Online Neural Speaker Diarization with Target Speaker Tracking

Weiqing Wang, Ming Li, *Senior Member, IEEE*

***Abstract*—This paper proposes an online target speaker voice activity detection (TS-VAD) system for speaker diarization tasks that does not rely on prior knowledge from clustering-based diarization systems to obtain target speaker embeddings. By adapting conventional TS-VAD for real-time operation, our framework identifies speaker activities using self-generated embeddings, ensuring consistent performance and avoiding permutation inconsistencies during inference. In the inference phase, we employ a front-end model to extract frame-level speaker embeddings for each incoming signal block. Subsequently, we predict each speaker's detection state based on these frame-level embeddings and the previously estimated target speaker embeddings. The target speaker embeddings are then updated by aggregating the frame-level embeddings according to the current block's predictions. Our model predicts results block-by-block and iteratively updates target speaker embeddings until reaching the end of the signal. Experimental results demonstrate that the proposed method outperforms offline clustering-based diarization systems on the DIHARD III and AliMeeting datasets. Additionally, this approach is extended to multi-channel data, achieving comparable performance to state-of-the-art offline diarization systems.**

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

I. Introduction

SPEAKER DIARIZATION is the task that assigns speaker labels to speech regions, serving as an important pre-processing pipeline for many downstream tasks like multi-speaker automatic speech recognition [1, 2] in different scenarios, e.g., broadcast news [3, 4, 5], meeting conversations [6, 7], conversational telephone speech [8, 9] and clinical diagnosis [10, 11], etc.

Early research in speaker diarization primarily focused on modularized systems, where audio signals are processed through a series of cascaded modules [12]. With advancements in speaker representation techniques [13, 14, 15], these modularized diarization systems have shown the potential to deliver commendable performance across various scenarios [16, 17]. Nevertheless, these systems often rely on clustering-based algorithms, which typically allocate audio samples to individual speaker clusters. This approach poses limitations when confronted with overlapping speech regions, as it struggles to handle instances where multiple speakers' voices are present within the same segment. Addressing this issue requires additional modules, such as overlap speech detection [18] and target-speaker voice activity detection (TS-VAD) [19]. Beyond the challenge of overlapping speech, each module in modularized systems is optimized independently, which makes it difficult to optimize the system as a whole.

To address the challenge of overlapping speech in modularized systems, a framework known as end-to-end neural diarization (EEND) was proposed [20, 21]. EEND aims to streamline the speaker diarization process within a single neural network, effectively managing overlapping speech and directly optimizing the diarization objective. Additionally, EEND exhibits adaptability, making it easy to transfer to diverse domains through fine-tuning with real-world data. However, EEND also has limitations, such as operating with a fixed maximum number of speakers. To address this, an extended EEND model incorporating an encoder-decoder-based attractor was developed [22]. This extension was specifically designed to handle scenarios with varying numbers of speakers, further enhancing the applicability of the EEND framework.

Both the modularized and end-to-end diarization systems mentioned earlier demonstrate promising performance on many datasets. However, they are designed to process pre-recorded audio and operate offline, meaning they must analyze the entire audio file at once to identify individual speakers. In several practical applications, such as meeting transcription systems and smart agents, low-latency speaker assignment is crucial [12]. Typically, in ASR and speaker diarization tasks, a latency of 1 to 2 seconds is considered low enough for online applications. Despite ongoing efforts to develop online speaker diarization systems, both within clustering-based approaches [23, 24] and neural network-based frameworks [25, 26, 27], achieving fully effective online diarization remains an unresolved challenge.

In an online modularized system, each module must function in an online manner, with a primary challenge being how to replace the offline clustering algorithm with an online one to process speaker representations in chronological order [28, 24]. Given this focus, most online modularized systems concentrate primarily on the clustering algorithm, often overlooking other modules, such as overlap speech detection.

In the online EEND framework, a primary challenge is the inconsistency in speaker order across different timestamps, as the speaker order can vary as a permutation of all existing speakers. During the training stage, this inconsistency can be addressed using a permutation-free objective [20, 29]. However, this approach limits the EEND's extension to online applications, as the order of speakers continually changes with new incoming signals. One solution is to use a buffer to store both prior inputs and results, allowing the alignment of current

Weiqing Wang and Ming Li are with the Department of Electrical and Computer Engineering, Duke University. Ming Li is also with the Suzhou Municipal Key Laboratory of Multimodal Intelligent Systems, Data Science Research Center, Duke Kunshan University.

Corresponding author: Ming Li, E-mail: ming.li369@dukekunshan.edu.cn

results with previous ones [26, 30, 31]. However, this typically requires a large buffer to achieve satisfactory performance.

As a post-processing method for offline modularized diarization systems, target-speaker voice activity detection (TS-VAD) is often employed to refine the diarization results obtained from clustering algorithms [19]. Given the target speaker representations and audio signals, TS-VAD can recognize overlapping speech in a block-wise manner. This inherent property suggests that TS-VAD could naturally be adapted as an online module. However, this potential is often overlooked since TS-VAD typically functions as a sub-module within modularized diarization systems. If target speaker representations can be obtained online, TS-VAD could operate in real-time. Furthermore, if TS-VAD itself generates the target speaker representations, the diarization output could be directly produced using only a single neural model.

In this paper, we propose an online framework for speaker diarization by adapting conventional TS-VAD for real-time operation, termed online TS-VAD (OTS-VAD). Similar to offline TS-VAD, OTS-VAD not only retrieves speaker activities for each speaker but also detects overlapping speech. Unlike the EEND framework, our proposed system determines speaker activities using target speaker embeddings, which are self-generated by the OTS-VAD model and continuously updated with incoming audio signals. As the order of these target speaker embeddings remains consistent, inference-phase inconsistencies are eliminated, requiring only a very small chunk size for inference. Consequently, our approach can operate with a small block size during inference that is similar to TS-VAD, ensuring a fast inference speed.

This paper builds upon our earlier research [32] focusing on the online speaker diarization task. The new contributions of this extension include:

- Enhancing the performance to state-of-the-art (SOTA) levels through improved training and inference methodologies.
- Incorporating a cross-channel attention mechanism to enhance multi-channel support, following existing methods in the literature to achieve improved consistency in performance.
- Replacing the front-end ResNet model with a lightweight Conformer, which indicates the possibility of reducing the model's size without significant performance trade-offs.

The organization of this paper is as follows. Section II reviews the offline and online methods for both modularized systems and end-to-end frameworks. Section III presents the proposed online TS-VAD method as well as the extensions. Section IV describes the experimental details and Section V presents the results, respectively. Section VI provides the conclusions.

II. Related Works

A. Modularized speaker diarization system

1) Offline method: The modularized speaker diarization system, often referred to as the traditional speaker diarization system, consists of several sub-modules. The process begins with voice activity detection (VAD), which filters out non-speech components, such as background noise, from the speech signal [33, 34, 35]. Next, the audio stream is segmented into speaker-consistent sections [36, 37], from which speaker embeddings are derived [38, 13]. To group these embeddings, clustering algorithms such as agglomerative hierarchical clustering (AHC) [39] or spectral clustering [40] are applied using a suitable similarity measurement technique. Pre- and post-processing modules, such as speech enhancement [41], speech separation [42], overlapping speech detection [18], and resegmentation [43], are sometimes included to improve performance. Additionally, two or more modules can be integrated into a single neural network; for example, a region proposal network can perform VAD, segmentation, and embedding extraction simultaneously [44].

2) Online method: While VAD, segmentation, and speaker embedding can be executed in real-time, the primary challenge in adapting the modularized system for online use lies in replacing the offline clustering technique with an online alternative. Notable approaches include adapted i-vectors with a transformation matrix [28] and modified clustering methods [23]. However, these methods can be inefficient for extended audio clips, as their time complexities scale linearly with the number of speaker segments. Other online clustering techniques, such as incremental clustering [45] and online spectral clustering [46], have also been proposed to achieve low-latency online diarization. Additionally, several supervised methods, such as UIS-RNN [25] and UIS-RNN-SML [47], replace segmentation and clustering with trainable models.

B. End-to-end Neural speaker diarization system

1) Offline method: End-to-end neural diarization (EEND) is an innovative framework that performs the entire speaker diarization process using a single deep neural network [20, 21]. Given a speech signal, it directly outputs diarization results, including recognition of overlapping speech. To enhance its versatility and address scenarios with an undefined number of speakers, an improved version known as EEND with Encoder-Decoder-Based Attractor (EDA) was introduced. This advancement not only improved performance but also expanded the framework's applicability. Further refinements include integrating front-end components like Conformer [48] and introducing EEND with Global and Local Attractors (GLA) [49], both of which have enhanced system efficacy on real datasets. In addition to its fully end-to-end framework, EEND can also be employed in modularized speaker diarization systems for tasks such as embedding extraction [50, 51, 52], overlap-aware resegmentation [53], and post-processing [54].

2) Online method: The EEND framework has also been widely explored for online speaker diarization tasks. Han et al. [27] proposed a block-wise EEND-EDA (BW-EDA-EEND) approach that uses hidden states from previous blocks to generate attractors sequentially. Another method to address the issue of speaker permutation ambiguity involves maintaining acoustic features within a speaker-tracing buffer [26, 30]. Building on this, Horiguchi et al. [31] enhanced EEND-GLA for online use by incorporating a speaker-tracing buffer.

Notably, this approach achieved state-of-the-art performance across various datasets, surpassing numerous offline modularized speaker diarization systems.

C. Target speaker voice activity detection

Target speaker tracking is widely used in various speech-related tasks to retrieve information about a specific speaker, including target speaker automatic speech recognition (TS-ASR) [55], target speaker speech separation [56], and target-speaker voice activity detection (TS-VAD) [19]. These tasks share a common approach: they utilize a speaker profile to focus on the speech of interest, thereby refining results for that particular speaker. Notably, this method remains effective even when the audio includes overlapping speech. Typically, the speaker profile is a pre-enrolled speaker embedding, containing speaker identity information like i-vectors [38] or x-vectors [13].

For speaker diarization, TS-VAD has been introduced to refine results derived from clustering-based diarization systems. It is particularly useful in scenarios with significant overlapping speech, where the goal is to detect the presence or absence of specific speakers. The concept of TS-VAD is rooted in personal VAD [57], which adapts the VAD process to the unique vocal characteristics of an individual speaker. However, personal VAD methods typically perform poorly in diarization tasks, as they consider only a single speaker and overlook inter-speaker relationships within the speech signal. In contrast, TS-VAD is designed to recognize the speech activities of multiple speakers present in any segment, incorporating constraints and relationships among speakers for a more comprehensive understanding. Consequently, TS-VAD provides enhanced performance and more reliable diarization outcomes.

The early version of TS-VAD employs a pre-enrolled i-vector as an acoustic footprint [19]. It takes a short speech segment and several i-vectors as input, evaluating whether each frame belongs to one or more speakers, as illustrated in Fig. 1. In this i-vector-based TS-VAD model, a CNN first processes the acoustic features to produce a temporal feature map. Each i-vector is then concatenated with each frame of the feature map and fed separately to a two-layer BiLSTM, generating a speaker detection (SD) feature for each i-vector. These SD features are subsequently concatenated and passed through a final BiLSTM layer to produce the posterior presence probabilities of each speaker. Remarkably, the i-vector-based TS-VAD approach demonstrates a notable reduction in diarization error rate (DER) compared to traditional clustering-based methods, providing a robust solution to the diarization challenge.

Consider N target speaker embeddings represented as $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N\}$, and the acoustic feature as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_L\}$. A typical TS-VAD model detects the speech activities of each speaker n as:

$$\mathbf{S}^{'}_n = [S^{'}_{n,1}, S^{'}_{n,2}, ..., S^{'}_{n,T}] = f_d(\begin{bmatrix}\mathbf{e}_n \\ \hat{\mathbf{e}}_1\end{bmatrix}, \begin{bmatrix}\mathbf{e}_n \\ \hat{\mathbf{e}}_2\end{bmatrix}, ..., \begin{bmatrix}\mathbf{e}_n \\ \hat{\mathbf{e}}_T\end{bmatrix}), \tag{1}$$

where f_d is the detection function that produces the speaker activities $\mathbf{S}^{'}_{n,t} \in R$ based on target speaker embedding $\mathbf{e}_n$ at time t, and $\hat{\mathbf{e}}_t$ is the frame-level embedding generated from the acoustic features by the front-end module. Here $\mathbf{T}$ is usually smaller than $\mathbf{L}$ as the input acoustic feature will be down-sampled to the frame-level embeddings by the front-end module. Subsequently, the speech activities from all speakers are concatenated, serving as the last layer's input. This generates refined speech activities $\mathbf{S} \in (0,1)^{N\times T}$ for all speakers:

$$\mathbf{S} = f_j(\begin{bmatrix} S^{'}_{1,1} & ... & S^{'}_{1,T} \\ ... & S^{'}_{n,t} & ... \\ S^{'}_{N,1} & ... & S^{'}_{N,T}\end{bmatrix}), \tag{2}$$

where f_j acts as the joint detection function that produces final speaker existence probabilities. As depicted in Fig. 1, f_d operates as the 2-layer BiLSTM, while f_j functions as the 1-layer BiLSTM. The front-end model designed for frame-level embedding extraction is constituted by a 4-layer CNN, and this is the typical i-vector TS-VAD framework. Here, $\mathbf{e}_n$ is iv_n and $\mathbf{S}^{'}_n = \left[\mathbf{S}^{'}_{n,1}, ..., \mathbf{S}^{'}_{n,T}\right]$ is SD_n as shown in Fig. 1.

The deployment of i-vectors in TS-VAD, while achieving some success, has shown limitations when applied to multi-domain data [58]. These findings prompted exploration of x-vectors as an alternative, although a straightforward replacement of i-vectors with x-vectors did not yield an immediate performance boost [19]. The likely cause of this unexpected result is that the shallow CNN used in the front end of the TS-VAD system may not be well-suited for efficiently processing the more complex, deep learning-based x-vector. To address this hypothesis, Wang et al. [59] replaced the TS-VAD front end with a pre-trained module specifically designed for x-vectors. This modified architecture demonstrated a more effective handling of x-vectors, resulting in improved performance compared to the i-vector-based system. Additionally, the robustness and generalization capability of x-vectors enabled this model to perform commendably on challenging datasets like DIHARD III, which includes diverse real-world scenarios.

Typically, TS-VAD can handle only a fixed number of speakers, as the final layer is generally a linear projection.

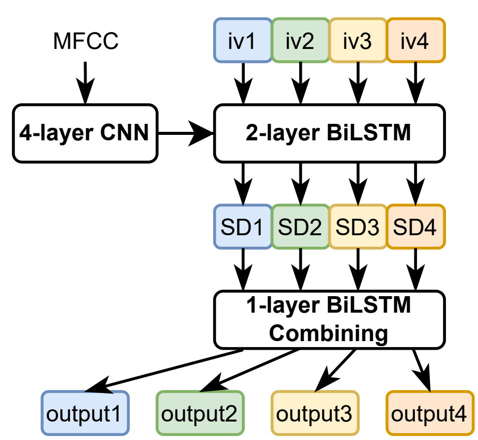


Fig. 1. The architecture of the i-vector-based TS-VAD [19]

.

A common approach is to define the output size based on the maximum anticipated number of speakers. This presents a challenge because, in real-world scenarios, the actual number of speakers is unknown. Furthermore, when the input audio contains fewer speakers than the maximum set, the TS-VAD model performs redundant computations, especially when certain speaker embeddings are initialized to zero or arbitrary values. To address this inefficiency, the terminal linear layer can be replaced with architectures capable of managing variable-length data. For instance, both LSTM [60] and Transformer [61] models have been implemented along the speaker dimension rather than the temporal one. Recently, Cheng et al. [62] proposed a sequence-to-sequence (Seq2Seq) architecture for TS-VAD, referred to as Seq2Seq TS-VAD. In this approach, frame-level representations and speaker embeddings are independently input into the encoder and decoder, eliminating the need for concatenation. Notably, the decoder generates an embedding of consistent dimensionality for each speaker's voice activity, regardless of the input feature duration. Additionally, the final linear layer offers greater flexibility, enabling voice activity predictions at finer temporal resolutions with modest computational demands.

Compared to end-to-end neural diarization (EEND), TS-VAD serves as a post-processing module within modularized speaker diarization systems. While EEND provides a comprehensive solution by learning speaker embeddings directly from audio waveforms to produce diarization results, TS-VAD refines outputs derived from clustering-based methods. Due to its reliance on prior knowledge from clustering-based diarization results, TS-VAD generally achieves superior performance to EEND on recent benchmarks [63], especially when the number of speakers is large. In the following section, we introduce our TS-VAD-based online framework, where the speaker embedding is self-generated by the TS-VAD model.

III. Online Target Speaker Voice Activity Detection

This section details the transformation from an offline approach to an online one, encompassing both the training and inference procedures.

A. Online TS-VAD

1) Architecture: Fig. 2 illustrates the architecture of the ResNet-based online TS-VAD (OTS-VAD). The model consists of two sub-modules: a front end for embedding extraction and a back end for target speaker detection. The front end comprises a ResNet followed by a frame-level global statistics pooling (GSP) layer, which condenses the statistics of each frame from the ResNet output, and is followed by a fully connected layer. The back end includes a Conformer-based encoder, a BiLSTM layer, and a linear layer.

Given an acoustic feature $\mathbf{X} \in \mathbb{R}^{H \times L}$, the ResNet transforms it into a temporal feature map $\mathbf{M} \in \mathbb{R}^{V \times \frac{H}{8} \times \frac{L}{8}}$. Here, V represents the number of channels, H is the feature's dimensionality, and L indicates the total number of frames. It is worth noting that the temporal resolution of this feature map is reduced by a factor of eight compared to the original input, attributed to ResNet's capability of downsampling by this magnitude. Subsequently, the frame-level GSP layer computes the mean and standard deviation over the feature dimension, thus generating a vector of dimension $2V$ for each frame. The succeeding fully-connected layer ingests the GSP's output, $\mathbf{P} \in \mathbb{R}^{2V \times T}$, producing frame-level embeddings, represented as $\hat{\mathbf{E}} = [\hat{\mathbf{e}}_1, ..., \hat{\mathbf{e}}_T] \in \mathbb{R}^{D \times T}$. Here, $T = \frac{L}{8}$ and D is the dimensionality of frame-level embeddings. Furthermore, D matches the dimension of the target speaker embeddings. The usage of frame-level GSP is to imitate the utterance-level GSP in the speaker recognition task [64]. This method interprets each window as a distinct utterance for which embeddings are extracted. By adopting this approach, the front-end module can effectively discern the speaker's identity for each frame, which invariably leads to enhanced performance.

After obtaining the frame-level embeddings $\hat{\mathbf{E}}$ and the target speaker embeddings, represented by $\mathbf{E} = [\mathbf{e}_1, ..., \mathbf{e}_N] \in \mathbb{R}^{N \times D}$, the back-end module predicts speech activities similarly to the offline TS-VAD model. Specifically, each target speaker embedding is replicated and concatenated with the frame-level embeddings. The initial encoder processes these concatenated embeddings separately for each target speaker, generating a primary decision level. It is important to note that these decisions may not strictly represent probabilities. Subsequently, the decisions for all speakers are concatenated and passed to the BiLSTM for refinement, allowing the model to incorporate inter-speaker information. Finally, a sigmoid-activated fully connected layer estimates the probabilities of speaker presence.

To adapt offline TS-VAD for online operation, it is crucial to obtain the target speaker embedding without relying on an additional speaker embedding extractor during both training and inference. Additionally, maintaining the consistency of the target speaker embedding is essential for the back-end module to accurately detect speech activities. In the following sections, we describe our approach to obtaining a consistent target speaker embedding throughout the training and inference processes.

2) Training process: As shown in Fig. 2, given an input $\mathbf{X}$ with length of $2L$, the initial step involves partitioning the input features uniformly into left and right segments $\mathbf{X}^l$ and $\mathbf{X}^r$, each having a length of L. After extracting the frame-level embeddings $\hat{\mathbf{E}}^l, \hat{\mathbf{E}}^r \in \mathbb{R}^{D \times T}$ for the left and right segments, the target speaker embedding for the n-th speaker can be obtained by:

$$\mathbf{e}_n = \frac{\hat{\mathbf{E}}^l \mathbf{Y}_n^l}{\sum_{i=1}^{T} Y_{n,i}^l}, \tag{3}$$

where $n \in \{1, 2, ...N\}$, i is the frame index, $T = \frac{L}{8}$ by downsampling from ResNet, $\mathbf{Y}_n^l \in \{0, 1\}^T$ is the corresponding label of n-th speaker in the left segment with the overlapping regions set to zero. Eq. 3 describes that target-speaker embedding can be obtained by averaging all non-overlapping frame-level embeddings for each respective speaker, as shown in Fig 3. In addition, due to the limit of the GPU memory, we cannot train the model with long-duration inputs. However, this will lead to a problem that

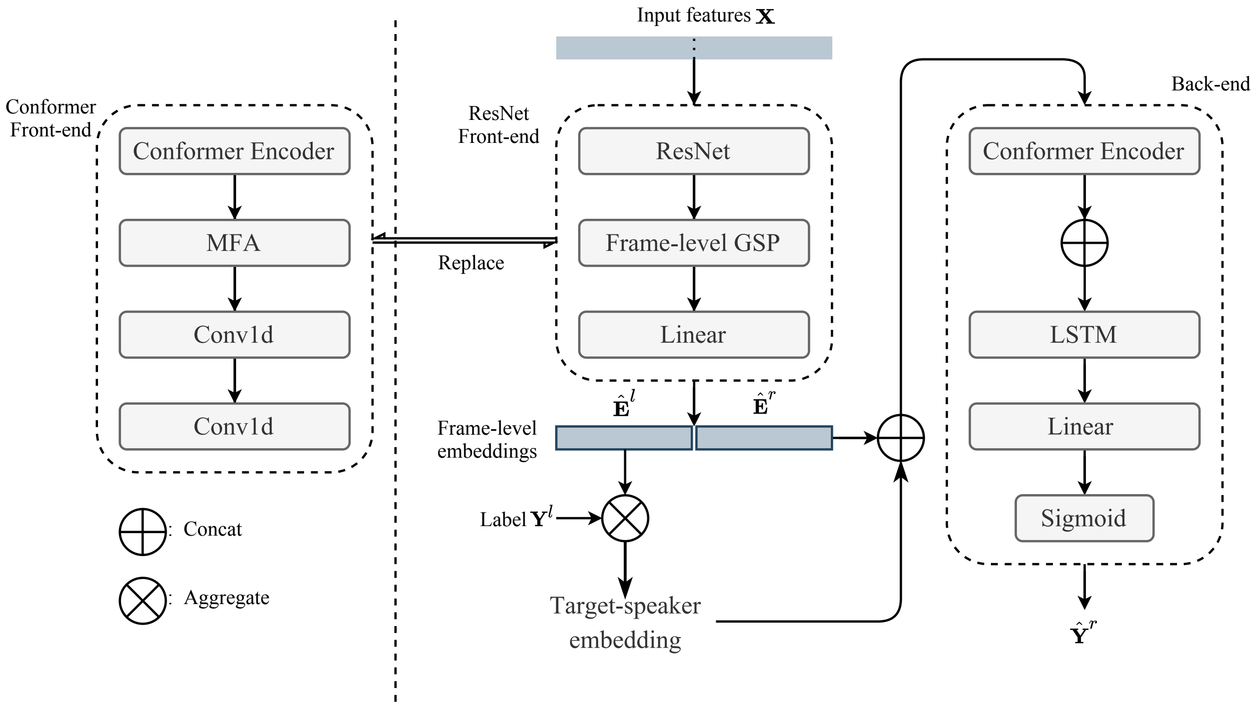


Fig. 2. The architecture of the proposed OTS-VAD. The right is the backbone with ResNet-based front-end, and the left is another Conformer-based front-end which can replace the ResNet-based front-end.

$\mathbf{X}^l$ only contains limited speakers with a short duration, and the back-end module only accepts insufficient target speaker embeddings during the training stage. Therefore, the model cannot deal with the input with lots of speakers. To cope with this, we replace the $\mathbf{X}^l$ with a randomly simulated signal with a probability of 0.5. Each speaker in the simulated signal has equal length of speech.

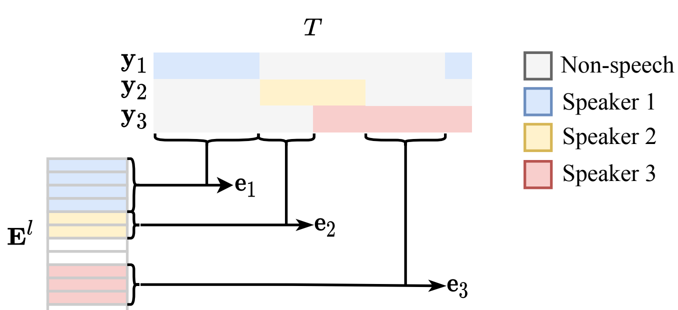


Fig. 3. Illustration of the target speaker embedding extraction/aggregation.

With the target speaker embedding $\mathbf{E}$ and frame-level embeddings $\hat{\mathbf{E}}^r$, the training process for the back end mirrors that of offline TS-VAD. Specifically, we repeat and concatenate the target speaker embeddings with the frame-level embeddings as input to the back end. Since we know the speaker order in the ground truth labels $\mathbf{Y} \in \{0,1\}^{N\times T}$, we maintain this order across all speaker embeddings. Consequently, the speaker order in the output $\hat{\mathbf{Y}} \in (0,1)^{N\times T}$ matches the labels, eliminating the need to address permutation ambiguity as required in the EEND framework. We then optimize the entire model using binary cross-entropy (BCE) loss.

It's important to mention that we exclude silent regions from the input audio beforehand, as shown in Fig. 3. This is because when there's an abundance of silence in the data, the aggregation method may struggle to gather sufficient target speaker embeddings within a relatively short segment for training purposes.

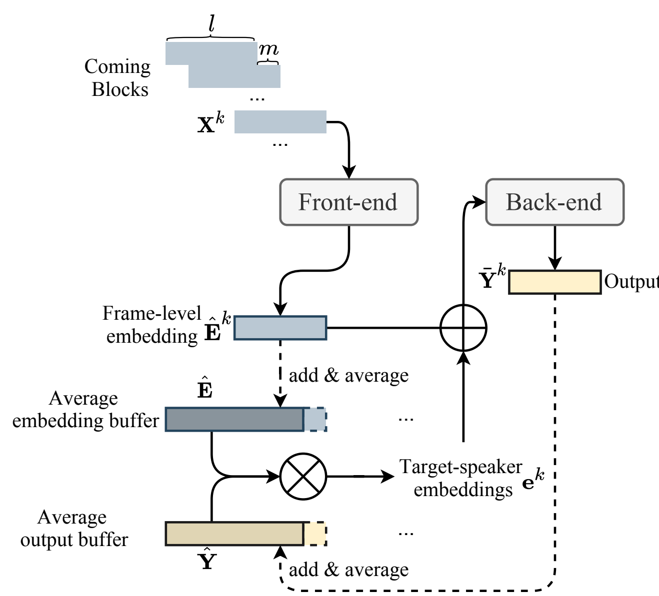


Fig. 4. Inference process of OTS-VAD with embedding buffer and output buffer. Dash line means updating the buffer after obtaining the output from current block.

3) Inference process: The inference process is quite different from the training process as the target speaker embeddings are accumulated during the inference process. Specifically, during inference, target speaker embeddings are aggregated incrementally as the OTS-VAD system systematically processes each successive block of audio. Here we also assume that the silence regions are removed in advance. For clarity, we assume that the features and output have the same time resolution, and we will explain the actual time resolution for all features and output in Section IV.

As shown in Fig. 4, the length and shift of each incoming block are denoted by l and m, respectively. Additionally, we maintain a frame-level embedding buffer, $\hat{\mathbf{E}} \in \mathbb{R}^{D\times T}$, and an output buffer, $\hat{\mathbf{Y}} \in (0,1)^{N\times T}$, to store the frame-level embeddings and corresponding outputs over time, where T is the total output length, D is the embedding size, and N is the preset number of speakers. Both buffers are initialized with zeros. Initially, the block size begins at m and gradually increases until there is enough signal to form a full block of size l. Thereafter, the block size remains fixed at l, advancing with a consistent shift of m. Notably, m also represents the latency of the OTS-VAD system.

For the first block, we believe that it only contains one speaker as m is set to a very small value. This assumption makes it very easy to initialize the system using the first block. The first block is fed into the front-end module, producing the initialized frame-level embeddings. In addition, we assign the output of the first speaker with one, and other values in the output are set to zero:

$$\hat{\mathbf{y}}^1_{n,t} = \begin{cases} 1 \quad , \quad n = 1 \\ 0 \quad , \quad n = 2,3,...,N \end{cases} \tag{4}$$

where $\hat{\mathbf{y}}^1$ is the output of the first block, n and t are the index of speaker and frame, respectively. Finally, both the frame-level embeddings and the output are added to both buffer at the corresponding position and averaged at the overlapping region between the blocks, as shown in Fig. 4.

For the following blocks, assume that we already have the averaged embeddings buffer $\hat{\mathbf{E}}$ and output buffer $\hat{\mathbf{Y}}$ until timestamp $t^{'}$, the n-th target speaker embedding can be easily computed by using the first $t^{'}$ frames of these two buffers:

$$\mathbf{e}_n = \frac{\hat{\mathbf{E}}\bar{\mathbf{Y}}_n^{\top}}{\sum_{t=1}^{t^{'}} \bar{Y}_{n,t}}, \tag{5}$$

where $\bar{\mathbf{Y}} \in \{0,1\}^{N\times l}$ is the output binarized from $\hat{\mathbf{Y}}$ with an **upper threshold** $thres_{\text{upper}}$. Here the upper threshold is usually greater than 0.5, which means that we only use the frame-level embedding with higher speaker existence probabilities for target speaker embedding extraction to ensure the purity of embeddings. Given the target speaker embeddings and frame-level embeddings, the back-end module can easily predict the output for current block. If a new speaker presents in the last m frames without overlapping speech, the OTS-VAD cannot recognize any speaker given previous target speaker embedding, thus all the values of the output are closed to zero in the last m frames. In this way, we can know that a new speaker appears, and we can assign the output with the value of the upper threshold for the new speaker in the last m frames. If the number of speakers has already reached the maximum, we keep the output unchanged. Usually, if we find that all values of the output in the last m frames are lower than a **lower threshold** $thres_{\text{lower}}$, we believe that there is a new speaker, where the lower threshold is set to no more than 0.5. Algorithm 1 shows the detailed inference process.

Latency is a crucial factor in online applications. With an unlimited number of incoming blocks, it becomes impractical to store all frame-level embeddings in the buffers, and the computational cost of calculating the target speaker embedding also becomes prohibitive. In such cases, we store only those frame-level embeddings in the buffer that have a high probability for at least one speaker, discarding the rest. This approach significantly reduces computational costs when processing extremely long audio signals.

Another solution for reducing the latency is to accumulate the frame-level embedding for each speaker and record the sum of all frame-level embeddings, referred to accumulated total embedding $\tilde{\mathbf{e}}_n$, as shown in Fig. 5. At the same time, the number of frames F is also recorded. Then the target speaker embedding $\mathbf{e}_n$ can be directly computed by:

$$\mathbf{e}_n^{k+1} = \frac{\tilde{\mathbf{e}}_n^k}{F^k}, \tag{6}$$

where k is the index of block and $\mathbf{e}_n^k$ is the target speaker embedding for inference of the k-th block. When there comes a series of frame-level embeddings $\hat{\mathbf{E}}^k = [\hat{\mathbf{e}}_1, ..., \hat{\mathbf{e}}_T] \in \mathbb{R}^{D\times T}$ and the corresponding binarized prediction $\bar{\mathbf{Y}}^k \in \{0,1\}^{N\times T}$ from the k-th block, the accumulated embedding and number of frame for n-th speaker can be updated by:

$$\begin{aligned} \tilde{\mathbf{e}}_n^k &= \tilde{\mathbf{e}}_n^{k-1} + \hat{\mathbf{E}}^k(\bar{\mathbf{Y}}_n^k)^{\top}, \\ F_n^k &= F_n^{k-1} + \sum_{t=0}^{T} \bar{Y}_{n,t}^k, \end{aligned} \tag{7}$$

where $\hat{\mathbf{E}}^k$ is the frame-level embeddings of k-th block and $\bar{\mathbf{Y}}_n^k$ is the n-th row of $\bar{\mathbf{Y}}^k$. Similar to the buffer-based inference process mentioned in Fig. 4, only the frame with the probabilities greater than the upper threshold will be used for updating. If the probabilities in the last M frames are lower than the lower threshold, we notify a new speaker coming by setting his/her probabilities to the upper threshold.

The first strategy uses a buffer to continuously store average frame-level embeddings and corresponding outputs. By retaining a comprehensive record of embeddings, this method enhances precision in target speaker inference, potentially improving overall system performance. However, a primary limitation of this approach is the increased computational time and memory requirements when processing extensive input features. In contrast, the second approach streamlines the inference process by calculating only the accumulated embeddings and their corresponding frame counts. While this method offers significant computational efficiencies, especially for long input sequences, it may lead to a slight reduction in performance accuracy. Consequently, the optimal choice between these strategies depends on the specific application

Algorithm 1 Inference process of OTS-VAD with embedding buffer (Fig. 4)

Require: Acoustic feature $\mathbf{X} \in \mathbb{R}^{H \times L}$; block length l; block shift m, num_of_speaker = 0,
procedure INFERENCE($\mathbf{X}$, l and m)
    Initialize $\hat{\mathbf{Y}} \in (0,1)^{N \times t'}$ and $\hat{\mathbf{E}} \in \mathbb{R}^{D \times t'}$ with empty tensor ($t' = 0$)
    Initialize ct $\in \mathbb{R}^{m}$ with $\vec{1}$ ▷ The frame count at each step

    for start, end, $\mathbf{X}^k$ in chunking($\mathbf{X}$, l, m) **do** ▷ Get $\mathbf{X}^k$ every m step
        # Block-wise inference
        $\mathbf{E}^k \leftarrow$ Front_end($\mathbf{X}^k$)
        if $\hat{\mathbf{Y}}$ is empty **then** ▷ Init: only one speaker in the first block with m frames
            $\hat{\mathbf{Y}}[0] \leftarrow \vec{1}$
            $\hat{\mathbf{E}} \leftarrow \hat{\mathbf{E}}^k$
        end if
        $\bar{\mathbf{Y}} \leftarrow$ binarize($\hat{\mathbf{Y}}$, upper_threshold) ▷ Take the binary output from predictions
        $\mathbf{e}^k \leftarrow \bar{\mathbf{Y}}$@$\hat{\mathbf{E}}$.transpose$(0, 1)/$torch.sum$(\bar{Y}, \text{dim} = 1, \text{keepdim=True})$
        $\hat{\mathbf{Y}}^k \leftarrow$ Back_end($\mathbf{E}^k$, $\mathbf{e}^k$)

        # Updating the memory
        if ($\hat{Y}$[:, end-m:end] < lower_threshold).all() and num_of_speaker < max_n_speaker **then**: ▷ Find a new speaker
            num_of_speaker $\leftarrow$ num_of_speaker + 1
            $\hat{Y}$[num_of_speaker, end-m:end] $\leftarrow \vec{1}$
        end if
        if ct.size(1) < end **then**
            $\hat{\mathbf{Y}}$[start:end-m] $\leftarrow \hat{\mathbf{Y}}$[start:end-m] $\times$ ct[start:end-m] ▷ Recover the accumulated values
            $\hat{\mathbf{E}}$[start:end-m] $\leftarrow \hat{\mathbf{E}}$[start:end-m] $\times$ ct[start:end-m]
            ct $\leftarrow$ padding_zeros(ct, m) ▷ Reserve space to save results from current block
            $\hat{\mathbf{Y}}_{\text{total}} \leftarrow$ padding_zeros($\hat{\mathbf{Y}}_{\text{total}}$, m)
            $\hat{\mathbf{E}}_{\text{total}} \leftarrow$ padding_zeros($\hat{\mathbf{E}}_{\text{total}}$, m)
            ct[start:end] $\leftarrow$ ct[start:end] + 1
            $\hat{\mathbf{Y}}$[start:end] $\leftarrow$ ($\hat{\mathbf{Y}}$[start:end] + $\hat{\mathbf{Y}}^k$) / ct[start:end] ▷ Adding and average
            $\hat{\mathbf{E}}$[start:end] $\leftarrow$ ($\hat{\mathbf{E}}$[start:end] + $\hat{\mathbf{E}}^k$) / ct[start:end] ▷ Adding and average
        end if
    end for
    Return binarize($\hat{\mathbf{Y}}$, 0.5)
end procedure

requirements, balancing accuracy with computational resource constraints.

B. Incorporation of MFA-Conformer as front-end

To further investigate the impact of a front end with fewer parameters, we replace the ResNet with a lightweight Conformer, originally developed for ASR tasks. Recent studies have shown that Conformer-based architectures can achieve excellent performance on speaker verification tasks through multi-scale feature aggregation [65, 66]. This architecture demonstrates better performance with fewer parameters compared to ResNet and ECAPA-TDNN [65].

Multi-scale feature aggregation (MFA) is a method that joins together output feature maps from every frame-level module within a speaker embedding network before pooling at the utterance-level. Research indicates that when used with TDNN-based networks, this method enhances performance, which suggests that features from the lower-levels offer valuable speaker-related information [15].

To incorporate the Conformer encoder into speaker verification tasks, the MFA-Conformer introduces an MFA module within the Conformer encoder [65]. This module combines all frame-level outputs from the Conformer blocks before passing them to the pooling layer. After obtaining this combined frame-level feature map, attentive statistics pooling is applied to generate an utterance-level representation [67]. The speaker embedding is then derived by applying batch normalization and a fully connected layer to this utterance-level representation. When replacing the ResNet with the MFA-Conformer in the OTS-VAD framework, we use the combined frame-level features before pooling as the frame-level embeddings, which are output by the MFA layer shown in Fig. 2. However, the time resolution of the outputs from the MFA layer is too high for use as frame-level embeddings, as processing these

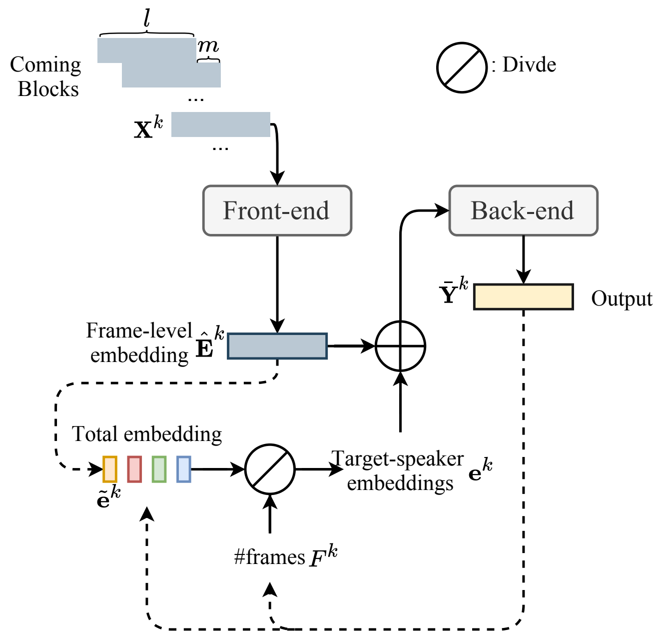


Fig. 5. Inference process of OTS-VAD with accumulated embedding and the total number of frames. Dash line means updating the accumulated embedding and the number of frames after obtaining the output from current block.

features would require substantial GPU memory in the back-end module. To address this, we add two 1D convolutional layers to reduce the time resolution from $\frac{1}{2}$ to $\frac{1}{8}$, with each layer halving the resolution. Finally, the output from the last 1D convolutional layer matches the time resolution of the ResNet-based front end.

C. Multi-channel Extension

Cross-channel attention mechanism has achieved significant success in multi-channel speech signal processing domains, including speech enhancement [68], speech separation [69, 70], speech recognition [71], and speaker diarization [72]. Their efficacy lies in their ability to comprehend non-linear contextual relationships across channels both temporally and contextually.

Given C channels of Fbank sequence denoted by $\mathcal{X} = (\mathbf{X}^1, ..., \mathbf{X}^C)$, the target-speaker embedding from N target speakers represented as $\mathcal{E} = (\mathbf{E}^1, ..., \mathbf{E}^C)$, where $\mathbf{E}^c = [\mathbf{e}^c_1, \mathbf{e}^c_2, ..., \mathbf{e}^c_N]$ are the target speaker embeddings extracted from the c-th channel, the corresponding target speaker decision is denoted by $\mathbf{Y} = (\mathbf{y}_1, ..., \mathbf{y}_T)$. Here, $\mathbf{X}_i \in \mathbb{R}^{L\times H}$ has L frames, $\mathbf{E}^c \in \mathbb{R}^{N\times D}$ signifies the D-dimensional speaker embedding from N target speakers in the c-th channel, and $\mathbf{y}_t \in \{0, 1\}^N$ encapsulates the target speaker decision at time step t with a dimension of N. The front-end ResNet takes the C-channels of Fbank sequence as input, producing the frame-level speaker embedding sequence, denoted by $\hat{\mathcal{E}} = (\hat{\mathbf{E}}^1, ..., \hat{\mathbf{E}}^C)$, where $\hat{\mathbf{E}}^c = [\hat{\mathbf{e}}^c_1, \hat{\mathbf{e}}^c_2, ..., \hat{\mathbf{e}}^c_T]$. Subsequent operations entail repeating the target-speaker embedding T times and the frame-level speaker embedding N times. Post these operations, the two speaker embeddings are concatenated in the embedding dimension, forming the tensor $\mathcal{G} = (\mathbf{G}^1, ..., \mathbf{G}^C)$, where $\mathbf{G}^c \in \mathbb{R}^{T\times N\times 2D}$.

The cross-channel self-attention mechanism ingests the concatenated speaker embedding $\mathbf{G}_{\text{in}}$:

$$\mathbf{G}_{\text{in}} = \text{concat}(\mathcal{G}) \in \mathbb{R}^{T\times N\times C\times 2D} \tag{8}$$

$$\mathbf{Q}^i = \mathbf{G}_{\text{in}}\mathbf{W}^i_Q + \mathbf{b}^i_Q \tag{9}$$

$$\mathbf{K}^i = \mathbf{G}_{\text{in}}\mathbf{W}^i_K + \mathbf{b}^i_K \tag{10}$$

$$\mathbf{V}^i = \mathbf{G}_{\text{in}}\mathbf{W}^i_V + \mathbf{b}^i_V, \tag{11}$$

with $\mathbf{Q}^i$, $\mathbf{K}^i$, and $\mathbf{V}^i$ representing the query, key, and value matrices, respectively, for the i$^{\text{th}}$ attention head. The weight matrices $\mathbf{W}^i \in \mathbb{R}^{M\times 2D}$ and bias vectors $\mathbf{b}^i \in \mathbb{R}^M$ correspond to the i$^{\text{th}}$ head. The scaled dot-product attention mechanism is then applied:

$$\text{Attention}(\mathbf{Q}^i, \mathbf{K}^i, \mathbf{V}^i) = \text{softmax}\frac{\mathbf{Q}^i(\mathbf{K}^i)^\intercal}{\sqrt{n}}\mathbf{V}^i, \tag{12}$$

where $n = M$. Subsequently, a position-wise feed-forward layer complemented by a ReLU activation function is employed, incorporating layer normalization and residual connections at each layer. The resultant output is denoted by $\mathbf{G}_{\text{out}} \in \mathbb{R}^{T\times N\times C\times 2D}$. This output is averaged across the channel dimension via a global average pooling layer:

$$\mathbf{G}' = \frac{1}{C}\sum_{i=1}^{C}\mathbf{G}_{\text{out},i}, \tag{13}$$

where $\mathbf{G}' \in \mathbb{R}^{T\times N\times 2D}$. The back-end model subsequently processes this aggregated speaker embedding $\mathbf{G}'$, akin to how the single-channel TS-VAD processes the concatenated speaker embedding.

IV. Experimental Setup

A. Dataset

The experiments are conducted on the DIHARD III dataset [73] and the AliMeeting dataset [74]. The DIHARD III dataset is a multi-domain dataset containing 34.15 hours of training data and 33.01 hours of evaluation data across 11 different domains. The number of speakers per recording ranges from 1 to 9, with an average speech overlap ratio of 10.75% in the development set and 9.37% in the evaluation set. The AliMeeting dataset is a far-field, 8-channel dataset comprising 118.75 hours of speech data, divided into 104.75 hours for training, 4 hours for evaluation, and 10 hours for testing. Each meeting session includes 2 to 4 participants, with a session duration of approximately 30 minutes. The average speech overlap ratio is 42.27% for the training set and 34.76% for the evaluation set.

Existing datasets for diarization tasks often lack sufficient complexity for effective multi-speaker learning, especially for deep learning-based speaker diarization models. Consequently, data simulation is essential to train a model with robust generalization capabilities. Previous research found that speaker change distribution follows a Poisson Stochastic Process (PSP) [75], which is commonly used to simulate training data for EEND [22]. Recently, a more comprehensive approach for

TABLE I
DER (%) AND JER (%) OF METHODS WITH RESNET AND CONFORMER FRONT-ENDS ON DIFFERENT DATASETS WITH THE EMBEDDING BUFFER-BASED INFERENCE PROCESS IN FIG. 4.

Dataset	Pretraining	Front-end	Block shift	Block length 2s DER	2s JER	4s DER	4s JER	8s DER	8s JER	16s DER	16s JER
DIHARD III Eval	✓	ResNet	0.4s	16.20	35.50	14.47	34.16	13.83	34.57	13.77	35.09
			0.8s	15.31	34.65	13.96	34.05	13.57	33.87	**13.31**	33.83
			1.6s	16.60	37.00	14.40	35.17	14.12	34.76	13.59	34.34
		Conformer	0.4s	16.39	40.45	15.55	39.33	15.77	41.05	16.30	42.22
			0.8s	16.18	40.93	15.66	40.08	**15.33**	40.10	15.86	40.98
			1.6s	16.97	38.45	16.01	41.71	16.36	42.21	17.02	42.82
DIHARD III Eval	✗	ResNet	0.4s	19.83	44.34	16.86	40.13	15.58	39.12	16.02	39.91
			0.8s	20.25	44.06	17.06	40.32	15.35	38.43	15.48	39.23
			1.6s	21.10	43.58	16.82	40.55	15.51	39.23	**14.67**	38.46
		Conformer	0.4s	22.54	48.69	20.57	46.70	19.79	47.73	20.91	50.65
			0.8s	22.74	49.91	20.17	45.73	18.11	44.23	18.63	46.09
			1.6s	26.56	49.49	19.60	45.30	**17.97**	43.85	18.28	44.68

simulating speaker diarization data was proposed, which considers various statistical properties to create data that reflects real-world patterns [76]. Our strategy is simpler compared to previous methods. For instance, to create our training data for the DIHARD III dataset, we use the ground truth labels from the DIHARD III development dataset. We first derive a label matrix $\mathbf{Y} \in 0,1^{N\times T}$ for each recording in the DIHARD III development dataset, where N represents the number of speakers and T signifies the duration of an utterance after excluding non-speech regions. The simulated data is then generated on-the-fly in an online manner, where we randomly select a segment according to the label and fill the active regions with continuous, non-overlapping speech segments from the VoxCeleb dataset. Similarly, for the AliMeeting dataset, the data is directly simulated from the AliMeeting training set.

We perform data augmentation with MUSAN [77] and RIRs corpus [78]. 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.

B. Network Configuration

For the ResNet-based OTS-VAD model, the front-end architecure is the same as the ResNet34 in [79]. After obtaining the temporal feature map, we apply the global statistics pooling over each frame of the feature map to produce the frame-level representation. Finally, the linear layer with 256 units projects the representation to frame-level embedding. For the back-end, it contains a 6-layer 8-head Conformer Encoder with a 512 dimensional feed-forward layer, and the dropout is set to 0.1. The hidden size of the single BiLSTM layer is 256. The fully-connected layer projects the output from BiLSTM to a N-dimensional vector, where N is the number of target speakers. In our experiment, N is set to 8 for DIHARD III dataset and 4 for Alimeeting dataset. The total number of parameters in the ResNet-based front-end is 20.54M.

For the Conformer-based OTS-VAD model, we employ the Conformer implemented by NeMo toolkit [80] and we follow the speaker verification training protocal in [66]. The NeMo Conformer is available in three variants: small, medium, and large. For our research, we opted for the 'small' variant[1], characterized by a convolution subsampling rate of 14 and a uniform kernel size of 31 within its convolution modules. This version encompasses 16 Conformer layers with an encoder dimension of 176, accommodates 4 attention heads, and comprises 704 linear hidden units. We only use the first 8 layers for the speaker verification pretraining. The features processed by MFA layer are then fed to two 1d convolution layers, each of which has kernel size of 3, stride of 2 and padding of 1. The back-end is the same as the ResNet-based OTS-VAD model. The total number of parameters in the Conformer-based front-end is 9.38M.

For the mutli-channel OTS-VAD, we employ the same ResNet34 as the front-end model. The back-end contains a cross-channel Conformer Encoder and a cross-frame Conformer Encoder, each of which has 3 Conformer layers and 4 heads. The decision to reduce the Encoder size stems from the substantial GPU memory demands associated with multi-channel data processing. While this reduction may lead to a slight performance decline, it is offset by the advantages conferred by analyzing multi-channel data.

C. Training and Inferring Details

1) Front-end pretraining: Both the ResNet and the Conformer front-end models are pretrained on the VoxCeleb 2 dataset [81]. For the ResNet model, we follow the training protocol mentioned in [79] and achieve 0.814% EER on the Vox-O trial. For the Conformer model, we follow the training protocol mentioned in [66], which has 0.65% EER on the Vox-O trial.

[1] https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_conformer_ctc_small

2) Training process: At the begining, the front-end module is initialized from a pretrained ResNet34- or Conformer-based speaker embedding model, and other parts are randomly initialized. The training process contains three different stage:

- Stage 1: Keep the front-end module frozen, and we only train the back-end module with the simulated data for 100,000 steps. The last model will be used for next stage.
- Stage 2: Unfreeze the front-end module and train the whole OTS-VAD model with 20% of the real data and 80% of the simulated data for 50,000 steps. The model will be validated every 500 steps and the best model with the lowest DER will be used for next stage.
- Stage 3: Continue to train the whole OTS-VAD model with only real training data for 50,000 steps. The model will be validated every 500 steps.

For DIHARD III, the real training data is DIHARD III development set. For Alimeeting, the real training data is Alimeeting training set.

3) Inference process: For inference, we tested different block lengths $l \in 2s, 4s, 8s, 16s$ and block shifts $m \in 0.4s, 0.8s, 1.6s$. Note that the block shift also represents the latency of the OTS-VAD model. The upper and lower thresholds were tuned by grid search with a fixed block length and shift, set to 16s and 0.8s, respectively. Specifically, the lower threshold was tuned between 0.1 and 0.4, and the upper threshold between 0.6 and 0.9, both with a step size of 0.05. After determining the thresholds that resulted in the lowest diarization error rate (DER), we applied these thresholds for inference with other block lengths and shifts. Finally, the output of each block was averaged, binarized using the optimal upper threshold (0.5), and converted to an RTTM file for scoring. During inference, an additional VAD module was needed to remove silent regions, and for simplicity, we used oracle VAD in our experiments.

Both two different inference processes mentioned in Fig. 4 and 5 are evaluated. For the buffer-based inference process, we only evaluate it on the DIHARD III dataset in Fig. 4. The reason is that the average duration of Alimeeting dataset is about 30 minutes, which requires large memory to saving the buffer and long time for inference. For the accumulation-based inference process, we provide comprehensive evaluation for both datasets.

4) Evaluation metric: The OTS-VAD model is evaluated on the evaluation set of both datasets, where we use the Diarization Error Rate (DER) and Jaccard Error Rate (JER) as the evaluation metric. For the DIHARD III dataset, we follow the evaluation protocol in the DIHARD III challenge [73], where no forgiveness collar will be applied to the reference segments prior to scoring. For Alimeeting, we follow the evaluation protocol in the Alimeeting challenge [74], where a forgiveness collar of 0.25 is employed. All overlapping speech will be evaluated.

V. Experimental Results and Discussion

A. Detailed performance with different block length and shift

Tab. I and II show the results of the OTS-VAD model with different front-ends on the DIHARD III dataset and the Alimeeting dataset with different inference strategies as mentioned in Fig. 4 and Fig. 5, where we evaluated the model with different block length and shift.

Overall, with the buffer-based inference process, the best DER performance on DIHARD III dataset is 13.31%. For the first channel of alimeeting dataset, the lowest DER is 5.05% and 4.96% for Eval and Test set, respectively. And, if incorporated with the multi-channel data, the DER will be further reduced to 4.71% and 4.50% for Alimeeting Eval and Test set, respectively.

For different front-end modules, as Tab. I and II show, ResNet front-end usually achieves better DER and JER scores compared to the Conformer front-end across most block lengths and block shifts. ResNet front-end achieves its best DER score at a block length of 8s or 16s.

For different inference processes, the buffer-based inference process shows a better performance than the accumulation-based inference process does in most conditions. The choice of the inference process might be application-specific, depending on the specific goals and constraints of the task. For the DIHARD III dataset, it only shows a tiny performance degradation, e.g., 13.65% compared with 13.31%.

Block length and shift also matter for a better performance. For DIHARD III, the best performance in Tab. I and II are typically achieved with an 8s or 16s block length. Conversely, for Alimeeting, longer block lengths consistently yield improved and stable performance. This contrast may be attributed to the heterogeneous domains of the DIHARD III dataset, making hyper-parameter tuning more difficult and resulting in less stable outcomes. Furthermore, inference with lower block shifts exhibit enhanced stability, while longer block shifts introduce greater performance variability.

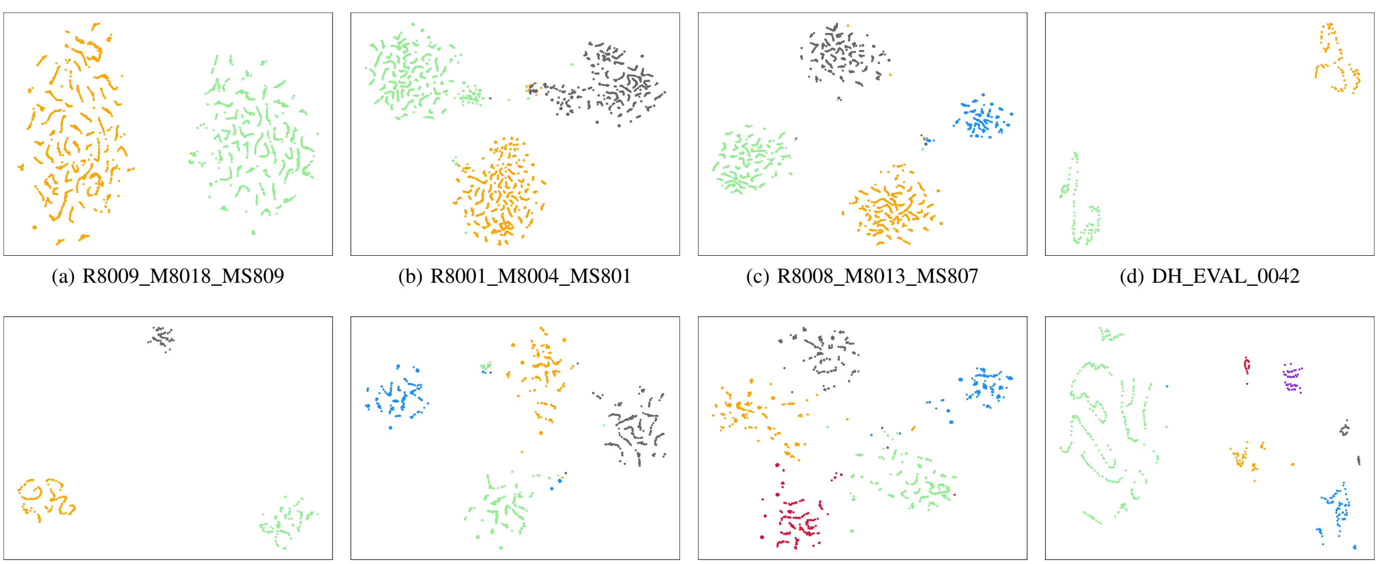
We also evaluate the multi-channel data on the cross-channel-based model, which improves the performance on the Eval and Test set by relatively 14% and 9%, repectively. In addition, it can provide a more stable performance under different condition, e.g., it can still show a satisfying performance with block length of 2s on the Test set. Compared to offline TS-VAD [82], the improvement from multi-channel extension in online TS-VAD is moderate. This is likely due to the reduced encoder size used to manage GPU memory requirements during training (3 layers, 4 heads vs. 6 layers, 8 heads). We believe performance could be further enhanced if the same encoder configuration were employed for multi-channel extension.

B. Comparison with other offline and online systems

Tab. III presents a comparative analysis of various speaker diarization systems' performances on the DIHARD III dataset, measured by Diarization Error Rates (DERs). Both offline and online systems with oracle VAD are included. Among offline systems, the Seq2Seq TS-VAD achieves state-of-the-art performance with a DER of 10.77%. For online systems, the ResNet-based OTS-VAD demonstrates the best performance, achieving a DER of 13.31%. We also evaluated the proposed model without oracle VAD by training an additional ResNet-based VAD model identical to that described in [79]. Results indicate

TABLE II
DER (%) AND JER (%) OF METHODS WITH RESNET AND CONFORMER FRONT-ENDS ON DIFFERENT DATASETS WITH THE ACCUMULATION-BASED INFERENCE PROCESS IN FIG. 5.

Dataset	**Pretraining**	**Front-end**	**Block shift**	**Block length**							
				2s		4s		8s		16s	
				DER	JER	DER	JER	DER	JER	DER	JER
DIHARD III Eval	✓	ResNet	0.4s	15.80	33.95	14.53	32.46	**13.65**	33.00	13.65	33.22
			0.8s	16.00	34.16	14.51	33.13	13.68	32.92	14.07	33.14
			1.6s	17.44	37.74	15.17	35.52	14.39	35.09	14.36	34.84
		Conformer	0.4s	19.34	36.27	17.18	35.65	17.90	37.46	17.63	38.84
			0.8s	19.00	36.03	**16.33**	34.78	16.56	36.71	17.24	38.56
			1.6s	18.58	37.55	16.89	36.98	16.88	39.11	17.23	40.51
DIHARD III Eval	✗	ResNet	0.4s	22.70	43.73	18.10	39.26	**15.80**	36.67	16.79	39.47
			0.8s	21.52	43.23	17.26	38.10	16.14	37.27	16.46	38.38
			1.6s	22.27	44.65	17.56	38.90	16.39	37.87	16.15	38.54
		Conformer	0.4s	30.21	47.96	22.43	43.48	20.95	44.07	21.65	47.07
			0.8s	27.51	47.39	21.91	42.77	**20.05**	43.84	21.10	47.24
			1.6s	27.35	50.09	21.61	43.84	20.18	44.75	21.76	47.94
Alimeeting Eval (1st Channel)	✓	ResNet	0.4s	16.77	25.66	10.22	21.92	7.26	18.00	**5.37**	15.32
			0.8s	15.92	26.83	10.27	21.93	8.57	19.85	5.67	15.80
			1.6s	12.60	23.41	10.06	22.26	8.62	20.06	8.43	19.75
		Conformer	0.4s	24.74	36.59	13.60	23.17	7.90	18.08	**5.05**	13.91
			0.8s	27.92	39.47	16.72	30.03	9.95	22.05	6.61	16.81
			1.6s	32.06	45.03	17.99	32.60	9.58	21.88	8.78	21.20
Alimeeting Test (1st Channel)	✓	ResNet	0.4s	15.62	26.40	11.46	21.14	12.39	19.05	6.87	14.65
			0.8s	13.84	25.26	13.74	21.26	8.86	14.82	**4.96**	12.54
			1.6s	14.32	27.77	11.25	20.50	7.69	13.81	5.46	13.57
		Conformer	0.4s	28.79	37.83	20.20	26.13	10.93	18.22	**6.38**	13.89
			0.8s	31.21	39.74	17.62	24.21	10.59	18.02	7.29	15.42
			1.6s	39.02	47.57	17.37	28.13	12.41	21.97	9.66	19.31
Alimeeting Eval (Multi Channels)	✓	ResNet	0.4s	16.68	28.90	9.32	19.31	8.97	19.09	5.26	13.05
			0.8s	16.27	27.67	8.77	18.03	9.06	19.79	**4.71**	12.03
			1.6s	12.38	23.81	10.51	22.51	9.22	20.11	4.81	12.17
Alimeeting Test (Multi Channels)	✓	ResNet	0.4s	8.69	19.10	6.48	15.65	5.50	13.80	4.97	12.16
			0.8s	10.70	21.25	9.38	18.15	8.08	13.05	**4.50**	12.95
			1.6s	12.61	23.31	11.14	19.80	7.07	13.04	4.50	12.60



(a) R8009_M8018_MS809 (b) R8001_M8004_MS801 (c) R8008_M8013_MS807 (d) DH_EVAL_0042

(e) DH_EVAL_0183 (f) DH_EVAL_0039 (g) DH_EVAL_0044 (h) DH_EVAL_0196

Fig. 6. t-SNE visualization of the target speaker embedding in each 16-second block during inference process, where the cosine is employed as the distance metric. (a), (b) and (c) are the samples from Alimeeting Eval dataset, each of which contains two, three and four speakers, respectively. (d), (e), (f), (g) and (h) are the samples from DIHARD III dataset, each of which contains $2 \sim 6$ speakers, respectively.

that the model remains robust, maintaining performance even with potentially inaccurate VAD results.

Tab. IV presents a comparison of our model with other systems on the AliMeeting dataset. For offline systems, we include the official baseline [74] and the winning system [82]. For online systems, we did not find other online systems evaluated on the AliMeeting dataset, but we can make a direct comparison with the offline TS-VAD system. On the AliMeeting evaluation set, both single-channel and multi-channel OTS-VAD demonstrate similar performance to the single-channel TS-VAD. On the test set, the single-channel TS-VAD appears to overfit, while the multi-channel OTS-VAD maintains strong performance compared to the offline multi-channel TS-VAD system.

TABLE III
DERs (%) OF DIFFERENT SYSTEMS ON THE DIHARD III DATASET.

Methods	Latency	# speakers ≤ 4	# speakers ≥ 5	Total
Offline + oracle VAD				
VBx + resegmentation [53]	-	N/A	N/A	16.2
Seq2Seq TS-VAD [62]	-	6.65	25.90	10.77
Online + oracle VAD				
Zhang et al. [24]	0.5s	N/A	N/A	19.57
Core samples selection [83]	2s	N/A	N/A	19.3
EEND-GLA-Large [31]	1s	8.85	38.86	14.70
Conformer-based OTS-VAD	0.8s	9.23	40.53	15.33
ResNet-based OTS-VAD	0.8s	**8.78**	**32.03**	**13.31**
Online + system VAD				
EEND-GLA-Large [31]	1s	14.81	45.17	20.73
ResNet-based OTS-VAD	0.8s	14.25	38.99	19.07

C. Importance of pretraining

We also evaluate if the speaker verification pretraining of the front-end module is necessary. As Tab. I and II show, the model trained from scratch exhibits poorer performance, particularly for very short block lengths. This suggests that, without speaker verification pretraining, the front end struggles to generalize well for frame-level embeddings.

In comparison, the Conformer architecture demonstrates lower performance than ResNet, both with and without pre-training. A potential reason may be the removal of the last attentive pooling layer and the addition of two Conv1D layers to adjust the time resolution from 20ms to 80ms. These Conv1D layers are not part of the pretrained model, which may affect performance.

D. Visualization of target speaker embedding

To evaluate how well the model learns frame-level embeddings to form the target-speaker embedding, we present t-SNE visualizations of the ResNet-based target speaker embeddings for each block with varying numbers of speakers in Fig. 6. Specifically, Figs. 6a, 6b, and 6c are from the AliMeeting dataset with 2 to 4 speakers, while Figs. 6d through 6h are from the DIHARD III dataset with 2 to 6 speakers. Each point in the figure represents the mean embedding of the frame-level embeddings within a block during the inference stage. As shown in Fig. 6, the target speaker embeddings are well separated, even without a specific training objective for embedding separation.

E. Training/inference time and real-time factor

Our experiments were conducted on an NVIDIA GeForce RTX 3090 GPU. Training took approximately 2 days using two GPUs for each single-channel dataset and 4 days with eight GPUs for the multi-channel dataset. For the accumulation-based inference process, we used a single NVIDIA GeForce RTX 3090 GPU. For the single-channel ResNet-based OTS-VAD, the inference time for a 16s block is about 0.064s, while for the multi-channel ResNet-based OTS-VAD, the inference time for a 16s block is approximately 0.186s. Given that latency corresponds to the block shift, the real-time factors (RTFs) can be calculated as $\frac{t}{m}$, where t is the average inference time per block and m is the block shift. The RTFs for single-channel and multi-channel OTS-VAD with a 0.4s block shift are 0.16 and 0.465, respectively, demonstrating that the proposed method is feasible for online inference.

VI. Conclusion

Here's a refined version of your paragraph for clarity and flow:

In this paper, we introduce an innovative neural framework, OTS-VAD, designed for online speaker diarization using a target speaker profile as guidance. Adapted from the offline TS-VAD framework, this online version employs self-generated target speaker embeddings to detect speaker activity. Evaluated on the DIHARD III and AliMeeting datasets, OTS-VAD consistently demonstrates strong performance across varied scenarios, establishing its reliability. Additionally, we enhanced the model's capabilities to handle multi-channel data, improving its resilience to different block lengths and shifts. We also transitioned from the ResNet front end to a more efficient Conformer encoder, reducing the model size with minimal performance degradation. Our results show that OTS-VAD not only meets but, in some cases, surpasses state-of-the-art benchmarks, approaching the performance of certain offline speaker diarization systems. Its real-time factor further suggests suitability for online applications.

However, the model has certain limitations. First, it currently lacks an integrated voice activity detection (VAD) component, which could impact applicability in real-world scenarios. An online VAD model could be easily incorporated with our proposed OTS-VAD model. During inference, we could detect speech frames with an online VAD model until enough frames are gathered to form a block for OTS-VAD inference. This could be achieved either by using an external online VAD model or by leveraging the output from the front end for speech frame detection. Additionally, the model's relatively large size and extended CPU inference time present challenges, though quantization techniques could help address these issues. Moving forward, our future work will focus on integrating a VAD module and optimizing the framework's size and inference strategy to better suit real-world applications.

TABLE IV
DERs (%) of different systems on the Alimeeting dataset, where oracle VAD is employed. SC refers to single channel and MC refers to multi channel

Methods	Latency	Alimeeting Eval set 2 spk	3 spk	4 spk	All	Alimeeting Test set 2 spk	3 spk	4 spk	All
Offline									
Official baseline [74]	-	4.84	10.90	23.18	15.24	3.00	6.36	30.59	15.60
SC TS-VAD [82]	-	0.89	6.63	5.47	4.11	-	-	-	-
MC TS-VAD [82]	-	0.61	3.16	3.05	**2.25**	0.15	5.19	4.37	**2.98**
Online									
ResNet-based SC OTS-VAD	0.8s	0.85	3.98	9.27	5.67	0.33	6.18	8.39	4.96
ResNet-based MC OTS-VAD	0.8s	0.77	4.04	7.45	**4.71**	0.43	11.60	4.73	**4.50**

References

[1] N. Kanda, C. Boeddeker, J. Heitkaemper, Y. Fujita, S. Horiguchi, K. Nagamatsu, and R. Haeb-Umbach, "Guided Source Separation Meets a Strong ASR Backend: Hitachi/Paderborn University Joint Investigation for Dinner Party ASR," in *Proc. Interspeech*, 2019, pp. 1248–1252.

[2] N. Kanda, Y. Gaur, X. Wang, Z. Meng, Z. Chen, T. Zhou, and T. Yoshioka, "Joint Speaker Counting, Speech Recognition, and Speaker Identification for Overlapped Speech of any Number of Speakers," in *Proc. Interspeech*, 2020, pp. 36–40.

[3] S. Meignier, D. Moraru, C. Fredouille, J.-F. Bonastre, and L. Besacier, "Step-by-step and integrated approaches in broadcast news speaker diarization," *Computer Speech & Language*, vol. 20, no. 2-3, pp. 303–330, 2006.

[4] J.-L. Gauvain, L. F. Lamel, and G. Adda, "Partitioning and transcription of broadcast news data," in *Proc. ICSLP*, 1998.

[5] S. Tranter and D. A. Reynolds, "Speaker diarisation for broadcast news," in *Proc. Odyssey*, 2004.

[6] Q. Jin and T. Schultz, "Speaker segmentation and clustering in meetings." in *Proc. Interspeech*, 2004, pp. 597–600.

[7] F. Yu, S. Zhang, Y. Fu, L. Xie, S. Zheng, Z. Du, W. Huang, P. Guo, Z. Yan, B. Ma *et al.*, "M2met: The icassp 2022 multi-channel multi-party meeting transcription challenge," in *Proc. ICASSP*, 2022, pp. 6167–6171.

[8] D. Liu and F. Kubala, "A cross-channel modeling approach for automatic segmentation of conversational telephone speech [automatic speech recognition applications]," in *Prof. ASRU*, 2003, pp. 333–338.

[9] P. Kenny, D. Reynolds, and F. Castaldo, "Diarization of telephone conversations using factor analysis," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 6, pp. 1059–1070, 2010.

[10] R. Lahiri, M. Kumar, S. Bishop, and S. Narayanan, "Learning domain invariant representations for child-adult classification from speech," in *Proc. ICASSP*, 2020, pp. 6749–6753.

[11] D. Bone, S. Bishop, R. Gupta, S. Lee, and S. S. Narayanan, "Acoustic-Prosodic and Turn-Taking Features in Interactions with Children with Neurodevelopmental Disorders," in *Proc. Interspeech*, 2016, pp. 1185–1189.

[12] 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," *Computer Speech & Language*, vol. 72, p. 101317, 2022.

[13] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep Neural Network Embeddings for Text-Independent Speaker Verification," in *Proc. Interspeech*, 2017, pp. 999–1003.

[14] 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.

[15] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in *Proc. Interspeech*, 2020, pp. 3830–3834.

[16] 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 *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6529–6533.

[17] Q. Lin, W. Cai, L. Yang, J. Wang, J. Zhang, and M. Li, "DIHARD II is Still Hard: Experimental Results and Discussions from the DKU-LENOVO Team," in *Proc. Odyssey*, 2020, pp. 102–109.

[18] K. Boakye, B. Trueba-Hornero, O. Vinyals, and G. Friedland, "Overlapped speech detection for improved speaker diarization in multiparty meetings," in *Proc. ICASSP*, 2008, pp. 4353–4356.

[19] 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.

[20] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe, "End-to-End Neural Speaker Diarization with Permutation-Free Objectives," in *Proc. Interspeech 2019*, 2019, pp. 4300–4304.

[21] Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with self-attention," in *Proc. ASRU*, 2019, pp. 296–303.

[22] S. Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and K. Nagamatsu, "End-to-End Speaker Diarization for an Unknown Number of Speakers with Encoder-Decoder Based Attractors," in *Proc. Interspeech*, 2020, pp. 269–273.

[23] D. Dimitriadis and P. Fousek, "Developing on-line speaker diarization system." in *Proc. Interspeech*, 2017, pp. 2739–2743.

[24] Y. Zhang, Q. Lin, W. Wang, L. Yang, X. Wang, J. Wang, and M. Li, "Low-Latency Online Speaker Diarization with Graph-Based Label Generation," in *Proc. Odyssey)*, 2022, pp. 162–169.

[25] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, "Fully supervised speaker diarization," in *Proc. ICASSP*, 2019, pp. 6301–6305.

[26] Y. Xue, S. Horiguchi, Y. Fujita, S. Watanabe, P. García, and K. Nagamatsu, "Online end-to-end neural diarization with speaker-tracing buffer," in *Proc. SLT*, 2021, pp. 841–848.

[27] E. Han, C. Lee, and A. Stolcke, "Bw-eda-eend: Streaming end-to-end neural speaker diarization for a variable number of speakers," in *Proc. ICASSP*, 2021, pp. 7193–7197.

[28] W. Zhu and J. Pelecanos, "Online speaker diarization using adapted i-vector transforms," in *Proc. ICASSP*, 2016, pp. 5045–5049.

[29] M. Kolbæk, D. Yu, Z.-H. Tan, and J. Jensen, "Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 10, pp. 1901–1913, 2017.

[30] Y. Xue, S. Horiguchi, Y. Fujita, Y. Takashima, S. Watanabe, L. P. G. Perera, and K. Nagamatsu, "Online Streaming End-to-End Neural Diarization Handling Overlapping Speech and Flexible Numbers of Speakers," in *Proc. Interspeech 2021*, 2021, pp. 3116–3120.

[31] S. Horiguchi, S. Watanabe, P. García, Y. Takashima, and Y. Kawaguchi, "Online neural diarization of unlimited numbers of speakers using global and local attractors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 706–720, 2023.

[32] W. Wang, M. Li, and Q. Lin, "Online Target Speaker Voice Activity Detection for Speaker Diarization," in *Proc. Interspeech*, 2022, pp. 1441–1445.

[33] N. Ryant, M. Liberman, and J. Yuan, "Speech activity detection on youtube using deep neural networks." in *Proc. Interspeech*. Lyon, France, 2013, pp. 728–731.

[34] S. Thomas, S. Ganapathy, G. Saon, and H. Soltau, "Analyzing convolutional neural networks for speech activity detection in mismatched acoustic conditions," in *Proc. ICASSP*, 2014, pp. 2519–2523.

[35] G. Gelly and J.-L. Gauvain, "Optimization of rnn-based speech activity detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 3, pp. 646–656, 2017.

[36] M. Senoussaoui, P. Kenny, T. Stafylakis, and P. Dumouchel, "A study of the cosine distance-based mean shift for telephone speech diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*,

vol. 22, no. 1, pp. 217–227, 2013.

[37] 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.

[38] 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.

[39] M. Diez, L. Burget, F. Landini, S. Wang, and H. Černockỳ, "Optimizing bayesian hmm based x-vector clustering for the second dihard speech diarization challenge," in *Proc. ICASSP*, 2020, pp. 6519–6523.

[40] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007.

[41] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *Proc. ICASSP*, 2015, pp. 708–712.

[42] 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 *Proc. ICASSP*, 2021, pp. 5824–5828.

[43] M. Diez, L. Burget, F. Landini, and J. Černockỳ, "Analysis of speaker diarization based on bayesian hmm with eigenvoice priors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 355–368, 2019.

[44] Z. Huang, S. Watanabe, Y. Fujita, P. García, Y. Shao, D. Povey, and S. Khudanpur, "Speaker diarization with region proposal network," in *Proc. ICASSP*, 2020, pp. 6514–6518.

[45] J. M. Coria, H. Bredin, S. Ghannay, and S. Rosset, "Overlap-aware low-latency online speaker diarization based on end-to-end local segmentation," in *Proc. ASRU*, 2021, pp. 1139–1146.

[46] W. Xia, H. Lu, Q. Wang, A. Tripathi, Y. Huang, I. L. Moreno, and H. Sak, "Turn-to-diarize: Online speaker diarization constrained by transformer transducer speaker turn detection," in *Proc. ICASSP*, 2022, pp. 8077–8081.

[47] E. Fini and A. Brutti, "Supervised online diarization with sample mean loss for multi-domain data," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7134–7138.

[48] Y. C. Liu, E. Han, C. Lee, and A. Stolcke, "End-to-End Neural Diarization: From Transformer to Conformer," in *Proc. Interspeech*, 2021, pp. 3081–3085.

[49] S. Horiguchi, S. Watanabe, P. Garcia, Y. Xue, Y. Takashima, and Y. Kawaguchi, "Towards neural diarization for unlimited numbers of speakers using global and local attractors," in *Proc. ASRU*, 2021, pp. 98–105.

[50] K. Kinoshita, M. Delcroix, and N. Tawara, "Integrating end-to-end neural and clustering-based diarization: Getting the best of both worlds," in *Proc. ICASSP*, 2021, pp. 7198–7202.

[51] ——, "Advances in Integration of End-to-End Neural and Clustering-Based Diarization for Real Conversational Speech," in *Proc. Interspeech*, 2021, pp. 3565–3569.

[52] K. Kinoshita, M. Delcroix, and T. Iwata, "Tight integration of neural-and clustering-based diarization through deep unfolding of infinite gaussian mixture model," in *Proc. ICASSP*, 2022, pp. 8382–8386.

[53] H. Bredin and A. Laurent, "End-To-End Speaker Segmentation for Overlap-Aware Resegmentation," in *Proc. Interspeech*, 2021, pp. 3111–3115.

[54] S. Horiguchi, P. Garcia, Y. Fujita, S. Watanabe, and K. Nagamatsu, "End-to-end speaker diarization as post-processing," in *Proc. ICASSP*, 2021, pp. 7188–7192.

[55] T. Moriya, H. Sato, T. Ochiai, M. Delcroix, and T. Shinozaki, "Streaming end-to-end target-speaker automatic speech recognition and activity detection," *IEEE Access*, vol. 11, pp. 13 906–13 917, 2023.

[56] T. Li, Q. Lin, Y. Bao, and M. Li, "Atss-Net: Target Speaker Separation via Attention-Based Neural Network," in *Proc. Interspeech*, 2020, pp. 1411–1415.

[57] S. Ding, Q. Wang, S.-Y. Chang, L. Wan, and I. Lopez Moreno, "Personal VAD: Speaker-Conditioned Voice Activity Detection," in *Proc. Odyssey*, 2020, pp. 433–439.

[58] 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.

[59] W. Wang, Q. Lin, D. Cai, and M. Li, "Similarity measurement of segment-level speaker embeddings in speaker diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2645–2658, 2022.

[60] C.-Y. Cheng, H.-S. Lee, Y. Tsao, and H.-M. Wang, "Multi-target extractor and detector for unknown-number speaker diarization," *IEEE Signal Processing Letters*, 2023.

[61] D. Wang, X. Xiao, N. Kanda, T. Yoshioka, and J. Wu, "Target speaker voice activity detection with transformers and its integration with end-to-end neural diarization," in *Proc. ICASSP*, 2023, pp. 1–5.

[62] M. Cheng, W. Wang, Y. Zhang, X. Qin, and M. Li, "Target-speaker voice activity detection via sequence-to-sequence prediction," in *Proc. ICASSP*, 2023, pp. 1–5.

[63] J. Huh, J. S. Chung, A. Nagrani, A. Brown, J.-w. Jung, D. Garcia-Romero, and A. Zisserman, "The voxceleb speaker recognition challenge: A retrospective," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 3850–3866, 2024.

[64] D. Cai, W. Cai, and M. Li, "Within-sample variability-invariant loss for robust speaker recognition under noisy environments," in *Proc. ICASSP*, 2020, pp. 6469–6473.

[65] Y. Zhang, Z. Lv, H. Wu, S. Zhang, P. Hu, Z. Wu, H. yi Lee, and H. Meng, "MFA-Conformer: Multi-scale Feature Aggregation Conformer for Automatic Speaker Verification," in *Proc. Interspeech*, 2022, pp. 306–310.

[66] D. Cai and M. Li, "Leveraging asr pretrained conformers for speaker verification through transfer learning and knowledge distillation," *arXiv preprint arXiv:2309.03019*, 2023.

[67] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive Statistics Pooling for Deep Speaker Embedding," in *Proc. Interspeech*, 2018, pp. 2252–2256.

[68] M. T. Ho, J. Lee, B.-K. Lee, D. H. Yi, and H.-G. Kang, "A Cross-Channel Attention-Based Wave-U-Net for Multi-Channel Speech Enhancement," in *Proc. Interspeech*, 2020, pp. 4049–4053.

[69] D. Wang, Z. Chen, and T. Yoshioka, "Neural Speech Separation Using Spatially Distributed Microphones," in *Proc. Interspeech*, 2020, pp. 339–343.

[70] D. Wang, T. Yoshioka, Z. Chen, X. Wang, T. Zhou, and Z. Meng, "Continuous speech separation with ad hoc microphone arrays," in *Proc. EUSIPCO*, 2021, pp. 1100–1104.

[71] F.-J. Chang, M. Radfar, A. Mouchtaris, and M. Omologo, "Multi-Channel Transformer Transducer for Speech Recognition," in *Proc. Interspeech*, 2021, pp. 296–300.

[72] S. Horiguchi, Y. Takashima, P. Garcia, S. Watanabe, and Y. Kawaguchi, "Multi-channel end-to-end neural diarization with distributed microphones," in *Proc. ICASSP*, 2022, pp. 7332–7336.

[73] N. Ryant, P. Singh, V. Krishnamohan, R. Varma, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "The third dihard diarization challenge," *arXiv preprint arXiv:2012.01477*, 2020.

[74] F. Yu, S. Zhang, Y. Fu, L. Xie, S. Zheng, Z. Du, W. Huang, P. Guo, Z. Yan, B. Ma, X. Xu, and H. Bu, "M2MeT: The ICASSP 2022 multi-channel multi-party meeting transcription challenge," in *Proc. ICASSP*, 2022.

[75] A. Vinciarelli, "Speakers role recognition in multiparty audio recordings using social network analysis and duration distribution modeling," *IEEE Transactions on Multimedia*, vol. 9, no. 6, pp. 1215–1226, 2007.

[76] T. J. Park, H. Huang, C. Hooper, N. Koluguri, K. Dhawan, A. Jukic, J. Balam, and B. Ginsburg, "Property-aware multi-speaker data simulation: A probabilistic modelling technique for synthetic data generation," *arXiv preprint arXiv:2310.12371*, 2023.

[77] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[78] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. ICASSP*, 2017, pp. 5220–5224.

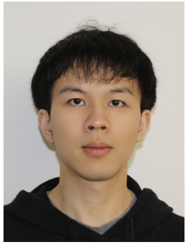
[79] W. Wang, D. Cai, X. Qin, and M. Li, "The dku-dukeece systems for voxceleb speaker recognition challenge 2020," *arXiv preprint arXiv:2010.12731*, 2020.

[80] O. Kuchaiev, J. Li, H. Nguyen, O. Hrinchuk, R. Leary, B. Ginsburg, S. Kriman, S. Beliaev, V. Lavrukhin, J. Cook *et al.*, "Nemo: a toolkit for building ai applications using neural modules," *arXiv preprint arXiv:1909.09577*, 2019.

[81] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *Proc. Interspeech*, 2017.

[82] W. Wang, X. Qin, and M. Li, "Cross-channel attention-based target speaker voice activity detection: Experimental results for the m2met challenge," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 9171–9175.

[83] Y. Yue, J. Du, M.-K. He, Y. Yeung, and R. Wang, "Online Speaker Diarization with Core Samples Selection," in *Proc. Interspeech*, 2022, pp. 1466–1470.



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 automatic piano transcription.

Ming Li received his Ph.D. in Electrical Engineering from University of Southern California in 2013. He is currently a Professor of Electrical and Computer Engineering at Duke Kunshan University. He is also an Adjunct Professor at School of Computer Science in Wuhan 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 200 papers and served as the member of IEEE speech and language technical committee, APSIPA speech and language processing technical committee. He was an area chair at Interspeech 2016, Interspeech 2018, Interspeech 2020, SLT2022 and Interspeech 2024. He is the technical program co-chair at Odyssey 2022 and ASRU 2023. He is an editorial member of Computer Speech and Language and APSIPA Transactions on Signal and Information Processing. Works co-authored with his colleagues have won first prize awards at Interspeech Computational Paralinguistic Challenges 2011, 2012 and 2019, ASRU 2019 MGB-5 ADI Challenge, Interspeech 2020 and 2021 Fearless Steps Challenges, VoxSRC 2021, 2022 and 2023 Challenges, ICASSP 2022 M2MeT Challenge, IJCAI 2023 ADD challenge, ICME 2024 ChatCLR challenge and Interspeech 2024 AVSE challenge. As a co-author, he has won the best paper award in DCOSS 2009 and ISCSLP 2014 as well as the best paper shortlist in Interspeech 2024. 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.