INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Speech Transformer with Speaker Aware Persistent Memory

Yingzhu Zhao[1,2*], *Chongjia Ni*[2], *Cheung-Chi Leung*[2], *Shafiq Joty*[1], *Eng Siong Chng*[1], *Bin Ma*[2]

[1]Nanyang Technological University, Singapore
[2]Machine Intelligence Technology, Alibaba Group

{srjoty, aseschng}@ntu.edu.sg
{yingzhu.zhao, ni.chongjia, cc.leung, b.ma}@alibaba-inc.com

Abstract

End-to-end models have been introduced into automatic speech recognition (ASR) successfully and achieved superior performance compared with conventional hybrid systems, especially with the newly proposed transformer model. However, speaker mismatch between training and test data remains a problem, and speaker adaptation for transformer model can be further improved. In this paper, we propose to conduct speaker aware training for ASR in transformer model. Specifically, we propose to embed speaker knowledge through a persistent memory model into speech transformer encoder at utterance level. The speaker information is represented by a number of static speaker i-vectors, which is concatenated to speech utterance at each encoder self-attention layer. Persistent memory is thus formed by carrying speaker information through the depth of encoder. The speaker knowledge is captured from self-attention between speech and persistent memory vector in encoder. Experiment results on LibriSpeech, Switchboard and AISHELL-1 ASR task show that our proposed model brings relative 4.7%-12.5% word error rate (WER) reductions, and achieves superior results compared with other models with the same objective. Furthermore, our model brings relative 2.1%-8.3% WER reductions compared with the first persistent memory model used in ASR.

Index Terms: speech transformer, persistent memory, speaker adaptation

1. Introduction

The past decade has witnessed extensive use of deep learning in various domains. For ASR, end-to-end models have brought a new state-of-the-art performance compared with conventional hybrid systems. From connectionist temporal classification (CTC) models [1, 2], attention-based encoder-decoder models [3], to recurrent neural network transducer (RNN-T) [2] and transformer models [4], end-to-end architecture has the advantage to be a single unified model with fast computation speed and fast development time.

Although ASR performance has greatly improved, *speaker mismatch* between training and test data will degrade ASR performance. There are several previous studies to address speaker mismatch problem, which can be categorized into feature adaptation and model adaptation. Feature adaptation works on acoustic features, either by normalizing acoustic features to be speaker-independent [5, 6, 7], or by bringing auxiliary speaker related knowledge (e.g. i-vector) into acoustic model [8, 9, 10, 11]. In [12], K. Vesely *et al.* replaces speaker i-vector with a summary vector of each utterance, and trains the summary vector together with the main model. [13] generates shifting and scaling parameters in layer normalization layer to adapt to acoustic variability. Model adaptation estimates the speaker-dependent parameters from speaker-independent model parameters with additional adaptation data. One of the methods is to retrain the whole model. L2 regularization [14], Kullback-Leibler divergence [15] and adversarial multitask learning [16] are used to avoid overfitting. To prevent from retraining a large model, only certain layers or subset of parameters are adapted [17, 18, 19]. In particular, [20, 21, 22] reparameterize each hidden unit (in fully-connected or convolutional neural network layers) with speaker-dependent amplitude function.

Recently, more and more speaker adaptation research has adopted end-to-end models. [11] learns a speaker representation for each speech time step, and then appends it along the feature dimension of encoder final layer output. However, given a single speech utterance from one speaker, it is unclear on the necessity to generate separate speaker representation for each speech time step. Besides, because [11] appends speaker representation to encoder output, it doubles the linear transformation parameter size at each decoder layer. Specific to these two points, we propose a speaker aware persistent memory model built on top of the speech transformer model [4] to address the speaker mismatch problem. Our method belongs to the feature adaptation category, where we concatenate speaker i-vectors to speech utterance, and apply this for each encoder layer, thus forming a persistent memory through the depth of encoder. Different from [11] which learns for each speech time step, our method learns utterance level speaker knowledge, which we believe is more reasonable. On top of that, our method does not generate a specific speaker knowledge vector to append to encoder output, but integrates speaker knowledge into encoder hidden state by attention computation. By this means, our method resolves the problem of introducing extra model parameters. Experiment results on LibriSpeech, Switchboard and AISHELL-1 ASR task show that our proposed model brings relative 4.7%-12.5% WER reductions.

2. Model architecture

We build on top of the speech transformer model [4] for speaker aware training. In this section, we will briefly review the speech transformer model first, and then introduce our proposed model.

2.1. Speech transformer

Speech transformer [4] builds on top of the transformer model [23] for ASR. Here we summarize a few key components of speech transformer model. For full details, please refer to [4]. There are N_e encoder layers and N_d decoder layers in the model. For a speech input sequence, it first applies two convolution layers with stride two to reduce hidden represen-

*Yingzhu Zhao is under the Joint PhD Program between Alibaba and Nanyang Technological University.

http://dx.doi.org/10.21437/Interspeech.2020-1281

tation length. A sinusoidal positional encoding is added to encode position information. Transformer encoder then computes hidden state representation of each position in parallel with a self-attention network. For the three inputs key, query and value, which are three distinct transformations of an input sequence, the multi-head attention network, which concatenates self-attention network h times as introduced in [23], is:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \quad (2)$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (3)$$

where h is the head number, $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $W^O \in \mathbb{R}^{hd_v \times d_{model}}$, $d_k = d_q = d_v = d_{model}/h$ in this paper.

Multi-head attention allows learning input representation from different subspaces concurrently. Layer normalization and residual connection are applied before and after multi-head attention network. Afterward, there is a position-wise feedforward network with rectified linear unit (ReLU) activation:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (4)$$

where $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$, and the biases $b_1 \in \mathbb{R}^{d_{ff}}$, $b_2 \in \mathbb{R}^{d_{model}}$.

Each layer in encoder consists of multi-head attention network and position-wise feedforward network. For speech transformer decoder, there is a third sublayer between multi-head attention and position-wise feedforward network. This is a multi-head cross attention computed over encoder and decoder output, where key and value vectors come from encoder, and query vector comes from decoder. To prevent self-attention network from attending to future positions in the decoder, a masking is applied.

2.2. Speaker aware persistent memory

Persistent memory was proposed by [24] to replace position-wise feedforward network in Eq. 4 by an *all-attention* network, and to capture general knowledge and non-contextual information about the task, but the similar concept was first proposed in the question answering task [25]. Different from them, our objective is to do speaker adaptation. We propose to perform speaker aware training by learning speaker specific knowledge in a similar manner as capturing general knowledge by the persistent memory model. We call it speaker aware persistent memory model. In particular, we randomly sample N speaker i-vectors $m_1, ..., m_N \in \mathbb{R}^{d_k}$ which form the speaker space. Persistent memory vectors M_k and M_v are learned transformation of speaker space:

$$M_k = Concat([U_k m_1, ..., U_k m_N]) \in \mathbb{R}^{N \times d_k} \quad (5)$$

$$M_v = Concat([U_v m_1, ..., U_v m_N]) \in \mathbb{R}^{N \times d_k} \quad (6)$$

where $U_k \in \mathbb{R}^{d_k \times d_k}$, $U_v \in \mathbb{R}^{d_k \times d_k}$.

Speaker specific knowledge of an utterance is learned through the attention with the persistent memory vectors. This is built on the assumption that the linear combinations of speaker space are enough to cover the speaker information space, i. e. a weighted sum of persistent memory vectors can represent any specific speaker knowledge. The persistent memory vectors M_k and M_v, together with $X = [x_1, ..., x_t]$ which

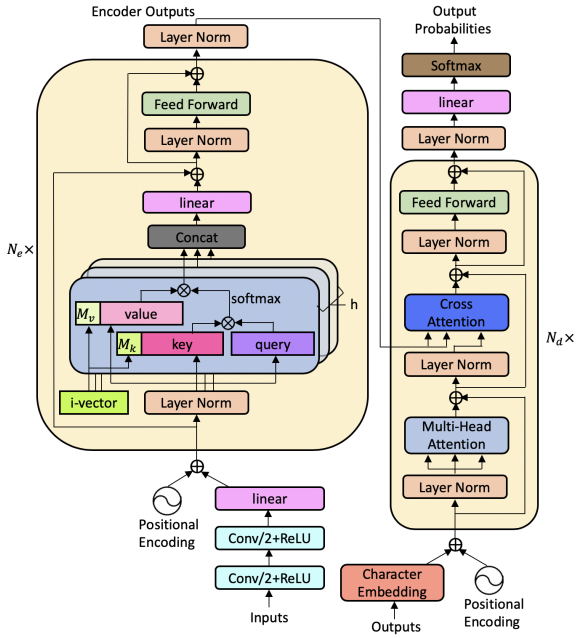

Figure 1: *Speaker aware persistent memory model. M_k and M_v from speaker i-vectors are concatenated to key and value vectors.*

are input vectors of self-attention network, form the new key and value vectors for self-attention computation as Eq. 9:

$$K_m = [k_1, ..., k_{t+N}] = Concat([W_k x_1, ..., W_k x_t], M_k) \quad (7)$$

$$V_m = [v_1, ..., v_{t+N}] = Concat([W_v x_1, ..., W_v x_t], M_v) \quad (8)$$

$$Attention(Q, K_m, V_m) = softmax(\frac{QK_m^T}{\sqrt{d_k}})V_m \quad (9)$$

In the original persistent memory model [24], it randomly initializes N pairs of vectors as M_k and M_v in Eq. 7 and Eq. 8, and these vectors are learnable. On the contrary, the N speaker i-vectors we use are fixed. Only the weight matrices associated with attention mechanism U_k and U_v are learnable. Furthermore, since our speaker aware persistent memory is not meant to play the same role as position-wise feedforward network, we leave the feedforward network same as the original transformer model. Given that M_k and M_v are shared across all layers, they form the persistent memory, and that is why we call it speaker aware persistent memory model. Figure 1 gives the overall framework of speaker aware persistent memory model.

There are three main advantages of our proposed model. First, we use randomly sampled N speaker i-vectors as the speaker space, and obtain any speaker knowledge from them. It solves the problem of having unknown speakers in the test data. It further relieves us from computing all speaker i-vectors of the training data. Second, attention with persistent memory vectors is computed together with the entire utterance. Therefore, it takes the entire utterance into consideration while capturing speaker information, which is different from [11] who obtains time step dependent speaker information. Last, [11] concatenates speaker embedding with encoder final layer output, and downscales the vector to original at decoder side. It doubles the parameter size of linear transformation in each decoder layer. In contrast, attention with persistent memory vectors is integrated into the self-attention network in our model, and there is no extra parameters of linear transformation in decoder.

Table 1: *Details of datasets used for experiments*

LibriSpeech	
Training set	100h (251 speakers)
Test_clean set	5.4h (20 males, 20 females)
Test_other set	5.1h (16 males, 17 females)
Switchboard	
Training set	300h (4804 speakers)
SWBD set	2.1h (40 speakers)
CallHome set	1.6h (40 speakers)
AISHELL-1	
Training set	150h (340 speakers)
Test set	5h (13 males, 7 females)

Table 2: *WER results of end-to-end speech recognition models on LibriSpeech 100h*

Model	Test_clean	Test_other
End-to-end (E2E) [29]	14.7	40.8
E2E with augmented data [30]	15.1	-
LAS [31]	12.9	35.5
Baseline	12.0	29.7

3. Experimental setup

3.1. Datasets

We conduct experiments to confirm the effectiveness of the proposed model on three publicly available datasets, including LibriSpeech [26], Switchboard [27] and AISHELL-1 [28]. LibriSpeech consists of 16kHz read English speech from audiobooks. Switchboard is a 300 hour corpus of conversational English telephone speech. AISHELL-1 is a 16kHz Chinese Mandarin speech corpus recorded by 400 speakers from different accent areas in China. The characteristics of the datasets are summarized in Table 1.

3.2. Training setup

We use PyTorch and Espnet [32] toolkit for our experiments. Input features are generated by 80-dimensional filterbanks with pitch on each frame, with a window size of 25ms shifted every 10ms. The acoustic features are mean and variance normalized. We exclude utterances longer than 3000 frames or 400 characters to keep memory manageable. For joint decoding of CTC and attention, the coefficient is 0.3 and 0.7 for CTC and attention respectively. The convolutional frontend before transformer encoder is two 2D convolutional neural network layers with filter size (3,2) and stride 2, each followed by a ReLU activation. The attention dimension d_{model} is 256, and the feed-forward network hidden state dimension d_{ff} is 2048. In the transformer structure, the number of attention heads h is 4, with $d_k = d_q = d_v = 64$ for each head, the number of encoder layers N_e is 12, the number of decoder layers N_d is 6, the initial value of learning rate is 5.0, the encoder and decoder dropout rate is 0.1. The input samples are shuffled randomly and trained with batch size 12. We use unigram sub-word algorithm with the vocabulary size capped to be 5000. For i-vector generation, we follow SRE08 recipe in Kaldi toolkit on the training data. I-vectors extracted are of dimension 100. They are then transformed to have the same dimension as speech vectors for concatenation. Hyperparameters, including the number of speaker i-vectors in the speaker space and the number of layers applied with speaker aware persistent memory are tuned using the LibriSpeech dataset. Our baseline model is competitive compared with other model results from Table 2.

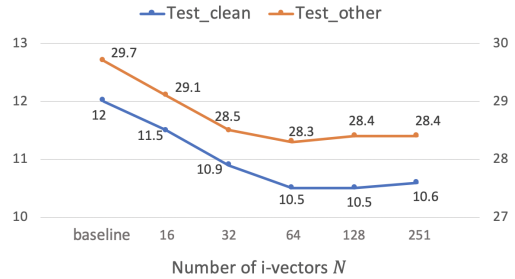

Figure 2: *Speaker aware persistent memory results (WER) applied on all encoder layers for different number of speaker i-vectors on LibriSpeech test dataset.*

Table 3: *Speaker aware persistent memory results (WER) with 64 speaker i-vectors applied on different layers on LibriSpeech test dataset*

Encoder layers applied	Test_clean	Test_other
Baseline	12.0	29.7
Lower half (1-6)	11.2	28.6
Higher half (7-12)	10.9	28.1
All (1-12)	10.5	28.3

4. Experimental results

4.1. Number of speaker i-vectors in the speaker space

We first investigate the effect of the number of speaker i-vectors N in the speaker space on the LibriSpeech dataset. Different numbers of speakers are randomly selected from the training data, ranging from 16 to 251 (all speakers). Speaker aware persistent memory is applied on all encoder layers. The results are shown in Figure 2. It can be seen that the performance improves with the number of i-vectors in the speaker space increasing from 16 to 64. This exemplifies the effectiveness of speaker aware persistent memory, that introducing speaker i-vector helps capture speaker knowledge. Furthermore, adding more speaker i-vectors gives more diverse speaker information. Afterward, increasing the number of i-vectors does not affect much. We believe this is because 64 i-vectors are good enough to capture enough speaker variation. Thus, increasing the number of i-vectors will neither add value further nor deteriorate the performance. We use 64 speaker i-vectors for subsequent experiments.

4.2. Number of layers applied with speaker aware persistent memory

Secondly, we look into which hidden layers should the speaker aware persistent memory be applied to. For these layers, there is a component of speaker knowledge in the hidden layer vectors. The results are summarized in Table 3. When we apply speaker aware persistent memory on all encoder layers (layer 1-12), it achieves the best result on the clean test set (Test_clean), and overall it brings relative 4.7%-12.5% WER reductions. Applying it on higher encoder layers (layer 7-12) performs better than applying it on lower encoder layers (layer 1-6). Compared with lower layers, encoder output from higher layers has more abstract and global information. In contrast, encoder output from lower layers comes from raw features and has more phonetic information. Thus, introducing i-vector will be of less use in the

Table 4: *State-of-the-art results of different algorithms*

Model	**Switchboard (WER)**		
	SWBD	**CallHome**	**All**
Baseline	8.8	18.0	13.4
Attn based [10]	-	-	15.5
BLHUC [22]	12.8	22.9	-
Persistent Memory [33]	8.8	17.5	13.2
Our work	**8.2**	**16.0**	**12.1**
Model	**AISHELL-1 (CER)**		
	Dev	**Test**	
Baseline	6.4	7.3	
SAST [11]	7.6	7.8	
Persistent Memory [33]	6.3	7.1	
Our work	**5.9**	**6.4**	
Model	**LibriSpeech 100h (WER)**		
	Test_clean	**Test_other**	
Baseline	12.0	29.7	
SAST [11]	11.4	28.4	
Persistent Memory [33]	11.3	28.9	
Our work	**10.5**	**28.3**	

lower encoder layers. We notice that the improvement on the noisy test set (Test_other) is moderate only. This is mainly because our 100h LibriSpeech training data is all clean data. The mismatch of training and test data deteriorates the performance. Similar deterioration on the LibriSpeech noisy test set is also observed in all other models to be presented in the next section.

4.3. Comparison with other state-of-the-art results

Thirdly, we show that our work outperforms other similar approaches on Switchboard, AISHELL-1 and LibriSpeech datasets in Table 4. We use 64 speaker i-vectors and apply to all layers with speaker aware persistent memory. For Switchboard dataset, where the training data contains both SWBD and CallHome data, our work brings 9.7% relative improvement over the baseline. Furthermore, here we also compare our model with the first persistent memory model used in ASR [33]. In [33], persistent memory vectors are randomly initialized and meant to capture general knowledge. Different from them, our model is to address the speaker mismatch issue. It can be seen that our model outperforms theirs by 8.3%. Similarly for AISHELL-1 dataset, our model brings 7.8%-12.3% relative improvement over the baseline. For LibriSpeech 100h dataset, our work has 4.7%-12.5% relative improvement. As mentioned earlier, due to the fact that LibriSpeech training data is clean data, all model's improvement on LibriSpeech noisy data is less significant.

4.4. Analysis of speaker aware persistent memory

Lastly, we would like to verify if our proposed speaker aware persistent memory really captures speaker knowledge using LibriSpeech dataset. WER reduction on all datasets could not prove or disprove from this perspective. Since we use the fixed speaker space, we plot the learnable attention scores between an utterance and persistent memory vector M_k, which is the learned transformation of 64 speaker i-vectors, after softmax computation from Eq. 9, averaged over all time steps of the utterance. The plot on top of Figure 3 shows attention scores of ten different utterances from the same speaker. Given the ten

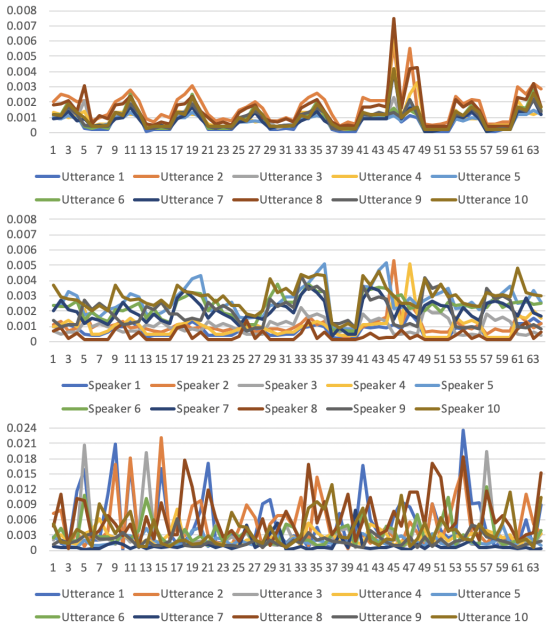

Figure 3: *Attention scores between ten utterances and persistent memory vector M_k. Top: All utterances from same speaker and M_k is learned transformation of 64 speaker i-vectors. Middle: All utterances from different speakers and M_k is learned transformation of 64 speaker i-vectors. Bottom: All utterances from same speaker and M_k contains 64 randomly initialized vectors [33]. Attention scores shown are after softmax computation and averaged over all time steps of each utterance.*

different utterances with the only common ground of coming from the same speaker, they show similar pattern of attention scores, which clearly shows that certain speaker knowledge is captured. On the other hand, in the middle of the figure, the attention scores of ten random utterances from ten different speakers are quite random. The dataset does not provide one common utterance from different speakers, so we have to select random utterances. We take average attention scores along the entire utterance, which will mitigate acoustic variation among different utterances to some extent. Here we also present the attention scores of ten utterances from the same speaker in a transformer model using randomly initialized M_k as in [33] at the bottom of the figure, which is meant to capture general knowledge. From the irregular plot, it does not seem to capture speaker-related knowledge. From here, we conclude that speaker aware persistent memory captures speaker knowledge.

5. Conclusions

In this paper, we have proposed a speaker aware persistent memory model to address speaker mismatch between training and test data in ASR. Our speaker aware persistent memory is composed of a number of static speaker i-vectors, which form the speaker space for speaker knowledge extraction. The two transformations of speaker i-vectors are concatenated to the key and value vectors of self-attention layer respectively, and are carried across multiple layers, thus form a persistent memory. Experiment results show that our model has 4.7%-12.5% relative improvement over baseline model for LibriSpeech dataset, 7.8%-12.3% relative improvement for AISHELL-1 dataset, and 9.7% for Switchboard dataset. In the future, we will try to address above issue without introducing external information such as i-vector.

6. References

[1] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *2015 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2015, pp. 167–174.

[2] A. Graves, "Sequence transduction with recurrent neural networks," in *International Conference of Machine Learning (ICML)*, 2012, pp. 235–242.

[3] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4845–4849.

[4] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5884–5888.

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

[6] N. Tomashenko and Y. Estève, "Evaluation of feature-space speaker adaptation for end-to-end acoustic models," in *LREC 2018*, 2018.

[7] T. Ochiai, S. Watanabe, S. Katagiri, T. Hori, and J. Hershey, "Speaker adaptation for multichannel end-to-end speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6707–6711.

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

[9] A. Senior and I. Lopez-Moreno, "Improving DNN speaker independence with i-vector inputs," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[10] J. Pan, D. Liu, G. Wan, J. Du, Q. Liu, and Z. Ye, "Online speaker adaptation for LVCSR based on attention mechanism," *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 183–186, 2018.

[11] Z. Fan, J. Li, S. Zhou, and B. Xu, "Speaker-aware speech-transformer," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 222–229.

[12] K. Vesely, S. Watanabe, K. Zmolikova, M. Karafiat, L. Burget, and J. H. Cernocky, "Sequence summarizing neural network for speaker adaptation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5315–5319.

[13] T. Kim, I. Song, and Y. Bengio, "Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition," in *INTERSPEECH 2017 – 18th Annual Conference of the International Speech Communication Association*, 2017.

[14] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7947–7951.

[15] 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 *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7893–7897.

[16] Z. Meng, J. Li, and Y. Gong, "Adversarial speaker adaptation," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5721–5725.

[17] 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 *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012.

[18] S. M. Siniscalchi, J. Li, and C.-H. Lee, "Hermitian polynomial for speaker adaptation of connectionist speech recognition systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2152–2161, 2013.

[19] L. Samarakoon and K. C. Sim, "Factorized hidden layer adaptation for deep neural network based acoustic modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 12, pp. 2241–2250, 2016.

[20] P. Swietojanski, J. Li, and S. Renals, "Learning hidden unit contributions for unsupervised acoustic model adaptation," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 24, no. 8, pp. 1450–1463, 2016.

[21] L. Samarakoon and K. C. Sim, "Subspace LHUC for fast adaptation of deep neural network acoustic models," in *INTERSPEECH 2016 – 17th Annual Conference of the International Speech Communication Association*, 2016, pp. 1593–1597.

[22] X. Xie, X. Liu, T. Lee, S. Hu, and L. Wang, "BLHUC: Bayesian learning of hidden unit contributions for deep neural network speaker adaptation," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5711–5715.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[24] S. Sukhbaatar, E. Grave, G. Lample, H. Jegou, and A. Joulin, "Augmenting self-attention with persistent memory," *arXiv preprint arXiv:1907.01470*, 2019.

[25] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1400–1409.

[26] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.

[27] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: telephone speech corpus for research and development," in *1992 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1992.

[28] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, "AISHELL-1: An open-source mandarin speech corpus and a speech recognition baseline," in *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*. IEEE, 2017.

[29] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "RWTH asr systems for librispeech: Hybrid vs attention," in *INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association*, 2019, pp. 231–235.

[30] A. Bérard, L. Besacier, A. C. Kocabiyikoglu, and O. Pietquin, "End-to-end automatic speech translation of audiobooks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[31] K. Irie, R. Prabhavalkar, A. Kannan, A. Bruguier, D. Rybach, and P. Nguyen, "On the choice of modeling unit for sequence-to-sequence speech recognition," in *INTERSPEECH 2019 – 20th Annual Conference of the International Speech Communication Association*, 2019.

[32] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "Espnet: End-to-end speech processing toolkit," in *INTERSPEECH 2018 – 19th Annual Conference of the International Speech Communication Association*, 2018, pp. 2207–2211.

[33] Z. You, D. Su, J. Chen, C. Weng, and D. Yu, "DFSMN-SAN with persistent memory model for automatic speech recognition," *arXiv preprint arXiv:1910.13282*, 2019.