still affected by the variability inherent in speech. One of the main sources of variability is the mismatch between speakers. Techniques proposed to alleviate this problem include using speaker-informed input features to the DNNs [1, 2], adapting the model structure [3, 4] and using auxiliary adaptation models or features [5, 6, 7, 8, 9, 10, 11, 12]. From a different perspective, adaptation methods can also be classified as supervised and unsupervised based on whether they use additional text or labels for the test data in addition to audio.

In input feature adaptation systems, features are normalized using a transform such as feature-space maximum likelihood linear regression (fMLLR) [13, 1] or the features are augmented with speaker specific features such as i-vectors [14, 2]. Other methods modify the speaker independent DNN model by introducing speaker adaptive layers [15]. For example, [16] investigates the use of learning an affine transform after LSTM activations at different layers of the network. Alternatively, the network structure is kept the same but the weights are adapted based on speakers [3]. Recently, auxiliary feature or auxiliary network based adaptation methods have become more popular as these methods usually require little or no adaptation data [10]. Such approaches extract speaker invariant intermediate features by adversarial training [8, 9]. In these systems, the auxiliary network performs speaker classification whereas the main network performs phone/senone classification. Auxiliary feature based systems are usually based on sequence summary vectors [17] and they are often applied only to the fully connected layers. However, recently some methods are extended for the adaptation of the LSTM layers. For example, in [10], the sequence summary idea is applied in an encoder-decoder based end-to-end framework.

This work is an extension of our previous work [12] on auxiliary network based speaker adaptation. The proposed system is based on an auxiliary network and it is an unsupervised speaker adaptation method for hidden layers of the neural network based acoustic models. The auxiliary network takes the hidden layer activation from the unadapted main senone classifier and tries to reconstruct speaker level mean of the activations at the output. Similar to [18, 19] where it has been shown that mean normalization can improve classification performance, in this paper, we perform normalization at the speaker level. Unlike training, we do not have access to the speaker labels and hidden layer activations to compute the means during testing. Therefore, we propose to use the auxiliary network to predict these means which do not require additional information from the test data other than the acoustic input. Instead of predicting the means directly, we aim at predicting the shift of the average activation from the global average. This allows us to extract the speaker specific component within the activation explicitly using the auxiliary network.

In the joint training of the main and auxiliary networks, bottleneck features from the auxiliary network are projected back into the hidden activation space and these vectors are used as offsets for the main network activations. Since the auxiliary network targets are speaker specific, we hypothesize that the auxiliary network will explicitly learn the speaker specific part of the activations. We therefore refer to our method as “adaptation with speaker aware offsets” (ASAO).

Compared to [12], here we demonstrate the flexibility of ASAO approach by showing that it is applicable to LSTM layers in addition to fully connected layers. The main difference is that we have whole utterances as input to the main network and LSTM layers produce a sequence of activations to be adapted. These allow us to make use of better context modeling capabilities of LSTM in the main network. We also perform our experiments on a different dataset and provide comparisons with an adversarial training approach. We show that we can get up to 2% absolute reduction in word error rate (WER).

Similar to our approach, Miao et al. [6] also use an auxiliary network but they apply the offset only to the input features rather than the hidden layers of the main network. And they use i-vectors as input to the auxiliary network rather than hidden layer activations. In [9], an adversarial multitask objective is used to extract speaker-invariant deep features. Here, we also

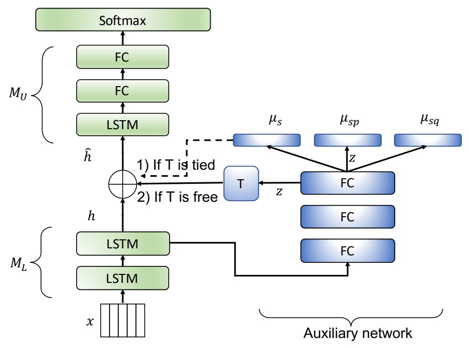


Figure 1: *Flowchart of the ASAO system with its auxiliary network*

use a multitask objective but our aim is to learn speaker dependent information explicitly through our adaptation network which is then used as an offset to get speaker invariant features. Veselý et al. [17] use summary vectors computed over the utterance as additional input features to their main network. In our study, we use speaker or phonetic level averages to summarize speaker and phonetic content but we use them as training targets for our auxiliary network instead of using them as input.

In Section 2 of the paper, we describe the method to extract speaker-aware offsets. We present the experimental setup and the results in Section 3. In Section 3.3, we compare our method to a previous work and we conclude the paper in Section 4.

2. Adaptation with Speaker-Aware Offset

The proposed ASAO system consists of two major components. The main network performs senone classification and the auxiliary network tries to reconstruct speaker and phonetic level averages of the hidden layer activations of the main network which is to be adapted. As shown in Fig. 1, there is also another component T which acts as a bridge between the auxiliary and the main networks by transforming the auxiliary network output and generating the offset for the main network. The main and auxiliary networks are described in detail below.

2.1. Main Network

For a duration N, let the input speech features to the network be denoted by $X = \{x_1, x_2, \dots, x_N\}$ and their corresponding senone labels by $Y = \{y_1, y_2, \dots, y_N\}$. A main senone classifier is trained first with inputs X and outputs Y. We then choose an adaptation layer l and divide the network into lower (M_L) and upper (M_U) parts. The main network outputs can hence be written as $M_U(M_L(x_n))$. The hidden layer activations h^l at layer l are then used as input to the auxiliary network which tries to learn the speaker dependent component of h^l. In this study, the main network consists of several LSTM layers followed by fully-connected layers. However, this does not restrict the choice of l because given an utterance of length N, we can compute the LSTM output for each time step n which is a vector that can be adapted as in adapting the fully connected layers.

2.2. Auxiliary Network

The aim of the auxiliary network is to extract speaker dependent information of the hidden layer activations h^l. Let the hidden layer activation at layer l for input at time n be denoted by h^l_n. Also let s_n, p_n and q_n denote the speaker, phone and senone label of input x_n, respectively. Average activations are then computed as in Eqs. 1-4. In these equations, $\mathbf{1}[.]$ is the indicator function that evaluates to 1 when its argument holds. The averages are computed over time instances which have the particular label or label pair. Similar to μ_s, we also define μ_p and μ_q.

$$\mu_g = \frac{1}{N}\sum_n h^l_n \tag{1}$$

$$\mu_s = \frac{1}{\sum_n \mathbf{1}[s_n = s]}\sum_n \mathbf{1}[s_n = s]h^l_n \tag{2}$$

$$\mu_{sp} = \frac{1}{\sum_n \mathbf{1}[s_n = s, p_n = p]}\sum_n \mathbf{1}[s_n = s, p_n = p]h^l_n \tag{3}$$

$$\mu_{sq} = \frac{1}{\sum_n \mathbf{1}[s_n = s, q_n = q]}\sum_n \mathbf{1}[s_n = s, q_n = q]h^l_n \tag{4}$$

In order to extract speaker specific component of h, we use the deviation of the speaker dependent mean activations from the global averages. Using the above definitions, the three linear output layers of the auxiliary network can be written as, $(\mu_s - \mu_g)$ that captures the deviation in the speaker mean from the global mean, $(\mu_{sp} - \mu_p)$ that captures speaker and phone level variation, and $(\mu_{sq} - \mu_q)$ that captures speaker and senone level variation. These three output layers share the auxiliary network parameters at the lower layers and differ only in their final layers. By parameter sharing and joint training of these outputs, we extract speaker dependent information from the network.

The hidden speaker dependent information for the n-th frame, z_n is computed from the last common hidden layer in the auxiliary network. Once z_n is computed, an affine transformation T is applied to the speaker dependent features z_n. We call these transformed features as speaker aware offsets and subtract them from the hidden layer activations of the main network as in Eq. (5). Thus, we aim at obtaining a speaker independent component of h^l such that better senone classification accuracy can be achieved in this speaker invariant space.

$$\widehat{h}^l_n = h^l_n - \mathrm{T}(z_n), \qquad n \in \{1, 2, \dots, N\} \tag{5}$$

Once the main network is augmented with this auxiliary network, the outputs of the main network can be written as $M_U(\widehat{h}^l_n)$.

2.3. Training Procedure

Initially, the main senone classifier is trained using the cross entropy objective. After choosing a layer l, we compute the average activations for different units as shown in Eqs. 1-4. These serve as training targets for the auxiliary network. Finally, the auxiliary network and the transform T are attached to the main network and joint training is performed. The overall training objective $\mathcal{L}$ is a linear combination of the cross entropy loss from the main network and the total mean squared error (MSE) from the three output layers of the auxiliary network.

$$\mathcal{L} = \mathcal{L}_{\text{xent}}(\widehat{y}, y) + \mathcal{L}_{\text{MSE}}(s) + \mathcal{L}_{\text{MSE}}(sp) + \mathcal{L}_{\text{MSE}}(sq) \tag{6}$$

In the above equation, $\widehat{y}$ and y denote the estimated and true senone sequences. The MSE terms can be described using the three auxiliary network outputs (m_s, m_{sp}, m_{sq}) at time n. For example, for (speaker, phone)-level output, we write

$$\mathcal{L}_{\text{MSE}}(sp) = \frac{1}{N}\sum_n ||m_{sp,n} - (\mu_{sp,n} - \mu_{p,n})||^2. \tag{7}$$

Table 1: *Training and heldout data settings*

# Train Spk	Utt/Spk	Train Dur.	# Heldout Spk	Heldout Dur.
150	40	21.5hr	1241	6.7 hr
600	10	21.9hr	1336	6.7 hr

3. Experiments

We performed our experiments on the Hub4 Broadcast News dataset [20, 21]. As the number of speakers in this dataset is large, it is likely that using all the speakers in training will lead to a speaker independent model that conceals the effect of adaptation. Therefore, we performed training under two conditions shown in Table 1. In the experiments, the total number of utterances was kept the same while the number of speakers and the number of utterances per speaker were adjusted. In all settings, training and heldout sets were disjoint in terms of speakers. We did not perform larger experiments with higher number of speakers for two reasons: 1) Having about 2000 speakers would inherently lead to a speaker independent model that would not necessarily require adaptation. 2) In our framework, since we computed all (speaker,senone)-level means to get the training targets for the auxiliary network, it would become computationally expensive to compute all the averages.

3.1. Architectures

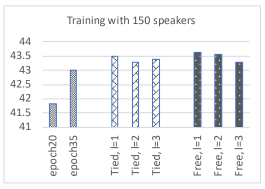
The input features in our experiments are 40-dimensional logmel features along with their deltas and delta-deltas, resulting in 120-dimensional input. The main network consists of three LSTM layers followed by two fully-connected layers with 256 and 512 nodes. LSTM layers are unidirectional and contain 128 cells. The number of output units is 2000 corresponding to each senone. The auxiliary network has three fully connected layers with 512, 256 and 128 units. Therefore, z has a dimension of 128. When we adapt the LSTM layers, each of the three output layers have a dimension of 128 as the LSTMs have 128 cells. All fully-connected layers except the one generating z have rectified linear unit nonlinearity.

The networks are trained using PyTorch [22] with Adam optimization using learning rate of 0.001. In the first stage, the main senone classifier is randomly initialized by Xavier method [23] and trained for 20 epochs. Then, the hidden features h and their speaker, (speaker, phone) and (speaker,senone) level averages are computed. In joint network training, the main network is initialized from the senone classifier and the auxiliary network targets are taken to be the average activations. Joint training is performed for 15 epochs. Feature extraction, HMM training and decoding are performed using the Attila toolkit [24].

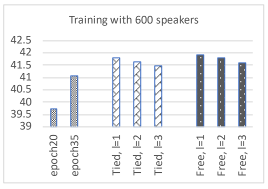
In joint training, the auxiliary network is utilized in two different ways (Fig. 1): The T matrix shares its weights with the speaker-level output layer of the auxiliary network, and therefore auxiliary network output is directly used as the offset or T is kept to be a general affine transformation of the last hidden layer of the auxiliary network z. In both cases, the offset will carry speaker dependent information.

3.2. Results

Fig. 2 shows the frame-level senone classification accuracy on the heldout data for the networks trained with logmel features. We compare the main network trained for 20 epochs (epoch20), with the two adaptation methods (T is Tied or Free) for the adaptation of different LSTM layers individually (l=1,2,3). Since joint training is performed for 15 epochs, we also compare the performance of the 35th epoch (epoch35) of the main network training without any adaptation. We present the results under two training conditions (150 vs 600 speakers).



(a) *Training with 150 speakers*



(b) *Training with 600 speakers*

Figure 2: *Senone classification accuracy of the main network on the heldout data before and after adaptation for logmel inputs*

In all adaptation experiments, we performed better than the initial unadapted model at epoch20 on the heldout set which resulted in 41.8% and 39.7% accuracy for 150 and 600-speaker conditions. For both 150 and 600 speaker conditions, the adapted models performed slightly better than the unadapted epoch35 model which achieved 43.0% and 41.1% on the heldout set in 150 and 600 speaker conditions, respectively. If we compare the performance of the adaptation of different LSTM layers, we found that the improvements over the unadapted model decrease as we go deeper in the network (l gets larger). This might be due to the fact that as we get deeper and therefore closer to the output level, the hidden features become less speaker dependent and more phone dependent. Therefore, the difference between h and $\hat{h}$ becomes smaller as we go deeper in the network and it reduces the effectiveness of adaptation. For the best performing layer (l=1), we achieved 43.6% and 41.9% in the 150 and 600-speaker conditions, respectively.

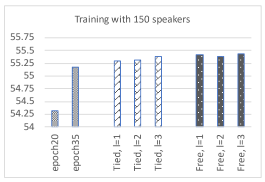
When we compare the performances of two architectures (T is Tied or Free), i.e. the second and third group of bars in Figs. 2a and 2b, we see that with the Free setup, a slight improvement over using Tied is observed. This result supports the observation made in [12] and suggests that adaptation with $\mathrm{T}(z)$ is better.

If we compare the results of two training conditions (Fig. 2a vs. 2b), in the 600-speaker case, our unadapted model has lower accuracy than 150-speaker condition possibly because of higher numbers of speakers in the heldout set. In the 600-speaker case, we see that the relative improvement in senone accuracy with adaptation is larger than that of the 150-speaker condition.

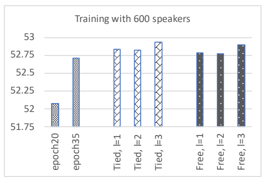
3.3. Comparison

We repeated similar experiments described above for the case where the inputs are already speaker adapted by fMLLR [13]. Now, the inputs are 40-d fMLLR features concatenated within ± 5 context. The main and auxiliary network structures are kept the same except for the input layer which has 440-d inputs rather than 120. The goal is to show if the proposed method is complementary to fMLLR based input adaptation by comparing the performances of the adapted and unadapted fMLLR models.

Most of the conclusions from logmel features also hold for this case. As shown in Figs. 3a-3b, adapted models are always better than epoch20 model. As compared to epoch35, we still observe increase in accuracy of the adapted models although the gains are much smaller. For the fMLLR case, we do not observe a general pattern among adapting different LSTM layers, in these experiments, adapting $l = 2$ or 3 can perform better than adapting $l = 1$. This might be due to the fact that input



(a) *Training with 150 speakers*



(b) *Training with 600 speakers*

Figure 3: *Senone classification accuracy of the main network on the heldout data before and after adaptation for fMLLR input*

Table 2: *Senone classification accuracy comparison of unadapted model with adversarial adaptation of [9] and ASAO*

	150	600
Epoch20	41.8	39.7
Epoch35	43.0	41.1
Adversarial	42.2	40.5
ASAO	**43.4**	**41.5**

features are now already speaker normalized so $l = 1$ is more speaker independent as compared to the logmel experiments. Since adaptation in these systems still leads to improvement, it can be concluded that the fMLLR features retain speaker information and the proposed ASAO method provides complementary speaker normalization.

As mentioned in Section 1, there are various studies that make use of an auxiliary network. Among those, the one that is closest to ours is the adversarial training method of [9] which tries to obtain speaker independent hidden layer activations using multitasking. In this system, the auxiliary network takes the hidden layer activation (similar to our h) as input but the outputs of the auxiliary network are speaker labels rather than speaker dependent means. Their goal is to maximize the senone classification accuracy of the main network while minimizing the speaker classification accuracy by gradient reversal. Compared to this work, they do not use an additional projection matrix T.

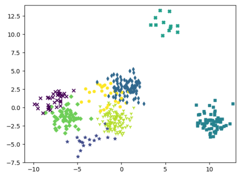
We use the pretrained senone classifier with logmel inputs and then attach the speaker classifying auxiliary network and perform joint training. Different from [9] which adapted only a fully connected network, we tried the same approach on LSTM layers. In the logmel based system, we adapted $l = 2$ for which our system has a disadvantage. Table 2 compares the senone classification performances of the unadapted model, adversarial adaptation and ASAO methods. It is shown that the proposed ASAO method leads to the highest classification accuracy.

According to Table 2, the adversarial training method improves the accuracy over the unadapted method at epoch20. Although the joint training is performed for 15 iterations, it cannot reach the performance of the epoch35 model of the unadapted main network. Therefore, we hypothesize that our ASAO method might be more effective for speaker normalization of the hidden layer activations of the main network.

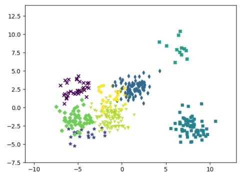
In Table 3, we present the WER on the heldout datasets for the unadapted model, adversarial training and best senone classifiers from the ASAO method. Since we see the largest gains when there is no explicit input adaptation, we include only the results for the logmel setup. For this setup, ASAO method achieves 9.2% and 6.7% relatively lower WER than the unadapted model in 150 and 600-speaker conditions, respectively, which corresponds to up to 2% absolute reduction in WER. It also performs better than adversarial training.

Table 3: *WER for 150 and 600 speaker training conditions under different models*

	150		600
Unadapted	21.8	Unadapted	26.9
Adversarial [9]	21.1	Adversarial	26.0
ASAO, Free, l=1	**19.8**	ASAO, Free, l=1	**25.1**



(a) h



(b) $\hat{h}$

Figure 4: *2D LDA projection of h and $\hat{h}$ for the logmel condition with ASAO, Free, l=1 model for a phone; colors represent different speakers*

In Fig. 4, we visualize the unnormalized (h) and normalized ($\hat{h}$) hidden activations of the network. We randomly select 10 heldout speakers and for each phone, we plot the 2D linear discriminant analysis (LDA) projection of the activations to separate the speaker classes. For the model with the largest improvement (150-speaker logmel condition, T Free, l=1 adaptation), we get Fig. 4 for a phone. In the figures, each color represents a different speaker. In the speaker normalized space, we expect and observe that the $\hat{h}$ activations related to a certain phone to be closer. For example, the speaker on top right or bottom right gets closer to the larger cluster on lower left.

4. Conclusions

In this work, we have presented a neural network based speaker adaptation scheme using an auxiliary network. This auxiliary network which is trained to reconstruct speaker, (speaker, phone), and (speaker, senone)-level averages, generates a speaker aware offset that is subtracted from the main network activations. The main advantage of the auxiliary network is that once it is trained, we do not need additional data for test speakers as the auxiliary network will automatically generate the speaker-aware offsets. We show that the proposed model can be used to adapt LSTM layers in addition to just fully-connected layers. Experimental results show that if the input features are speaker independent logmel features, adapting lower layers of the network is more helpful. Using a free projection (T matrix) is shown to be better than tying it to the speaker dependent output layer of the auxiliary network. We also show that ASAO can slightly improve the senone classification accuracy when the inputs to the main network are already speaker adapted fMLLR features showing that the two methods are complementary. We compare our multitask learning with the adversarial training method of [9] and show that ASAO is more effective in speaker normalization by achieving up to 2% absolute reduction in WER.

5. Acknowledgments

This work is supported in part by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network.

6. References

[1] S. P. Rath, D. Povey, K. Veselỳ, and J. Cernockỳ, “Improved feature processing for deep neural networks.” in *Interspeech*, 2013, pp. 109–113.

[2] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, “Speaker adaptation of neural network acoustic models using i-vectors.” in *ASRU*, 2013, pp. 55–59.

[3] P. Swietojanski and S. Renals, “Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models,” in *IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 171–176.

[4] 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 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6359–6363.

[5] 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 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 7942–7946.

[6] Y. Miao, H. Zhang, and F. Metze, “Speaker adaptive training of deep neural network acoustic models using i-vectors,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 11, pp. 1938–1949, 2015.

[7] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, “Fast adaptation of deep neural network based on discriminant codes for speech recognition,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1713–1725, 2014.

[8] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim *et al.*, “English conversational telephone speech recognition by humans and machines,” *arXiv preprint arXiv:1703.02136*, 2017.

[9] Z. Meng, J. Li, Z. Chen, Y. Zhao, V. Mazalov, Y. Gong *et al.*, “Speaker-invariant training via adversarial learning,” *arXiv preprint arXiv:1804.00732*, 2018.

[10] M. Delcroix, S. Watanabe, A. Ogawa, S. Karita, and T. Nakatani, “Auxiliary feature based adaptation of end-to-end asr systems,” *Interspeech*, pp. 2444–2448, 2018.

[11] X. Cui, V. Goel, and G. Saon, “Embedding-based speaker adaptive training of deep neural networks,” *Interspeech*, pp. 122–126, 2017.

[12] L. Sarı and M. Hasegawa-Johnson, “Speaker adaptation with an auxiliary network,” in *Machine Learning in Speech and Language Processing Workshop (MLSLP)*, 2018.

[13] M. J. Gales, “Maximum likelihood linear transformations for hmm-based speech recognition,” *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[14] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-end factor analysis for speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[15] F. Seide, G. Li, X. Chen, and D. Yu, “Feature engineering in context-dependent deep neural networks for conversational speech transcription,” in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2011, pp. 24–29.

[16] M. Kitza, R. Schlüter, and H. Ney, “Comparison of blstm-layer-specific affine transformations for speaker adaptation,” *Interspeech*, pp. 877–881, 2018.

[17] K. Veselỳ, S. Watanabe, K. Žmolíková, M. Karafiát, L. Burget, and J. H. Černockỳ, “Sequence summarizing neural network for speaker adaptation,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5315–5319.

[18] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[19] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

[20] “1996 English Broadcast News Speech (HUB4),” https://catalog.ldc.upenn.edu/LDC97S44.

[21] “1997 English Broadcast News Speech (HUB4),” https://catalog.ldc.upenn.edu/LDC98S71.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in *NIPS Autodiff Workshop*, 2017.

[23] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[24] H. Soltau, G. Saon, and B. Kingsbury, “The IBM Attila speech recognition toolkit,” in *IEEE Spoken Language Technology Workshop*. IEEE, 2010, pp. 97–102.