Online Speaker Adaptation Using Memory-Aware Networks for Speech Recognition

Jia Pan, Genshun Wan, Jun Du, and Zhongfu Ye

***Abstract*—In our previous work, we introduced our attention-based speaker adaptation method, which has been proved to be an efficient online speaker adaptation method for real-time speech recognition. In this paper, we present a more complete framework of this method named memory-aware networks, which consists of the main network, the memory module, the attention module and the connection module. A gate mechanism and a multiple-connections strategy are presented to connect the memory with the main network in order to take full advantage of the memory. An auxiliary speaker classification task is provided to improve the accuracy of the attention module. The fixed-size ordinally forgetting encoding method is used together with average pooling to gather both short-term and long-term information. Furthermore, instead of only using traditional speaker embeddings such as i-vectors or d-vectors as the memory, we design a new form of memory called residual vectors, which can represent different pronunciation habits. Experiments on both the Switchboard and AISHELL-2 tasks show that our method can perform online speaker adaptation very well with no additional adaptation data and with only a relative 3% increase in decoding computation complexity. Under the cross-entropy criterion, our method achieves a relative word error rate reduction of 9.4% and 8.3% compared to that of the speaker-independent model on the Switchboard task and the AISHELL-2 task, respectively, and approximately 7.0% compared to that of the traditional d-vector-based speaker adaptation method.**

***Index Terms*—Speaker adaptation, speech recognition, neural network, memory-aware networks.**

I. Introduction

RECENTLY, the accuracy of automatic speech recognition (ASR) has been greatly improved by the use of deep neural network (DNN) acoustic models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [1]–[4]. However, the performance is still unsatisfactory if the acoustic condition of the test data is mismatched to that of the training data. The adaptation technique is a useful way to alleviate the mismatch between training and testing conditions, and speaker adaptation is one of the most widely used and investigated adaptation techniques at present. In the past few years, many speaker adaptation methods have been proposed to improve the performance of ASR. Most of these methods fall into two categories: model adaptation and speaker adaptive training (SAT).

Model adaptation methods carry out adaptation by fine-tuning the speaker-independent (SI) model using adaptation data, so the fine-tuned model becomes a speaker-dependent (SD) model. Given a large quantity of annotated adaptation data, the SD model can achieve a significant performance improvement over the SI model. However, the gain wears off quickly when the quantity of annotated adaptation data decreases. Moreover, the performance of the SD model may be worse than that of the SI model if very few adaptation data are available. To alleviate the problem of overfitting, much effort has been made. One method is restricting the number of SD parameters. In [5]–[8], an additional layer is inserted into the SI model, and only the parameters of the additional layer are tuned during adaptation, while all the other parameters are fixed. Analogously, in [9], [10], an additional amplitude parameter is defined for each hidden unit. The amplitude parameters are tied for each speaker and are learned while keeping the other parameters fixed. In [11], the authors find that only parts of the weights of recurrent neural networks should be retrained to reduce the number of free parameters. In [12], only the parameters of batch normalization layers are updated during adaptation. In [13], [14], the weight matrix is factorized into three matrices via singular value decomposition (SVD), and only the diagonal matrix is taken as adaptation parameters. Similarly, in [15], each delta matrix between the adapted and SI models is factorized into two low-rank matrices via SVD to further reduce the number of SD parameters. Another method is conservative training. In [16], the Kullback-Leibler divergence (KLD) is adopted to keep the output of the SD model close to that of the original SI model. In [17], the L2 prior regularization proves to be helpful for improving the generalization of adapted models. An auxiliary monophone classification task is adopted in [18] to suppress the influence of prediction errors when the adaptation data are not annotated. The authors in [19] apply adversarial learning to regularize the distribution of deep hidden features in the SD model to be similar to that of the SI model during adaptation.

Speaker adaptive training methods try to obtain a compact model that converts original features into speaker-normalized features. These methods are called speaker adaptive training because the adaptation process is mostly performed during the training stage. One method to perform speaker adaptive training

Manuscript received October 21, 2019; revised February 14, 2020; accepted March 3, 2020. Date of publication March 12, 2020; date of current version April 1, 2020. This work was supported in part by the National Key R&D Program of China under Grant 2017YFB1002202, in part by the National Natural Science Foundation of China under Grants 61671422 and U1613211, in part by the Key Science and Technology Project of Anhui Province under Grant 17030901005, and in part by the Huawei Noah's Ark Lab. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ozlem Kalinli. *(Corresponding author: Jun Du.)*

The authors are with the University of Science and Technology of China, Hefei 230052, China (e-mail: jiapan@mail.ustc.edu.cn; gswan@mail.ustc.edu.cn; jundu@ustc.edu.cn; yezf@ustc.edu.cn).

Digital Object Identifier 10.1109/TASLP.2020.2980372

is using auxiliary features that contain information about speakers. In [20], feature-space maximum likelihood linear regression (f-MLLR) transformations are estimated to perform speaker adaptation on a hidden Markov model with a Gaussian mixtures (GMM-HMM) acoustic model, and then, the f-MLLR transformed features are taken as input for a DNN-based acoustic model. In [21]–[24], speaker i-vectors or bottleneck vectors are obtained using a pretrained speaker recognition model. Then, acoustic features concatenated with the corresponding speaker vectors are fed to a DNN-based acoustic model. The authors in [25]–[27] use speaker codes to represent speaker characteristics. The speaker codes are learned together with the acoustic model. To make speaker embeddings play a more important role in adaptation, [28], [29] try to generate the SD parameters via a controller network that takes speaker embeddings as input, and the controller network is shared among all speakers. In [30], several canonical transformations are presented for each hidden layer of a DNN-based acoustic model, and the interpolation weights to combine the canonical transformations are taken as SD parameters. The canonical transformations are learned together with the interpolation weights. A similar method is also used in [31]. Another method to perform speaker adaptive training is using an adversarial learning scheme. Similar to the methods used in domain adaptation [32]–[34], in [35], a DNN-based acoustic model and a speaker classification model are jointly optimized via adversarial learning. In [36], a reconstruction network is trained to predict the input speaker i-vector. The mean-squared error loss of the i-vector reconstruction and the cross-entropy loss of the acoustic model are jointly optimized through adversarial multitask learning. In [37], domain-private components and an auxiliary task to reconstruct the original speech feature are designed to improve the degree of the domain-invariance and the ASR performance of the acoustic model.

Despite the progress, when we perform speaker adaptation in some real-world dictation tasks, there remain two main problems. First, online speaker adaptation is necessary to control the latency of ASR. Most of the model adaptation methods and some speaker adaptive training methods need to tune the adaptation parameters via the backpropagation algorithm with the current utterance, so they cannot implement speaker adaptation with low latency. In [38], the authors use the click-through data of a user to generate the SD model. However, in most cases, there are insufficient adaptation data, especially for new speakers, and the mismatch between the previous click-through data and the current utterance may largely degrade the performance of adaptation. Second, the adaptation data are scarce. In an extreme case, when one uses a speech recognition service for the first time, only the current utterance can be used as adaptation data. There are few speaker adaptation methods that can work well under this circumstance. The i-vector-based speaker adaptation methods are the mainstream approaches that can meet both requirements, but their performance is unsatisfactory because the i-vector is unreliable for very short segments. In our previous work [39], we proposed an attention-based speaker adaptation method. In this approach, the i-vectors for the speakers in the training set are extracted and stored as a static memory in advance. Then, at each frame, the nearest i-vectors are selected from the memory via the attention mechanism, and the combination of these i-vectors is connected with the main network to provide information about the current speaker. Experiments on the Switchboard corpus have shown that the proposed method could achieve a decent performance improvement compared to that of the i-vector-based speaker adaptation method without the need of additional adaptation data and with only a limited increase in decoding time.

In this study, we extend the conference version of our previous work with the following new contributions. First, we propose a more complete framework named memory-aware networks (MANs) with several improvements: 1) Instead of concatenation, a gate mechanism and a multiple-connections strategy are presented to connect the memory with the main network such that the main network could take full advantage of the memory to generate speaker-normalized representations. 2) An auxiliary speaker classification task is added to force the attention module to provide more accurate information about the speaker. 3) The fixed-size ordinally forgetting encoding (FOFE) method is used together with average pooling so that both short-term information and long-term information are utilized to further improve the accuracy of the attention module. 4) Instead of using speaker i-vectors, which are not directly related to the speech recognition task, we design a new form of memory that consists of residual vectors (r-vectors). We calculate the residual vector between the ground truth and the posterior probability estimated by the acoustic model at each frame and then obtain the average residual vector for each speaker. The cluster centers of the speaker-level residual vectors are taken as a memory. Therefore, the speaker-level residual vectors can both contribute to the loss function of the speech recognition task and preserve discrimination between speakers with different pronunciation habits. Second, we conduct a series of comprehensive experiments on both the Switchboard and AISHELL-2 tasks. The quality of the aggregated speaker vector that is generated by the attention module is verified via visualization of the distances between the aggregated speaker vectors and embedding vectors of different speakers. The results show that the proposed approach achieves a decent performance improvement compared to that of the traditional d-vector-based online speaker adaptation method.

The rest of the paper is organized as follows. Section II gives an overview of the framework of MANs. In Section III, we elaborate the details of each module. In Section IV, we report and analyze the experimental results. Finally, we conclude our findings in Section V.

II. The Framework of MANs

The framework of MANs is presented by extending the attention-based speaker adaptation method proposed in our previous work [39]. The key point of the method is generating embedding vectors for the current speaker via a combination of the embedding vectors of similar speakers. With the help of the attention mechanism and the direct guidance of the ASR objective function, the generated speaker embeddings can provide useful information for ASR. The major advantage of the method is that it works very well without additional adaptation data, and it achieves a performance comparable to that of speaker adaptation methods, which use considerable adaptation data. In

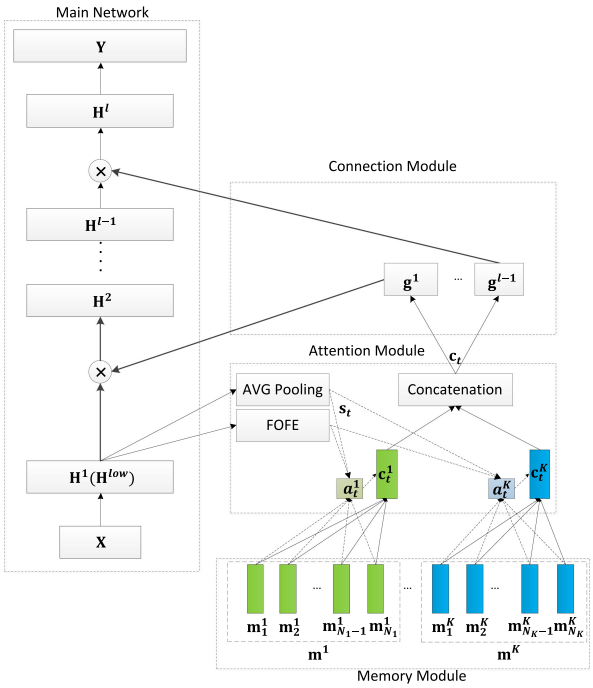


Fig. 1. The framework of MANs, including the memory module, the main network, the attention module and the connection module.

this study, we present a more complete framework and make it robust to the model structure. As shown in Fig. 1, the architecture of MANs consists of four parts, namely, the main network, the memory module, the attention module and the connection module. The details are introduced below.

A. The Main Network

The main network plays two roles in the framework of MANs: acoustic modeling and providing information for the attention module. The role of acoustic modeling is to remove the speaker and other variabilities from the model except for the phoneme information, while the other role is to preserve the speaker information that is necessary for the attention module. To resolve conflicts, we propose that the main network is composed of two parts. The lower part consists of the hidden layers near the input layer, and the upper part consists of the hidden layers near the output layer. The lower part contains some long-term information in addition to the speaker information because the phone-irrelevant information is removed gradually. Accordingly, the outputs of the last layer of the lower part are provided for the attention module. The upper part takes the outputs of the attention module as input to generate a speaker-normalized representation for speech recognition. Determining how to divide the main network is important especially for very deep neural networks, which will be investigated in our experiments.

The main network can be any type of deep neural network used for acoustic modeling, including feedforward neural networks, CNNs and RNNs. CNNs and RNNs are preferred because of their excellent ability of sequence representation. Given an utterance with T speech frames, the acoustic features are represented by $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, where each $\mathbf{x}_t$ represents the feature vector at the frame t. The corresponding outputs of the l-th hidden layer of the main network are denoted as $\mathbf{H}^l = \{\mathbf{h}_1^l, \mathbf{h}_2^l, \ldots, \mathbf{h}_T^l\}$.

B. The Memory Module

The memory module is one of the most important parts of MANs because it provides additional information for adaptation. In this study, the memory module is more flexible; it may contain an arbitrary number of memories rather than only one memory, as in our previous work [39]. Each memory consists of a group of vectors. The vectors in a memory are easily distinguished from each other by its corresponding speaker. To build a memory, a speaker recognition model should be trained in advance. For example, we can train a traditional i-vector system based on the recipe of a universal background model with GMM (GMM-UBM). Then, speaker embeddings are obtained through the pretrained speaker recognition model. Finally, a clustering algorithm such as K-means [40] is adopted to control the number of vectors in a memory, and the cluster centers are taken as a memory.

It is worth mentioning that the speakers included in a memory are not limited to those included in the training set of the acoustic model. This is important because the training dataset of large-scale real-world ASR tasks usually consists of diverse sources of data, and not all the utterances in the training dataset have information about the speaker's identity. This issue prevents traditional SAT methods from working well. Our experiment demonstrates that the performance does not degrade if we remove the vector corresponding to the speaker of the current utterance from the memory during the training stage.

Different memories can be distinguished by the different speaker recognition models that are used to obtain the speaker embeddings. For example, we can build one memory with speaker i-vectors [41] and another with d-vectors [42]. Because the i-vectors are obtained from a GMM-UBM recipe and the d-vectors are obtained from an end-to-end recipe, the information contained in them may be complementary, and the performance of speaker adaptation may be improved with both memories. Different memories can be distinguished by different signal channels alternatively. For example, one memory may include speaker embeddings under far-field scenes, while another memory may contain speaker embeddings under near-field scenes. This configuration of memories gives the acoustic model better compatibility with different scenes. In this paper, we design a new form of memory to enhance the complementarity among different memories, which will be elaborated in Section III.

A memory module containing K kinds of memories is denoted by $\mathbf{M} = \{\mathbf{m}^1, \mathbf{m}^2, \ldots, \mathbf{m}^K\}$, in which $\mathbf{m}^k$ represents the k-th memory. Assuming that the k-th memory has N_k vectors, the memory is denoted by $\mathbf{m}^k = \{\mathbf{m}_1^k, \mathbf{m}_2^k, \ldots, \mathbf{m}_{N_k}^k\}$, in which $\mathbf{m}_i^k$ represents the i-th vector in the k-th memory.

C. The Attention Module

The attention module is equipped to select the vectors that are most similar to the current speech segment from each memory. It consists of two blocks: the information gathering block and the combination block. The information gathering block is used

to obtain information about the current speech segment. The more speaker information that the block obtains, the better the accuracy the attention module can achieve. However, if we use a complex model such as a neural network for information gathering, we may face the overfitting problem. Therefore, the average pooling layer described in the following formula is suitable due to its simplicity.

$$\mathbf{s}_t = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{h}_\tau^{low} \tag{1}$$

In Eq. (1), $\mathbf{h}_\tau^{low}$ is the output of the last layer in the lower part of the main network at the frame τ. Usually, one phoneme only lasts approximately 0.1 seconds, so with t increasing, more long-term information, such as speaker and channel information, can be preserved by $\mathbf{s}_t$, and the phoneme information becomes blurred. Instead of the average pooling layer, some other layers can be adopted alternatively. For example, we use FOFE together with average pooling for information gathering. The details will be introduced in Section III.

The combination block aims to select the vectors most similar to the current speech segment from each memory and combine them into a vector named the aggregated speaker vector. The additive attention mechanism [43] is adopted to learn the similarity scores between $\mathbf{s}_t$ and each vector in a memory, as described in the following formula.

$$e_{t,i}^k = \mathbf{v}^{k\top}\tanh(\mathbf{W}^k\mathbf{s}_t + \mathbf{U}^k\mathbf{m}_i^k) \tag{2}$$

In Eq. (2), $e_{t,i}^k$ is the attention value scoring the similarity between $\mathbf{s}_t$ and $\mathbf{m}_i^k$. The matrices $\mathbf{W}^k$ and $\mathbf{U}^k$, the vector $\mathbf{v}^k$ are the parameters of the attention model for memory k. We choose the additive instead of multiplicative attention [44] because we find that the additive attention can achieve higher accuracy on some other tasks such as end-to-end speech recognition, and we think this conclusion can be extended to the current task.

The attention values are normalized through the logistic sigmoid operation instead of the softmax operation to avoid the sparsity problem [45].

$$a_{t,i}^k = 1/(1+\exp(-e_{t,i}^k)) \tag{3}$$

We try to replace the logistic sigmoid operation by $tanh$ or linear functions to relax restrictions on the range of normalized attention values, but achieve no gains, so we still use the logistic sigmoid operation as a default.

The normalized attention values are used to compute a weighted sum of the vectors in each memory as follows:

$$\mathbf{c}_t^k = \sum_{i=1}^{N_k} a_{t,i}^k\mathbf{m}_i^k \tag{4}$$

In Eq. (4), $\mathbf{c}_t^k$ is the aggregated speaker vector from the k-th memory at the frame t. The aggregated speaker vectors from all memories are concatenated to form a total aggregated speaker vector, which can be denoted as $\mathbf{c}_t$.

D. The Connection Module

The connection module connects the aggregated speaker vector with the main network, and the main network generates speaker-normalized representations using the speaker information in the aggregated speaker vector. A simple method is concatenating the aggregated speaker vector with the outputs of the last layer in the lower part of the main network as $\hat{\mathbf{h}}_t^{low} = [\mathbf{h}_t^{low\top}, \mathbf{c}_t{}^\top]^\top$. However, $\mathbf{h}_t^{low}$ and $\mathbf{c}_t$ belong to two different feature spaces, and it is difficult for the upper part of the main network to further process the concatenated vector, especially when using deep CNNs as the main network. For deep CNNs, if the lower part of the main network consists of few layers, the receptive field of the last layer of the lower part will not be wide enough and long enough, so the output feature maps of this layer can only represent local and short-term features. However, the speaker information contained in the aggregated speaker vector is usually considered long-term features. Therefore, the above simple connection method may affect the extraction of spatial structure information from the time-frequency feature maps by the upper part of the main network. In this study, we propose a multiple-gated-connections mechanism to solve this issue. The details are described in Section III.

III. Implementation Details of MANs

In this section, we introduce the details of the methods mentioned above, including the new form of memory with r-vectors, FOFE, multiple-gated-connections strategies and the auxiliary speaker recognition task.

A. The Memory With R-Vectors

As shown in the previous section, the speaker i-vectors or d-vectors extracted by a pretrained speaker recognition model can be regarded as a memory. The i-vector approach is first introduced for speaker recognition in [41]. In this method, a large GMM with C full-covariance Gaussian components is trained as a UBM to represent the distribution of acoustic features, denoted as

$$\mathbf{x} \sim \sum_{c=1}^{C} w_c\mathcal{N}(\mathbf{x};\mu_c,\Sigma_c) \tag{5}$$

In Eq. (5), μ_c, Σ_c and w_c represent the mean, covariance and weight of the k-th Gaussian component, respectively, and $\mathbf{x}$ represents the acoustic feature vector. Then, the mean vectors of all Gaussian components are concatenated as a supervector, and the supervector is adapted for each speaker as the following expression:

$$\mathbf{z}_s = \mathbf{z} + \mathbf{T}\mathbf{w}_s \tag{6}$$

In Eq. (6), $\mathbf{z}$ is the supervector of the UBM, $\mathbf{T}$ is a matrix with low rank called the loading matrix, and $\mathbf{w}_s$ is the i-vector for speaker s. $\mathbf{T}$ and $\mathbf{w}_s$ are learned jointly using a maximum a posteriori (MAP) criterion. Normally, linear discrimination analysis (LDA) [46] is subsequently performed to reduce the dimension and improve the discrimination of i-vectors.

The d-vector approach [42] uses neural networks to generate speaker embeddings. The neural network is trained by speaker discriminative criteria such as the cross-entropy or triplet loss [47]. After training, the output of the last hidden layer is obtained to produce a frame-level speaker representation, and

Algorithm 1: The Extraction Process of R-Vectors.

for n in number of utterances: **do**
  for t in number of frames of utterance n: **do**
    $\mathbf{r}_t^n = \hat{\mathbf{y}}_t^n$ - $\mathbf{y}_t^n$
  end for
end for
for d in number of mono-phones: **do**
  $\mathbf{R}_t^n[d] = \sum_{j:phone(j)=d} \mathbf{r}_t^n[j]$
  if $\mathbf{R}_t^n[d] < threshold$ **then**
    $\mathbf{R}_t^n[d] = 0$
  end if
end for
for s in number of speakers: **do**
  $N_s =$ Number of frames for speaker s
  $\mathbf{E}_s = \frac{1}{N_s} \sum_{j:speaker(j)=s} \sum_t \mathbf{R}_t^n$
end for
return $\mathbf{E}_s$

all the frame-level representations are then averaged to form an utterance-level speaker embedding called a d-vector.

Both the i-vectors and the d-vectors have a low intraspeaker variance to guarantee the accuracy of speaker classification, so the short-term information such as phoneme information is not well contained in these traditional speaker embeddings. However, the speaker information contained in the output of the lower part of the main network is weak because the task is ASR. Furthermore, the speech segment is not long enough for the attention module to select similar speaker vectors. Is there a type of speaker embedding that contains adequate speaker information and works well on short speech segments?

People usually have their own pronunciation habits, e.g., accents. In a certain accent, there are fixed patterns of pronunciation errors. For example, in a southern Chinese accent, 'n' is usually mispronounced as 'l'. These pronunciation habits can distinguish between different speakers and are not explicitly considered in the design of i-vectors or d-vectors. Inspired by these considerations, we design a new speaker embedding method called r-vectors. The extraction process of r-vectors is divided into four steps. First, we calculate the posterior probabilities of triphone states using a pretrained SI acoustic model and obtain the residual vector between the ground-truth one-hot vector and the posterior probability vector at each speech frame of each utterance in the training data. Second, we convert the residual vector from a state-level to phone-level vector by summing up values of the states that belong to the same phone. This conversion is necessary to reduce the dimensionality of the residual vector. Third, to enhance the discrimination of the residual vector and highlight the pronunciation error pairs, a threshold is set to prune the unrepresentative values. Finally, the frame-level residual vectors belonging to the same speaker are averaged to obtain the speaker-level residual vector. The frame-level residual vector may be influenced by many factors, such as environmental noise, but the utterances of one speaker may happen in different scenes, so the speaker-level residual vector is more robust than the frame-level vector, and to some extent, can represent pronunciation habits. The procedure is described in Algorithm 1.

The speaker-level r-vectors are clustered via the K-means algorithm, and the cluster centers are taken as a memory. Unlike the i-vectors or d-vectors, r-vectors are clustered using the Euclidean distance as the distance measure rather than the cosine distance. In addition to relaxing the requirement of a large quantity of data, another advantage of r-vectors is that the contained information of pronunciation error pairs may help the acoustic model correct pronunciation errors. We do not care much about the performance of speaker classification for r-vectors because our main task is speech recognition. Due to the very different characteristics of speaker embeddings, it is necessary for us to use both r-vectors and i-vectors/d-vectors as memories.

B. Fixed-Size Ordinally Forgetting Encoding

In the information gathering block of the attention module, an average pooling is often used to obtain long-term information. The average pooling means that all the frames have the same importance. However, the final task of the aggregated speaker vector is to recognize the current speech frame, so nearby frames should have more importance than the remote frames. To realize this idea, we introduce fixed-size ordinally forgetting encoding [48] into our framework. FOFE can uniquely encode any variable-length sequence into a fixed-size code with an emphasis on near-term tokens. The key point of FOFE is the forgetting factor. With a suitable factor, FOFE can achieve considerable gains on many tasks, such as language modeling. Another advantage of FOFE is that it is a nonparametric model, and it is not necessary to consider the problem of overfitting. The process of FOFE is described by the following formula.

$$\mathbf{s}_t = \sum_{\tau=0}^{t-1} \alpha^\tau \mathbf{h}_{t-\tau}^{low} \tag{7}$$

In Eq. (7), α is the forgetting factor, which is constrained to (0, 1). Moreover, when we use r-vectors as a memory, FOFE is more important because both FOFE and r-vectors focus on local information. In general, average pooling pays close attention to global information, while FOFE focuses on near-term information, so we can use them together via the multihead attention mechanism introduced in [49].

C. Multiple Gated Connections

In our previous work, the aggregated speaker vector is only connected to the first layer of the upper part of the main network. When deep convolutional neural networks are adopted as the main network, this simple connection mechanism seems a bit inappropriate. Using this connection mechanism, the first convolution layer in the upper part of the main network will add the linearly transformed aggregated speaker vector to the output feature maps pixel by pixel. If the receptive field of the layer is not wide enough, this operation means adding global information to local information, and it is a burden for the later convolution layers to process both types of information together. To solve this issue, similar to the approach in [50], we propose a channelwise gate mechanism described by the following formula.

$$\hat{\mathbf{h}}_{t,(u,v)}^l = \mathbf{h}_{t,(u,v)}^l \odot \mathbf{g}^l = \mathbf{h}_{t,(u,v)}^l \odot \sigma(\mathbf{W}^l \mathbf{c}_t + \mathbf{b}) \tag{8}$$

In Eq. (8), $\mathbf{g}^l$ is the vector denoting the gates for the output channels of the l-th convolutional layer. $\mathbf{h}^l_{t,(u,v)}$ is the outputs of the l-th convolutional layer for pixel (u, v), so it is an M-dimensional vector, where M is the number of output channels. $\sigma(\bullet)$ denotes the logistic function that operates on each element of a vector. $\odot$ means the elementwise product operation. The parameter $\mathbf{W}^l$ is a matrix used to transform $\mathbf{c}_t$ into an M-dimensional vector. In other words, a channelwise and elementwise product operation is adopted. Each feature map has a gate, and all the elements in a feature map share the gate. $\hat{\mathbf{h}}^l_{t,(u,v)}$ is used as the input of the next layer.

Inspired by ResNet [51], we add direct connections between the aggregated speaker vector and layers of the upper part of the main network. These direct connections may alleviate the gradient vanishing problem and further improve the effect of the aggregated speaker vector. Combining the gate mechanism with the direct-connections mechanism, such connection strategy is called multiple gated connections.

D. The Auxiliary Speaker Classification Task

Through the above introduction, the aggregated speaker vector is automatically generated using the attention mechanism. We cannot guarantee that the aggregated speaker vectors are close for the same speaker and distinguished for different speakers. This fact may increase the risk of overfitting, especially when the speaker vectors in the memory are obtained using a small quantity of data. Under this circumstance, the attention model may learn to measure the similarity via some noise rather than the speaker information. In our previous work [39], a recurrent attention mechanism was proposed to guarantee that the attention values do not change greatly between consecutive frames within an utterance, but there was no constraint on the attention values among utterances belonging to the same speaker.

In this study, we propose an auxiliary speaker recognition task to force the aggregated speaker vectors to have a strong ability of speaker classification. The triplet loss function and cross-entropy loss function are usually used for speaker classification tasks. We choose the cross-entropy loss function for the auxiliary speaker classification task because of its convenience for training. Specifically, a softmax regression classifier is added on top of the aggregated speaker vector in the training step to implement the speaker classification task and is removed in the inference step. The final loss function at the cross-entropy training stage is described by the following formula:

$$\begin{aligned}\mathcal{L}(\theta_{asr}, \theta_{sre}, \theta_{share}) = &\sum_{n=1}^{N}\sum_{t=1}^{T_n} \log p_{\theta_{asr},\theta_{share}}(y_t^n|\mathbf{x}_t^n, \mathcal{M}) \\ &+ \lambda \sum_{n=1}^{N}\sum_{t=1}^{T_n} \log p_{\theta_{sre},\theta_{share}}(s^n|\mathbf{x}_t^n, \mathcal{M})\end{aligned} \tag{9}$$

In Eq. (9), the first item is the cross-entropy loss for the speech recognition task. $\mathbf{x}_t^n$ and y_t^n indicate the acoustic feature vector and the triphone state label for frame t in utterance n. T_n is the number of frames of utterance n, and N is the number of total utterances in the training set. θ_{asr}, θ_{sre} and θ_{share} denotes the model parameters for ASR, speaker classification and shared parts, respectively. The last item is the cross-entropy loss for the auxiliary speaker classification task, and s^n represents the speaker label for utterance n. $\mathcal{M}$ means the memory. λ is a hyperparameter to balance these two tasks.

IV. Experiments and Result Analysis

A. Experimental Setup

We evaluated the performance of the proposed approach on both English and Mandarin large vocabulary continuous speech recognition (LVCSR) tasks; the details of these tasks are illustrated in Table I.

The Switchboard (SWB) task [52] is a benchmark for English LVCSR, so we chose it as the main task. The training data consist of the 20-hour English CALLHOME training dataset and 309-hour Switchboard-I training dataset, including a total of 5110 speakers. In the following experiments, we mainly used the SWB part of the NIST 2000 Hub5 evaluation set as the test set, which contains 1831 utterances from 40 speakers. We also performed an overall evaluation on the CALLHOME part. The 13-dimensional perceptual linear prediction (PLP) features and their first- and second-order derivatives were extracted to train a GMM-HMM ASR system, which was used to obtain the state-level alignments for training deep neural networks. These features were preprocessed with speaker-level mean and variance normalization. The cross-word triphone GMM-HMM system with 8991 tied states and 360,000 Gaussian components was trained with the maximum likelihood criterion. A trigram language model was trained on the 2000 h Fisher-corpus transcripts with a dictionary of approximately 39,000 words for testing.

We chose the AISHELL-2 task as our Mandarin LVCSR task. The AISHELL-2 task is an elaborately prepared public dataset [53]. It consists of 1000 hours of clean audio segments recorded via an iPhone channel from 1991 speakers. There are 1293 speakers with slight northern accents, 678 speakers with southern accents and 20 speakers with other accents during the recording. The development set contains 2500 utterances from 5 speakers, and the test set contains 5000 utterances from 10 speakers. Each speaker has approximately half an hour of audio segments. To generate the state-level alignments, a GMM-HMM system with 9004 tied states and 40 Gaussian components per state was trained using the Mel-frequency cepstral coefficient (MFCC) features. For the ASR setup, the vocabulary contained approximately 121,000 words, and the language model was a 3-gram model.

B. Establishment of Baselines

In this subsection, we experimented with a VGG-like model architecture [54] based on the frame-level cross-entropy criterion. The details of the model architecture are reported in Table II. The inputs of the model were the 40-dimensional log Mel-scale filter-bank features processed with mean and variance normalization. The speech was analyzed using a 25-ms Hamming window with a 10-ms fixed frame rate. The architecture of the model consisted of many convolutional layers and pooling

TABLE I
THE DETAILS OF THE SPEECH RECOGNITION TASKS

Task name	Training set size	# of training speakers	Testing set size	# of testing speakers	# of tied-states	Vocabulary size	Language model
Switchboard	329 hours	5110	1831 utterances	40	8991	39,000	3-gram
AISHELL-2	1000 hours	1991	5000 utterances	10	9004	121,000	3-gram

TABLE II
THE DETAILS OF THE MODEL ARCHITECTURE

Stage	Output (c,h,w)[a]	Kernel size
conv0	(64,40,T)	(3x3)
pooling0	(64,40,T/2)	(1x2)
conv1-4	(64,40,T/2)	(3x3)
pooling1	(64,20,T/4)	(2x2)
conv5-8	(128,20,T/4)	(3x3)
pooling2	(128,10,T/4)	(2x1)
conv9-12	(256,10,T/4)	(3x3)
pooling3	(256,5,T/4)	(2x1)
conv13-16	(512,5,T/4)	(3x3)
pooling4	(512,3,T/4)	(2x1)
conv17	(2048,1,T/4)	(3x3)
deconv(SWB)	(8991,1,T)	(1x4)
deconv(AISHELL-2)	(9004,1,T)	(1x4)

[a] c, h, and w mean the number of channels and the height and width of the feature map, respectively.

TABLE III
PERFORMANCE OF THE BASELINE MODELS ON THE SWB TASK

Method	WER(%)	WERR
SI baseline	13.8	–
SD baseline (i-vectors, speaker level)	13.3	3.6%
SD baseline (d-vectors, speaker level)	13.0	5.8%
SD baseline (i-vectors, utterance level)	13.7	0.7%
SD baseline (d-vectors, utterance level)	13.5	2.2%

layers. Each convolutional layer was equipped with a standard ReLU activation function. Because of the pooling in the time domain, the temporal resolution has been reduced to one quarter. Accordingly, a deconvolution layer with a stride of four was used as the output layer to recover the time resolution. We shuffled the utterances in the training data and grouped them into minibatches with a limit of 2048 frames per minibatch to speed up training. Stochastic gradient descent was used as the optimizer, and the initial learning rate was set to 0.02. At the beginning of each epoch, the learning rate was halved if the accuracy of the validation set did not increase, and the training process was stopped after 12 epochs. All experiments in our study were performed using the CAFFE toolkit [55] and run on a server equipped with 4 Tesla P40 GPUs.

As mentioned in the opening chapter, speaker i-vectors or d-vectors can be taken as additional inputs for a convolutional neural network acoustic model to perform speaker adaptive training. To obtain the i-vectors, a UBM with 512 Gaussian components was trained first using all the training data and the 39-dimensional PLP features mentioned above. Then, 300-dimensional i-vectors were extracted and further compressed to 64 dimensions via LDA followed by length normalization. Regarding the d-vectors, the network included five convolutional layers. The utterances belonging to the same speaker were concatenated and split into audio segments, each of which had 500 frames. 64-dimensional log Mel-scale filter-bank features were taken as input. The kernel size of each convolutional layer was set to 3×3, and the stride was set to one. Each convolutional layer was connected to a max pooling layer with a stride of 2×2. The cross-entropy loss and triplet loss were employed for training. After training, an average pooling was used to obtain the speaker representation vector. The speaker representation vectors processed via length normalization were taken as the d-vectors.

The i-vector- and d-vector-based SD models were evaluated at both the speaker and utterance levels. The utterance-level i-vectors or d-vectors were extracted from each utterance separately during both training and testing steps. The speaker-level i-vectors or d-vectors were extracted using all the utterances from the same speaker. Table III reports the word error rate (WER) of baseline models on the SWB task. The d-vector-based SD model achieves better results than the i-vector-based model does at both levels, with a 0.2%–0.3% absolute gain. The performance at the utterance level is much worse than that at the speaker level. The best SD model at the utterance level only achieves a relative WER reduction (WERR) of 2.2% compared to that of the SI model. This result is because the utterance-level i-vectors or d-vectors are not robust due to the random utterance length.

C. Experiments With MANs

In this subsection, we evaluated the performance of MANs on the SWB task.

1) Memory With r-Vectors: We first evaluated the effectiveness of the memory with r-vectors based on the MANs framework. For extraction of the r-vectors, we calculated the output posterior probability of the SI baseline model for each frame and then obtained the frame-level residual vector. Next, the residual vector was converted from a state-level to a phone-level vector to reduce the dimensionality to 45. To enhance the discrimination of the residual vector and highlight the pronunciation error pairs, a threshold value 0.2 was used to prune unrepresentative values. Speaker-level residual vectors were obtained by averaging over all the frame-level residual vectors belonging to the same speaker. All 5110 speaker-level residual vectors in the training set were clustered into 128 classes via the K-means algorithm.

To verify that the r-vectors can be regarded as high-frequency pronunciation errors corresponding to accents or pronunciation habits, we show the r-vectors of some speakers in Fig. 2. The boxes in red represent phonemes for which it is easy to make

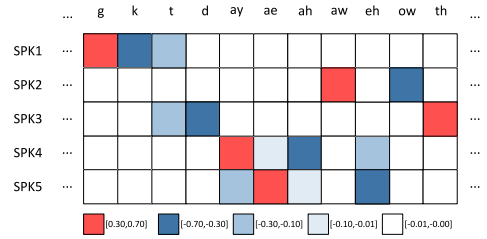


Fig. 2. Illustration of speaker-level r-vectors.

TABLE IV
PERFORMANCE COMPARISON FOR THE MEMORY WITH R-VECTORS

Method	WER (%)	WERR
SI baseline	13.8	–
MANs (i-vectors)	13.2	4.3%
MANs (d-vectors)	13.2	4.3%
MANs (r-vectors)	13.4	2.9%
MANs (i-vectors+d-vectors)	13.1	5.1%
MANs (i-vectors+r-vectors)	13.0	5.8%
MANs (d-vectors+r-vectors)	13.0	5.8%

mistakes, while the boxes in blue represent the high-frequency incorrect phonemes. For speaker 2 in Fig. 2, we can observe that 'aw' is often recognized as 'ow'. This phenomenon often happens for speakers with Canadian accents when they speak words such as 'about'. For speaker 4, 'ay' is often recognized as 'ah'. This phenomenon represents a pronunciation habit for some speakers with southern American accents. They often speak more quickly, so when they say 'my,' it sounds like 'mah'. This result implies that the speaker-level residual vectors can be taken as a means of representing speaker information.

For comparison, MANs taking the i-vectors or the d-vectors as the memory were also trained. The results reported in Table IV show that MANs taking the i-vectors as the memory achieve a relative 4.3% WER reduction over the SI baseline model and a relative 3.6% WER reduction over the utterance-level i-vector-based speaker adaptation method. This result proves the effectiveness of MANs. Though the gap is narrowed when replacing the i-vectors by the d-vectors, the performance is still comparable to that of the speaker-level d-vector-based speaker adaptation method and is better than that of the utterance-level d-vector-based speaker adaptation method. Compared to the memory with i-vectors or d-vectors, the memory with r-vectors has no significant advantage. However, the r-vectors pay more attention to local information, so the combination of the memories with d-vectors and r-vectors shows high complementarity. As listed in Table IV, the combination of the memories with i-vectors and d-vectors leads to a slight performance increase compared to that using them individually because they are more consistent in terms of speaker information, but the combination of the memories with r-vectors and i-vectors or d-vectors achieves better performance, with a total absolute gain of 0.2% compared to that of i-vectors or d-vectors only.

2) FOFE: In the information gathering block, FOFE is taken as a substitute for average pooling to obtain context information. The main difference between average pooling and FOFE is the forgetting mechanism. We experimented with a forgetting factor varying between [0.5, 0.8] because using an excessively large or small forgetting factor may hurt the performance. The experimental results have shown that the best choice of forgetting factor is approximately 0.7, so we set $\alpha = 0.7$ for the remaining experiments in this paper.

TABLE V
PERFORMANCE OF THE MANS WITH FOFE

Method	WER (%)	WERR
SI baseline	13.8	–
MANs (d-vectors, Average pooling)	13.2	4.3%
MANs (d-vectors, FOFE)	13.1	5.1%
MANs (d-vectors, FOFE+Average pooling)	13.0	5.8%

TABLE VI
PERFORMANCE OF THE MANS WITH MULTIPLE GATED CONNECTIONS

Method	WER(%)	WERR
SI baseline	13.8	–
SD baseline (d-vectors, utterance level)	13.5	2.2%
SD baseline (d-vectors, utterance level, first layer of each block, gated)	13.4	2.9%
MANs (d-vectors, first layer, simple)	13.2	4.3%
MANs (d-vectors, first layer, gated)	13.1	5.1%
MANs (d-vectors, last layer, gated)	13.3	3.6%
MANs (d-vectors, first layer of each block, simple)	13.2	4.3%
MANs (d-vectors, first layer of each block, gated)	13.0	5.8%
MANs (d-vectors, each layer, gated)	13.0	5.8%

As shown in Table V, replacing average pooling with FOFE can lead to outperforming the basic MANs, with an absolute gain of 0.1%. When we take multihead attention into consideration, which means that FOFE and average pooling are employed as different heads, an extra absolute WER reduction of 0.1% is achieved. The two heads pay attention to different audio segments: FOFE focuses on local information, and average pooling focuses on global information. This complementarity is similar to the combination of the memories with d-vectors and r-vectors introduced above.

3) Multiple Gated Connections: In the above experiments, the aggregated speaker vectors were attached to the output of the first hidden layer simply, which may result in an inflexible and insufficient implementation of memory. We replaced it by the multiple-gated-connections strategy to capture more detailed and important features from the memory. In our experiments, we mainly investigated two types of multiple connections: connection to the first layers of each convolutional block and connection to all convolutional layers.

We first evaluated the effect of the gated connection strategy. As reported in Table VI, if the aggregated speaker vector is only connected to the first layer, it is found that the gated connection mechanism obtains improvement compared with the simple connection. When the memory is connected to the last convolutional layer instead of the first layer, the performance is much worse. This result indicates the importance of determining how to divide

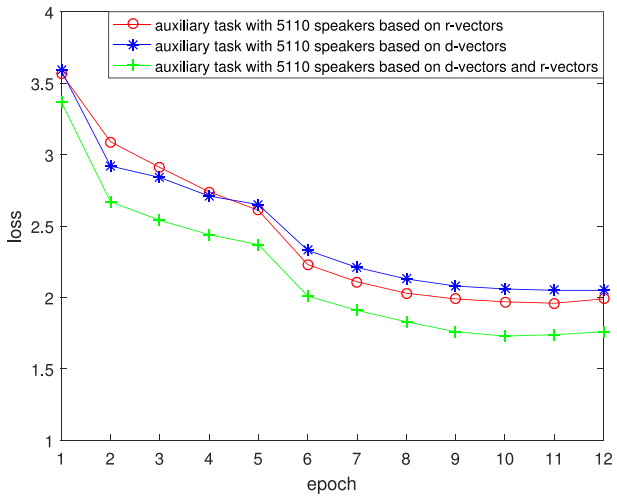


Fig. 3. Illustration of the losses of different tasks.

the main network. We then evaluated the effect of the multiple-gated-connections strategy. When we connected the aggregated speaker vector to the first layers of each convolutional block (layer conv0, conv1, conv5, conv9, conv13) without changing the connection type, there is no improvement at all. That is, the aggregated speaker vector does not make any contribution to the top layers. However, multiple gated connections yield positive effects, which indicates that the gated connection mechanism can make better utilization of the aggregated speaker vector, and accordingly, the multiple-connections strategy is useful. A total absolute WERR of 0.2% is obtained using the multiple-gated-connections strategy when the aggregated speaker vector is connected to the first layer of each block. An absolute WERR of 0.1% is also obtained when we used the multiple-gated-connections strategy to the traditional d-vector-based speaker adaptation recipe, and it further proves the effectiveness of the multiple-gated-connections strategy. When we attempted to connect the aggregated speaker vector to all the convolutional layers of the model (from layer conv0 to layer conv16), no more improvement was observed, so in the following experiments, we only connected the aggregated speaker vector to the first layers of each block.

4) Auxiliary Task: To constrain the attention values among the utterances belonging to the same speaker, we add an auxiliary speaker recognition task with the cross-entropy loss. We evaluated the performance of the auxiliary task with different targets and different memories.

The losses of the auxiliary speaker recognition task for MANs with different memories are shown in Fig. 3. We found that the loss can reach a relatively low value when we used r-vectors and d-vectors together. This result further verified the complementarity between r-vectors and d-vectors. As reported in Table VII, by using both d-vectors and r-vectors as memories, the auxiliary speaker recognition task can force the attention module to capture speaker information more accurately and achieves an absolute WER reduction of 0.1%, but according to the "p-value" of 0.104 reported in the Table X, the improvement is not obvious. Note that we found that a good choice of the scaling factor of the auxiliary task lies in [0.2, 0.5], so in the following experiments, the factor was set to 0.3. We compared the performance of auxiliary tasks with the 5110 speakers or the 128 speaker clusters as the classification target, and there was no difference, so we chose the former for the following experiments.

TABLE VII
PERFORMANCE OF THE MANS WITH THE AUXILIARY TASK

Method	WER (%)	WERR
SI baseline	13.8	–
MANs (d-vectors)	13.2	4.3%
MANs (d-vectors, auxiliary task[1])	13.2	4.3%
MANs (d-vectors, auxiliary task[2])	13.2	4.3%
MANs (d-vectors+r-vectors)	13.0	5.8%
MANs (d-vectors+r-vectors, auxiliary task[2])	12.9	6.5%

[1]Auxiliary task with 128 speaker clusters as targets.
[2]Auxiliary task with 5110 speakers as targets.

TABLE VIII
PERFORMANCE OF THE MANS WHEN THE CORRESPONDING SPEAKER IS NOT CONTAINED IN THE MEMORY

Method	WER (%)	WERR
SI baseline	13.8	–
MANs (d-vectors)	13.2	4.3%
MANs (d-vectors, removed)	13.2	4.3%

TABLE IX
PERFORMANCE OF THE MANS WITH DIFFERENT NORMALIZATION FUNCTIONS FOR ATTENTION VALUES

Method	WER (%)	WERR
SI baseline	13.8	–
MANs (d-vectors, sigmoid)	13.2	4.3%
MANs (d-vectors, tanh)	13.3	3.6%
MANs (d-vectors, linear)	13.7	0.7%

5) Other Experiments: Furthermore, we evaluated the performance when we removed the corresponding cluster center of current utterance from the memory. The results in Table VIII showed that there was no performance degradation, which implies that we can generate an accurate speaker embedding even when the speakers involved in the memory are different from the speakers involved in the training data of the acoustic model. Thus, we can perform speaker adaptive training without the speaker's identity information of each utterance in the training data, which is impossible for traditional speaker adaptive training methods.

We also evaluated the performance of different activation functions for the attention values. The attention values for all of the above experiments are normalized through the logistic sigmoid operation. To relax the restriction on the range of normalized attention values, we change the normalization function to $tanh$ and linear functions. As shown in Table IX, neither $tanh$ nor linear functions can achieve further improvements.

6) Significance Tests: The significance test is a two-tailed test with the null hypothesis that there is no performance difference between the two systems. The "p-value" is the main

TABLE X
THE RESULTS OF SIGNIFICANCE TESTS OF OUR METHODS

Method	WER(%)	p-value
MANs (d-vectors)	13.2	–
MANs (d-vectors + r-vectors)	13.0	<0.001
MANs (d-vectors, FOFE+Average pooling)	13.0	0.002
MANs (d-vectors, Multiple-gated-connections)	13.0	<0.001
MANs (d-vectors + r-vectors, Auxiliary task)	12.9	0.104[1]

[1]Compared to MANs (d-vectors + r-vectors).

indicator of significance tests, and reflects the degree of support for the null hypothesis. The smaller "p-value" is, the bigger significant differences between two systems are. We adopted the "Matched Pair Test" method mentioned in [56] to calculate the "p-value" of each of the above methods. The "p-values" of different models with the cross-entropy criterion are given in Table X. The results in Table X imply that there is a high probability that our methods are able to achieve a different performance compared to the vanilla MANs.

D. Overall Results

First, we summarized these methods, including the combination of the r-vectors and the d-vectors, the combination of FOFE and average pooling, the multiple-gated-connections strategy and the auxiliary speaker classification task. To make the results more convincing, we also presented the results on the CALLHOME part of the NIST 2000 Hub5 evaluation set. In order to verify the robustness of our methods, we also reported the results under the sequence-level discriminative training and the results on a Bi-directional long short-term memory (BLSTM) model. The BLSTM model consisted of 3 bidirectional layers with 2048 cells per layer (1024 per direction). The sequence-level discriminative training was achieved through the maximum mutual information (MMI) objective function [57] smoothed by adding the scaled gradient of the cross-entropy loss [58].

From Table XI, it can be observed that under the cross-entropy criterion, when tested with all the proposed approaches, the WER of MANs is 12.5%, representing a relative WER reduction of 9.4% over the SI model and a relative WER reduction of 5.3% over MANs without these methods. The gains of MANs for the CALLHOME part is slightly smaller than the gains for the SWB part. The acoustic models trained via the MMI objective function can achieve a relative WER reduction between 6% and 9% over the acoustic models trained via the cross-entropy criterion. The gains of MANs under the MMI criterion are slightly smaller than those under the cross-entropy criterion. The gains of MANs on BLSTMs are comparable to the gains on CNNs.

Second, we evaluated the performance of MANs under noisy scenarios. Data augmentation was performed using a simulator, adding different types of noise with an SNR ranging from 5 to 20 dB. The noise sources were obtained from the dataset of the CHIME4 challenge [59], and contained four types of noise: bus, cafe, pedestrian area and street junction. This doubled the amount of training data. Each utterance of the test set also added with random noise at a random SNR. To verify that MANs has the ability to perform environment adaptation, two memories were used, one from the raw training data and the other from the noisy training data. It is worth noting that the model of the d-vectors is not retrained with simulated training data.

TABLE XI
PERFORMANCE OF THE MANS WITH ALL OF THE PROPOSED IMPROVEMENTS UNDER DIFFERENT CONFIGURATIONS

Method		SWB		CALLHOME	
		WER(%)	WERR	WER(%)	WERR
CNN CE	SI baseline	13.8	–	23.3	–
	MANs (d-vectors)	13.2	4.3%	22.4	3.9%
	MANs (all methods)	12.5	9.4%	21.4	8.2%
CNN MMI	SI baseline	12.9	–	21.8	–
	MANs (d-vectors)	12.4	3.9%	21.0	3.7%
	MANs (all methods)	11.8	8.5%	20.1	7.8%
BLSTM CE	SI baseline	14.1	–	23.5	–
	MANs (d-vectors)	13.4	5.0%	22.6	3.8%
	MANs (all methods)	12.7	9.9%	21.7	7.7%
BLSTM MMI	SI baseline	13.1	–	22	–
	MANs (d-vectors)	12.5	4.6%	21.1	4.1%
	MANs (all methods)	11.9	9.2%	20.4	7.3%

TABLE XII
PERFORMANCE OF THE MANS UNDER NOISY SCENARIOS

Method	SWB RAW		SWB NOISY	
	WER(%)	WERR	WER(%)	WERR
SI baseline	13.2	–	19.2	–
SD baseline (d-vectors, utterance level)	12.9	2.3%	19.2	0.0%
MANs (one memory)	11.9	9.8%	18.1	5.7%
MANs (two memories)	12.0	9.1%	17.5	8.9%

The results with the cross-entropy criterion are given in Table XII. From Table XII, we can see that the performance has an overall improvement on the raw test set due to data augmentation. The gain of the traditional d-vector-based speaker adaptation method decreases significantly on the noisy test set. This may because that the model of the d-vectors is not retrained with noisy data. The result is similar for MANs with only one memory from the raw training data. However, when we added a new memory from the noisy training data, the loss of gain on the noisy test set is back with only a slight decrease in gain on the raw test set. This implies that our methods have the ability of environment adaptation.

Third, we evaluated the performance of MANs with all the methods mentioned above on the AISHELL-2 task to verify the generalization ability. The experimental details of the AISHELL-2 task were similar to those of the SWB task. The results with the cross-entropy criterion are given in Table XIII. As reported in Table XIII, the WER of the MANs with all of the proposed improvements is 6.6%, representing a relative WER reduction of 8.3% over the SI model and a relative WER

TABLE XIII
PERFORMANCE OF MANS ON THE AISHELL-2 DATASET

Method	WER (%)	WERR
SI baseline	7.2	–
SD baseline (d-vectors, speaker level)	6.9	4.2%
SD baseline (d-vectors, utterance level)	7.1	1.4%
MANs (d-vectors)	6.9	4.2%
MANs (all methods)	6.6	8.3%

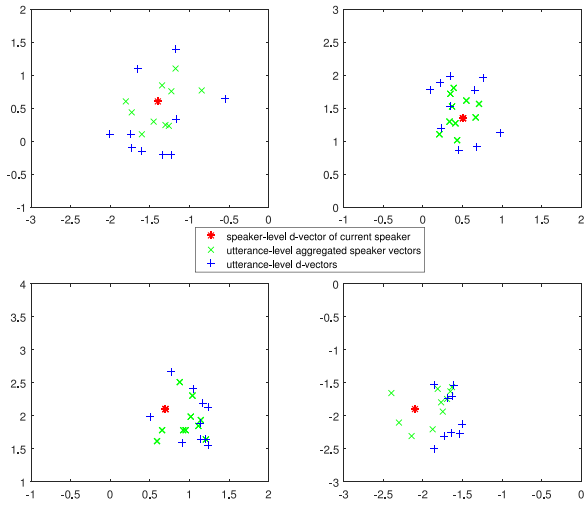


Fig. 4. t-SNE of the aggregated speaker vectors in the training set.

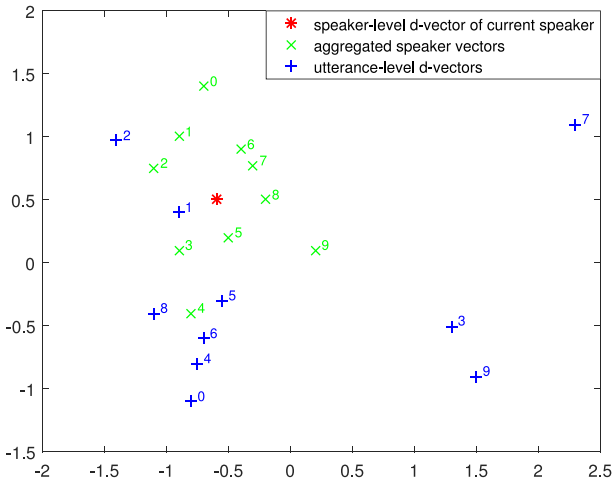


Fig. 5. t-SNE of the aggregated speaker vectors in the test set.

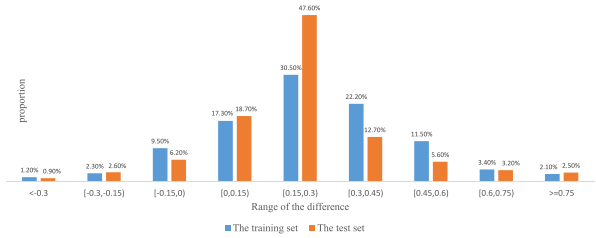


Fig. 6. The histograms of the differences in the distance to the speaker-level d-vectors between the aggregated speaker vectors and the utterance-level d-vectors the in the training and test set.

reduction of 7.0% over the utterance-level d-vector-based SD baseline model.

Finally, We calculated the computational complexity of MANs and found only a 3% increase over the SI model. This computational complexity does not affect the real-time quality of our model. We also calculated the computational complexity of the d-vector-based SD model. It has a 31% increase in computational complexity compared to that of the SI model and may lead to an increase in the decoding time.

E. Analysis of the Aggregated Speaker Vector

To verify the effectiveness of the aggregated speaker vectors, we compared them with the utterance-level d-vectors with t-distributed stochastic neighbor embedding (t-SNE) [60] on both training data and test data. As one of the dimension reduction methods for data visualization, t-SNE can provide better visualization than conventional dimension reduction can by relieving the so-called crowding problem. We first randomly picked four speakers from the training set and one speaker from the test set and then obtained the aggregated speaker vector for each frame of each utterance of each speaker. We averaged all the frame-level aggregated speaker vectors in an utterance together to obtain the utterance-level aggregated speaker vector. For comparison, traditional utterance-level d-vectors were also extracted for each utterance of each speaker.

We first showed the cohesiveness of the utterance-level speaker vectors. Top-10 utterance-level d-vectors and aggregated speaker vectors that were closest to the speaker-level d-vector were selected. As shown in Fig. 4, the aggregated speaker vectors are closer to each other than the utterance-level d-vectors. In other words, the aggregated speaker vectors are more cohesive than d-vectors. We then demonstrated the superiority of the aggregated speaker vector. We randomly picked one speaker from the test set and randomly picked 10 utterances of the speaker. The utterance-level d-vectors and aggregated speaker vectors of these ten utterances were obtained. As shown in Fig. 5, the aggregated speaker vector is closer to the speaker-level d-vector than the utterance-level d-vector for 90% of utterances. To solidify the conclusion, we calculated the Euclidean distance between the utterance-level speaker vector and speaker-level d-vector for all the utterances and all the speakers in the training and test set and compared the distance of the utterance-level d-vectors with the distance of the aggregated speaker vectors for each speaker. The differences in the distance in the training and test set are illustrated with histograms respectively. As shown in Fig. 6, for 87% of the speakers in the training set and 90.3% of the speakers in the test set, the aggregated speaker vectors are closer to the speaker-level d-vectors than the utterance-level d-vectors. This again demonstrates the superiority of the aggregated speaker vector.

V. CONCLUSION

In this paper, we have presented an online speaker adaptation method for speech recognition using memory-aware networks. A new form of speaker embeddings called r-vectors was designed and taken as a memory. Three improvements, the

multiple-gated-connections strategy, the auxiliary speaker classification task and the fixed-size ordinally forgetting encoding method were provided to improve the robustness and effectiveness of the framework, and a detailed study was performed to show why our method worked well.

The results on the SWB and AISHELL-2 tasks verified the effectiveness of our methods. Under the cross-entropy criterion, our approach achieved relative word error rate reductions of 9.4% and 8.3% compared to those of the SI model on the SWB and AISHELL-2 tasks, respectively, with only a 3% increase in the decoding computation complexity. Under the cross-entropy criterion, our method yielded relative word error rate reductions of approximately 7.0% compared to that of the utterance-level d-vector-based speaker adaptation method with a fair comparison and 4.0% compared to that of the speaker-level d-vector-based speaker adaptation method, which uses extra adaptation data.

Future work will evaluate the performance when extra adaptation data are available, and we will also evaluate the performance of language or channel adaptive training with MANs. Experiments on larger tasks will also be considered.

REFERENCES

[1] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[2] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, 2014, pp. 338–342.

[3] O. Abdel-Hamid, A. Rahman Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2012, pp. 4277–4280.

[4] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. Lecun, "Very deep multilingual convolutional neural networks for LVCSR," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 4955–4959.

[5] J. Neto *et al.*, "Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system," in *Proc. EUROSPEECH*, 1995, pp. 2171–2174.

[6] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. D. Mori, "Linear hidden transformations for adaptation of hybrid ANN/HMM models," *Speech Commun.*, vol. 49, no. 10, pp. 827–835, 2007.

[7] B. Li and K. C. Sim, "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems," in *Proc. Interspeech*, 2010, pp. 526–529.

[8] 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 *Proc. IEEE Spoken Lang. Technol.*, 2012, pp. 366–369.

[9] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proc. IEEE Spoken Lang. Technol.*, 2015, pp. 171–176.

[10] 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 *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 5711–5715.

[11] C. Liu and Y. Wang, "Investigations on speaker adaptation of LSTM RNN models for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 5020–5024.

[12] Z. Wang and D. Wang, "Unsupervised speaker adaptation of batch normalized acoustic models for robust ASR," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 4890–4894.

[13] L. Samarakoon and K. Sim, "Factorized hidden layer adaptation for deep neural network based acoustic modeling," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 24, no. 12, pp. 2241–2250, Dec. 2016.

[14] L. Samarakoon, B. Mak, and K. Sim, "Learning effective factorized hidden layer bases using student-teacher training for LSTM acoustic model adaptation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 5954–5958.

[15] 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 *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 6359–6363.

[16] 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 *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 7893–7897.

[17] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 7947–7951.

[18] Z. Huang, J. Li, S. M. Siniscalchi, I. Chen, and J. Wu, "Rapid adaptation for deep neural networks through multi-task learning," in *Proc. Interspeech*, 2015, pp. 3625–3629.

[19] Z. Meng, J. Li, and Y. Gong, "Adversarial speaker adaptation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 5721–5725.

[20] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. IEEE Autom. Speech Recognit. Understanding*, 2011, pp. 24–29.

[21] P. Karanasou, Y. Wang, M. J. Gales, and P. C. Woodland, "Adaptation of deep neural network acoustic models using factorised i-vectors," in *Proc. Interspeech*, 2014, pp. 2180–2184.

[22] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proc. IEEE Autom. Speech Recognit. Understanding*, 2013, pp. 55–59.

[23] Y. Miao, H. Zhang, and F. Metze, "Speaker adaptive training of deep neural network acoustic models using i-vectors," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 23, no. 11, pp. 1938–1949, Nov. 2015.

[24] P. Cardinal, N. Dehak, Y. Zhang, and J. Glass, "Speaker adaptation using the i-vector technique for bottleneck features," in *Proc. Interspeech*, 2015, pp. 2867–2871.

[25] 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 *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 7942–7946.

[26] S. Xue, O. Abdel-Hamid, H. Jiang, and L. Dai, "Direct adaptation of hybrid DNN/HMM model for fast speaker adaptation in LVCSR based on speaker code," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 6389–6393.

[27] 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 Trans. Audio, Speech Lang. Process.*, vol. 22, no. 12, pp. 1713–1725, Dec. 2014.

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

[29] Y. Zhao, J. Li, S. Zhang, L. Chen, and Y. Gong, "Domain and speaker adaptation for cortana speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 5984–5988.

[30] T. Tan, Y. Qian, M. Yin, Y. Zhuang, and K. Yu, "Cluster adaptive training for deep neural network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 4335–4329.

[31] C. Wu and M. J. Gales, "Multi-basis adaptive neural network for rapid adaptation in speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 4315–4319.

[32] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by back-propagation," in *Proc. JMLR Workshop. Conf. Proc.*, 2015, vol. 37, pp. 1180–1189.

[33] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *Proc. Neural Inf. Process. Syst.*, 2016, pp. 343–351.

[34] Z. Meng, J. Li, Y. Gong, and B. Juang, "Adversarial teacher-student learning for unsupervised domain adaptation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 5949–5953.

[35] Z. Meng *et al.*, "Speaker-invariant training via adversarial learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 5969–5973.

[36] G. Saon *et al.*, "English conversational telephone speech recognition by humans and machines," in *Proc. Interspeech*, 2017, pp. 132–136.

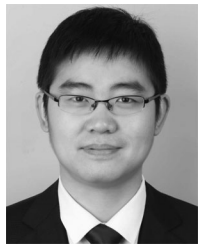
[37] Z. Meng, Z. Chen, V. Mazalov, J. Li, and Y. Gong, "Unsupervised adaptation with domain separation networks for robust speech recognition," in *Proc. IEEE Autom. Speech Recognit. Understanding*, 2017, pp. 214–221.

[38] Y. Zhao, J. Li, J. Xue, and Y. Gong, "Investigating online low-footprint speaker adaptation using generalized linear regression and click-through data," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 4310–4314.

[39] J. Pan, D. Liu, G. Wan, J. Du, Q. Liu, and Z. Ye, "Online speaker adaptation for LVCSR based on attention mechanism," in *Proc. Asia-Pacific Signal Inf. Process. Assoc.*, 2018, pp. 183–186.
[40] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.
[41] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 19, no. 4, pp. 788–798, May 2011.
[42] E. Variani, X. Lei, E. McDermott, I. Lopez, Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 4052–4056.
[43] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.
[44] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.
[45] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. Neural Inf. Process. Syst.*, 2015, pp. 577–585.
[46] R. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
[47] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. Conf. Comput. Vision Pattern Recognit.*, 2015, pp. 815–823.
[48] J. Sanu, M. Xu, H. Jiang, and Q. Liu.,"Word embeddings based on fixed-size ordinally forgetting encoding," in *Proc. Empirical Methods Natural Lang. Process.*, 2017, pp. 310–315.
[49] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[50] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3942–3951.
[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Comput. Vision Pattern Recognit.*, 2016, pp. 770–778.
[52] J. Godfrey, E. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 1992, pp. 517–520.
[53] J. Du, X. Na, X. Liu, and H. Bu, "Aishell-2: Transforming mandarin asr research into industrial scale," 2018, *arXiv:1808.10583*.
[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[55] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[56] D. S. Pallet, W. M. Fisher, and J. G. Fiscus, "Tools for the analysis of benchmark speech recognition tests," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 1990, pp. 97–100.
[57] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted MMI for model and feature-space discriminative training," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 2008, pp. 4057–4060.
[58] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 6664–6668.
[59] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *Comput. Speech. Lang.*, vol. 46, pp. 535–557, 2017.
[60] L. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.

Jia Pan received the B.S. and M.S. degrees in 2006 and 2009, respectively, from the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China, where he is currently working toward the Ph.D. degree. Since 2009, he has been with iFlytek Research on speech recognition and spoken dialogue systems. His current research interests include speech recognition and machine learning.

Genshun Wan received the B.S. and M.S. degrees in communication and information systems from Jiangsu University, Zhenjiang, China, in 2012 and 2015, respectively. He is currently working toward the Ph.D. degree with the University of Science and Technology of China, Hefei, China. Since 2015, he has been with iFlytek Research on speech recognition. His current research interests include speech recognition and machine learning.



Jun Du received the B.Eng. and Ph.D. degrees from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2004 and 2009, respectively. From 2004 to 2009, he was with iFlytek Speech Lab, USTC. During this period, he was an Intern twice for nine months in Microsoft Research Asia, Beijing, China. In 2007, he was also a Research Assistant for six months with the Department of Computer Science, University of Hong Kong. From July 2009 to June 2010, he worked with iFlytek Research on speech recognition. From July 2010 to January 2013, he joined MSRA as an Associate Researcher, working on handwriting recognition, OCR, and speech recognition. Since February 2013, he has been with the National Engineering Laboratory for Speech and Language Information Processing, USTC.

Zhongfu Ye received the B.E. and M.S. degrees from the Hefei University of Technology, Hefei, China, in 1982 and 1986, respectively, and the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 1995. He is currently a Professor with the University of Science and Technology of China, Hefei, China. His current research interests include statistical and array signal processing, and image processing.