Sequence-to-Sequence Neural Diarization with Automatic Speaker Detection and Representation

Ming Cheng, Yuke Lin, Ming Li, *Senior Member, IEEE*

***Abstract*—This paper proposes a novel Sequence-to-Sequence Neural Diarization (S2SND) framework to perform online and offline speaker diarization. It is developed from the sequence-to-sequence architecture of our previous target-speaker voice activity detection system and then evolves into a new diarization paradigm by addressing two critical problems. 1) Speaker Detection: The proposed approach can utilize partially given speaker embeddings to discover the unknown speaker and predict the target voice activities in the audio signal. It does not require a prior diarization system for speaker enrollment in advance. 2) Speaker Representation: The proposed approach can adopt the predicted voice activities as reference information to extract speaker embeddings from the audio signal simultaneously. The representation space of speaker embedding is jointly learned within the whole diarization network without using an extra speaker embedding model. During inference, the S2SND framework can process long audio recordings blockwise. The detection module utilizes the previously obtained speaker-embedding buffer to predict both enrolled and unknown speakers' voice activities for each coming audio block. Next, the speaker-embedding buffer is updated according to the predictions of the representation module. Assuming that up to one new speaker may appear in a small block shift, our model iteratively predicts the results of each block and extracts target embeddings for the subsequent blocks until the signal ends. Finally, the last speaker-embedding buffer can re-score the entire audio, achieving highly accurate diarization performance as an offline system. Experimental results show that our proposed S2SND framework achieves new state-of-the-art diarization error rates (DERs) for online inference on the DIHARD-II (24.41%) and DIHARD-III (17.12%) evaluation sets without using oracle voice activity detection. At the same time, it also refreshes the state-of-the-art performance for offline inference on these benchmarks, with DERs of 21.95% and 15.13%, respectively.**

***Index Terms*—Speaker Diarization, Online Speaker Diarization, Sequence-to-Sequence Neural Diarization**

I. Introduction

SPEAKER diarization aims to split the conversational audio signal into segments with labeled identities, solving the problem of "Who-Spoke-When" [1]. It is the core front-end speech processing technique in various downstream tasks like multi-speaker speech recognition, etc [2].

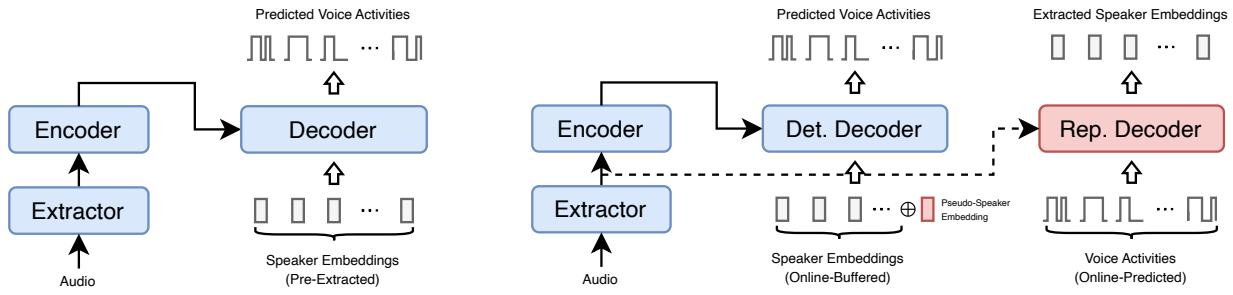
Early speaker diarization studies have widely investigated the cascaded methods that process audio signals through a series of independent modules [3]–[9]. Later, End-to-End Neural Diarization (EEND) methods [10]–[13] are proposed to estimate multiple speakers' voice activities as multi-label classification, where the end-to-end model architecture can be directly optimized by the permutation-invariant training (PIT) [14]. Also, Target-Speaker Voice Activity Detection (TSVAD) approaches [9], [15], [16] combine the advantages of cascaded methods and end-to-end neural networks. A typical TSVAD-based system requires a prior diarization system (e.g., the cascaded method) to extract each speaker's acoustic footprint as the speaker enrollment. Then, a neural network-based module predicts all speakers' corresponding voice activities. This two-stage framework demonstrates promising performance in popular benchmarks such as DIHARD-III [17] and VoxSRC21-23 [18]–[21].

However, the diarization systems mentioned above are natively designed to process pre-recorded audio offline, which means they cannot satisfy scenarios with low latency demand (e.g., real-time meeting transcription) [1]. For online speaker diarization, cascaded methods must modify all the built-in components to be capable of online inference, especially the inherent clustering algorithms [22], [23]. Online EEND systems are implemented by only replacing the network architecture [24] or using a buffer to trace the previous input-output pairs [25]–[27]. However, the speaker permutation problem is prone to be affected by the increasing number of speakers in long-form audios, which remains a challenge that has yet to be fully addressed. Although the recent FS-EEND [28] method can determine the speaker permutation according to their appearance order in online scenarios, the error accumulation with inference time may become a new issue. As the post-processing approach, TSVAD models natively process the audio signals blockwise except for acquiring pre-extracted speaker embeddings from the initial stage. Therefore, online TSVAD methods [29], [30] are proposed to enable self-generated speaker embeddings during blockwise inference. However, these existing methods must be integrated with another online VAD system to help detect the presence of new speakers. The practical use of them is relatively difficult.

Fig. 1 illustrates the progress of our speaker diarization research. In our previous work [16], the Sequence-to-Sequence Target-Speaker Voice Activity Detection (Seq2Seq-TSVAD) framework has been proposed for offline-only speaker diarization. It mainly consists of three modules, shown in Fig. 1a. First, the extractor obtains frame-level speaker embedding features from the raw audio. Next, the encoder processes long-term dependencies between frame-level features for the speaker diarization task. Finally, the decoder takes multiple speaker embeddings as reference information to predict the target-speaker voice activities, which has a one-to-one correspondence between the input order of speaker embeddings

Ming Cheng, Yuke Lin and Ming Li are with the School of Computer Science, Wuhan University, Wuhan 430072, China, and also with Suzhou Municipal Key Laboratory of Multimodal Intelligent Systems, Digital Innovation Research Center, Duke Kunshan University, Kunshan 215316, China.

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



(a) Seq2Seq-TSVAD (offline-only) [16] (b) S2SND (online/offline)

Fig. 1. Overview of our speaker diarization frameworks from offline-only to online/offline scenarios: (a) Previous Sequence-to-Sequence Target-Speaker Voice Activity Detection (Seq2Seq-TSVAD) framework; (b) Newly proposed Sequence-to-Sequence Neural Diarization (S2SND) framework. *Det.* and *Rep.* denote the abbreviations of detection and representation, respectively. The red parts indicate the new modules added in the S2SND model compared to the Seq2Seq-TSVAD model.

and the output order of voice activities. As a classical TSVAD-based method, the input speaker embeddings for the Seq2Seq-TSVAD model should be extracted by a prior diarization system, which restricts it from being an online system.

To tackle the above problem, this paper proposes a novel Sequence-to-Sequence Neural Diarization (S2SND) framework compatible with online and offline inference, shown in Fig. 1b. The S2SND framework is still built upon the sequence-to-sequence architecture, with some improvements. First, The detection decoder works similarly to the single decoder in Fig. 1a. Differently, we introduce a pseudo-speaker embedding to represent the unknown speaker without pre-extracted embedding, which is a kind of masked speaker prediction technique described in Sec. III-B1. Second, we add a new representation decoder to take multiple target-speaker voice activities as reference information to predict speaker embeddings, which is a kind of target-voice speaker embedding extraction technique described in Sec. III-B2. In this way, the S2SND framework can adopt partial speaker embeddings to predict complete voice activities and then extract the missed speaker embedding simultaneously. By traversing the input audio signal, the S2SND model can predict target-speaker voice activities of each coming audio block in real-time operation and progressively gather new speaker embeddings for the subsequent blocks. After the first-pass diarization, the collected speaker embeddings can also be used to re-decode the entire audio as an offline system.

Our proposed S2SND framework does not need unsupervised clustering or permutation-invariant training, making it fundamentally different from previous diarization systems. Therefore, we name it a new neural diarization approach. The contributions are summarized below.

1) We propose a novel masked speaker prediction method. One of the input speaker embeddings may be randomly erased during training. Then, the model learns to associate the output of the masked speaker with a learnable pseudo-speaker embedding, solving the one-to-one mapping problem between input speaker embeddings and output voice activities.
2) We propose a novel target-voice speaker embedding extraction method. In contrast to the previous TSVAD method, it utilizes the predicted voice activities as reference information to extract target embeddings from the input audio further. The embedding space of speaker detection and representation is jointly learned.
3) A simple but effective knowledge distillation strategy is developed to explore the potential of our proposed method when meeting large-scale data. We evaluate our approach on several widely-used datasets, outperforming previous state-of-the-art results in various online and offline evaluation settings.
4) The designed framework combines characteristics of both EEND and TSVAD methods. It is not only clustering-free and PIT-free, but also can utilize the end-to-end neural network to discover possible unknown speakers. Meanwhile, the use of target embeddings maintains recognized speaker identities consistent across different blocks in long audio, which is usually the advantage of TSVAD-based methods.

II. Related Works

A. Offline Diarization

The cascaded speaker diarization consists of several components. 1) Voice Activity Detection (VAD) [31] removes non-speech regions from the audio. 2) Speech regions are divided into shorter segments [32], [33]. 3) Speaker embeddings (e.g., i-vectors [34], x-vectors [35]) are extracted from the speech segments and clustered into different identities by K-Means [6], AHC [33], SC [7], or others. 4) Post-processing techniques for overlapped speech regions can be optionally implemented [36], [37]. The number of output speakers is determined by the clustering algorithms.

End-to-End Neural Diarization (EEND) [10], [11] predicts multiple speakers' voice activities by formulating the diarization problem as a multi-label classification task. The original EEND models have a fixed number of output speakers restricted by their network architecture. Although using Encoder-Decoder based Attractor (EDA) [12], [13] can infer the variable number of speakers. In practice, the number of output speakers is still capped by the training data [38]. To

solve this problem, integrating the end-to-end and clustering approach is a promising direction. For example, EEND-vector clustering (EEND-VC) [39]–[41] deploys an EEND model for shortly divided audio blocks and addresses the inter-block speaker permutation ambiguity by clustering of speaker embeddings. EEND-GLA [27], [42] computes local attractors for each short block and determines speaker correspondence based on similarities between inter-block attractors. Also, several extensions of EEND are proposed from the aspects of network architecture [43]–[45], objective function design [46], [47], self/semi-supervised learning [48], [49], and so on.

Target-Speaker Voice Activity Detection (TSVAD) [15] is also effective. It relies on a prior diarization system to extract each speaker's acoustic footprint (i-vector) as enrollment. Then, the TSVAD model uses speech features (e.g., MFCC) and extracted i-vectors to output target speaker voice activities according to the enrollment order. Later, He et al. [50] adapt the model to handle a variable number of speakers by setting a maximum speaker limit and producing null voice activities for zero-padded ones. Sequential models (e.g., LSTM [51] and Transformer [52]) are implemented on the speaker dimension of model input to manage a variable number of speakers. To explore more discriminative speaker embeddings as an alternative to i-vector, Wang et al. [9] replace the front-end of the TSVAD model with a pre-trained extractor tailored for frame-level x-vectors. This modification demonstrates superior performance than a simple swap of i-vectors for x-vectors in early attempt [15]. Furthermore, the TSVAD framework has been investigated in various aspects (e.g., multi-channel signal [53], multi-modal system [54]–[56], joint inference with ASR [57], generative approach [58]).

In addition, several studies have explored using voice activity information to guide speaker embedding extraction for the downstream tasks, including both clustering-based and TSVAD-based diarization systems [59], [60]. However, these methods treat embedding extraction as a separate stage without jointly optimizing it with diarization objectives. Also, they are inherently designed for offline inference and thus cannot be directly extended to low-latency online scenarios.

B. Online Diarization

In an online scenario, the diarization system must make continual decisions on each audio frame while the conversation continues. This paradigm is crucial for low-latency applications such as real-time conversation transcription.

To extend cascaded methods to online inference, all built-in modules (e.g., voice activity detection, speech segmentation, speaker embedding extraction) must be executed in real time. Several techniques (e.g., UIS-RNN [61], UIS-RNN-SML [62]) replace the speech segmentation and speaker clustering with supervised neural networks. As the most critical component, online speaker clustering attracts much research interest, e.g., modified clustering [22], [63], PLDA-scoring [64], and clustering guided embedding extractor training [65]. However, their time complexity will increase with the number of speech segments, resulting in inadequate performance for long audio.

The extension of end-to-end approaches to online diarization are broadly divided into two directions. The first is to train models that can convey information during block-wise or frame-wise inference to address the speaker permutation ambiguity. For instance, BW-EDA-EEND [24] adopts Transformer-XL [66] with recursive hidden states to take block-wise inputs, where the hidden states obtained from the previous blocks are used to generate attractors of the current block. Liang et al. [28] propose the frame-wise online EEND (FS-EEND) to adaptively update speaker attractors frame by frame, which has a lower inference latency. In this direction, online diarization models are easily optimized in a fully end-to-end manner. However, independent network architectures are required rather than offline diarization models. If both offline and online diarization models are needed, the deployment costs will be largely increased. The second direction is to modify offline models for online inference. Speaker-tracing buffer (STB) [25], [26] is proposed to maintain the preceding results of EEND models during online inference. It makes the order of output speakers consistent without changing the network architecture. On top of this direction, EEND-GLA [27], [42] further integrates local and global attractors with STB for online inference, achieving state-of-the-art performance on multiple datasets. It is reported that STB can minimize the inference latency using a small block size and outperform BW-EDA-EEND [26], [27]. Nonetheless, this approach demands extra computations because every past frame in the buffer must be re-computed for each new block.

On the success of offline TSVAD methods [17]–[20], diverting the TSVAD framework for online inference is also a promising direction. In offline scenarios, TSVAD methods usually serve as post-processing to refine cascaded diarization results [15]. After obtaining target-speaker embedding from the initial stage, TSVAD models process audio signals block-wise. This property implies that TSVAD models are naturally adapted to online inference if target-speaker embeddings are acquired in real time. Therefore, Wang et al. [29] firstly present the online TSVAD framework and then adapt it to multi-channel data [30]. Chen et al. [67] design a dictionary learning module across different frequency bands in multi-channel data to reduce the inference cost. Nevertheless, two problems prevent existing online TSVAD methods [29], [30], [67] from practical use. 1) They assume the input audio to contain at least one active speaker at all times. During inference, unenrolled speakers are determined once the models do not detect any active voice activity using enrolled speaker embeddings. Thus, an additional VAD module is required to remove all the silent regions in the input audio. The VAD errors might severely impact the final system output. This kind of bypass approach does not fundamentally solve the problem of new speaker detection. 2) They utilize local speaker labels within each recording to optimize target embedding extraction, where the power of global speaker modeling is not fully exploited. In contrast, current advanced speaker verification techniques are mainly based on unique speaker identities over the whole training set [68].

Notably, the concepts of TSVAD and EEND families are becoming closer. In early TSVAD systems [9], [15], [16], speaker embeddings are typically acoustic footprints extracted by speaker verification models (e.g., i-vectors [34],

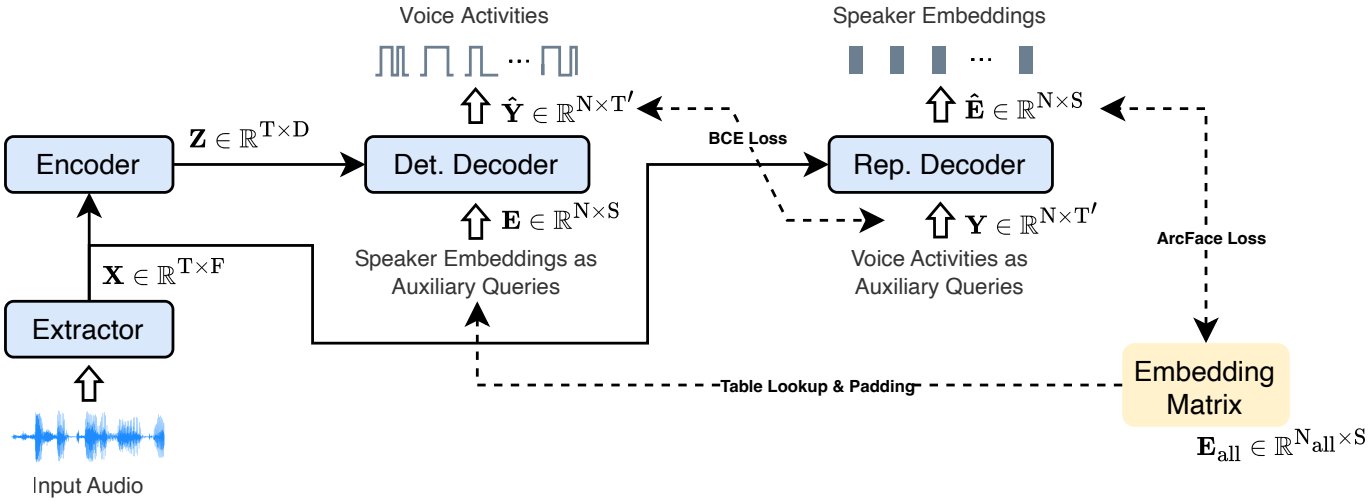


Fig. 2. The Sequence-to-Sequence Neural Diarization (S2SND) framework. *Det.* and *Rep.* denote the abbreviations of detection and representation, respectively.

x-vectors [35]). Then, works of [29], [30] turn to generate speaker embeddings within TSVAD models. On the other hand, the attractors used in EEND systems [12], [13] are also a kind of local speaker embeddings within each audio block. Recent studies of [27], [28] begin to constrain the speaker similarity of attractors across different audio blocks. Obviously, using a set of embedding vectors to represent speaker identities has been widely adopted with different terminologies. Therefore, in this work, we aim to take advantage of both EEND and TSVAD methods to propose the new S2SND framework, achieving state-of-the-art performance on various multi-scenario datasets.

III. Sequence-to-Sequence Neural Diarization

A. Architecture

The proposed S2SND framework takes the sequence-to-sequence architecture used in our previous offline method [16] with several modifications, shown in Fig. 2.

1) Extractor: The ResNet-based [69] model is adopted as the front-end extractor. The audio signal is firstly transformed into log Mel-filterbank energies. Then, it is fed into the extractor with segmental statistical pooling (SSP) [9] to obtain frame-level speaker embeddings $\mathbf{X} \in \mathbb{R}^{\mathrm{T}\times\mathrm{F}}$, where T and F denote the length and dimension of extracted feature sequence. An additional linear layer is employed to align the output dimension F with the input dimension of subsequent encoder and decoder modules, omitted to plot for clarity. This process converts raw audio signals into a sequence of neural network-based features.

2) Encoder: The Conformer-based [70] model is employed as the encoder to process the frame-level speaker embeddings. The input feature sequence is firstly added with sinusoidal positional encodings [71] and then fed into the encoder to obtain output feature sequence $\mathbf{Z} \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}}$, where D is the attention dimension used in the encoder. This process further takes long-term dependencies between frame-level speaker embeddings for the diarization task.

3) Decoder: The decoder block retains the main layout of the original Speaker-wise Decoder (SW-D) [16] with a few changes, shown in Fig. 3. There are four parts of the input. First, the feature embeddings come from the extractor output $\mathbf{X}$ or encoder output $\mathbf{Z}$. Second, the positional embeddings are the same as those used in the encoder. Third, as a decoder module is usually composed of several basic decoder blocks stacked together, the input decoder embeddings at the first block are initialized by zeros and then processed by the following blocks. After the last output block, a simple linear transformation is adopted to obtain the desirable output dimension. Lastly, the auxiliary queries play the role of reference information in multi-speaker tasks, which can be either target-speaker embeddings or voice activities. The detailed design of the decoder block is described below.

- The cross-attention layer is placed before the self-attention layer. As the input embeddings of the first decoder block are initialized as zeros, there is no useful information for the self-attention layer at first.
- We define $\mathrm{F_q}(\cdot)$ and $\mathrm{F_k}(\cdot)$ to denote the fusion operations for input queries and keys, respectively. Let N denote the preset maximum number of speakers that the model can handle simultaneously. $\mathbf{X}_{\mathrm{dec}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and $\mathbf{Q}_{\mathrm{aux}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}'}$ represent the decoder embeddings and auxiliary queries, respectively. The $\mathrm{F_q}(\cdot)$ operation is described as:

$$\mathbf{Q} = \mathbf{X}_{\mathrm{dec}} + \mathrm{Linear}_{\mathbb{R}^{\mathrm{D}'}\to\mathbb{R}^{\mathrm{D}}}(\mathbf{Q}_{\mathrm{aux}})/\sqrt{D}, \tag{1}$$

where the linear transformation is deployed to align the dimension of queries and keys with the weight factor $1/\sqrt{D}$. Similarly, let $\mathbf{X}_{\mathrm{fea}} \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}}$ and $\mathbf{K}_{\mathrm{pos}} \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}'}$ represent the feature embeddings and positional embeddings with the length of T. The $\mathrm{F_k}(\cdot)$ operation is described as:

$$\mathbf{K} = \mathbf{X}_{\mathrm{fea}} + \mathrm{Linear}_{\mathbb{R}^{\mathrm{D}'}\to\mathbb{R}^{\mathrm{D}}}(\mathbf{K}_{\mathrm{pos}})/\sqrt{D}. \tag{2}$$

The fused queries $\mathbf{Q}$ and keys $\mathbf{K}$ are fed into the cross-attention layer with a Pre-LayerNorm method. Compared

with the previous concatenation fusion, this additive fusion is more straightforward without expanding the output dimension of queries and keys.
- If the input auxiliary queries are speaker embeddings, they must be $L2$-normalized to lie in a hypersphere. Otherwise, if the input auxiliary queries are voice activities, they do not need to undergo any normalization.

Based on the same structure of the decoder block, we introduce two decoders responsible for different functions. Let $\mathbf{E} \in \mathbb{R}^{\mathrm{N}\times\mathrm{S}}$ denote the given speaker embeddings with the number of N and the dimension of S. The ground truth of their target voice activities is denoted as a binary matrix $\mathbf{Y} \in \{0,1\}^{\mathrm{N}\times\mathrm{T}'}$, where $y_{n,t'}$ represents the speaking existence of the n-th speaker at time t'. The detection decoder utilizes encoder output $\mathbf{Z}$ as feature embeddings and speaker embeddings $\mathbf{E}$ as auxiliary queries to obtain the predicted voice activities $\hat{\mathbf{Y}} \in \{0,1\}^{\mathrm{N}\times\mathrm{T}'}$. In contrast, the representation decoder utilizes extractor output $\mathbf{X}$ as feature embeddings and voice activities $\mathbf{Y}$ as auxiliary queries to obtain the extracted speaker embeddings $\hat{\mathbf{E}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{S}}$. Two decoders perform inverse tasks to predict target-speaker voice activities and extract speaker embeddings simultaneously.

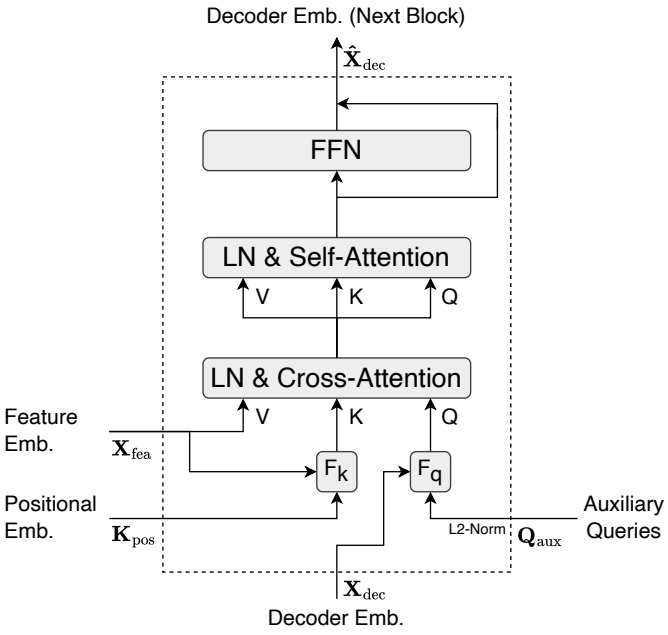


Fig. 3. The structure of the modified Speaker-wise Decoder. For clarity, the residual connections between attention layers are omitted from the plot. The abbreviation of *LN* refers to the layer normalization applied before the QKV inputs for attention modules.

B. Training Process

The ground truth of $\mathbf{Y}$ is obtained from the adopted dataset during training. However, $\mathbf{E}$ is not directly available because the embedding space must be learned by neural networks. To overcome this problem, we adopt a learnable embedding matrix $\mathbf{E}_{\mathrm{all}} \in \mathbb{R}^{\mathrm{N_{all}}\times\mathrm{S}}$ as the target embeddings of all speakers in the training data. N_{all} and S represent the total number of speakers and embedding dimension, respectively. Each row vector denotes one specific speaker embedding, which is randomly initialized as a unit vector with a magnitude (norm) of 1. Given $n \leq N_{\mathrm{all}}$, the n-th target-speaker embedding in the training data is obtained by $\mathbf{E}_{\mathrm{all}}(n,:)$. Meanwhile, the n-th speaker label in the training data is denoted by a N_{all}-dim one-hot vector with zeros everywhere except its n-th value will be 1. During training, given an input audio block with N_{loc} speaker labels $\mathbf{S}_{\mathrm{loc}} \in (0,1)^{\mathrm{N_{loc}}\times\mathrm{N_{all}}}$, the input speaker embeddings for the detection decoder are obtained by $\mathbf{S}_{\mathrm{loc}} \cdot \mathbf{E}_{\mathrm{all}} \in \mathbb{R}^{\mathrm{N_{loc}}\times\mathrm{S}}$, which is a simple table look-up operation using matrix multiplication. Also, $\mathbf{E}_{\mathrm{all}}$ is used as the training objectives of output speaker embeddings from the representation decoder. We propose the following approaches to jointly optimize $\mathbf{E}_{\mathrm{all}}$ with the whole diarization model.

1) Masked speaker prediction: The masked language modeling (MLM) technique has been validated in natural language processing [72], conducted by randomly masking some words in the input text and then training the model to predict the masked words. Similarly, we introduce a masked speaker prediction method into speaker diarization. During training, one of the input speaker embeddings for each audio block will be randomly masked. The model learns to identify whether there is a person speaking in the audio block but without the given speaker embedding. To achieve this goal, two padding strategies are implemented.

The first strategy is to pad the input speaker embeddings $\mathbf{E}$ using a learnable pseudo-speaker embedding $\mathbf{e}_{\mathrm{pse}} \in \mathbb{R}^{\mathrm{S}}$, where $\mathbf{e}_{\mathrm{pse}}$ is initialized with zeros and optimized during training. For each training data, a probability is 0.5 that one existing speaker label $\mathbf{s}_n \in \mathbf{S}$ will be randomly selected as the masked one. Accordingly, the speaker embedding $\mathbf{e}_n$ will be removed from $\mathbf{E}$ and the ground-truth of target voice activities $\mathbf{y}_n \in \mathbf{Y}$ will be re-assigned to the output of pseudo-speaker embedding. In this way, the model is trained to utilize the pseudo-speaker embedding to capture any unenrolled speaker's voice activities.

The second strategy is to pad the input speaker embeddings $\mathbf{E}$ using a learnable non-speech embedding $\mathbf{e}_{\mathrm{non}} \in \mathbb{R}^{\mathrm{S}}$, where $\mathbf{e}_{\mathrm{non}}$ is initialized with zeros and optimized during training. We define speaker capacity as the maximum number of speakers (embeddings) that the model can process simultaneously. By setting the speaker capacity to a relatively large value N, the pseudo-speaker embedding $\mathbf{e}_{\mathrm{pse}}$ accounts for one, and there are also N_{loc} existing speaker embeddings. As usually $(N_{\mathrm{loc}}+1) \ll N$, the left $N - N_{\mathrm{loc}} - 1$ vacancies will be randomly filled up with 50% non-speech embeddings and 50% speaker embeddings from who are not appearing in the current audio block. Accordingly, their ground-truth voice activities are silent. In this way, the input dimension of a mini-batched training data is aligned, and the model is trained to distinguish valid and invalid speaker embeddings for the given audio block and how to assign target voice activities to the corresponding speakers.

Finally, the output $\hat{\mathbf{Y}}$ from the detection decoder is optimized to minimize its binary cross-entropy (BCE) loss with

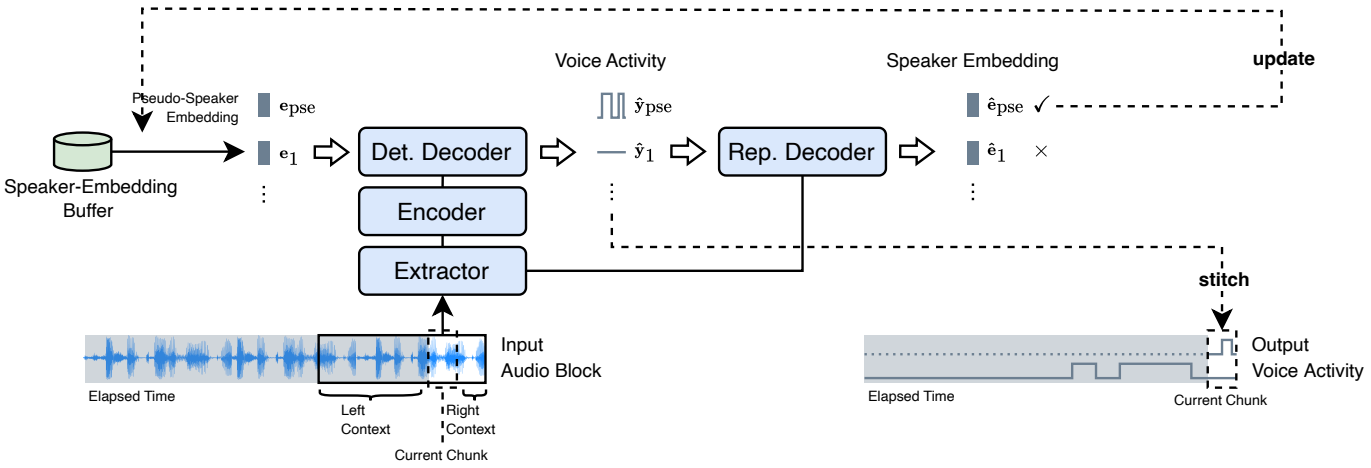


Fig. 4. The inference diagram of the Sequence-to-Sequence Neural Diarization (S2SND) framework. *Det.* and *Rep.* denote the abbreviations of detection and representation, respectively.

$\mathbf{Y}$, which is described as follows:

$$\mathcal{L}_{\text{bce}} = -\frac{1}{N \times T'} \sum_{n=1}^{N} \sum_{t'=1}^{T'} [y_{n,t'} \log(\hat{y}_{n,t'}) + (1 - y_{n,t'}) \log(1 - \hat{y}_{n,t'})], \tag{3}$$

where $\hat{y}_{n,t'} = \hat{\mathbf{Y}}(n, t')$ is the predicted speaking probability of the n-th speaker at time t'. And $y_{n,t'} = \mathbf{Y}(n, t')$ is its ground-truth label.

2) Target-voice speaker embedding extraction: As speaker embeddings can be used as reference information to extract target speaker voice activities, why can't voice activities be used as reference information to extract target speaker embeddings from multi-talker audio signals? Although a voice activity pattern may be the same between two speakers in a short window, it is not a big issue because the single-speaker speech usually occupies most of the time in real conversational data. Following this idea, we propose the target-voice speaker embedding extraction method, an inverse function of target-speaker voice activity detection.

The ArcFace [73] loss is employed between $\hat{\mathbf{E}}$ and the embedding matrix $\mathbf{E}_{\text{all}}$, which is described as follows:

$$\mathcal{L}_{\text{arc}} = \frac{1}{N} \sum_{n=1}^{N} -\log \frac{e^{\alpha \cdot \cos(\theta_n + m)}}{e^{\alpha \cdot \cos(\theta_n + m)} + \sum_{i=1, i \neq \phi(n)}^{N_{all}} e^{\alpha \cdot \cos \theta_i}}. \tag{4}$$

In this formula, n is the local speaker index for the current model output within the maximum speaker capacity, where $1 \leq n \leq N$. Let $\phi(n)$ denote the mapping from the local speaker index n to its corresponding global speaker index within all training data, where $1 \leq \phi(n) \leq N_{\text{all}}$. In code implementation, ϕ could be easily recorded when preparing training data. Then, θ_n is the angle between the n-th extracted speaker embedding $\hat{\mathbf{e}}_n \in \hat{\mathbf{E}}$ and its target embedding $\mathbf{e}_{\phi(n)} \in \mathbf{E}_{\text{all}}$. θ_i is the angle between the n-th extracted speaker embedding $\hat{\mathbf{e}}_n \in \hat{\mathbf{E}}$ and its non-target embedding $\mathbf{e}_i \in \mathbf{E}_{\text{all}}$, where $1 \leq i \leq N_{\text{all}}$ and $i \neq \phi(n)$ controls that θ_i is only implemented on negative pairs in contrastive learning. α and m are the re-scale factor and additive angular margin penalty, respectively.

The total training loss is the sum of $\mathcal{L}_{\text{bce}}$ in Eq. 3 and $\mathcal{L}_{\text{arc}}$ in Eq. 4. Using a learnable embedding matrix as the bridge between the built-in decoders, the embedding space of speaker detection and representation is jointly optimized in an end-to-end manner.

C. Inferring Process

Fig. 4 demonstrates the inference diagram of our proposed S2SND framework. Once the training is finished, the embedding matrix will no longer be needed. Instead, a speaker-embedding buffer is initialized as an empty dictionary to store speaker embeddings extracted during inference. Then, the model processes the input audio block by block and progressively updates the speaker-embedding buffer.

1) Data preparation: The input audio is cut into blocks of fixed length L, where L should be identical to the preset during training. To reduce the latency of model output, we introduce a smaller unit: chunk. As shown in the left part of Fig. 4, each input audio block contains three regions: left context, current chunk, and right context. The chunk length is set to L_{chunk}, representing the period corresponding to each inference step. The left and right context lengths are set to L_{left} and L_{right}, respectively. A sliding window method is applied to move the current chunk on the audio stream with the chunk shift equal to the chunk length. For each inference, the model takes the input audio block containing the current chunk and its contexts as long as $L = L_{\text{left}} + L_{\text{chunk}} + L_{\text{right}}$. The absence of left context is padded with zeros at the beginning of the inference until the acquired audio signal is available to compose an entire block. Furthermore, since acquiring the right context needs to await an extra period, the algorithmic latency of model inference should be the sum of L_{chunk} and L_{right}.

Assume that speaker capacity is set to N during training and N_{loc} identities are currently enrolled in the speaker-embedding buffer. The input speaker embeddings for the detection decoder consist of three different sources. The first part is always

kept for the pseudo-speaker embedding $\mathbf{e}_{\text{pse}} \in \mathbb{R}^{\text{S}}$. The second part consists of the target embeddings enrolled in the current speaker-embedding buffer, denoted as $\mathbf{E}_{\text{buf}} = [\mathbf{e}_1\ \mathbf{e}_2\ \cdots\ \mathbf{e}_{\text{N}_{\text{loc}}}]^\top \in \mathbb{R}^{\text{N}_{\text{loc}}\times\text{S}}$. The third part is padded by the non-speech embedding $\mathbf{e}_{\text{non}} \in \mathbb{R}^{\text{S}}$ with the number of $N - N_{\text{loc}} - 1$, denoted as $\mathbf{E}_{\text{non}} = [\mathbf{e}_{\text{non}}\ \mathbf{e}_{\text{non}}\ \cdots\ \mathbf{e}_{\text{non}}]^\top \in \mathbb{R}^{(\text{N}-\text{N}_{\text{loc}}-1)\times\text{S}}$. Overall, the total input speaker embeddings are concatenated by $\mathbf{E} = \left[\mathbf{e}_{\text{pse}}\ \mathbf{E}_{\text{buf}}^\top\ \mathbf{E}_{\text{non}}^\top\right]^\top \in \mathbb{R}^{\text{N}\times\text{S}}$. Therefore, the dimension of input speaker embeddings during inference is consistent with the preset during training.

2) Decoding Procedure: The first decoding stage takes the given speaker embeddings as reference information to predict multiple speakers' voice activities from the detection decoder. As the input order of speaker embeddings determines the output order of target voice activities, the predicted target-speaker voice activities also have three parts. Let T' indicate the number of timestamps in a given speaker's prediction. The first part is $\hat{\mathbf{y}}_{\text{pse}} \in \mathbb{R}^{\text{T}'}$, which represnets the predicted result corresponding to the pseudo-speaker embedding $\mathbf{e}_{\text{pse}}$. The second part is made of the predicted results corresponding to the buffered embeddings $\mathbf{E}_{\text{buf}}$, denoted as $\hat{\mathbf{Y}}_{\text{buf}} = [\hat{\mathbf{y}}_1\ \hat{\mathbf{y}}_2\ \cdots\ \hat{\mathbf{y}}_{\text{N}_{\text{loc}}}]^\top \in \mathbb{R}^{\text{N}_{\text{loc}}\times\text{T}'}$. The third part is padded by the predicted results corresponding to the non-speech embeddings $\mathbf{E}_{\text{non}}$, denoted as $\hat{\mathbf{Y}}_{\text{non}} = [\hat{\mathbf{y}}_{\text{non}}\ \hat{\mathbf{y}}_{\text{non}}\ \cdots\ \hat{\mathbf{y}}_{\text{non}}]^\top \in \mathbb{R}^{(\text{N}-\text{N}_{\text{loc}}-1)\times\text{T}'}$. Overall, the total predicted target-speaker voice activities are concatenated by $\hat{\mathbf{Y}} = \left[\hat{\mathbf{y}}_{\text{pse}}\ \hat{\mathbf{Y}}_{\text{buf}}^\top\ \hat{\mathbf{Y}}_{\text{non}}^\top\right]^\top \in \mathbb{R}^{\text{N}\times\text{T}'}$. Furthermore, $\hat{\mathbf{Y}}_{\text{non}}$ are invalid results because they belong to padded contents to maintain the fixed dimension of predicted target-speaker voice activities.

The second decoding stage takes the predicted voice activities as reference information to extract multiple speakers' embeddings from the representation decoder. Similarly, the input order of voice activities determines the output order of target speaker embeddings. First, $\hat{\mathbf{e}}_{\text{pse}} \in \mathbb{R}^{\text{S}}$ represents the extracted result corresponding to $\hat{\mathbf{y}}_{\text{pse}}$. Second, $\hat{\mathbf{E}}_{\text{buf}} = [\hat{\mathbf{e}}_1\ \hat{\mathbf{e}}_2\ \cdots\ \hat{\mathbf{e}}_{\text{N}_{\text{loc}}}]^\top \in \mathbb{R}^{\text{N}_{\text{loc}}\times\text{S}}$ denotes the extracted results corresponding to $\hat{\mathbf{Y}}_{\text{buf}}$. Third, $\hat{\mathbf{E}}_{\text{non}} = [\hat{\mathbf{e}}_{\text{non}}\ \hat{\mathbf{e}}_{\text{non}}\ \cdots\ \hat{\mathbf{e}}_{\text{non}}]^\top \in \mathbb{R}^{(\text{N}-\text{N}_{\text{loc}}-1)\times\text{S}}$ denotes the extracted results corresponding to $\hat{\mathbf{Y}}_{\text{non}}$. Overall, the total extracted target-speaker embeddings are concatenated by three parts, denoted as $\hat{\mathbf{E}} = \left[\hat{\mathbf{e}}_{\text{pse}}\ \hat{\mathbf{E}}_{\text{buf}}^\top\ \hat{\mathbf{E}}_{\text{non}}^\top\right]^\top \in \mathbb{R}^{\text{N}\times\text{S}}$.

Furthermore, let $\hat{\mathbf{y}} = [y_1, y_2, \ldots, y_{T'}]$ indicates the predicted voice activities of a given speaker in $\hat{\mathbf{Y}}$, we define the operation $W : \mathbb{R}^{\text{T}'} \to \mathbb{R}, \text{W}(\hat{\mathbf{y}}) = \sum_{t'=1, t'\notin\mathbf{Overlap}}^{T'} \hat{y}_{t'}$ to count the non-overlapped speaking time in the given $\hat{\mathbf{y}}$. As the quality of embedding extraction may be easily affected by each speaker's active speaking time and overlapping status, the longer single-speaking time for each speaker usually results in better embedding extraction, which can be used as an additional embedding weight. Applying the function W to each predicted target-speaker voice activity in $\hat{\mathbf{Y}}$, the weight of each extracted target-speaker embedding is calculated one by one. First, $\hat{w}_{\text{pse}} \in \mathbb{R}$ represents the weight corresponding to $\hat{\mathbf{e}}_{\text{pse}}$. Second, $\hat{\mathbf{w}}_{\text{buf}} = [\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_{\text{N}_{\text{loc}}}]^\top \in \mathbb{R}^{\text{N}_{\text{loc}}}$ denotes the weights corresponding to $\hat{\mathbf{E}}_{\text{buf}}$. Third, $\hat{\mathbf{w}}_{\text{non}} = [\hat{w}_{\text{non}}, \hat{w}_{\text{non}}, \ldots, \hat{w}_{\text{non}}]^\top \in \mathbb{R}^{\text{N}-\text{N}_{\text{loc}}-1}$ denotes the weights corresponding to $\hat{\mathbf{E}}_{\text{non}}$. Overall, the total weights of extracted target-speaker embeddings are concatenated by three parts, denoted as $\hat{\mathbf{w}} = \left[\hat{w}_{\text{pse}}\ \hat{\mathbf{w}}_{\text{buf}}^\top\ \hat{\mathbf{w}}_{\text{non}}^\top\right]^\top \in \mathbb{R}^{\text{N}}$.

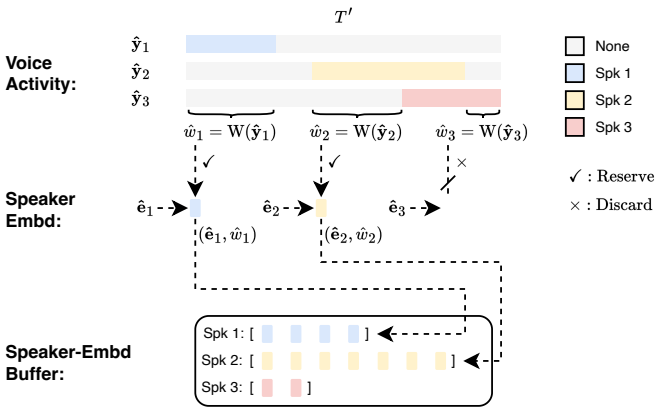


Fig. 5. Updating strategy of the speaker-embedding buffer.

We adopt two thresholds denoted as τ_1 and τ_2, respectively. If $\hat{w}_{\text{pse}} > \tau_1$, it means that an unenrolled speaker is detected and the extracted embedding is qualified to be reserved. The results of $\hat{\mathbf{y}}_{\text{pse}}$ and $\hat{\mathbf{e}}_{\text{pse}}$ will be assigned a new speaker label. Otherwise, $\hat{\mathbf{y}}_{\text{pse}}$ and $\hat{\mathbf{e}}_{\text{pse}}$ will be discarded as invalid results. Also, each $\hat{\mathbf{e}}_n \in \hat{\mathbf{E}}_{\text{buf}}$ represents the extracted target-speaker embedding corresponding to the target-speaker voice activity $\hat{\mathbf{y}}_n$, where $1 \le n \le N_{\text{loc}}$. If $\hat{w}_n > \tau_2$, the results of $\hat{\mathbf{y}}_n$ and $\hat{\mathbf{e}}_n$ will be reserved. Otherwise, $\hat{\mathbf{e}}_n$ will be discarded to prevent the unreliable speaker embedding from polluting the buffer. However, $\hat{\mathbf{y}}_n$ can still be adopted because it is predicted by target-speaker embeddings buffered previously. Lastly, valid results of the predicted voice activities will be stitched onto their preceding predictions in the elapsed time. It must be noticed that only the output region belonging to the current chunk is adopted as new predictions in every inference, which ensures the temporal causality of online inference. Valid results of the extracted speaker embeddings are updated in the speaker-embedding buffer to infer the next audio block.

3) Buffer Updating: Fig. 5 illustrates the updating strategies for selecting and buffering target-speaker embeddings at the end of each inference. In this example, both $\hat{w}_1$ and $\hat{w}_2$ exceed the preset threshold for reserving, but $\hat{w}_3$ is discarded. In the dictionary-based speaker-embedding buffer, the keys represent the enrolled speaker labels, and the corresponding values contain lists of embedding-weight pairs, respectively. Each reserved speaker embedding and its weight are appended into the buffer according to the key of the speaker label. When inferring the next audio block, each speaker's target embedding for model input will be the weighted average of all the buffered results. To formally describe this procedure, let $\left\{\hat{\mathbf{e}}_n^1, \hat{\mathbf{e}}_n^2, \ldots, \hat{\mathbf{e}}_n^{K_n}\right\}$ and $\left\{\hat{w}_n^1, \hat{w}_n^2, \ldots, \hat{w}_n^{K_n}\right\}$ denote the embeddings and weights of the n-th speaker in the buffer, where K_n is the number of embeddings. The aggregation of

Algorithm 1 Pseudocode of online inference in the Python-like style.

```
"""
- Extractor(), Encoder(), Det_Decoder(), Rep_Decoder(): neural network modules in S2SND models
- W(): calculating embedding weight
Inputs
- blocks: a sequence of input audio blocks
- e_pse/e_non: pseudo-speaker/non-speech embedding
- tau_1/tau_2: threshold for pseudo-speaker/enrolled-speaker embedding weight
- lc/lr: number of output VAD frames belonging to the current chunk / right context
- N: speaker capacity
- S: embedding dimension
Outputs
- dia_result: predicted target-speaker voice activities
- emb_buffer: extracted speaker embeddings
"""

dia_result = {} # initial diarization result
emb_buffer = {} # initial speaker-embedding buffer
num_frames = 0 # number of predicted VAD frames

for audio_block in blocks: # load the next audio block
   emb_list = [e_pse] # initialize input speaker embedding list & put pseudo-speaker embedding
   spk_list = [len(emb_buffer)+1] # initialize input speaker labels & create new speaker label

   for spk_id in emb_buffer.keys(): # obtain each enrolled target-speaker embedding
      e_sum = torch.zeros(S) # embedding vector, shape: S
      w_sum = 0 # embedding weight, scalar
      for e_i, w_i in emb_buffer[spk_id]:
         e_sum += w_i*e_i
         w_sum += w_i
      emb_list.append(e_sum/w_sum) # append weighted speaker embedding
      spk_list.append(spk_id) # append speaker label

   while len(emb_list) < N: # pad input embeddings to tensor with the length of N
      emb_list.append(e_non)
   emb_tensor = torch.stack(emb_list)

   X = Extractor(audio_block) # forward extractor, output shape: T x F
   X_hat = Encoder(X) # forward encoder, output shape: T x D
   Y_hat = Det_Decoder(X_hat, emb_tensor) # forward detection decoder, output shape: N x T'
   E_hat = Rep_Decoder(X, Y_hat) # forward representation decoder, output shape: N x S

   y_pse = Y_hat[0] # predicted pseudo-speaker voice activity, shape: T'
   e_pse = E_hat[0] # extracted pseudo-speaker embedding, shape: S
   w_pse = W(y_pse) # embedding weight: scalar
   if w_pse > tau_1:
      elapsed_y = torch.zeros(num_frames) # create the elapsed result as zeros
      current_y = y_pse[-(lc+lr):-lr] # cut the current chunk result from the block output
      new_id = spk_list[0] # get speaker id
      dia_result[new_id] = torch.cat(elapsed_y, current_y) # store diarization result
      emb_buffer[new_id] = [(e_pse, w_pse)] # store embedding-weight pair

   for n in range(1, len(S)):
      y_n = Y_hat[n] # predicted enrolled voice activity, shape: T'
      e_n = E_hat[n] # extracted enrolled embedding, shape: S
      w_n = W(y_n) # embedding weight, scalar
      spk_id = spk_list[n] # get speaker id
      dia_result[spk_id] = torch.cat(dia_result[spk_id], y_n[-(lc+lr):-lr]) # stitch diarization result
      if w_n > tau_2:
         emb_buffer[spk_id].append((e_n, w_n)) # append embedding-weight pair

   num_frames += lc # update
```

target-speaker embedding is calculated as follows:

$$\bar{\mathbf{e}}_n = \frac{\sum_{k=1}^{K_n} (\hat{w}_n^k \cdot \hat{\mathbf{e}}_n^k)}{\sum_{k=1}^{K_n} \hat{w}_n^k}. \tag{5}$$

Algorithm 1 summarizes the pseudocode of online inference in a Python-like style. A live audio signal is fed into the proposed model by a sliding window approach. The neural network detects if a new speaker appears in each coming audio block by itself, eliminating the use of any prior system (e.g., the cascaded diarization). Then, it finishes the target-speaker voice activity detection and embedding extraction for the following audio blocks. In such blockwise processing, the predictions are output immediately as an online diarization system.

In addition, our proposed framework can achieve better performance through a rescoring mechanism. After the online inference, the speaker-embedding buffer will collect all target-speaker embeddings from the full audio recording. If intermediate features of the extractor and encoder are cached during the first-pass inference, the final speaker-embedding buffer can be used to fastly re-decode the audio, which acts as an offline diarization system. Beneficial to the co-designed training and inferring techniques, our proposed framework adapts to both online and offline inference modes.

IV. Experimental Settings

A. Datasets

To train the S2SND models with numerous speaker identities, we introduce two speaker corpora for data simulation. The first corpus is the widely-used VoxCeleb2 [74] with over 1 million utterances for 6,112 identities. The second corpus is the recently released VoxBlink2 [75] with approximately 10 million utterances for 111,284 identities. We employ the FSMN-VAD module in FunASR [76] toolkit to remove non-speech regions from the raw audio, purifying the data as much as possible. Then, the simulated data is generated in an on-the-fly manner during training. First, the single-speaker utterance is independently created by alternately concatenating the source speech and silent (zero-padded) segments, where each segment length is randomly sampled from a uniform distribution of 0-4 seconds. Second, we randomly mix utterances of 1-3 speakers from the corpora, which follows the same implementation in our previous works [16], [55].

The models pretrained by simulated data are further adapted and evaluated on real multi-domain datasets: DIHARD-II [77] and DIHARD-III [78], respectively. The DIHARD-II dataset includes 11 conversational scenarios (e.g., interview, clinical, restaurant), with 23.81 hours of development set and 22.49 hours of evaluation set. We select the first 153 recordings (80%) of the original development set for model adaptation, namely the *dev153* set. The last 39 recordings (20%) remain for validation, namely the *dev39* set. The DIHARD-III dataset is the next edition of the DIHARD-II dataset in a series of speaker diarization challenges, with 34.15 hours of development set and 33.01 hours of evaluation set. Similarly, we select the first 203 recordings (80%) of the original development set for model adaptation, namely the *dev203* set. The last 51 recordings (20%) remain for validation, namely the *dev51* set. The statistics of both simulated and real datasets are described in Table I.

TABLE I
Statistics of datasets used in our experiments. The overlap ratios of simulated data are estimated on 250,000 randomly generated samples.

Dataset	Split	Num. Speakers	Num. Recordings	Overlap Ratio
On-the-fly Simulation	sim1spk	1	-	0.00%
	sim2spk	2	-	28.01%
	sim3spk	3	-	39.66%
	total	1-3	-	22.56%
DIHARD-II [77]	dev153	1-10	153	9.78%
	dev39	1-9	39	9.73%
	eval	1-9	194	8.90%
DIHARD-III [78]	dev203	1-10	203	10.83%
	dev51	1-8	51	10.37%
	eval	1-9	259	9.37%

B. Network Configurations

1) Pretrained extractor: As the pretrained front-end extractor can effectively facilitate the model to learn the identity information in target-speaker embeddings, we pretrain three speaker embedding extractors with similar network architecture but different model sizes and training data. The first two extractors are both based on the ResNet-34 model, while their residual blocks have respective channels of $\{32, 64, 128, 256\}$ and $\{64, 128, 256, 512\}$, namely the ResNet34-32ch and ResNet34-64ch. After adding the global statistical pooling (GSP) [35] and linear projection layer with the output dimension of 256, these two extractors are trained on the VoxCeleb2 [74] dataset by the ArcFace ($\alpha = 32, m = 0.2$) [73] classifier. We also introduce the third ResNet-152 model trained on the VoxBlink2 [75] dataset to explore the potential of large model size and training data. The ResNet34-32ch, ResNet34-64ch, and ResNet-152 models have 5.45M, 21.53M, and 58.14M parameters, respectively. Accordingly, they obtain 1.17%, 0.81%, and 0.34% equal error rates (EERs) on the Vox-O [79] trial.

2) S2SND model: For the entire S2SND model, we propose two versions with different numbers of parameters. The first is named S2SND-Small. Its extractor is based on the ResNet34-32ch model. The following encoder and decoder adopt 256-dim attentions with 8 heads and 512-dim feedforward layers. The second is named S2SND-Medium. Its extractor is based on the ResNet34-64ch model. The encoder and decoder are changed to 384-dim attentions with 8 heads and 768-dim feedforward layers. The other configurations for the two models are identical. All encoders and decoders have 4 blocks. The kernel size of convolutions in Conformer blocks is set to 15. In total, the parameters in the S2SND-Small and S2SND-Medium models are 16.56M and 45.96M, respectively. Because the number of parameters of the ResNet-152 extractor is too large, even twice that of the ResNet34-64ch extractor, the heavy parameters will take too much time to do the experiments and bring high demand for computing cost during real-time inference. We only use the ResNet-152 extractor as the teacher model of knowledge distillation [80] to improve the current models described in the following paragraph.

C. Training and Inferring Details

1) Training details: All training audio is split into fixed-length blocks and normalized with a mean of 0 and a standard deviation of 1. Specifically, the block length in this work is set to 8 seconds. The input acoustic features are 80-dim log Mel-filterbank energies with a frame length of 25 ms and a shift of 10 ms. Also, we apply the additive noise from Musan [81] and reverberation from RIRs [82] as audio augmentation. As suggested by our previous findings [16], [55], the temporal resolution (duration per frame-level prediction) of system output is directly set to 10 ms for precise option. The speaker capacity N is adopted as 30, a relatively large number that can adequately cover the maximum number of speakers in most datasets.

When the number of speakers in a given audio block cannot reach N, absent positions will be padded as described in Sec. III-B1. Lastly, all the input target-speaker embeddings are randomly shuffled to make the model invariant to speaker order. Accordingly, the ground truth labels for target-speaker

voice activity detection and embedding extraction must also be re-assigned based on their shuffled results. Then, the whole model is optimized by AdamW [83] optimizer with the binary cross entropy (BCE) loss and ArcFace ($\alpha = 32, m = 0.2$) [73] loss depicted in Fig. 2. Using 8 $\times$ NVIDIA RTX-3090 GPUs with a batch size of 16, we investigate two multi-stage training strategies as follows.

The first training strategy follows our previous work [16], containing three different stages starting from the pretrained extractor. In each stage, the model will be validated every 500 steps. The checkpoint with the lowest diarization error rate on the adopted validation set will be used for the next stage.

- Stage 1: We copy and freeze the weights of a pretrained speaker embedding model to initialize the front-end extractor. Only simulated data is used to train the back-end modules for 100,000 steps with a learning rate of *1e-4*.
- Stage 2: The front-end extractor is unfrozen. The whole S2SND model is adapted by 80% of the simulated data and 20% of the real data from the specific dataset, taking around 75,000 steps.
- Stage 3: The learning rate is decayed to *1e-5* for fine-tuning the whole S2SND model, taking around 50,000 steps.

In this work, we also explore the second kind of training strategy based on knowledge distillation, shown in Fig. 6. The pretrained ResNet-152 model is employed as the teacher extractor. The original input audio will be copied to feed the student and teacher extractors during training. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T] \in \mathbb{R}^{\mathrm{T}\times\mathrm{F}}$ denote the output of the student extractor, where T is the time axis and F is the feature axis. Comparatively, the output of the teacher extractor is represented as $\mathbf{X}' = [\mathbf{x}'_1, \ldots, \mathbf{x}'_T] \in \mathbb{R}^{\mathrm{T}\times\mathrm{F}}$. Then, we employ a frame-wise cosine similarity loss between two extractors, which is described as:

$$\mathcal{L}_{\mathrm{distill}} = 1 - \frac{1}{T}\sum_{t=1}^{T}\frac{\mathbf{x}_t \cdot \mathbf{x}'_t}{\|\mathbf{x}_t\| \cdot \|\mathbf{x}'_t\|}, \tag{6}$$

where $\mathbf{x}_t$ and $\mathbf{x}'_t \in \mathbb{R}^{\mathrm{F}}$ represent the frame-level speaker embedding extracted by the student and teacher extractors at time t, respectively. By minimizing $\mathcal{L}_{\mathrm{distill}}$, the representation space of $\mathbf{X}$ is forced to align with $\mathbf{X}'$, which means the knowledge in the larger teacher extractor transfers into the smaller student extractor. Later, $\mathbf{X}$ and $\mathbf{X}'$ are fed into the shared encoder and decoder modules as same as the regular training framework described in Sec. III-B. The original ground-truth labels are also copied to supervise the two output branches.

During distillation, the total training loss is the sum of $\mathcal{L}_{\mathrm{bce}}$ in Eq. 3, $\mathcal{L}_{\mathrm{arc}}$ in Eq. 4, and $\mathcal{L}_{\mathrm{distill}}$ in Eq. 6. There are also three training stages, similar to the pretraining strategy.

- Stage 1: We initialize the weights of the student extractor from scratch and freeze the pretrained teacher extractor. Only simulated data is used to train the student extractor and shared encoder-decoder modules for 100,000 steps with a learning rate of *1e-4*.
- Stage 2: The teacher extractor is unfrozen. All weights in Fig. 6, including the student extractor, teacher extractor, and shared encoder-decoder modules, are adapted by 80% of the simulated data and 20% of the real data from the specific dataset, taking around 75,000 steps.
- Stage 3: The learning rate is decayed to *1e-5* for finetuning based on Stage 2, taking around 50,000 steps.

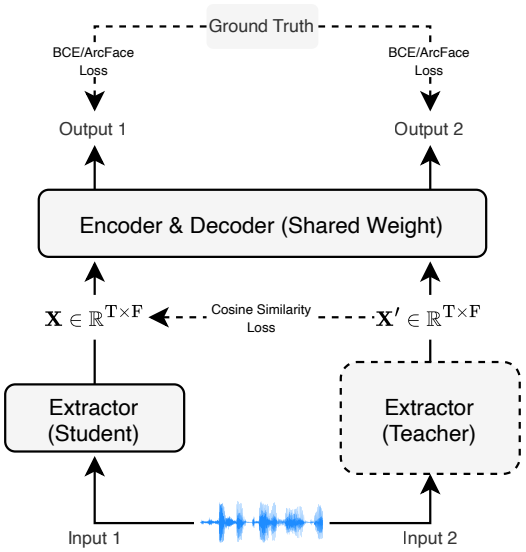


Fig. 6. Illustration of the training strategy based on knowledge distillation.

2) Inferring details: The inferring process follows the Sec. III-C. The thresholds τ_1 and τ_2 are determined using grid search on the validation set of the specific dataset. By adjusting the proportion of the current chunk and its contexts in the input audio block, the online diarization system can be flexibly inferred at different latencies. The algorithmic latency is the sum of chunk length L_{chunk} and right-context length L_{right}. As the shift of the sliding window is equal to the chunk length, a smaller L_{chunk} can decrease the system latency but bring intensive computing. The right context represents the use of future information. A larger L_{right} may result in more accurate prediction but increase the system latency. The impacts of different settings are investigated in the experimental results.

D. Evaluation Metric

The diarization error rate (DER) is used as the evaluation metric without collar tolerance. The S2SND models are tested on evaluation sets of DIHARD-II [77] and DIHARD-III [78] datasets. For a fair comparison, the Oracle VAD information can revise the diarization results as a post-processing approach [13] if sometimes the evaluation condition allows.

V. Results

A. Evaluation of S2SND Models

Table II illustrates the performance of our proposed S2SND models with different training and inferring conditions. The effects of model size, simulation corpus, training strategy, and various combinations of chunk and right-context lengths are shown step by step. Browsing the DER results on DIHARD-II and DIHARD-III datasets, several consequences are found as follows.

TABLE II
PERFORMANCE OF S2SND MODELS ON DIHARD-II AND DIHARD-III EVALUATION SETS WITH VARIOUS TRAINING AND INFERRING CONDITIONS. THE DIARIZATION ERROR RATES (DERS) ARE REPORTED WITHOUT ORACLE VAD AND COLLAR TOLERANCE.

ID	Model Size	Simulation Corpus	Training Strategy	Chunk Length	Right-Context Length	Algorithmic Latency	DIHARD-II Eval		DIHARD-III Eval	
							Online DER (%)	Offline DER (%)	Online DER (%)	Offline DER (%)
S1	Small	VoxCeleb2	Pretraining	0.48s	-	0.48s	27.79	23.74	20.55	16.28
S2				0.48s	0.16s	0.64s	26.11	23.85	18.53	16.33
S3				0.64s	-	0.64s	27.54	24.07	19.72	16.36
S4				0.64s	0.16s	0.80s	25.79	23.52	18.33	16.33
S5	Small	VoxBlink2	Pretraining	0.48s	-	0.48s	27.39	22.80	20.45	16.51
S6				0.48s	0.16s	0.64s	25.72	22.97	18.44	16.38
S7				0.64s	-	0.64s	26.93	23.05	19.55	16.32
S8				0.64s	0.16s	0.80s	25.70	23.17	18.36	16.45
S9	Small	VoxBlink2	Distillation	0.48s	-	0.48s	27.87	23.88	21.33	17.70
S10				0.48s	0.16s	0.64s	26.71	24.36	19.42	17.51
S11				0.64s	-	0.64s	27.58	24.28	20.67	17.57
S12				0.64s	0.16s	0.80s	26.21	24.29	19.23	17.57
S13	Medium	VoxCeleb2	Pretraining	0.48s	-	0.48s	27.57	23.78	20.61	16.81
S14				0.48s	0.16s	0.64s	25.57	23.61	18.82	16.97
S15				0.64s	-	0.64s	27.00	23.83	20.08	16.83
S16				0.64s	0.16s	0.80s	25.78	23.89	18.43	16.79
S17	Medium	VoxBlink2	Pretraining	0.48s	-	0.48s	27.79	24.09	20.13	16.04
S18				0.48s	0.16s	0.64s	26.02	23.86	18.10	15.77
S19				0.64s	-	0.64s	27.10	23.77	19.30	15.83
S20				0.64s	0.16s	0.80s	25.55	23.59	17.99	15.93
S21	Medium	VoxBlink2	Distillation	0.48s	-	0.48s	26.13	21.95	19.11	15.14
S22				0.48s	0.16s	0.64s	24.41	22.17	17.33	15.30
S23				0.64s	-	0.64s	25.44	22.07	18.46	15.13
S24				0.64s	0.16s	0.80s	24.48	22.26	17.12	15.23

The lowest online and offline DERs of each model size are highlighted by the gray background.

1) Across all experimental groups, under the same conditions (e.g., model size, simulation corpus, and training strategy), the longer chunk and right-context lengths can generally result in lower online DERs. Especially, using right-context information means that future information is exploited when predicting each chunk, which leads to a more significant impact. On the other hand, although the chunk length has less influence on the DERs, adjusting it can help maintain the algorithmic latency constant while increasing the right-context length. For instance, *S2* has lower online DERs than *S3*, even though their total latencies are equal. These phenomena are also shown in all the other experimental groups. To avoid too large system latency during online inference, the chunk and right-context lengths used in our experiments are selected to be relatively small and close values. Furthermore, the longer chunk and right-context lengths do not exhibit apparent advantages in offline DERs. The rescoring mechanism updates the diarization output over the whole recording, which already takes global information. Its offline performance is insensitive to the context length of the first-pass online inference.
2) Comparing *S1-4*, *S5-8*, and *S9-12*, we evaluate the small model with different simulation corpora and training strategies. It can be seen that the VoxBlink2 corpus containing larger speaker identities (111k+) does not result in significant and consistent improvement over VoxCeleb2 (6k+). Also, the training strategy of knowledge distillation slightly downgrades the performance compared to the pretraining strategy. It is speculated that the small model with few parameters cannot fully exploit the large simulation corpus and knowledge distillation.
3) Comparing *S13-16*, *S17-20* and *S21-24*, we evaluate the medium model with different simulation corpora and training strategies again. In this case, the combination of VoxBlink2 corpus and knowledge distillation demonstrates overwhelming advantages over others. All the lowest DERs for medium model on two datasets are obtained in *S21-24*. When increasing the number of model parameters, the newly introduced distillation strategy can successfully empower the usage of large speaker identities in the simulation corpus.

Overall, for the S2SND-Small model, the best online DERs on DIHARD-II and DIHARD-III datasets are 25.70% and 18.33%, and the best offline DERs on the two datasets are 22.80% and 16.28%, respectively. For the S2SND-Medium model, the best online DERs on DIHARD-II and DIHARD-III datasets are 24.41% and 17.12%, and the best offline DERs on two datasets are 21.95% and 15.13%, respectively. To summarize, the combination of a small simulation corpus and pretraining strategy is the better choice for the small model. When a large simulation corpus is available, adopting the medium model and distillation strategy can achieve better DER performance.

TABLE III
COMPARISONS OF S2SND MODELS WITH OTHERS ON THE DIHARD-II EVALUATION SET.

Method	Latency (s)	DER (%)
Online		
EEND-EDA + FW-STB [26]	1.00	36.00
EEND-EDA + Improved FW-STB [27]	1.00	33.37
Overlap-aware Speaker Embeddings [63]	1.00	35.10
EEND-GLA-Small + BW-STB [27]	1.00	31.47
EEND-GLA-Large + BW-STB [27]	1.00	30.24
S2SND-Small (S8 in Table II)	0.80	25.70
S2SND-Medium (S22 in Table II)	0.64	**24.41**
Online (with oracle voice activity detection)		
UIS-RNN-SML [62]	1.00	27.30
EEND-EDA + FW-STB [26]	1.00	25.80
EEND-EDA + Improved FW-STB [27]	1.00	24.67
Core Samples Selection [84]	1.00	23.10
EEND-GLA-Small + BW-STB [27]	1.00	23.26
EEND-GLA-Large + BW-STB [27]	1.00	21.92
NAVER System [85]	0.50	21.60
S2SND-Small (S8 in Table II) + Oracle VAD	0.80	**18.07**
S2SND-Medium (S22 in Table II) + Oracle VAD	0.64	18.65
Offline		
EEND-EDA [13]		29.57
+ Iterative Inference+ [13]		28.52
EEND-GLA-Small [27]		29.31
EEND-GLA-Large [27]		28.33
BUT System [36] †		27.11
+ EEND Post-Processing [86]		26.88
AED-EEND [87]		25.92
+ Embedding Enhancer [87]		24.64
S2SND-Small (S5 in Table II)		22.80
S2SND-Medium (S21 in Table II)		**21.95**
Offline (with oracle voice activity detection)		
EEND-EDA [13]		20.54
+ Iterative Inference+ [13]		20.24
VBx [8]		18.55
BUT System [36] †		18.42
S2SND-Small (S5 in Table II) + Oracle VAD		15.84
S2SND-Medium (S21 in Table II) + Oracle VAD		**15.34**

† Winning system on Track 1&2 of the DIHARD-II Challenge.

TABLE IV
COMPARISONS OF S2SND MODELS WITH OTHERS ON THE DIHARD-III EVALUATION SET.

Method	Latency (s)	DER (%)
Online		
Overlap-aware Speaker Embeddings [63]	1.00	27.60
EEND-EDA + Improved FW-STB [27]	1.00	25.09
EEND-GLA-Small + BW-STB [27]	1.00	22.00
EEND-GLA-Large + BW-STB [27]	1.00	20.73
ResNet-based OTS-VAD [30]	0.80	19.07
S2SND-Small (S4 in Table II)	0.80	18.33
S2SND-Medium (S24 in Table II)	0.80	**17.12**
Online (with oracle voice activity detection)		
Zhang et al. [23]	0.50	19.57
Core Samples Selection [84]	1.00	19.30
NAVER System [85]	0.50	19.05
EEND-EDA + Improved FW-STB [27]	1.00	18.58
EEND-GLA-Small + BW-STB [27]	1.00	15.82
EEND-GLA-Large + BW-STB [27]	1.00	14.70
ResNet-based OTS-VAD [30]	0.80	13.31
S2SND-Small (S4 in Table II) + Oracle VAD	0.80	13.07
S2SND-Medium (S24 in Table II) + Oracle VAD	0.80	**11.88**
Offline		
EEND-EDA [13]		21.55
+ Iterative Inference+ [13]		20.69
Pyannote.audio v3.1 [88]		21.30
DiaPer [45]		20.30
EEND-GLA-Small [27]		20.23
EEND-GLA-Large [27]		19.49
VBx + Overlap-aware Resegmentation [37]		19.30
USTC-NELSLIP System [17] †		16.78
ANSD-MA-MSE [60]		16.76
EEND-M2F [89]		16.07
S2SND-Small (S1 in Table II)		16.28
S2SND-Medium (S23 in Table II)		**15.13**
Offline (with oracle voice activity detection)		
EEND-EDA [13]		14.91
+ Iterative Inference+ [13]		14.42
Hitachi-JHU System [90]		11.58
USTC-NELSLIP System [17] †		11.30
ANSD-MA-MSE [60]		11.12
Seq2Seq-TSVAD [16]		10.77
MIMO-TSVAD [55]		**10.10**
S2SND-Small (S1 in Table II) + Oracle VAD		11.13
S2SND-Medium (S23 in Table II) + Oracle VAD		10.37

† Winning (fusion) system on Track 1&2 of the DIHARD-III Challenge.

B. Comparison with Other Existing Methods

We select the lowest online and offline DERs for each model size on DIHARD-II and DIHARD-III datasets as the representative results, highlighted by the gray background in Table II. To fairly compare with some existing methods, the corresponding results of post-processing by Oracle VAD [13] are also provided.

Table III compares our proposed methods with the previous state-of-the-art results on the DIHARD-II dataset. In the online scenario, our proposed methods obtain the lowest DERs of 18.07% and 24.41% with and without Oracle VAD, respectively. Regarding algorithmic latency, our proposed methods still have a significant advantage over others when Oracle VAD is not used. In the offline scenario, our proposed methods obtain the lowest DERs of 15.34% and 21.95% with and without Oracle VAD, respectively. Generally, our best results significantly outperform previous state-of-the-art systems in all scenarios. Notably, our best online DER (24.41%) is even lower than the previous best offline system (24.64%) [87] and the winning system (27.11%) [36] of the DIHARD-II Challenge.

Table IV compares our proposed methods with the previous state-of-the-art results on the DIHARD-III dataset. In the online scenario, our proposed methods obtain the lowest DERs of 11.88% and 17.12% with and without Oracle VAD, respectively. In the offline scenario, our proposed methods obtain the lowest DERs of 10.37% and 15.13% with and without Oracle VAD, respectively. Except for the offline result with Oracle VAD, our best results significantly outperform previous state-of-the-art systems in all other scenarios. Nevertheless, the DER (10.10%) of MIMO-TSVAD [55] comes from our earlier study designed for offline scenarios, which adopts the audio block of 32 seconds to provide extended context but is not suitable for online inference. Last but not least, our best online DER (17.12%) is very close to the previous best offline system (16.07%) [89] and the winning fusion system (16.78%) [17] of the DIHARD-III Challenge.

C. Investigation of Speaker Counting Ability

The previous speaker diarization systems mainly utilize unsupervised clustering [6]–[9], permutation-invariant train-

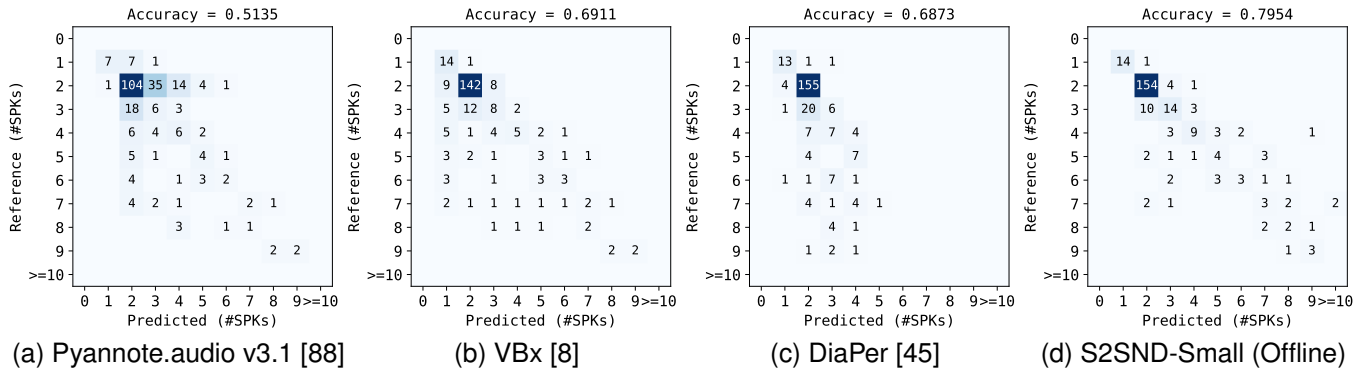


Fig. 7. Confusion matrices for speaker counting on the DIHARD-III evaluation set. The Pyannote.audio v3.1, VBx, and DiaPer results are provided by their respective authors. For our trained S2SND-Small model, the same settings as *S4* in Table II are adopted. Oracle VAD is not used. Accuracy is calculated as the number of recordings in which all speakers are correctly predicted, divided by the total number of recordings.

ing [10]–[13], or their combination to determine the unknown number of speakers in the input audio. In our proposed S2SND framework, speakers are detected by traversing the entire audio using the masked speaker prediction mechanism, which is clustering-free. Also, it only adds one unknown speaker each time to avoid the increasing complexity problem of the permutation-invariant training.

Fig. 7 depicts the confusion matrices for speaker counting obtained by different methods on the DIHARD-III evaluation set. Due to space limitations, only offline performances without Oracle VAD are shown. We select three representative systems from different technical routes for comparison. The first Pyannote.audio v3.1 [88] is a hybrid method of supervised end-to-end diarization and unsupervised clustering. The second VBx [8] is a well-known clustering-based diarization method, where the shown performance is reproduced as the baseline in the third EEND-based method (DiaPer [45]). As a result, the S2SND-Small model exhibits the more balanced predictions with the highest accuracy of 79.54%, proving the speaker counting ability of our proposed S2SND framework.

D. Computing Efficiency

Table V illustrates the computing efficiency of the S2SND models. First, the total number of model parameters is an essential measurement. Second, as mentioned in Sec IV-C2, the chunk length (L_{chunk}) determines the shift of the sliding window in our settings. Given the amount of floating-point operations for processing each window as Δ_{flops}, the Floating-Point Operations Per Second (FLOPS) is calculated as $\Delta_{\text{flops}}/L_{\text{chunk}}$. A smaller chunk length (window shift) means more windows must be processed per unit time, leading to more FLOPS because of more intensive computing. On the contrary, a larger chunk length (window shift) is more computationally economical, but the system latency will increase. Third, the Real-time Factor (RTF) is calculated as the time to process each recording divided by the recording length, where all tests are based on the computer with Intel(R) Xeon(R) E5-2660 CPU @ 2.60GHz and NVIDIA RTX-3090 GPU.

When using GPU inference, the maximum RTF of 0.22 is adequate for real-time applications. Also, when using CPU inference, the maximum RTF of 0.60 is not an excellent

TABLE V
COMPUTING EFFICIENCY REGARDING THE NUMBER OF PARAMETERS, FLOATING-POINT OPERATIONS PER SECOND (FLOPS), AND REAL-TIME FACTOR (RTF).

Model	**Params (M)**	**FLOPS (G)**	**RTF-GPU**	**RTF-CPU**
S2SND-Small				
$L_{\text{chunk}} = 0.48s$	16.56	78.83	0.19	0.34
$L_{\text{chunk}} = 0.64s$	16.56	59.12	0.14	0.21
S2SND-Medium				
$L_{\text{chunk}} = 0.48s$	45.96	308.89	0.22	0.60
$L_{\text{chunk}} = 0.64s$	45.96	231.67	0.15	0.39

performance, but it is less than 1, which means the system operation is still in real-time. Our calculations of RTFs include all the time the speaker diarization system runs, not only the neural network inference but also the actual data I/O, signal preprocessing, buffer update, etc. Therefore, the RTFs tested on the CPU are not much larger than that of the GPU, especially for the small model size. From the perspective of computing efficiency, our proposed S2SND models are not outstanding. Nevertheless, they achieve promising diarization performance (as shown in Tables III and IV) with improved speaker counting performance (as shown in Fig. 7). In our future work, we will further improve the computational efficiency.

Furthermore, comparing computing efficiency depends on various measurement criteria and hardware platforms. For instance, EEND-GLA-Small [27] has only 6.4M parameters. However, it additionally relies on clustering of relative speaker embeddings, which has $\mathcal{O}(n^3)$ time complexity but cannot be counted into FLOPS on the GPU device. OTS-VAD [30] employs an external VAD module to remove silent regions from the original audio signal. The preprocessing time (e.g., VAD) is not involved in the RTFs reported by the authors. It is hard to compare different studies in those aspects fairly. Thus, this paper does not list the computing efficiency comparisons with other methods.

VI. Conclusions

This paper proposes a novel Sequence-to-Sequence Neural Diarization (S2SND) framework to tackle online and offline speaker diarization in a unified model. The S2SND models can automatically detect and represent an unknown number of speakers in the input audio signal using the well-designed training and inferring process. Experimental results show that the proposed S2SND framework obtains new state-of-the-art DERs across all online and offline inference scenarios. Nevertheless, the proposed models also have limitations. The large model size and computing cost still present challenges for real-time inference on edge devices without GPUs. In the future, we will further improve the current approach regarding both precision and speed, prompting speaker diarization to wide industrial applications.

Acknowledgments

This research is funded in part by the National Natural Science Foundation of China (62171207), Yangtze River Delta Science and Technology Innovation Community Joint Research Project (2024CSJGG01100), Science and Technology Program of Suzhou City (SYC2022051) and Guangdong Science and Technology Plan (2023A1111120012). Many thanks for the computational resource provided by the Advanced Computing East China Sub-Center.

References

[1] 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.

[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. H. Shum, N. Dehak, R. Dehak, and J. R. Glass, "Unsupervised methods for speaker diarization: An integrated and iterative approach," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2015–2028, 2013.

[4] 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, 2014.

[5] G. Sell and D. Garcia-Romero, "Speaker diarization with plda i-vector scoring and unsupervised calibration," in *Proc. SLT*, 2014, pp. 413–417.

[6] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker diarization with lstm," in *Proc. ICASSP*, 2018, pp. 5239–5243.

[7] 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.

[8] 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*, vol. 71, p. 101254, 2022.

[9] 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.

[10] 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, pp. 4300–4304.

[11] 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.

[12] 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.

[13] S. Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and P. García, "Encoder-decoder based attractors for end-to-end neural diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1493–1507, 2022.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] W. Wang, D. Cai, Q. Lin, L. Yang, J. Wang, J. Wang, and M. Li, "The dku-dukeece-lenovo system for the diarization task of the 2021 voxceleb speaker recognition challenge," *arXiv preprint arXiv:2109.02002*, 2021.

[19] W. Wang, X. Qin, M. Cheng, Y. Zhang, K. Wang, and M. Li, "The dku-dukeece diarization system for the voxceleb speaker recognition challenge 2022," *arXiv preprint arXiv:2210.01677*, 2022.

[20] M. Cheng, W. Wang, X. Qin, Y. Lin, N. Jiang, G. Zhao, and M. Li, "The dku-msxf diarization system for the voxceleb speaker recognition challenge 2023," in *Proc. NCMMSC*, J. Jia, Z. Ling, X. Chen, Y. Li, and Z. Zhang, Eds. Springer Nature Singapore, 2024, pp. 330–337.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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, pp. 3116–3120.

[27] 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.

[28] D. Liang, N. Shao, and X. Li, "Frame-wise streaming end-to-end speaker diarization with non-autoregressive self-attention-based attractors," in *Proc. ICASSP*, 2024, pp. 10 521–10 525.

[29] W. Wang, M. Li, and Q. Lin, "Online target speaker voice activity detection for speaker diarization," in *Proc. INTERSPEECH*, 2022, pp. 1441–1445.

[30] W. Wang and M. Li, "Online neural speaker diarization with target speaker tracking," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[31] 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 *Proc. ICASSP*, 2018, pp. 5549–5553.

[32] M. Hrúz and Z. Zajíc, "Convolutional neural network for speaker change detection in telephone speaker diarization system," in *Proc. ICASSP*, 2017, pp. 4945–4949.

[33] G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe, and S. Khudanpur, "Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge," in *Proc. INTERSPEECH*, 2018, pp. 2808–2812.

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

[35] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *Proc. ICASSP*, 2018, pp. 5329–5333.

[36] F. Landini, S. Wang, M. Diez, L. Burget, P. Matějka, K. Žmolíková, L. Mošner, A. Silnova, O. Plchot, O. Novotný, H. Zeinali, and J. Rohdin, "But system for the second dihard speech diarization challenge," in *Proc. ICASSP*, 2020, pp. 6529–6533.

[37] H. Bredin and A. Laurent, "End-to-end speaker segmentation for overlap-aware resegmentation," in *Proc. INTERSPEECH*, 2021, pp. 3111–3115.

[38] Y. Takashima, Y. Fujita, S. Watanabe, S. Horiguchi, P. García, and K. Nagamatsu, "End-to-end speaker diarization conditioned on speech activity and overlap detection," in *Proc. SLT*, 2021, pp. 849–856.

[39] 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.

[40] ——, "Advances in integration of end-to-end neural and clustering-based diarization for real conversational speech," in *Proc. INTERSPEECH*, 2021, pp. 3565–3569.

[41] 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.

[42] S. Horiguchi, S. Watanabe, P. García, 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.

[43] M. Rybicka, J. Villalba, N. Dehak, and K. Kowalczyk, "End-to-end neural speaker diarization with an iterative refinement of non-autoregressive attention-based attractors," in *Proc. INTERSPEECH*, 2022, pp. 5090–5094.

[44] Y. Fujita, T. Komatsu, R. Scheibler, Y. Kida, and T. Ogawa, "Neural diarization with non-autoregressive intermediate attractors," in *Proc. ICASSP*, 2023, pp. 1–5.

[45] F. Landini, M. Diez, T. Stafylakis, and L. Burget, "Diaper: End-to-end neural diarization with perceiver-based attractors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 3450–3465, 2024.

[46] Y.-R. Jeoung, J.-Y. Yang, J.-H. Choi, and J.-H. Chang, "Improving transformer-based end-to-end speaker diarization by assigning auxiliary losses to attention heads," in *Proc. ICASSP*, 2023, pp. 1–5.

[47] D. Palzer, M. Maciejewski, and E. Fosler-Lussier, "Improving neural diarization through speaker attribute attractors and local dependency modeling," in *Proc. ICASSP*, 2024, pp. 11 911–11 915.

[48] Y. Dissen, F. Kreuk, and J. Keshet, "Self-supervised speaker diarization," in *Proc. INTERSPEECH*, 2022, pp. 4013–4017.

[49] Y. Takashima, Y. Fujita, S. Horiguchi, S. Watanabe, L. P. G. Perera, and K. Nagamatsu, "Semi-supervised training with pseudo-labeling for end-to-end neural diarization," in *Proc. INTERSPEECH*, 2021, pp. 3096–3100.

[50] M. He, D. Raj, Z. Huang, J. Du, Z. Chen, and S. Watanabe, "Target-speaker voice activity detection with improved i-vector estimation for unknown number of speaker," in *Proc. INTERSPEECH*, 2021, pp. 3555–3559.

[51] 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*, vol. 30, pp. 638–642, 2023.

[52] 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.

[53] W. Wang, X. Qin, and M. Li, "Cross-channel attention-based target speaker voice activity detection: Experimental results for the m2met challenge," in *Proc. ICASSP*, 2022, pp. 9171–9175.

[54] M. Cheng, H. Wang, Z. Wang, Q. Fu, and M. Li, "The whu-alibaba audio-visual speaker diarization system for the misp 2022 challenge," in *Proc. ICASSP*, 2023, pp. 1–2.

[55] M. Cheng and M. Li, "Multi-input multi-output target-speaker voice activity detection for unified, flexible, and robust audio-visual speaker diarization," *arXiv preprint arXiv:2401.08052*, 2024.

[56] Y. Jiang, R. Tao, Z. Chen, Y. Qian, and H. Li, "Target speech diarization with multimodal prompts," *arXiv preprint arXiv:2406.07198*, 2024.

[57] W. Wang, D. Cai, M. Cheng, and M. Li, "Joint inference of speaker diarization and asr with multi-stage information sharing," in *Proc. ICASSP*, 2024, pp. 11 011–11 015.

[58] Z. Chen, B. Han, S. Wang, Y. Jiang, and Y. Qian, "Flow-tsvad: Target-speaker voice activity detection via latent flow matching," *arXiv preprint arXiv:2409.04859*, 2024.

[59] S. Horiguchi, T. Moriya, A. Ando, T. Ashihara, H. Sato, N. Tawara, and M. Delcroix, "Guided speaker embedding," in *Proc. ICASSP*, 2025, pp. 1–5.

[60] M.-K. He, J. Du, Q.-F. Liu, and C.-H. Lee, "Ansd-ma-mse: Adaptive neural speaker diarization using memory-aware multi-speaker embedding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1561–1573, 2023.

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

[62] E. Fini and A. Brutti, "Supervised online diarization with sample mean loss for multi-domain data," in *Proc. ICASSP*, 2020, pp. 7134–7138.

[63] 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.

[64] A. Sholokhov, N. Kuzmin, K. A. Lee, and E. S. Chng, "Probabilistic back-ends for online speaker recognition and clustering," in *Proc. ICASSP*, 2023, pp. 1–5.

[65] Y. Chen, G. Cheng, R. Yang, P. Zhang, and Y. Yan, "Interrelate training and clustering for online speaker diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 1352–1364, 2024.

[66] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. ACL*. Association for Computational Linguistics, 2019, pp. 2978–2988.

[67] W. Chen, T. T. Anh, X. Zhong, and E. S. Chng, "Enhancing low-latency speaker diarization with spatial dictionary learning," in *Proc. ICASSP*, 2024, pp. 11 371–11 375.

[68] S. Wang, Z. Chen, K. A. Lee, Y. Qian, and H. Li, "Overview of speaker modeling and its applications: From the lens of deep speaker representation learning," *arXiv preprint arXiv:2407.15188*, 2024.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[70] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. INTERSPEECH*, 2020, pp. 5036–5040.

[71] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, vol. 30, 2017.

[72] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019, pp. 4171–4186.

[73] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proc. CVPR*, 2019.

[74] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Proc. INTERSPEECH*, 2018, pp. 1086–1090.

[75] Y. Lin, M. Cheng, F. Zhang, Y. Gao, S. Zhang, and M. Li, "Voxblink2: A 100k+ speaker recognition corpus and the open-set speaker-identification benchmark," in *Proc. INTERSPEECH*, 2024, pp. 4263–4267.

[76] Z. Gao, Z. Li, J. Wang, H. Luo, X. Shi, M. Chen, Y. Li, L. Zuo, Z. Du, and S. Zhang, "Funasr: A fundamental end-to-end speech recognition toolkit," in *Proc. INTERSPEECH*, 2023, pp. 1593–1597.

[77] 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.

[78] 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.

[79] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Proc. INTERSPEECH*, 2017, pp. 2616–2620.

[80] G. Hinton, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

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

[82] 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.

[83] I. Loshchilov, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[84] 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.

[85] Y. Kwon, H.-S. Heo, B.-J. Lee, Y. J. Kim, and J.-W. Jung, "Absolute decision corrupts absolutely: Conservative online speaker diarisation," in *Proc. ICASSP*, 2023, pp. 1–5.

[86] S. Horiguchi, P. García, Y. Fujita, S. Watanabe, and K. Nagamatsu, "End-to-end speaker diarization as post-processing," in *Proc. ICASSP*, 2021, pp. 7188–7192.
[87] Z. Chen, B. Han, S. Wang, and Y. Qian, "Attention-based encoder-decoder end-to-end neural diarization with embedding enhancer," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 1636–1649, 2024.
[88] A. Plaquet and H. Bredin, "Powerset multi-class cross entropy loss for neural speaker diarization," in *Proc. INTERSPEECH*, 2023, pp. 3222–3226.
[89] M. Härkönen, S. J. Broughton, and L. Samarakoon, "Eend-m2f: Masked-attention mask transformers for speaker diarization," in *Proc. INTERSPEECH*, 2024, pp. 37–41.
[90] S. Horiguchi, N. Yalta, P. Garcia, Y. Takashima, Y. Xue, D. Raj, Z. Huang, Y. Fujita, S. Watanabe, and S. Khudanpur, "The hitachi-jhu dihard iii system: Competitive end-to-end neural diarization and x-vector clustering systems combined by dover-lap," *arXiv preprint arXiv:2102.01363*, 2021.

Ming Cheng is currently a Ph.D. candidate in Computer Science at Wuhan University. He received his Master's degree in Electrical and Electronic Engineering from The University of Hong Kong and Bachelor's degree in Measuring and Control Technologies and Instruments from China Jiliang University. His research interests include speech signal processing and multimodal behavior analysis.

Yuke Lin is currently a Master's student in Computer Science at Wuhan University. He received his Bachelor's degree in Computer Science from Wuhan University. His research interests include speech signal processing (e.g., speaker verification and diarization).

Ming Li (Senior Member, IEEE) received his Ph.D. in Electrical Engineering from University of Southern California in 2013. He is currently a Professor of Electronical 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 processing. 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, the editorial board member of the IEEE/ACM Transactions on Audio, Speech, and Language Processing and Computer Speech & Language. He is an area chair at Interspeech 2016, 2018, 2020, 2024 and 2025 as well as the technical program co-chair of Odyssey 2022 and ASRU 2023. 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, ICASSP 2023 MISP challenge, IJCAI 2023 ADD challenge, ICME 2024 ChatCLR challenge and Interspeech 2024 AVSE 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.