Similarity Measurement of Segment-level Speaker Embeddings in Speaker Diarization

Weiqing Wang, Qingjian Lin, Danwei Cai, *Student Member, IEEE*, Ming Li, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a neural-network-based similarity measurement method to learn the similarity between any two speaker embeddings, where both previous and future contexts are considered. Moreover, we propose the segmental pooling strategy and jointly train the speaker embedding network along with the similarity measurement model. Later, this joint training framework is further extended to the target-speaker voice activity detection (TS-VAD), with only slight modification in the network architecture. Experimental results of the DIHARD II, DIHARD III and VoxConverse datasets show that our clustering-based system with the neural similarity measurement achieves superior performance to recent approaches on all three datasets. In addition, the segment-level TS-VAD method further improves the clustering-based results and achieves DER of 16.48%, 11.62% and 4.39% on the DIHARD II, DIHARD III and VoxConverse datasets, respectively.**

***Index Terms*—Speaker diarization, speaker verification, target-speaker voice activity detection**

I. Introduction

SPEAKER DIARIZATION is a process that addresses the problem of "who spoke when" in multi-speaker audio data [1, 2, 3]. A conventional speaker diarization system consists of several independent modules, as shown in Fig. 1. First, a voice activity detection (VAD) module removes the non-speech region [4, 5, 6, 7]. Second, the detected speech regions are partitioned into short segments by speaker change detection (SCD) [8, 9], uniform segmentation [10], or agglomerative hierarchical clustering-based (AHC) segmentation [11]. Then, a speaker embedding like the i-vector [12] or x-vector [13] can be extracted from these segments and the segment-wise similarity can be measured by the cosine distance or probabilistic linear discriminate analysis (PLDA). Finally, these segments are grouped by the speaker identity using different clustering method, e.g., K-means [14], AHC [15], spectral clustering [16, 17] and path integral clustering [18].

As the conventional modular speaker diarization system assumes that each segment only contains one speaker, it cannot handle an overlapped speech where two or more speakers appear simultaneously. Therefore, some pre- and post-processing modules are proposed, focusing on the overlapped speech to improve the performance further. For pre-processing, the speech separation has shown a promising performance in both multi-channel [19, 20] and single-channel [11] speaker diarization tasks. Although the single-channel blind source separation may produce "leakage" of speech, this can be solved by the leakage filtering method [11]. For post-processing, overlap detection [21] can slightly improve the performance by assigning the overlapped region with the closest two speakers. Recently, many researchers have focused on multi-speaker speech processing, where the models can detect the information of a specific speaker given the corresponding acoustic footprint. This method has been applied to many different tasks, including target speech recognition [22], target speech extraction [23, 24], and target speech detection [25]. Inspired by these works, Medennikov et al. [26] proposed the target-speaker voice activity detection (TS-VAD) to extract the frame-level posterior probability given the corresponding target speaker's i-vector, which afforded great success on a highly overlapped speech signal. Recently, Wang et al. [27] further employed this i-vector-based TS-VAD method on a multi-domain dataset and achieved an excellent performance. The model is separately trained for each domain, making this a domain-dependent method.

In addition, other pre- and post-processing modules that are not for overlap handling can also improve the diarization performance. Pre-processing methods like speech enhancement [28, 29] and dereverberation [30] have shown satisfying performance in some situations, especially for a multi-channel signal [31]. For post-processing, resegmentation can refine the clustering-based diarization result [32]. Stolcke et al. [33] proposed Diarization Output Voting Error Reduction (DOVER)

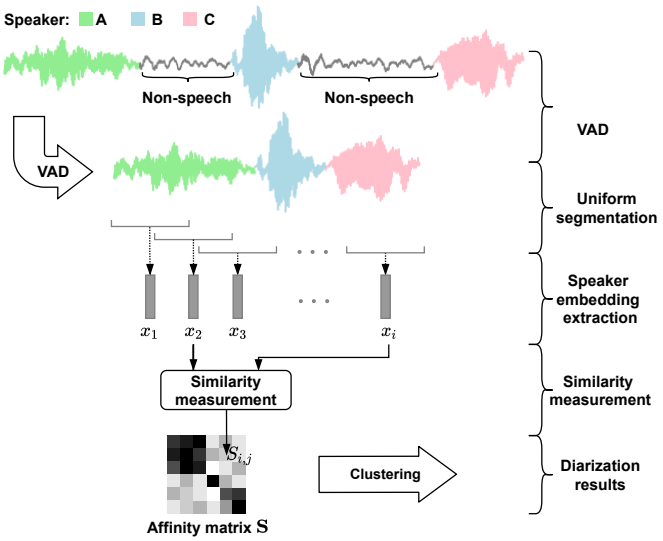


Fig. 1. An example of the conventional speaker diarization system

Weiqing Wang, Danwei Cai and Ming Li are with the Department of Electrical and Computer Engineering at Duke University. Ming Li is also with Data Science Research Center at Duke Kunshan University. Qingjian Lin was with the Data Science Research Center at Duke Kunshan University.
Corresponding author: Ming Li, E-mail: ming.li369@duke.edu

to fuse several different diarization systems. Moreover, the modified DOVER [11] and DOVER-lap [34] have shown a reduction of DER in many different challenges.

Although, each of the modules mentioned above is necessary for a good performance of the diarization system, the most important part is still the speaker embedding extraction. In recent years, many clustering-based speaker diarization systems made use of speaker embeddings extracted from uniformly segmented utterances. Nevertheless, this method also has potential limitations:

1) In most clustering-based methods, only local similarities with pairs of embeddings are considered, where each embedding is extracted only from a short segment. These speaker embeddings are not necessarily speaker homogeneous, which makes it difficult to cluster the speaker identities. In addition, the sequential context information is completely ignored during clustering, but the contents of conversations are usually highly structured.
2) CNN-based architectures like ResNet are currently popular for embedding extraction, but there is the problem with the zero-padding in the case of diarization that people usually ignore. In such case, the frame-level speaker information near the segment boundary contains less information than the central frame because the padding zeros near the borders contain no information [35]. This phenomenon is common when the segment is very short as is the case in the speaker diarization task.

Although some post-processing methods can incorporate the temporal continuity and smooth the transition among clusters, the improvements are relatively moderate [18]. In addition, Wisniewksi et al. [36] employed structured prediction for online speaker diarization, but only the structural information from the forward direction was considered.

In this paper, a two-stage diarization system is employed, where the first stage is a clustering-based method and the second stage is a TS-VAD method that refines the results of the first stage. For the first stage, we propose a neural network-based method in place of PLDA or cosine similarity to measure the similarity between speaker embeddings in the conventional clustering-based method. Unlike many AHC-based methods that ignore the temporal information, we use the BiLSTM or self-attention to extract the similarity between a sequence of pairs of speaker embeddings. In addition, for any dataset, our method only needs a fixed threshold as a hyperparameter for the spectral clustering to produce the final diarization results, whereas the AHC needs to carefully tune the threshold for different datasets and different speaker embedding extractors. We also jointly train the embedding extractor and the model of similarity measurement to obtain better performance. For the second stage, an x-vector-based TS-VAD method is proposed to find the overlapped speech while making use of the results from the first stage. We extract an x-vector for each speaker based on the clustering produced in the first stage, and detect the voice activities for each speaker given the corresponding x-vector. As TS-VAD can find the voice activities of the target speaker even for overlapped speech regions, it can always produce a better result than the clustering-based method if the number of speakers is correctly estimated in the first stage. Compared with the i-vector-based TS-VAD method [27], which needs to separately train a model for each domain, the x-vector-based TS-VAD model can be directly trained on multi-domain data and produce better performance, making it more robust for difficult data. Therefore, the proposed TS-VAD method is a domain-independent strategy by employing a more discriminatively trained x-vector, whereas the i-vectors are more domain-dependent.

This paper extends our previous works [16, 17] on similarity measurement using long-short term memory (LSTM) and self-attention. The new contributions from this work are:

- Joint training of the speaker embedding network and the network for measuring similarity between embeddings using a speaker diarization objective.
- Successfully applying the x-vector to the TS-VAD task, reformulating the TS-VAD as an embedding concatenation-based method, and introducing the pooling strategy to TS-VAD at the segment level.
- Applying a two-speaker TS-VAD method for only overlap detection, which also shows good performance.

The rest of this paper is organized as follows. Section II details the proposed similarity measurement with segmental pooling. Section III introduces the segment-level TS-VAD. Section IV provides the experimental details. Section V provides the experimental results. Section VI provides the conclusion of this paper.

II. Similarity Measurement

In general, a conventional modular speaker diarization system contains VAD, speaker embedding extraction, similarity measurement, and clustering. In this section, we mainly focus on DNN-based speaker embedding extraction and similarity measurement under the conventional modular speaker diarization framework.

A. Speaker Embedding Extraction with Segmental Pooling

This paper follows the ResNet-style x-vector introduced in [37, 38]. As shown in Fig. 2, the front-end extractor takes the acoustic features as the input and produces a CNN feature map. Later, a pooling layer aggregates the temporal information in this CNN feature map over time and generates a fixed-length representation. Finally, the penultimate layer produces the utterance-level representation.This entire network is trained for speaker classification, where the classes correspond to the training speakers.

As the pooling layer aggregates the temporal information into a fixed-length representation, the design of the pooling layer can directly influence the network performance. Several pooling layers have been explored under this ResNet-based framework and proved to be effective, including temporal average pooling (TAP), temporal statistical pooling (TSP), self-attentive pooling (SAP), and learnable dictionary encoding (LDE) layers [39, 40, 37].

In the speaker diarization task, the input signals are commonly broken into several short segments by uniform segmentation with a fixed window length and window shift. Next, the speaker embeddings are extracted from these segments to represent the identity of the speaker, as Fig. 3a shows. However, the receptive field of most speaker recognition networks is longer than the length of the segments, e.g., the receptive field of the ResNet34 used in this paper is over 2 seconds for each frame [35]. This means that the CNN feature map produced by the front-end extractor contains less information when the frames are close to the borders for short segments, whereas the central frames contain more information. This is caused by the zero-padding and ignoring the actual context.

Therefore, we propose segmental pooling (SP) to perform the uniform segmentation and pooling operation in a single step, where the segmentation is directly performed on the CNN feature map rather than the audio waveform. First, the audio signal is split into different speech segments based on the oracle VAD label, and we extract the CNN feature map for each speech segment with the front-end ResNet34. Therefore, the zero-padding is applied only at the boundaries of these (often relatively long) segments defined by VAD. As Fig. 3b shows, we then uniformly split the CNN feature map with a fixed window length and window shift, and we feed these segmented feature maps into the pooling layer to obtain a fixed-length representation for each segment. Finally, the feed-forward network generates the speaker embedding. Note that the parameters of the front-end extractor and the feed-forward network are not changed. We only perform segmentation on the feature maps before the pooling layer rather than on the input signals. Note that the speaker embeddings extracted this way are not affected by any zero-padding unless they are close to the boundary of the original VAD-based segment. As a consequence, each such embedding effectively "sees" a larger temporal context through the ResNet34 receptive field (3.72s in our experiments) as compared to the embeddings extracted from segmented speech signal (1.28s). In fact, except for the VAD-based segment boundaries where the zero-padding matters, the segmental pooling could be simulated by switching off the zero-padding over time and extracting embeddings from 3.72s speech signal segments.

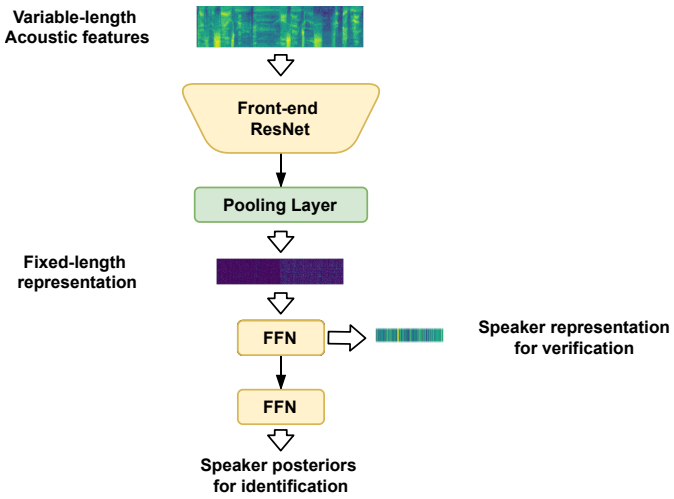


Fig. 2. The architecture of the speaker embedding network

TABLE I
THE NETWORK ARCHITECTURE FOR EMBEDDING EXTRACTION, WHERE $\mathbf{C}$(KERNAL SIZE, STRIDE) DENOTES THE CONVOLUTIONAL LAYER, $[\cdot]$ DENOTES THE RESIDUAL BLOCK; L RELATES TO THE DURATION OF THE SPEECH AND L RELATES TO THE NUMBER OF FREQUENCY BINS OF THE MEL SPECTROGRAM.

Layer	**Output Size**	**Structure**
Input	$1 \times F \times L$	-
Conv1	$32 \times F \times L$	$\mathbf{C}(3 \times 3, 1)$
Residual layer 1	$32 \times F \times L$	$\begin{bmatrix}\mathbf{C}(3 \times 3, 1)\\ \mathbf{C}(3 \times 3, 1)\end{bmatrix} \times 3$
Residual layer 2	$64 \times \frac{F}{2} \times \frac{L}{2}$	$\begin{bmatrix}\mathbf{C}(3 \times 3, 2)\\ \mathbf{C}(3 \times 3, 1)\end{bmatrix} \begin{bmatrix}\mathbf{C}(3 \times 3, 1)\\ \mathbf{C}(3 \times 3, 1)\end{bmatrix} \times 3$
Residual layer 3	$128 \times \frac{F}{4} \times \frac{L}{4}$	$\begin{bmatrix}\mathbf{C}(3 \times 3, 2)\\ \mathbf{C}(3 \times 3, 1)\end{bmatrix} \begin{bmatrix}\mathbf{C}(3 \times 3, 1)\\ \mathbf{C}(3 \times 3, 1)\end{bmatrix} \times 5$
Residual layer 4	$256 \times \frac{F}{8} \times \frac{L}{8}$	$\begin{bmatrix}\mathbf{C}(3 \times 3, 2)\\ \mathbf{C}(3 \times 3, 1)\end{bmatrix} \begin{bmatrix}\mathbf{C}(3 \times 3, 1)\\ \mathbf{C}(3 \times 3, 1)\end{bmatrix} \times 2$
Pooling layer	512	Statistics pooling
Embedding	128	Fully connected layer

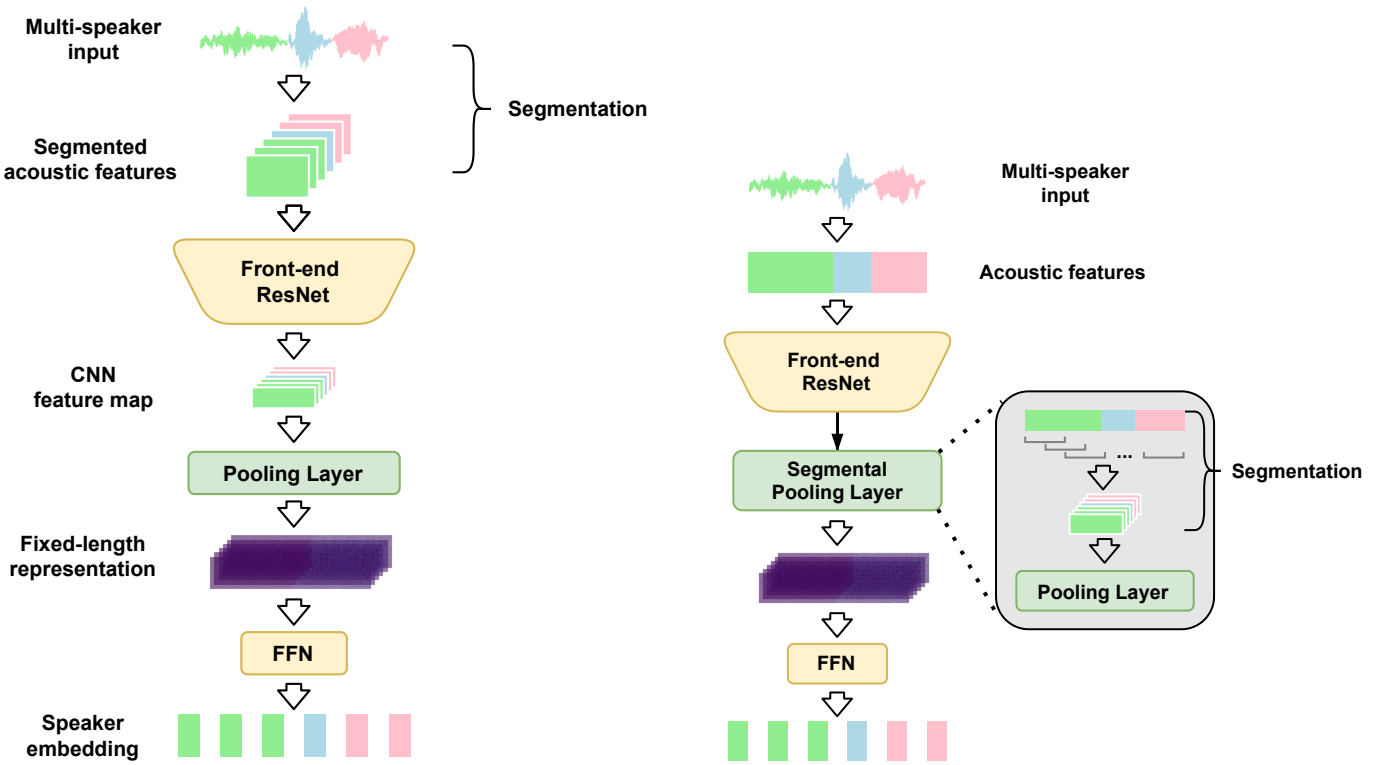
We employ a segmental statistical pooling (SSP) layer after the segmentation [39]. Consider an acoustic feature matrix $\mathbf{A} \in \mathbb{R}^{F \times L}$. The front-end extractor f_{fe} can extract a CNN feature map $\mathbf{M} \in \mathbb{R}^{C \times H \times T}$, where F and L are the dimensions and length of the acoustic feature, respectively; C is the number of channels, and H and T are the height and width of the CNN feature map, respectively. Usually, $H = \frac{F}{8}$ and $T = \frac{L}{8}$ in our ResNet34-based front-end extractor; thus, the CNN feature map can be considered as a downsampling from the acoustic feature. Next, we uniformly split the CNN feature map into short segments with a length T=16 frames and a shift 8 frames. The dimensions of each segment feature map $\mathbf{M}_i$ are $C \times H \times T^{'}$, where $T^{'}$ is the segment length. Finally, we perform statistical pooling on each channel of the segmented CNN feature map. Let $\mathbf{M}_i^c \in \mathbb{R}^{H \times T^{'}}$ denote the feature map of the c-th channel; the SSP layer aggregates the two-dimensional feature map as follows:

$$\mu_i^c = \frac{1}{HT^{'}} \sum_{h=1}^{H} \sum_{t=1}^{T^{'}} \mathbf{M}_{i,h,t}^c, \boldsymbol{\mu}_i = [\mu_i^1, ..., \mu_i^C]^\intercal \tag{1}$$

$$\sigma_i^c = \sqrt{\frac{1}{HT^{'}} \sum_{h=1}^{H} \sum_{t=1}^{T^{'}} (\mathbf{M}_{i,h,t}^c - \boldsymbol{\mu}_i)^2}, \boldsymbol{\sigma}_i = [\sigma_i^1, ..., \sigma_i^C]^\intercal, \tag{2}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are the mean and standard deviation vector of the i^{th} segment, respectively, and both of them are C-dimensional vectors. Finally, the pooling result for the i^{th} segmented CNN feature map is the concatenation of the $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$.

B. Similarity Measurement



(a) The uniform segmentation with speaker embedding extraction in the conventional speaker diarization system

(b) The segmental pooling strategy, where the segmentation is performed after the front-end feature extractor

Fig. 3. Performing segmentation in different stages of speaker embedding extraction

In the conventional modular speaker diarization system, the similarities between different segments are computed in a pairwise manner [21]. However, conversations between different speakers are usually highly structured. To take the sequential information into consideration, we propose a similarity measurement method colorredusing a recurrent neural network, where both previous and following segments are considered to improve the performance. In this section, we introduce our work using BiLSTM and self-attention for the supervised similarity measurement, which is the baseline in our experiments.

1) BiLSTM: The LSTM network architecture was originally developed in [41], but the early version of LSTM can only capture the previous context. Later, Schuster et al. [42] proposed BiLSTM, which utilizes the information from both the previous and future context. BiLSTM has shown success in automatic speech recognition (ASR) [43] and speech synthesis [44]. We also employ the BiLSTM to measure the similarity between two speaker embeddings with the previous and future context included.

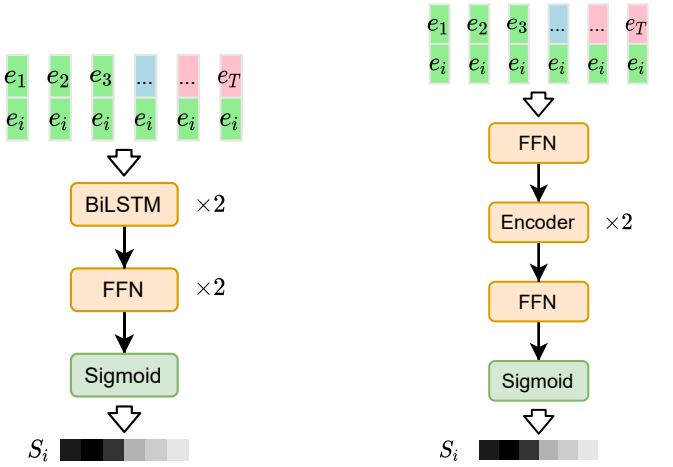
Considering an affinity matrix $\hat{\mathbf{S}} \in \{0,1\}^{T\times T}$ for an embedding sequence $\mathbf{E} = [\mathbf{e}_1^\intercal, \mathbf{e}_2^\intercal, ..., \mathbf{e}_T^\intercal]$, where $\hat{\mathbf{S}}_{i,j}$ is 1 when $\mathbf{e}_i$ and $\mathbf{e}_j$ are from the same speaker and 0 otherwise, our goal is to predict $\mathbf{S}_{i,j}$ so as to generate a complete affinity matrix. We concatenate the speaker embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ as the input of the network, for example $\left[\mathbf{e}_i^\intercal, \mathbf{e}_j^\intercal\right]^\intercal$. As the BiLSTM can deal with the sequential data, we can obtain each row $\mathbf{S}_i$ in one pass through the sequence:

$$\mathbf{S}_i = [\mathbf{S}_{i,1}, \mathbf{S}_{i,2}, ..., \mathbf{S}_{i,T}] = f_l(\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_i\end{bmatrix}, \begin{bmatrix}\mathbf{e}_2\\ \mathbf{e}_i\end{bmatrix}, ..., \begin{bmatrix}\mathbf{e}_T\\ \mathbf{e}_i\end{bmatrix}), \quad (3)$$

where f_l is the BiLSTM network. The complete affinity matrix $\mathbf{S}$ can be obtained by stacking all rows $\mathbf{S}_i$ together.

The architecture of the BiLSTM network is the same as the network in [16]. As shown in Fig. 4a, a two-layer BiLSTM takes a sequence of speaker embeddings as input, and two fully-connected layers with a sigmoid function predict the probability that two embeddings correspond to the same speaker, which is one row of the affinity matrix. As will be pointed out in Section III, this process is similar to the target-speaker voice activity (TS-VAD) method [26], where the target speaker embedding is the average of many "target segment embeddings" $\mathbf{e}_i$ from the same speaker.

2) Self-attention: As an alternative to the BiLSTM-based architecture, we also use the Transformer-based model [45] for the similarity measurement. The inputs and training objectives are the same as for the BiLSTM-based model. BiLSTM layers are replaced with one fully connected layer followed by two Transformer self-attention encoder layers. Unlike in [45], we do not use the positional encoding as it resulted in performance degradation. Finally, fully connected layers with a sigmoid function predict the similarities. Fig. 4b shows the architecture of the self-attention-based model. The encoder contains 2 layers, and each layer contains 2 heads with 1024 attention units for each head.



(a) The BiLSTM network (b) The self-attentive network

Fig. 4. The architectures of the networks for similarity measurement

C. Post-processing for Predicted Affinity Matrix

1) Affinity matrix partitioning: In practice, the embedding sequence can be long, which requires a large amount of memory. With the LSTM model, we cannot handle long sequences because of memory limitations. Therefore, we split the embedding sequences into several fixed-length overlapped short sub-sequences, and the affinity matrix is also broken into several blocks. Specifically, if we have N sub-sequences, there should be N^2 blocks in the affinity matrix, where the i$^{\text{th}}$ and j$^{\text{th}}$ sub-sequences can form the block in the corresponding position. After we obtain all blocks, we merge all blocks by placing them in the corresponding location, where the overlapped regions are averaged to produce the complete affinity matrix. This way, we can process each affinity block as a mini-batch and then combine these blocks to form the complete affinity matrix.

2) Affinity matrix refinement: As the affinity matrix is processed row by row, it is not symmetric and contains a lot of noise. Hence, the affinity matrix can be smoothed, symmetrized, and enhanced as follows [14]:

- Symmetrization: $\mathbf{Y}_{i,j} = \max(\mathbf{S}_{i,j}, \mathbf{S}_{j,i})$
- Diffusion: $\mathbf{Y} \leftarrow \mathbf{Y}\mathbf{Y}^\intercal$
- Row-wise max normalization: $\mathbf{S}^{'}_{i,j} = \frac{\mathbf{Y}_{i,j}}{\max \mathbf{Y}_i}$

where $\mathbf{Y}_i$ is the i^{th} row of matrix $\mathbf{Y}$.

3) Spectral clustering: After we obtain the refined affinity matrix $\mathbf{S}^{'}$, we employ spectral clustering to obtain the diarization results as mentioned in [46] as follows:

1) Construct the affinity matrix $\mathbf{S}^{'} \in \mathbb{R}^{n \times n}$ and set all diagonal entries to 0.
2) Generate the normalized Laplacian matrix $\mathbf{L}_{\text{norm}}$:

$$\mathbf{L} = \mathbf{D} - \mathbf{S}^{'},$$

$$\mathbf{L}_{\text{norm}} = \mathbf{D}^{-1}\mathbf{L},$$

where $\mathbf{L}$ is the Laplacian matrix, $\mathbf{D}$ is a diagonal matrix and and $\mathbf{D}_{i,i} = \sum_{j=1}^{n} \mathbf{S}^{'}_{i,j}$.
3) Compute the eigenvalues λ and corresponding eigenvectors $\mathbf{u}$ of $\mathbf{L}_{\text{norm}}$.
4) Compute the number of cluster k. In experiments, we employ a threshold β and find the number of eigenvalues lower than β as k.
5) Find the largest k eigenvalues $\lambda_1, ..., \lambda_k$ and corresponding eigenvectors $\mathbf{u}_1, ..., \mathbf{u}_k$.
6) Let $\mathbf{U} \in \mathbb{R}^{n \times k}$ containing $\mathbf{u}_1, ..., \mathbf{u}_k$ as columns.
7) Cluster the row vectors of $\mathbf{U}$ by the k-means algorithm.

D. Data Augmentation

If only given limited data, our similarity measurement model is easily overfitted to some particular region in the speaker embedding space. One common solution for speech data augmentation is to add background noise and reverberation to the training data, which can increase the model robustness for noisy data. However, this method cannot generate data for new unseen speakers, and the model will lack generalization ability with only hundreds of training speakers. For this purpose, we employ an embedding-level data augmentation strategy, which can rotate all the speaker embeddings to another region of the embedding space without changing the inter-class and intra-class distance. This augmentation method was first proposed in [47].

Assume that the speaker embedding sequence $\mathbf{E} = [\mathbf{e}_1^\intercal, \mathbf{e}_2^\intercal, ..., \mathbf{e}_T^\intercal] \in \mathbb{R}^{D \times T}$ is L2-normalized for each $\mathbf{e}_i$. The D-dimensional speaker embeddings can be rotated by an orthonormal transformation:

$$\mathbf{E}' = \mathbf{R}\mathbf{E} = [\mathbf{R}\mathbf{e}_1^\intercal, \mathbf{R}\mathbf{e}_2^\intercal, ..., \mathbf{R}\mathbf{e}_T^\intercal], \tag{4}$$

where $\mathbf{R} \in \mathbb{R}^{D \times D}$ are random orthonormal basis. Fig. 5 shows an example of the random orthonormal transformation.

After performing the on-the-fly data augmentation on the speaker embeddings, we can generate large-scale samples that have the potential to span the whole speaker embedding space with only limited real data. The model can focus on learning the difference between speaker embeddings instead of remembering the speaker information. Therefore, we can reduce the overfitting risk by applying this augmentation method. We only perform the data augmentation when training the similarity measurement model on the training set instead of fine-tuning to the development dataset as described in Sec.

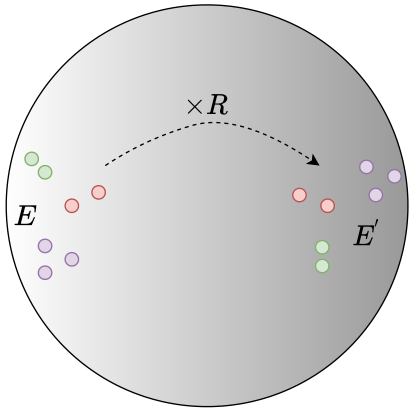


Fig. 5. An example of the random orthonormal transformation on the speaker embedding sphere.

IV-A2. The reason is that the model cannot adapt to some specific region of the development dataset if we randomly rotate the speaker embedding space when fine-tuning.

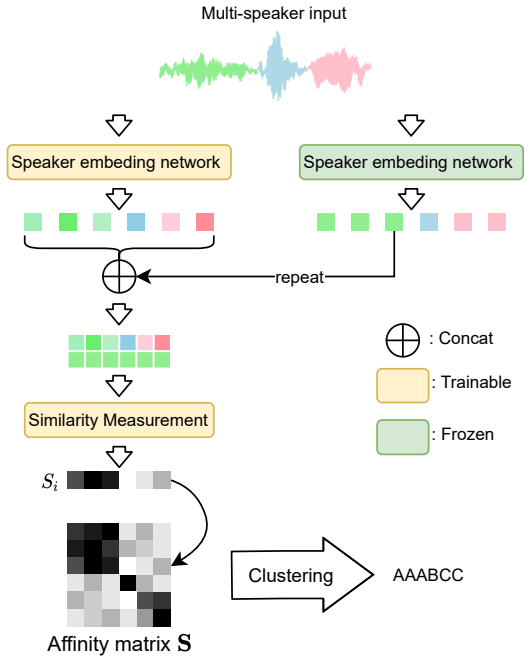


Fig. 6. The framework of joint training

E. Joint Training

Although the embedding-level data augmentation can improve the generalization ability of the similarity measurement model and avoid overfitting, we still like to fine-tune the model to a specific domain by training with a lower learning rate on a domain-specific data. In our previous work [16, 17], we only fine-tuned the model for the similarity measurement but kept the speaker embedding network unchanged. As Fig. 6 shows, the entire network consists of two sub-models: the front-end speaker embedding network and the back-end similarity measurement model, where the speaker embedding network is the same as that in Fig. 3b, and the similarity measurement model is the same as that in Fig. 4a.

The joint training framework is only employed in the fine-tuning stage. Therefore, we do not perform the data augmentation for the embeddings. During the fine-tuning stage, the networks in the yellow block in Fig. 6 are jointly trained and updated, whereas the parameters of the green block in Fig. 6 are frozen.

Unlike the conventional modular speaker diarization system, in which each module is optimized individually, the speaker embedding model in our framework is optimized with the objective of the similarity measurement, which connects two modules and improves the system performance. Under this joint training framework, the right speaker embedding network is a pre-trained segment-level speaker embedding extractor, and the left is a trainable segment-level speaker embedding extractor. If we employ the right network to extract an embedding of a known speaker, this model becomes a target-speaker voice activity detector (TS-VAD), as Fig. 7 shows.

III. Segment-Level TS-VAD

In the original TS-VAD [26], the model outputs a sequence of probabilities of the target speaker presence for each time frame, which is a frame-level method. Although the frame-level TS-VAD system has shown good performance on many different datasets [26], sometimes we do not need such a high resolution in time. The information about a single speaker can be aggregated at the segment level and thereby reduce the computational complexity. It is difficult for a back-end network to learn from a long sequence, and frame-level prediction also contains a lot of noise.

A. Segment-level TS-VAD

In Eq. 3, $\mathbf{e}_i$ can also be considered as a segment-level target speaker embedding, where the speaker identity is unknown. Therefore, we need to perform clustering on the affinity matrix in the first stage to obtain the initial diarization results. In the second stage, we extract "target speaker embedding $\mathbf{e}_t$" for each speaker identified in the first stage. Similarly to Eq. 3, we evaluate the similarity between each speech segment and each target speaker, which can be interpreted as target-speaker voice activity detection (TS-VAD):

$$\mathbf{Q}_t = [\mathbf{Q}_{t,1}, \mathbf{Q}_{t,2}, ..., \mathbf{Q}_{t,T}] = f_l(\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_t\end{bmatrix}, \begin{bmatrix}\mathbf{e}_2\\ \mathbf{e}_t\end{bmatrix}, ..., \begin{bmatrix}\mathbf{e}_T\\ \mathbf{e}_t\end{bmatrix}), \quad (5)$$

where $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_T$ are the segment embeddings extracted from the pre-trained speaker embedding model. The output $\mathbf{Q}_t$ can represent the features that contain the information of the specific speaker t. Later, these decision states from different

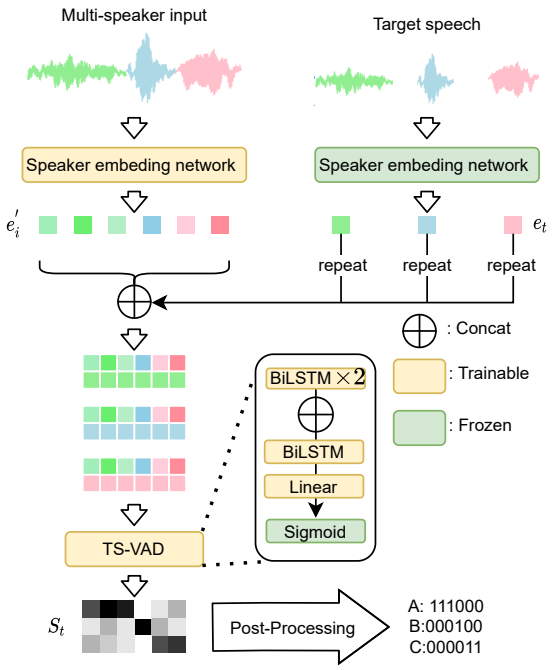


Fig. 7. The framework of TS-VAD

speakers are combined for the subsequent layers to further extract the binary decisions for all speaker.

Although Eq. 5 defines the general formulation of TS-VAD, we cannot directly perform TS-VAD with the segment embeddings from a pre-trained model $\mathbf{e}_1, ..., \mathbf{e}_T$ for several reasons:

1) The speaker embedding model is trained using single speaker utterances (i.e. without any overlapped speech) and therefore, the TS-VAD model may not perform well if the target speaker is in an overlapped speech region.
2) While the pre-trained speaker embedding model can extract good quality embeddings from the whole utterances, it may not be able to perform well for short-duration segments.

To improve the multi-speaker information extraction, we keep the target speaker embedding $\mathbf{e}_t$ and replace the segment embedding $\mathbf{e}_i$ with $\mathbf{e}_i^{'}$, where $\mathbf{e}_i^{'}$ comes from the jointly optimized speaker embedding network, as Fig. 7 shows. The networks for embedding extraction are the same as that of the joint training framework and the only difference is the architecture of the similarity measurement model.

Fig. 7 shows the architecture of our SP-based TS-VAD system, which is similar to the joint training framework in Section II-E. First, a pre-trained ResNet extracts the target speaker embedding $\mathbf{e}_t$ and another trainable ResNet extract the segment embeddings by SP. For training, the target speaker embedding is extracted from all non-overlapping speech of the target speaker. Next, the target speaker embedding and the segment embeddings are concatenated as Eq. 5 shows. We follow the method introduced in [26], where N target speaker embeddings are extracted simultaneously and concatenated with the segment embeddings separately, as Fig. 7 shows. Finally, two BiLSTM block followed by a fully connected layer predict the N speaker presence probabilities. The first block containing 2 BiLSTM layers separately processes the concatenated embeddings for each target speaker and the second combines the output from the first BiLSTM block. The hidden size is 128 for all BiLSTM layers. The fully-connected layer with sigmoid function maps the output of the second BiLSTM layer to the presence probability for each of the N speakers. We obtain the final diarization results by thresholding on the outputs of the TSVAD model, which is the post-processing shown in Fig 7.

During the training stage, the target speaker embedding is obtained from all the non-overlapping speech according to the ground-truth label. During the inference stage, we can only extract the target speaker embedding from clustering-based results, and the overlapped speech is also included as the clustering-based method cannot find the overlap. In addition, as the target speech is long for some recordings, we have to break up the speech signal into several small chunks, separately extract the speaker embedding for each chunk, and finally average these speaker embeddings.

For training, we extract the speaker embedding for each chunk, so that we have several different speaker embeddings for one speaker. Next, for each speaker in every batch, we randomly select no more than 10 speaker embeddings and average these embeddings as the target-speaker embedding for TS-VAD training. This can provide a more diverse target-speaker embedding in each batch and can improve the robustness of the model. And we imitate this process in the inference stage. We also tried only extracting one embedding for each speaker for inference, and the results are slightly worse.

B. Segment-level TS-VAD as Overlap Detection

Although TS-VAD can detect the overlapped speech and reduce the missed speaker error, false alarms may increase. Therefore, DER improvement from the overlap detection is moderate. To tackle this problem, we can update the clustering-based results only in the overlapped speech regions. First, we extract the target speaker embedding for each speaker according to the clustering-based results. Next, we select two target speaker embeddings and use a TS-VAD model to find the speech regions for each of these two speakers. Finally, we only update the overlapped speech region between these two speakers. We iteratively select all combinations of two speakers and update the overlapped regions in the clustering-based results. The architecture of this TS-VAD model is the same as that shown in Fig 7, except that the output dimension is 2, in that it only predicts the binary decisions for two speakers. During the inference stage, we only select several of the most talkative speakers and discard other speakers for two reasons: (1) Speakers with a long speech time are more likely to have overlapped speech; (2) We reduce the time of inference, as we only run the inference $\binom{\#\text{selected spk}}{2}$ times for each recording.

Unlike previous overlap detection methods that first find the overlapped regions and then assign the closest two speakers to these regions [21], the TS-VAD-based overlap detection can get the overlapped regions between any two speakers, which is more accurate. In addition, we do not need to preset the number of speakers based on the maximum number of the speakers in the dataset as the original TS-VAD does. Therefore, this TS-VAD overlap detector is more flexible. However, a limitation also exists. This method is not so efficient, as it needs to iteratively evaluate the TS-VAD many times depending on the number of speaker pairs. That is why we only select several of the most talkative speakers for the inference.

IV. Experimental Setup

We perform all experiments with oracle VAD. All non-speech regions are removed in the training, development and evaluation data.

A. Dataset and Evaluation Metric

1) Data Simulation: Most of the current datasets for diarization tasks are not difficult enough for multi-speaker learning with no more than 10% overlapped speech. Although the DIHARD dataset is difficult, it is not enough to train a neural network with only tens of hours of data. Hence, data simulation is important to train a model with good generalization ability.

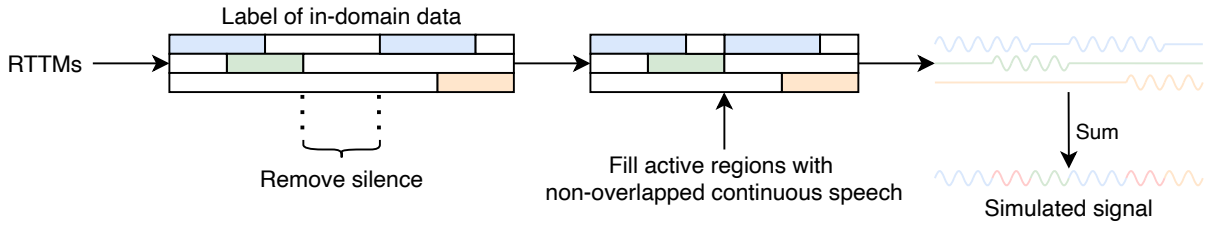


Fig. 8. The data simulation process

We generate the training data using the ground truth labels of the DIHARD development dataset. We first extract the label matrix $\mathbf{L} \in \{0,1\}^{N\times T}$ for each recording in the DIHARD development dataset, where N is the number of speakers and T is the length of an utterance with the non-speech regions removed. Next, according to the labels of each speaker, we fill the active regions with a speech of a single speaker from the LibriSpeech or DIHARD development dataset. Finally, we sum all the speech segments to produce one simulated recording. This simulation strategy can generate a huge amount of data similar to the DIHARD development dataset, but the speakers are different. Fig. 8 shows the process of data simulation.

For a single-domain dataset like LibriSpeech, we can directly generate the data with the above simulation strategy. However, if we use the DIHARD development dataset as an audio resource for simulation, the speakers in each simulated recording should come from the same domain. Otherwise, the model may learn to classify the domain information instead of the speaker identity. Finally, all of the simulated data from different domains are used for training the TS-VAD model.

2) Dataset for Training:

- Voxceleb 1 & 2 [48]: For speaker embedding model pre-training.
- AMI [49], ICSI [50], ISL (LDC2004S05), NIST (LDC2004S09), SPINE1&2 (LDC2000S87, LDC2000S96, LDC2001S04, LDC2001S06, LDC2001S08): For similarity measurement pre-training in Section II-B and II-E.
- LibriSpeech [51]: For 16kHz data simulation and TS-VAD pre-training.
- Switchboard [52] & NIST SRE 2004, 2005, 2006, 2008: For 8kHz telephone data simulation and TS-VAD pre-training.
- MUSAN & RIRs [53]: For waveform data augmentation.

3) Dataset for Finetuning / Evaluation: We employ the VoxConverse [54], DIHARD II [55], and DIHARD III dataset [56] for evaluation. Both the DIHARD II and DIHARD III datasets contain 11 domains , and VoxConverse is a single-domain dataset. Almost all domains contain 16kHz data, except the conversational telephone speech (CTS) data in DIHARD III is originally in 8 kHz. For each dataset, the corresponding development dataset is used as the fine-tuning dataset, and the details will be discussed in subsequent sections for each task.

4) Evaluation Metric: We use the Diarization Error Rate (DER) and Jaccard Error Rate (JER) as the evaluation metric. For the DIHARD dataset, we follow the evaluation protocol in the DIHARD challenge, where no forgiveness collar is applied to the reference segments prior to scoring, and overlapping speech is evaluated. For DIHARD III, only full set is considered for evaluation. For VoxConverse, we follow the evaluation protocol in the VoxSRC challenge, where a forgiveness collar of 0.25 is employed. We do **NOT** use any evaluation dataset for training or fine-tuning.

B. Speaker Embedding Extraction

The front-end extractor is ResNet34, where the widths (number of channels) of the residual blocks are {32, 64, 128, 256}. The temporal statistic pooling (TSP) layer computes the mean and standard deviation of the output feature maps and projects the variable length input to a fixed-length vector. Next, a fully connected layer predicts the 128-dimensional speaker embedding, with a dropout of 0.5. Finally, we use the ArcFace [57] (s=32,m=0.2) as the objective for the training.

We perform data augmentation with MUSAN and RIRs. For the MUSAN corpus, ambient noise, music, television, and babble noise are used for the background additive noise. For the RIRs corpus, only impulse responses from small and medium rooms are employed to perform convolution with training data.

The acoustic features are 80-dimensional log Mel-filterbank energies with a frame length of 25 ms and a frameshift of 10 ms. The acoustic features are mean-normalized before being fed into the network. We train two speaker embedding models. One is trained with the 16-kHz Voxceleb 1&2 dataset, which is for VoxConverse data and non-CTS data in DIHARD II and DIHARD III. Another is trained with the 8kHz downsampled Voxceleb 1&2 dataset, which is used in the TS-VAD model traing for CTS data. The Equal Error Rate (EER) on the Voxceleb 1 original test set is reported in Table II.

C. Similarity Measurement

1) Baseline: For the baseline, we extract the speaker embedding from the uniform segmented signal, where the window length is 1.28 s, and the window shift is 0.64 s.

TABLE II
THE EER OF THE SPEAKER EMBEDDING MODEL ON VOXCELEB-1 ORIGINAL TEST SET

Training data	EER (%)
Voxceleb 16k	1.23
Voxceleb 8k	1.79

The training input is an embedding sequence containing 64 continuous speaker embeddings. The overlapping segments are also involved for training, where the labels of these segments are the most talkative speaker. Therefore, for an overlapping segment, if a speaker talks more than other speakers, we label the segment with that speaker. If they talk for the same duration, we label the segment with a random speaker. For the same pair of speakers, we always label their segments with the same speaker.

The training process for the baseline contains two steps. First, we train the BiLSTM/self-attention-based network on the pre-training dataset introduced in Section IV-A2 for 100 epochs, where the development set is used for validation. Next, we fine-tune the model on the development set for 100 epochs. We use the Adam [58] optimizer with the binary cross-entropy (BCE) function and a learning rate of 0.001 for training and 0.0001 for finetuning. The average DER of the last ten epochs is reported as the final result.

In the inference stage, the embedding sequence is first broken into several sub-sequences with a length of 64 and a shift of 32. Next, we process each pair of sub-sequences and generate a 64×64 sub-matrix and merge all sub-matrices to form the complete affinity matrix, as mentioned in Section II-C1. Finally, spectral clustering is employed to get the final diarization results.

2) Data Augmentation: The data augmentation introduced in Section II-D is only employed in the training stage, where the dataset mentioned in Section IV-A2 is the training set. During the fine-tuning stage, we do not employ this data augmentation. The probability of performing data augmentation for each data sample is set to 0.4.

3) Segmental Pooling: With the speaker embedding network parameters unchanged, we replace the TSP with the SSP to extract the speaker embeddings from the uniformly segmented CNN feature map. The temporal resolution of the CNN feature map before the pooling is eight times less than for the input features, as shown in Table I. Therefore, the frame rate of the feature map is 80 ms. At the CNN feature map level, we set the segment length to 16 and the segment shift to 8 so that the window length is 1.28 s and the window shift is 0.64 s, which is the same as the configuration of the baseline system. The training and the inference processes are the same as those of the baseline, except that the embedding with SP can see more context than the baseline system does. "More context" means that the segments with SP have seen the information in 3.72s as compared to 1.28s for the baseline. That is why the system with SP can show better performance than the baseline system.

4) Joint Training: The purpose of the joint training is not only to adapt the back-end classifier to the dev set but also to transfer the front-end extractor to the dev set. Therefore, we only perform joint training on the development set of DIHARD II, DIHARD III and VoxConverse dataset. The front-end extractor is the pre-trained speaker embedding model mentioned in Section IV-B, with the pooling layer replaced by SSP. The back-end classifier is the pre-trained model mentioned in Section IV-C1. Then, we jointly fine-tune these models using three steps:

- Step 1: Keep the front-end network frozen, and only train the back-end BiLSTM for the similarity measurement until the convergence.
- Step 2: Jointly train the two networks for five epochs with a low learning rate.
- Step 3: Keep the front-end network frozen again and train the back-end network for 100 epochs.

The learning rate of the back-end BiLSTM is set to 0.001 for each training step, and the learning rate of the front-end extractor is set to 0.00001 in step 2. The optimizer is Adam, and the loss function is BCE loss. In step 3, the training process is the same as that in Section IV-C1, where we extract the speaker embedding with the adapted front-end ResNet and fine-tune the back-end network for 100 epochs.

D. Target-speaker Voice Activity Detection

1) Training process: In Section II, we can only slightly change the speaker embedding model with a low learning rate, as a large learning rate will destroy the generalization ability of this well-trained network. However, for TS-VAD, the speaker embedding model needs to be able to learn about overlapped speech. Therefore, we train the model in four stages. The optimizer is Adam, and the loss function is BCE loss in all stages; the only difference is the training dataset and learning rate.

- Stage 1: First, we copy the parameters of the pre-trained speaker embedding model to the front-end extractor and keep the front-end extractor frozen. Next, we train the TS-VAD model for ten epochs, with a learning rate of 0.0001. For non-CTS data, we use the simulated Librispeech dataset. For CTS data, we use the simulated telephone speech data.
- Stage 2: The configuration is the same as stage 1, except the front-end extractor is unfrozen.
- Stage 3: We continue to train the model for ten epochs, with a learning rate of 0.0001 on the simulated DIHARD dataset.
- Stage 4: We train the model with a learning rate of 0.00001 on the real development set for 100 epochs. We take two recordings from each domain for validation.

The number of target speakers (N) is set to 8 for the original TS-VAD model. During the inference stage, we only select the eight most talkative speakers to extract the target speaker embedding and discard other speakers, but the clustering-based results from the discarded speakers are kept. If the number of speakers in the clustering-based results is less than N, we use zero-vectors as the invalid speaker embeddings.

For the two-speaker TS-VAD as overlap detection, we only select the five most talkative speakers, which results in $\binom{5}{2} = 10$ times the inference for each recording. Finally, speaker pair prediction labels of the overlapped speech regions are assigned to the clustering-based results.

2) Ablation Study of the Segmental Pooling for TS-VAD: To determine how segmental pooling can influence the TS-VAD performance, we explore several different pooling sizes. We test different pooling sizes at the CNN feature map level from 1 frame to 8 frames (80 ms $\sim$ 640ms) to determine which

TABLE III
DER AND JER (%, ±STD) OF DIFFERENT SIMILARITY MEASUREMENT MODELS ON DIHARD II DATASET. EMBD-AUG IS THE DATA AUGMENTATION PERFORMED ON THE SPEAKER EMBEDDING, SP IS SEGMENTAL POOLING, AND JT DENOTES JOINT TRAINING.

Model	**DIHARD II**					
	Dev		**Eval**		**Eval (+dev adapt)**	
	DER	**JER**	**DER**	**JER**	**DER**	**JER**
BiLSTM	24.15±0.41	47.89±0.15	25.59±0.34	49.43±0.16	19.92±0.01	46.70±0.01
+ embd aug	17.63±0.09	43.72±0.32	18.26±0.18	43.36±0.30	18.12±0.01	43.43±0.03
+ SP	17.54±0.13	42.60±0.76	17.81±0.17	42.17±0.30	17.80±0.00	**42.77±0.01**
+ JT	-	-	-	-	**17.76±0.03**	42.83±0.01
+ embd aug (3.72s)	-	-	-	-	18.99±0.03	44.99±0.05
Self-att	20.97±0.67	46.69±0.23	22.48±0.59	47.47±0.20	19.99±0.04	46.13±0.08
+ embd aug	18.17±0.10	44.03±0.57	18.71±0.23	43.89±0.39	18.42±0.01	43.12±0.01
+ SP	18.28±0.26	43.29±0.46	18.76±0.22	43.28±0.40	**18.00±0.01**	**42.12±0.02**
+ embd aug (3.72s)	-	-	-	-	20.25±0.08	46.30±0.17
Official baseline [55]	-	-	-	-	25.99	59.51
winning system (Clustering) [21]	-	-	-	-	18.21	-

TABLE IV
DER AND JER (%, ±STD) OF DIFFERENT SIMILARITY MEASUREMENT MODELS ON DIHARD III DATASET. EMBD-AUG IS THE DATA AUGMENTATION PERFORMED ON THE SPEAKER EMBEDDING, SP IS SEGMENTAL POOLING, AND JT DENOTES JOINT TRAINING. THE WINNING SYSTEM SHOWN IN GRAY IS OBTAINED AS A FUSION OF MULTIPLE SYSTEMS.

Model	**DIHARD III**					
	Dev		**Eval**		**Eval (+dev adapt)**	
	DER	**JER**	**DER**	**JER**	**DER**	**JER**
BiLSTM	21.03±0.26	40.92±0.14	20.11±0.26	39.62±0.17	17.03±0.01	37.45±0.03
+ embd aug	16.30±0.08	36.17±0.22	15.85±0.13	34.63±0.32	15.62±0.01	34.45±0.01
+ SP	16.29±0.15	35.50±0.50	15.61±0.09	33.98±0.34	15.45±0.02	**33.91±0.03**
+ JT	-	-	-	-	**15.18±0.02**	34.04±0.03
+ embd aug (3.72s)	-	-	-	-	17.48±0.01	36.67±0.06
Self-att	19.99±0.52	40.35±0.20	19.20±0.40	38.26±0.14	16.84±0.07	36.81±0.06
+ embd aug	16.86±0.11	37.42±0.48	16.04±0.16	35.34±0.45	15.82±0.01	34.74±0.06
+ SP	16.69±0.19	36.45±0.35	16.00±0.18	34.57±0.27	**15.65±0.01**	**34.06±0.04**
+ embd aug (3.72s)	-	-	-	-	17.76±0.03	37.96±0.19
Official baseline [56]	-	-	-	-	19.25	42.45
winning system (Clustering) [59]	-	-	-	-	15.77	-
winning system (Fusion) [59]	-	-	-	-	11.30	-

TABLE V
DER AND JER (%, ±STD) OF DIFFERENT SIMILARITY MEASUREMENT MODELS ON VOXCONVERSE DATASET. EMBD-AUG IS THE DATA AUGMENTATION PERFORMED ON THE SPEAKER EMBEDDING, SP IS SEGMENTAL POOLING, AND JT DENOTES JOINT TRAINING.

Model	**VoxConverse**					
	Dev		**Eval**		**Eval (+dev adapt)**	
	DER	**JER**	**DER**	**JER**	**DER**	**JER**
BiLSTM	12.75±0.31	41.79±0.55	17.25±0.33	53.31±0.61	12.52±0.08	39.78±0.05
+ embd aug	4.53±0.07	24.27±0.55	7.04±0.14	39.85±1.10	6.67±0.02	33.81±0.08
+ SP	4.41±0.10	24.28±0.60	5.82±0.22	39.56±1.04	**4.63±0.03**	**31.46±0.15**
+ JT	-	-	-	-	4.74±0.00	32.12±0.01
Self-att	10.56±0.30	41.11±1.19	13.43±0.34	55.28±1.18	10.21±0.06	41.48±0.07
+ embd aug	6.91±0.22	35.62±0.93	9.64±0.22	51.43±0.97	7.28±0.02	36.03±0.05
+ SP	6.08±0.21	32.80±1.12	7.81±0.31	48.81±1.26	**5.69±0.03**	**33.43±0.08**

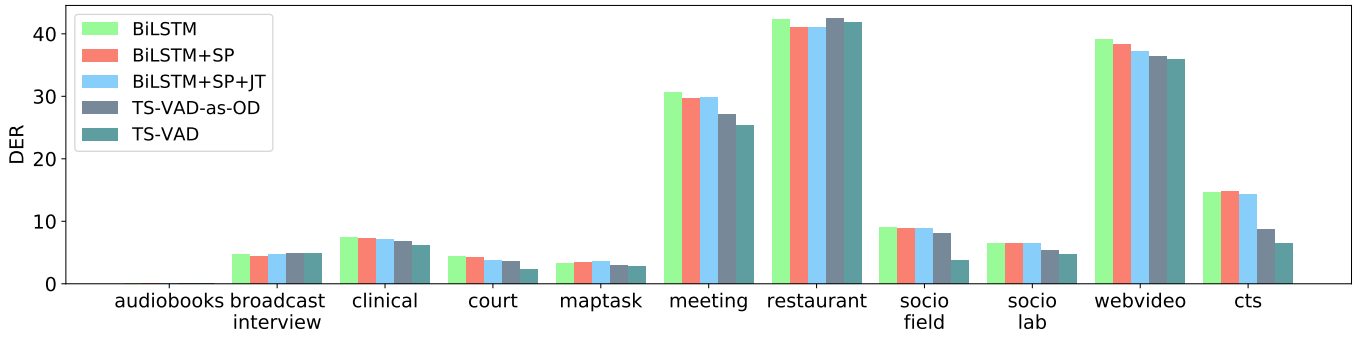


Fig. 9. DER (%) on different domains in the DIHARD III dataset. SP and JT denote segmental pooling and joint training, respectively. The random rotation augmentation on the speaker embedding is performed.

pooling size produces the best performance. A small pooling size may not contain enough information, while a large pooling size may include multiple speakers in a single embedding and result in performance degradation.

V. Experimental Results and Discussion

A. Similarity Measurement

Table III, IV, V shows the DERs of the similarity measurement models on the DIHARD II, DIHARD III, and VoxConverse datasets, respectively. For the evaluation set, both the results before and after adaptation are reported. The columns of Dev and Eval show the results before fine-tuning, and the columns of Eval (+dev adapt) show the results after fine-tuning on the development set. The first row is the result of the baseline without embedding rotation augmentation (Embd-Aug) and SP. The DER is calculated from the average results of the models from the last ten epochs to reduce the variance. Results of BiLSTM model show that the joint training (JT) with segmental pooling achieves the lowest DER, 17.76% on DIHARD II, 15.18% on DIHARD III. Generally, the embedding augmentation can always improve the performance for both BiLSTM and self-attention-based model, and SP can further improve the performance. Compared with BiLSTM, the self-attention-based network shows better performance if we directly train the model without augmentation, which is similar to the findings in our previous work [17]. However, when the augmentation is employed, BiLSTM shows better performance than the self-attention-based network. In addition, we also show the adapted DERs of the system with embedding augmentation for the 3.72-second segment with a shift of 0.64 second in Table III and IV, as the embedding with SP can see more context (3.72s) than its length (1.28s) through the ResNet34 receptive field. When using a long segment with embedding augmentation, it shows degraded performance as the resolution of the long segments is lower than that of the short segments.

From the results, we find that the segmental pooling shows good improvement on all datasets, which proves that the embedding with segmental pooling contains more speaker information than the embedding of the segmented acoustic features. Joint training further improves the performance on the DIHARD dataset, but it is only slightly better on DIHARD II. This may be because the DIHARD III contains 8kHz CTS data, and joint fine-tuning can help the speaker embedding network to slightly adapt to the 8kHz data. In addition, the performance worsens on the VoxConverse dataset. The reason is that the speaker embedding model is well trained on the Voxceleb dataset, which already has good generalization ability on the VoxConverse data. It is not easy to use the joint training framework to fine-tune the speaker embedding model from one domain to another similar domain.

Compared with the official baseline and the previous winning systems, our method shows better performance. As we do not perform any post-processing and the output is directly generated by spectral clustering, we only compare our results with the clustering-based results in the DIHARD III challenge. Both winning systems empoly AHC-based clustering methods, and VB or VBx resegmentation is performed [21, 27, 32]. We do not show the winning system of the VoxConverse dataset, as we use the oracle VAD in our experiments, but the winning system used the system VAD in the challenge.

B. TS-VAD

Table VI shows the DERs of our TS-VAD system on the DIHARD II, DIHARD III, and VoxConverse datasets, and Table VII shows the DERs of TS-VAD as overlap detection. For DIHARD II and VoxConverse, as all recordings are 16kHz, we only train one TS-VAD model with the speaker embedding model trained on the 16kHz Voxceleb dataset. For DIHARD III, we use the same speaker embedding model and the TS-VAD model for 16kHz data. But for 8kHz CTS data, we use another speaker embedding model and TS-VAD model trained on the 8kHz data for inference, and the results are finally merged together.

During the inference stage, we directly use the clustering-based results from the joint training BiLSTM model as the initialization of the TS-VAD system. As some speakers talk for a long time, and the speech signal is too long for the model to extract the target speaker embedding, the regions that contain a single speaker are first selected and uniformly broken into 16-second segments. Then, we extract the speaker embedding for all segments and take the mean of all embeddings as the target speaker embedding. The outputs of the TS-VAD model are the segment-level probabilities for each speaker. Next, we apply a threshold of 0.8 on the probabilities to get the binary decision. Since we use the oracle VAD, if the probability of

TABLE VI
DER (%) OF THE SEGMENT-LEVEL TS-VAD MODELS ON EVALUATION DATASET (N=8, FULLY ASSIGNED)

Pooling Size	DIHARD II				DIHARD III				VoxConverse			
	MISS(%)	FA(%)	SpkErr(%)	DER(%)	MISS(%)	FA(%)	SpkErr(%)	DER(%)	MISS(%)	FA(%)	SpkErr(%)	DER(%)
s=1 (80ms)	8.2	0.7	7.6	**16.48**	6.6	0.7	4.4	**11.62**	1.0	0.2	3.7	4.95
s=2 (160ms)	8.1	1.4	7.7	17.20	6.1	1.6	4.4	12.09	1.0	0.2	3.8	5.04
s=4 (320ms)	8.1	1.4	8.3	17.83	6.3	1.8	4.9	12.97	1.0	0.3	3.5	**4.72**
s=8 (640ms)	8.2	1.7	9.2	19.06	6.7	2.1	5.9	14.67	1.0	0.3	3.6	4.78
Clustering	9.7	0.0	8.1	17.76	9.5	0.0	5.7	15.18	1.6	0.0	3.0	4.57
winning system (TS-VAD)	-	-	-	-	-	-	-	12.30 [27]	-	-	-	-

TABLE VII
DER (%) OF THE SEGMENT-LEVEL TS-VAD AS OVERLAP DETECTION ON THE EVALUATION DATASET (N=2, PARTIALLY ASSIGNED OVERLAPPED REGION)

Pooling Size	DIHARD II				DIHARD III				VoxConverse			
	MISS(%)	FA(%)	SpkErr(%)	DER(%)	MISS(%)	FA(%)	SpkErr(%)	DER(%)	MISS(%)	FA(%)	SpkErr(%)	DER(%)
s=1 (80ms)	8.0	1.0	8.1	**17.19**	5.7	1.5	5.7	**12.89**	1.1	0.3	3.1	**4.39**
s=2 (160ms)	7.9	2.1	7.9	17.94	5.4	2.8	5.4	13.57	1.0	0.5	3.0	4.49
s=4 (320ms)	8.2	1.9	7.9	17.98	5.7	2.8	5.3	13.77	1.1	0.3	3.0	4.40
s=8 (640ms)	8.3	1.7	8.0	18.00	6.2	3.1	5.3	14.49	1.0	0.5	3.0	4.52
Clustering	9.7	0.0	8.1	17.76	9.5	0.0	5.7	15.18	1.6	0.0	3.0	4.57

a speech frame is lower than the threshold for all speakers, we will assign the speaker with the largest probability to this frame.

Results in Table VI show that our domain-independent TS-VAD method achieves a DER of 16.48% on DIHARD II and 11.62% on DIHARD III when we directly use the output from the original TS-VAD model as the results. For the DIHARD datasets, both results are better than the clustering-based results. However, for VoxConverse dataset, the result becomes worse. The reason is that the Voxconverse contains fewer overlapped speech regions than the DIHARD dataset, and the clustering-based method already has a good performance on the VoxConverse dataset. As Table VI shows, the reduction in MISS cannot compensate for the increase in FA and SpkErr.

Table VII shows the results of two-speaker TS-VAD as overlap detection. As we only assign the detected overlapped speech region to the clustering-based results, it shows better performance than the original TS-VAD method on the VoxConverse dataset. However, it becomes worse on the DIHARD dataset.

We also present the DERs with different pooling sizes in each table. The frame rate of the feature map from the front-end extractor is 80 ms, as the time resolution of the feature map is eight times less than for the input features. Next, we employ segmental pooling on the feature maps with the different number of frames. The number of original frames aggregated into each segment is 1, 2, 4, and 8, and the corresponding segment-level frame rate is 80 ms, 160 ms, 320 ms, and 640 ms, respectively. The results show that the DER becomes worse as the pooling size becomes larger on the DIHARD dataset. However, for the VoxConverse dataset, the pooling size does not show many differences in the final results. The reason is that the collar used for VoxConverse scoring is 0.25, and we do not need a high resolution to obtain the same good performance. Therefore, for the dataset that gets scored without a collar, a smaller pooling size shows good performance; otherwise a large pooling size may be better.

We also show DERs on different domains of the DIHARD III dataset in Fig. 9. It can be found that the significant improvement in DIHARD III mainly comes from the CTS dataset, as it only contains two speakers, and almost half of the errors are overlapped speech. We finally reduce the DER on CTS data in the DIHARD III eval set from 14.67% to 6.47%. As the DIHARD challenge is a multi-domain task, our domain-independent method cannot be optimized on each domain without using in-domain data, especially on the extremely noisy and overlapped domain, e.g., restaurant, as shown in Fig. 9. Note that we train our model on the multi-domain data, and we do not use any evaluation dataset for fine-tuning. Still, we achieve better performance than the previous domain-dependent i-vector-based TS-VAD model, in which the evaluation data is used for fine-tuning [27].

VI. CONCLUSION

In this paper, we introduced our neural similarity measurement method, which showed superior performance compared to previous systems on the DIHARD and VoxConverse datasets. We have proposed the segmental pooling strategy for similarity measurement enabling the embeddings to see more context, which leads to an improve the performance. Later, we unfroze the parameters of the speaker embedding network and train it jointly with the similarity measurement objective, which further improved performance. This joint training strategy was a general framework for both similarity measurement and TS-VAD at the segment level, which could be easily employed in both tasks with the help of segmental pooling. Therefore, we extended this framework to TS-VAD and explored the influence of different pooling sizes. Results showed that the TS-VAD method significantly reduced the MISS in DER.

In the future, we will explore different pooling strategies, including weighted pooling, attentive pooling, and learnable

dictionary encoding. In addition, we will extend this framework to more tasks like end-to-end speaker diarization.

REFERENCES

[1] S. E. Tranter and D. A. Reynolds, "An Overview of Automatic Speaker Diarization Systems," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1557–1565, 2006.

[2] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, "Speaker Diarization: A Review of Recent Research," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 356–370, 2012.

[3] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, "A Review of Speaker Diarization: Recent Advances with Deep Learning," *arXiv preprint arXiv:2101.09624*, 2021.

[4] C. Wooters, J. Fung, B. Peskin, and X. Anguera, "Towards Robust Speaker Segmentation: The ICSI-SRI Fall 2004 Diarization System," in *Proceedings of Fall 2004 Rich Transcription Workshop (RT-04F)*, 2004, p. 23.

[5] R. Zazo, T. N. Sainath, G. Simko, and C. Parada, "Feature Learning with Raw-Waveform CLDNNs for Voice Activity Detection," in *Proc. Interspeech 2016*, pp. 3668–3672.

[6] F. Eyben, F. Weninger, S. Squartini, and B. Schuller, "Real-life Voice Activity Detection with LSTM Recurrent Neural Networks and an Application to Hollywood Movies," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 483–487.

[7] S.-Y. Chang, B. Li, G. Simko, T. N. Sainath, A. Tripathi, A. van den Oord, and O. Vinyals, "Temporal Modeling Using Dilated Convolution and Gating for Voice-activity-detection," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5549–5553.

[8] M. Hrúz and Z. Zajíc, "Convolutional Neural Network for Speaker Change Detection in Telephone Speaker Diarization System," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4945–4949.

[9] R. Yin, H. Bredin, and C. Barras, "Speaker Change Detection in Broadcast TV Using Bidirectional Long Short-Term Memory Networks," in *Proc. Interspeech 2017*, pp. 3827–3831.

[10] G. Sell and D. Garcia-Romero, "Speaker Diarization with PLDA i-vector Scoring and Unsupervised Calibration," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, pp. 413–417.

[11] X. Xiao, N. Kanda, Z. Chen, T. Zhou, T. Yoshioka, S. Chen, Y. Zhao, G. Liu, Y. Wu, J. Wu *et al.*, "Microsoft Speaker Diarization System for the VoxCeleb Speaker Recognition Challenge 2020," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5824–5828.

[12] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[13] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN Embeddings for Speaker Recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5329–5333.

[14] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker Diarization with LSTM," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5239–5243.

[15] G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe *et al.*, "Diarization is Hard: Some Experiences and Lessons Learned for the JHU Team in the Inaugural DIHARD Challenge," in *Proc. Interspeech 2018*, pp. 2808–2812.

[16] Q. Lin, R. Yin, M. Li, H. Bredin, and C. Barras, "LSTM Based Similarity Measurement with Spectral Clustering for Speaker Diarization," in *Proc. Interspeech 2019*, pp. 366–370.

[17] Q. Lin, Y. Hou, and M. Li, "Self-Attentive Similarity Measurement Strategies in Speaker Diarization," in *Proc. Interspeech 2020*, pp. 284–288.

[18] P. Singh and S. Ganapathy, "Self-Supervised Representation Learning With Path Integral Clustering for Speaker Diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1639–1649, 2021.

[19] C. Boeddecker, J. Heitkaemper, J. Schmalenstroeer, L. Drude, J. Heymann, and R. Haeb-Umbach, "Front-end Processing for the CHiME-5 Dinner Party Scenario," in *Proc. CHiME 2018 Workshop on Speech Processing in Everyday Environments*, pp. 35–40.

[20] T. Yoshioka, H. Erdogan, Z. Chen, X. Xiao, and F. Alleva, "Recognizing Overlapped Speech in Meetings: A Multichannel Separation Approach Using Neural Networks," in *Proc. Interspeech 2018*, pp. 3038–3042.

[21] F. Landini, S. Wang, M. Diez, L. Burget, P. Matějka, K. Žmolíková, L. Mošner, A. Silnova, O. Plchot, O. Novotnỳ *et al.*, "But System for the Second Dihard Speech Diarization Challenge," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6529–6533.

[22] N. Kanda, S. Horiguchi, R. Takashima, Y. Fujita, K. Nagamatsu, and S. Watanabe, "Auxiliary Interference Speaker Loss for Target-Speaker Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 236–240.

[23] K. molkov, M. Delcroix, K. Kinoshita, T. Higuchi, A. Ogawa, and T. Nakatani, "Speaker-Aware Neural Network Based Beamformer for Speaker Extraction in Speech Mixtures," in *Proc. Interspeech 2017*, 2017, pp. 2655–2659.

[24] M. Delcroix, K. Zmolikova, K. Kinoshita, A. Ogawa, and T. Nakatani, "Single channel target speaker extraction and recognition with speaker beam," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5554–5558.

[25] S. Ding, Q. Wang, S.-y. Chang, L. Wan, and I. L. Moreno, "Personal vad: Speaker-conditioned voice activity detection," *arXiv preprint arXiv:1908.04284*, 2019.

[26] I. Medennikov, M. Korenevsky, T. Prisyach, Y. Khokhlov, M. Korenevskaya, I. Sorokin, T. Timofeeva, A. Mitrofanov, A. Andrusenko, I. Podluzhny, A. Laptev, and A. Romanenko, "Target-Speaker Voice Activity Detection: A Novel Approach for Multi-Speaker Diarization in a Dinner Party Scenario," in *Proc. Interspeech 2020*, pp. 274–278.

[27] Y. Wang, M. He, S. Niu, L. Sun, T. Gao, X. Fang, J. Pan, J. Du, and C.-H. Lee, "USTC-NELSLIP system description for DIHARD-III challenge," *arXiv preprint arXiv:2103.10661*, 2021.

[28] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech Enhancement Based on Deep Denoising Autoencoder," in *Proc. Interspeech 2013*, pp. 436–440.

[29] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A Regression Approach to Speech Enhancement Based on Deep Neural Networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2014.

[30] L. Drude, J. Heymann, C. Boeddeker, and R. Haeb-Umbach, "NARA-WPE: A Python Package for Weighted Prediction Error Dereverberation in Numpy and Tensorflow for Online and Offline Processing," in *ITG Symposium on Speech Communication*, 2018, pp. 1–5.

[31] R. Haeb-Umbach, S. Watanabe, T. Nakatani, M. Bacchiani, B. Hoffmeister, M. L. Seltzer, H. Zen, and M. Souden, "Speech Processing for Digital Home Assistants: Combining Signal Processing with Deep-learning Techniques," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 111–124, 2019.

[32] F. Landini, J. Profant, M. Diez, and L. Burget, "Bayesian HMM Clustering of x-vector Sequences (VBx) in Speaker Diarization: Theory, Implementation and Analysis on Standard Tasks," *Computer Speech & Language*, p. 101254, 2021.

[33] A. Stolcke and T. Yoshioka, "DOVER: A Method for Combining Diarization Outputs," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 757–763.

[34] D. Raj, L. P. Garcia-Perera, Z. Huang, S. Watanabe, D. Povey, A. Stolcke, and S. Khudanpur, "DOVER-Lap: A Method for Combining Overlap-aware Diarization Outputs," in *2021 IEEE Spoken Language Technology Workshop (SLT)*, pp. 881–888.

[35] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 4905–4913.

[36] G. Wisniewksi, H. Bredin, G. Gelly, and C. Barras, "Combining Speaker Turn Embedding and Incremental Structure Prediction for Low-Latency Speaker Diarization," in *Proc. Interspeech 2017*, pp. 3582–3586.

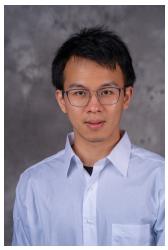
[37] W. Cai, J. Chen, and M. Li, "Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2018)*, pp. 74–81.

[38] W. Cai, J. Chen, J. Zhang, and M. Li, "On-the-fly Data Loader and Utterance-level Aggregation for Speaker and Language Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1038–1051, 2020.

[39] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep Neural Network-based Speaker Embeddings for End-to-end Speaker Verification," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 165–170.

[40] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan,

and Z. Zhu, "Deep Speaker: an End-to-end Neural Speaker Embedding System," *arXiv preprint arXiv:1705.02304*, 2017.
[41] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[42] M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
[43] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid Speech Recognition with Deep Bidirectional LSTM," in *2013 IEEE workshop on automatic speech recognition and understanding*, pp. 273–278.
[44] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "TTS Synthesis with Bidirectional LSTM Based Recurrent Neural Networks," in *Fifteenth annual conference of the international speech communication association*, 2014.
[45] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
[46] U. Von Luxburg, "A Tutorial on Spectral Clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.
[47] Q. Li, F. L. Kreyssig, C. Zhang, and P. C. Woodland, "Discriminative Neural Clustering for Speaker Diarisation," in *2021 IEEE Spoken Language Technology Workshop (SLT)*, pp. 574–581.
[48] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale Speaker Verification in the Wild," *Computer Speech & Language*, vol. 60, p. 101027, 2020.
[49] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal *et al.*, "The AMI Meeting Corpus: A Pre-announcement," in *International workshop on machine learning for multimodal interaction*. Springer, 2005, pp. 28–39.
[50] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke *et al.*, "The ICSI Meeting Corpus," in *2003 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. I–I.
[51] 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)*, pp. 5206–5210.
[52] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone Speech Corpus for Research and Development," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 1. IEEE Computer Society, 1992, pp. 517–520.
[53] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," *arXiv:1510.08484*, 2015.
[54] J. S. Chung, J. Huh, A. Nagrani, T. Afouras, and A. Zisserman, "Spot the Conversation: Speaker Diarisation in the Wild," in *Proc. Interspeech 2020*, 2020, pp. 299–303.
[55] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "The Second DIHARD Diarization Challenge: Dataset, Task, and Baselines," in *Proc. Interspeech 2019*, pp. 978–982.
[56] N. Ryant, P. Singh, V. Krishnamohan, R. Varma, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "The Third DIHARD Diarization Challenge," in *Proc. Interspeech 2021*, pp. 3570–3574.
[57] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive Angular Margin Loss for Deep Face Recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.
[58] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR*, 2015.
[59] Y.-X. Wang, J. Du, M. He, S.-T. Niu, L. Sun, and C.-H. Lee, "Scenario-Dependent Speaker Diarization for DIHARD-III Challenge," in *Proc. Interspeech 2021*, pp. 3106–3110.



Weiqing Wang received the B.S. (2018) in Computer Science from Sun Yat-sen University, and he is currently a Ph.D. student in Department of Electrical and Computer Engineering at Duke University. His research interests focus on speaker diarization. Before joining Duke ECE, he was a research assistant at Duke Kunshan University (2018-2019), working on music signal processing.

Qingjian Lin received the M.S. (2020) in Electronics and Communications Engineering from Sun Yat-sen University, advised by Prof. Ming Li. From 2019-2020, he was a research intern at Duke Kunshan University. After leaving university, he joined Lenovo and worked as a speech signal processing researcher in AI Lab. His research interests include speaker diarization, speaker verification and target speaker separation.

Danwei Cai is pursuing his Ph.D. degree in electrical and computer engineering at Duke University. He received his bachelor degree in software engineering and master degree in electronics and communication engineering from Sun Yet-Sen University in China. His primary research interests are in the area of speech processing, including speech recognition, speaker recognition, speaker diarization, and computational linguistics.

Ming Li received his Ph.D. in Electrical Engineering from University of Southern California in 2013. He is currently an Associate Professor of Electrical and Computer Engineering at Duke Kunshan University. He is also a research scholar at Department of Electrical and Computer Engineering at Duke University. His research interests are in the areas of audio, speech and language processing as well as multimodal behavior signal analysis and interpretation. He has published more than 140 papers and served as the member of IEEE speech and language technical committee, CCF speech dialogue and auditory processing technical committee, CAAI affective intelligence technical committee, APSIPA speech and language processing technical committee. He was the area chair of speaker and language recognition at Interspeech 2016, 2018 and 2020. Works co-authored with his colleagues have won first prize awards at Body Computing Slam Contest 2009, Interspeech Computational Paralinguistic Challenge 2011, 2012 and 2019, ASRU 2019 MGB-5 Challenge, Interspeech 2020 and 2021 Fearless Steps Challenge, VoxSRC 2021 Challenge, ICASSP 2022 M2MeT Challenge. He received the IBM faculty award in 2016, the ISCA Computer Speech and Language 5-years best journal paper award in 2018 and the youth achievement award of outstanding scientific research achievements of Chinese higher education in 2020. He is a senior member of IEEE.