Speaker Diarization with Deep Learning: Refinement, Online Extension and ASR Integration

by

Weiqing Wang

Department of Electrical and Computer Engineering
Duke University

Date: November 27, 2023
Approved:

Ming Li, Co-Supervisor

Xin Li, Co-Supervisor

Christ Richmond

Henry Pfister

Stacy Tantum

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University
2023

Abstract

Speaker Diarization with Deep Learning: Refinement, Online Extension and ASR Integration

by

Weiqing Wang

Department of Electrical and Computer Engineering
Duke University

Date: November 27, 2023

Approved:

Ming Li, Co-Supervisor

Xin Li, Co-Supervisor

Christ Richmond

Henry Pfister

Stacy Tantum

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University
2023

Abstract

As speech remains an essential mode of human communication, the necessity for advanced technologies in speaker diarization has risen significantly. Speaker diarization is the process of accurately annotating individual speakers within an audio segment, and this dissertation explores within this domain, systematically addressing three prevailing challenges through intertwined strands of investigation.

Initially, we focus on the intricacies of overlapping speech and refine the conventional diarization systems with the sequential information integrated. Our approach not only recognizes these overlapping segments but also discerns the distinct speaker identities contained within, ensuring that each speaker is precisely categorized.

Transitioning from the challenge of overlapping speech, we then address the pressing need for real-time speaker diarization. In response to the growing need for low-latency applications in various fields, such as smart agents and transcription services, our research adapts traditional systems, enhancing them to function seamlessly in real-time applications without sacrificing accuracy or efficiency.

Lastly, we turn our attention to the vast reservoir of the potential that lies within contextual and textual data. Incorporating both audio and text data into speaker diarization not only augments the system's ability to distinguish speakers but also leverages the rich contextual cues often embedded in conversations, further improving the overall diarization performance.

Through a coherent and systematic exploration of these three pivotal areas, the

dissertation offers substantial contributions to the field of speaker diarization. The research navigates through the challenges of overlapping speech, real-time application demands, and the integration of contextual data, ultimately presenting a refined, reliable, and efficient speaker diarization system poised for application in diverse and dynamic communication environments.

Contents

List of Tables

List of Figures

List of Abbreviations

AHC	Agglomerative Hierarchical Clustering
ASR	Automatic Speech Recognition
BCE	Binary Cross-entropy
BiLSTM	Bidirectional Long Short-Term Memory
BIC	Bayesian Information Criterion
CNN	Convolutional Neural Networks
CTS	Conversational Telephone Speech
DER	Diarization Error Rate
DIHARD	Diarization Is HARD
DNN	Deep Neural Network
EEND	End-to-end Neural Diarization
EER	Equal Error Rate
EDA	Encoder-Decoder-Based Attractor
FA	False Alarm
GMM	Gaussian Mixture Model
GLA	Global and Local Attractors
GLR	Generalized Likelihood Ratio
HMM	Hidden Markov Model
JER	Jaccard Error Rate
JFA	Joint Factor Analysis

JT	Joint Training
LDE	Learnable Dictionary Encoding
LSTM	Long Short-Term Memory
LSTM-SC	LSTM with Spectral Clustering
MC	Multi Channel
MFA	Multi-scale Feature Aggregation
MISS	Missed Detection
MSDD	Multi-Scale Diarization Decoder
NIST	National Institute of Standards and Technology
OTS-VAD	Online Target-speaker Voice Activity Detection
PLDA	Probabilistic Linear Discriminant Analysis
SAD	Speech Activity Detection
SAP	Self-attentive Pooling
SC	Spectral Clustering
SCD	Speaker-change Detection
SD	Speaker Diarization
Seq2Seq	Sequence-to-Sequence
SOTA	State-of-the-art
SP	Segmental Pooling
STD	Standard Deviation
SW-D	Speaker-wise Decoder
TAP	Temporal Average Pooling
TDNN	Time-delay Neural Network
TS-ASR	Target Speaker Automatic Speech Recognition
TS-VAD	Target-speaker Voice Activity Detection
TSP	Temporal Statistical Pooling

UBM	Universal Background Mode
VAD	Voice Activity Detection
VBx	Bayesian HMM clustering of x-vector sequences
VoxSRC	VoxCeleb Speaker Recognition Challenge
RTFs	Real-Time Factors

Acknowledgements

Embarking on this PhD journey at Duke University has been both challenging and enriching. Such a monumental task would not have been achievable without the steadfast support of numerous individuals who have been instrumental in my academic and personal journey.

Foremost, I owe a debt of gratitude to my esteemed advisors, Prof. Xin Li and Prof. Ming Li, whose guidance and unwavering belief in my capabilities have shaped my research endeavors. I am grateful for their invaluable insights and for introducing me to the world of research, having a profound impact on my academic trajectory.

I extend heartfelt thanks to my dissertation committee members, Prof. Christ D Richmond, Prof. Henry D. Pfister, and Prof. Stacy L. Tantum, for their invaluable feedback, critical reviews, and constructive suggestions. Their expertise and insights have significantly enriched the quality of this work.

Collaboration has been the backbone of my research experience. I am deeply appreciative of the brilliant minds I had the privilege to work alongside. Specifically, I wish to acknowledge Zexin Cai, Danwei Cai and Ming Cheng. The synergy, camaraderie, and intellectual exchanges with each one of them have been invaluable.

Finally, to my family, and especially my parents, words fall short in expressing my gratitude. Your unconditional love, sacrifice, and belief in me have been my guiding light. Your constant support, even from afar, has been the foundation upon which all my achievements stand. To each one of you: Thank you.

Chapter 1

Introduction

Speech is the most convenient, efficient, and natural way that people communicate with each other, as it is the main resource from which humans get information. With the Internet and multimedia technology development, intelligent voice speech technology has been widely employed in daily life. However, as communications between humans are always complicated, it is not easy to directly apply these technologies to a real-life scenario. Therefore, accurately detecting the boundary of the different speakers in speech and annotating the speech signal based on the speaker's identity has recently become an increasingly important research topic, and this is so-called speaker diarization.

Speaker diarization, which keeps a record of events in a diary, can label audio recordings with classes that correspond to the speaker's identity [2]. Here, the term "diarize" means to make a note or keep an event in a diary. Therefore, it is a process designed to answer the question: "Who spoke when?" Imagine a recorded meeting or conversation with multiple participants, as Figure 1.1 shows. The aim of speaker diarization is to automatically segment and label this recording, demarcating which sections belong to which speaker without having any prior knowledge of how many speakers there are or their specific identities.

Different from speaker recognition, which seeks to identify the actual identities

of the speakers, speaker diarization is content merely to differentiate, to assign different labels to different speakers, ensuring that segments from the same speaker are consistently labeled.

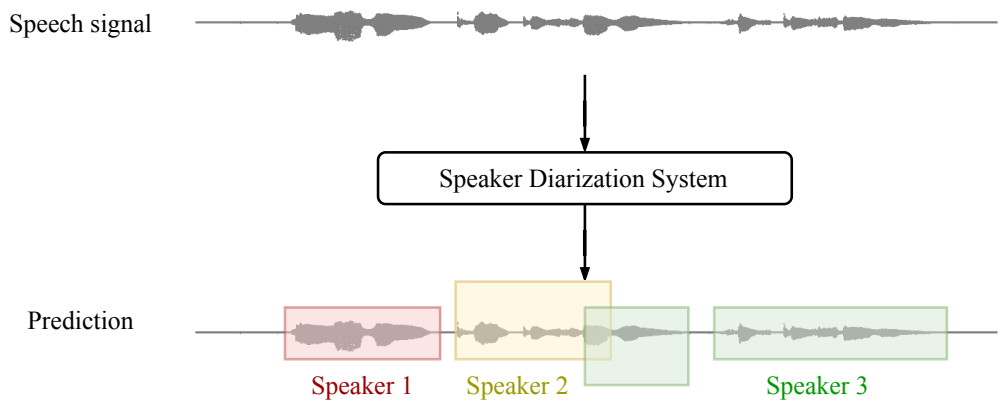


Figure 1.1: What is speaker diarization? Given an input signal, the speaker diarization system should annotate the speech signal based on the speaker's identity.

In its early days, speaker diarization was treated as an isolated task, somewhat detached from the broader realm of speech processing. However, with the continuous advancements in speech recognition, the role of speaker diarization has become increasingly intertwined with many speech applications. For instance, the output derived from a diarization process can be fed into an automatic speech recognition (ASR) system to enhance its accuracy. This collaboration between diarization and other speech technologies becomes more pertinent, especially as we witness a surge in voice-activated systems from autonomous vehicles to smart homes that increasingly demand voice assistant technologies.

In today's world, where vast amounts of audio data are being stored, there's a pressing need for automated methods to sift through this data and extract pertinent information. This has fueled interest in areas like speech recognition, speaker recognition, and, of course, speaker diarization. The real-world implications of speaker diarization are varied and vast. It aids in audio indexing, enhances speech recognition

systems, and has found relevance in human-computer and human-robot interactions.

To advance research in speaker diarization, the National Institute of Standards and Technology (NIST) organized the international Rich Transcription Evaluation series from 2002 to 2009 [3]. Following NIST's initiative, the Diarization is Hard I (DIHARD I) challenge was launched in the spring of 2018 [4]. This was succeeded by the DIHARD II [5] and DIHARD III [6] challenges in 2019 and 2020, respectively. Furthermore, the VoxCeleb Speaker Recognition Challenge (VoxSRC) has been continuously organized over the past five years, aiming to recognize speakers from unscripted speech samples [7, 8, 9, 10, 11]. Collectively, these competitions have enhanced the visibility of speaker diarization research and bolstered the robustness of diarization systems.

1.1 Research Motivation

Considering the system architecture, two primary types of speaker diarization systems are prevalent in the field: the modular system, which consists of multiple separate sub-modules, and the end-to-end system, which directly optimizes the diarization objective to yield results. The modular speaker diarization system divides audio recordings into short segments, which are then grouped based on the speaker's identity. Although it is the conventional approach of speaker diarization, this method has maintained its dominance in the field for several years due to it robustness. However, the system often overlooks the sequential data of speech segments during the clustering process. In addition, if a speech segment includes multiple speakers speaking simultaneously, it may not be accurately clustered. This is because the method typically assumes each segment has only one speaker. The end-to-end speaker diarization system, unlike the modular one, can produce the diarization results using only one single neural network as well as find the overlapping regions that contain

multiple speakers. However, this kind of model has to deal with the problem of speaker permutations, which requires a huge computational cost when the number of speakers is large. The detailed background of these two systems will be discussed in Chapter 2.

Depending on the immediacy of the application, diarization can be categorized as an offline or an online method. Actually, the aforementioned diarization systems, both the modular one and the end-to-end one, show promising performance on many datasets, but they are both designed to process pre-recorded audio and operate offline, meaning that they can analyze the entire audio file at once to identify the individual speakers. However, there are several practical applications, such as meeting transcription systems or smart agents, where the demand for low latency in speech processing is paramount [2]. Despite ongoing efforts to develop online speaker diarization systems, both within the realm of clustering-based approaches [12, 13] and neural network-based diarization frameworks [14, 15, 16], this remains a challenge that has yet to be fully overcome.

Depending on the modality of input data, diarization can be classified into an unimodal and a multimodal system. Multimodal data can significantly influence the performance of many systems. In this context, "modality" refers to the specific format in which certain information is stored. In the traditional system structures for speaker diarization, audio inputs are processed sequentially across the diarization components without considering the ASR performance, as speaker diarization usually serves as a pre-processing module for ASR. However, in certain scenarios, textual data can offer critical insights into speaker identification. For instance, distinguishing between a doctor and a patient becomes straightforward based solely on the content of their conversation. What's more, the joint modeling of speaker diarization and ASR may bring improvement for both sides, not only resulting in better ASR performance but also benefiting from ASR artifacts to enhance diariza-

tion performance.

Given the challenge mentioned above, the solutions presented in my research can be distilled into three main objectives:

1. Designing a refined modular speaker diarization system that considers sequential information. Furthermore, this system effectively identifies overlapping speech regions, where multiple speakers talk concurrently, without encountering speaker permutation challenges.

2. Adapting the speaker diarization system for real-time and online applications.

3. Integrating contextual/text information into the speaker diarization process for better performance.

These objectives are systematically presented in Chapters 3 through 5, respectively. To clearly present the relationship between each chapter, the taxonomy of the system presented in each chapter is shown in Table 1.1.

Table 1.1: Taxonomy of the objectives in Chapters 3, 4 and 5.

	Architecture		**Immediacy**		**Modality**	
	Modular	End-to-end	Offline	Online	Unimodal	Multimodal
Chapter 3	✓	-	✓	-	✓	-
Chapter 4	-	✓	-	✓	✓	-
Chapter 5	✓	✓	✓	✓	-	✓

1.2 Dissertation Outline

The remainder of this dissertation is organized as follows:

Chapter 2 provides an overview of the conventional pipelines of speaker diarization systems as well as the end-to-end model. The standard evaluation metric for the speaker diarization task is also included.

Chapter 3 addresses the challenges faced by conventional modular speaker diarization systems, particularly in handling overlapping speech involving multiple speakers. Through a two-stage diarization system, the study first employs a neural network approach for enhanced speaker embedding extraction and similarity measurement in the initial stage. The subsequent stage introduces a target-speaker voice activity detection (TS-VAD) technique to identify overlapping speech. This methodology integrates the outcomes of the first stage, refining voice activity detection for each speaker based on their corresponding identity profile, which significantly outperforms standard clustering methods.

Chapter 4 continues to refine the model introduced in Chapter 3, presenting a fully end-to-end framework for real-time speaker diarization, which can identify speaker activities and detect overlapping speech in an online manner. This model continually updates diarization results using incoming audio signals, maintaining consistent speaker order, and requires minimal memory during inference due to its small block size. The work is further extended to multi-channel data for better performance.

Chapter 5 further improves the model presenting in both Chapter 3 and Chapter 4, proposing a joint speaker diarization and ASR method by utilizing a pretrained ASR model. The method explores both offline and online scenarios, emphasizing the jointly modeling of ASR and speaker diarization for improved results.

Chapter 6 concludes the dissertation.

Chapter 2

Background: Pipelines of Speaker Diarization

This chapter provides an overview of key pipelines vital for crafting robust speaker diarization systems. We begin in Section 2.1 by introducing the evaluation metric of speaker diarization. In Section 2.2, we outline the sub-modules of the conventional modular speaker diarization system, which forms the foundation for Section 3. Section 2.3 delves into the end-to-end speaker diarization system and explains its mechanism.

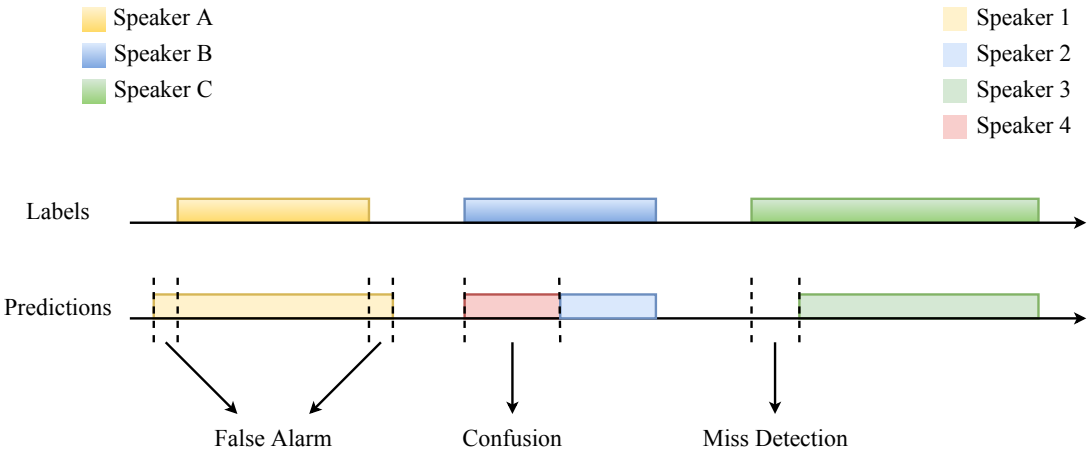


Figure 2.1: False alarm, miss detection and speaker confusion.

2.1 Evaluation Metric

2.1.1 Diarization error rate

The accuracy of a speaker diarization system is evaluated using the diarization error rate (DER) [17]. The DER encapsulates three distinct error categories: False alarm (FA) of speech, missed detection (MISS) of speech, and speaker confusion regarding speaker labels.

$$\text{DER} = \frac{\text{FA} + \text{MISS} + \text{Confusion}}{\text{Total duration}}. \tag{2.1}$$

Sometimes, a 0.25s "no score" collar is established around every reference segment boundary. This is to counteract potential discrepancies due to inconsistent notations and human inaccuracies in the reference transcript. DER is the prevalent evaluation metric in speaker diarization research and will be the primary metric in this dissertation.

2.1.2 Jaccard error rate

The Jaccard error rate (JER) was initially presented during the DIHARD II evaluation [5]. JER's aim is to assess each speaker impartially. Contrary to the DER, which evaluates an entire utterance collectively, JER first calculates error rates for individual speakers and then averages them. The computation of JER for an individual speaker is detailed as follows:

$$\text{JER}_i = \frac{\text{FA}_i + \text{MISS}_i}{\text{TOTAL}_i}, \tag{2.2}$$

where i is the index of the i-th speaker, and TOTAL_i is the union of the i-th speaker's speaking time in the reference transcript and the i-th speaker's speaking time in the hypotheses. Finally, the total JER is calculated by averaging all JER_i:

$$\text{JER} = \frac{1}{N} \sum_{i=1}^{N_{ref}} \text{JER}_i, \tag{2.3}$$

with N_{ref} denoting the total number of speakers in the reference transcript. It's worth noting that the speaker confusion observed in DER is accounted for in the FA_i portion when calculating JER. Given that JER employs a union operation between the reference and hypotheses, its value never surpasses 100%. In contrast, DER can substantially exceed this percentage. While DER and JER are closely related, JER may trend higher than usual if certain speakers predominate in a given audio recording.

2.2 Modular Speaker Diarization System

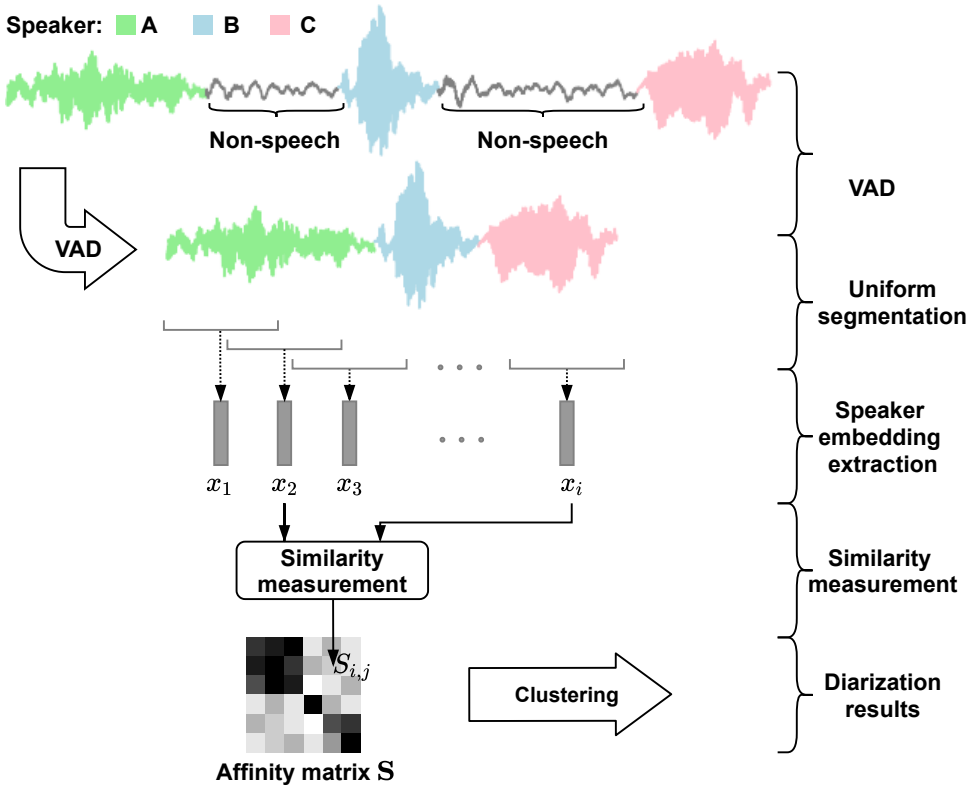


Figure 2.2: An example of the conventional modular speaker diarization system

A conventional modular speaker diarization system is composed of several distinct sub-modules, as illustrated in Figure 2.2. Initially, a voice activity detection (VAD) module filters out non-speech regions [18, 19, 20, 21]. Subsequently, the

identified speech portions are divided into short segments either by speaker-change point detection (SCD) [22, 23] or uniform segmentation [24]. From these segments, a speaker representation containing identity details is derived, enabling the calculation of segment-wise similarity to construct an affinity matrix. These segments are then grouped by speaker identity using various clustering algorithms, such as K-means [25], agglomerative hierarchical clustering (AHC), or spectral clustering [26, 27].

2.2.1 Voice activity detection

Voice activity detection (VAD), sometimes termed speech activity detection (SAD), discerns speech from non-speech elements. This includes silence, abrupt sounds like door closures and mouse clicks, and non-verbal noises produced by speakers, such as laughter, breaths, and coughs. Common VAD strategies determine if an input frame contains speech. Examples of these approaches include statistical methods like Gaussian Mixture Models (GMMs) [28] and Hidden Markov Models (HMMs) [29, 30], as well as deep learning techniques employing convolutional neural networks (CNN) [31] and long short-term memory (LSTM) [32].

In the context of speaker diarization, VAD's efficiency greatly influences the system's overall accuracy. This is because inaccuracies in VAD can either unnecessarily classify silent regions as speech or omit speech portions [33]. VAD's importance extends beyond not only speaker diarization but also speaker and speech recognition systems. Given its foundational role, inaccuracies in VAD can introduce errors that resonate throughout the entire process. Within speaker diarization studies, it's commonplace to report DER using an "oracle VAD" setup, indicating system outputs that match ground truth VAD outputs. Conversely, outputs generated with a voice activity detector are termed as "system VAD" outputs [2]. Usually, most of the FA and MISS in DER come from errors in the VAD system.

2.2.2 Segmentation

In speaker diarization, speech segmentation divides the input audio into segments, ensuring each segment is predominantly from a single speaker. Typically, there are two primary segmentation approaches: speaker-change point detection and uniform segmentation.

Earlier diarization systems primarily relied on speaker-change point detection. This method employs hypothesis testing on a sliding window split into two parts. The null hypothesis asserts that there is no speaker transition between the sections, which only requires a single model. On the other hand, the alternative hypothesis suggests a speaker transition, necessitating separate models for the two sections. Metrics such as the Generalized Likelihood Ratio (GLR) [34] and Bayesian Information Criterion (BIC) [35] are often employed for this detection. A limitation of this approach is the variability in segment lengths, potentially introducing inconsistencies in speaker representation and compromising its quality.

Uniform segmentation, on the other hand, divides the audio input using a fixed window and overlap, as depicted in Figure 2.2. This ensures a consistent duration for the diarization output. However, this method presents challenges, primarily the trade-off between segment length and speaker information. While segments must be short to likely pertain to a single speaker, they also need to encapsulate enough acoustic data for accurate speaker identification [2].

2.2.3 Speaker representation

In speaker diarization systems, speaker representation is essential to assess the similarity between speech segments. The primary objective is to convert a variable-length input signal into a fixed-length vector for easy comparison. This vector encapsulates speaker identity while filtering out extraneous information such as background noise

and channel mismatch. Currently, both statistical method like i-vector [36], as well as deep learning-based method like x-vector [37], shows excellent performance for speaker recognition and speaker diarization task.

i-vector

Before the emergence of speaker representations like i-vector [36] and x-vector [37], Gaussian Mixture Model-based Universal Background Models (GMM-UBM) [38] were introduced in acoustic feature-based speaker recognition tasks. UBMs typically consist of large GMMs, representing speaker-independent distribution of acoustic features. GMM-UBM models, which include mixture weights, mean values, and covariance matrices, were used to compare a speaker-adapted GMM against a speaker-independent one for verification purposes. However, GMM-UBM systems struggled with intersession variability [39], where the speaker enrollment process not only captured speaker-specific attributes but also unwanted channel noise and environmental nuisances.

Joint Factor Analysis (JFA) [39, 40] was introduced to mitigate these variability issues by separately modeling inter-speaker and channel/session variability. Building on JFA, Dehak et al. [36] proposed a method that combines channel and speaker information into a combined variability space. This approach led to the development of the i-vector, serving as a speaker representation vector. Each speaker and channel could now be modeled more effectively. The advent of i-vectors marked a significant advancement in speaker representation, leading to their widespread use not only in speaker recognition but also in numerous speaker diarization studies [41, 42, 43]. It has dominated this field for several years before the development of neural network-based speaker representations.

x-vector

The field of speaker diarization has seen significant developments with the integration of deep learning approaches for speaker representation. Initially introduced in the context of face recognition [44, 45], deep neural network (DNN)-based representation learning uses neural architectures to transform input signals (images or audio clips) into dense vector representations. This mapping relies on the nonlinear modeling capabilities of DNNs, enabling the network to learn this mapping without specifying particular components or factors, a marked contrast to traditional factor analysis models.

Unlike the component-based JFA, the parameters in DNN models for speaker embedding extraction are less explainable. Additionally, DNN-based learning does not rely on predefined probabilistic models, such as GMM-UBM, for processing acoustic features. This approach enhances efficiency, particularly during the inference phase, by avoiding computationally intensive operations like matrix inversion, which are typical in factor-analysis-based methods. Consequently, representation learning and inference speeds have improved compared to traditional methods.

A notable neural network-based speaker representation is the d-vector [46]. It employs stacked filter bank features, including context frames, and is trained using cross-entropy loss. Speaker representation vectors, or d-vectors, are derived from the final fully connected layer.

Further advancements in DNN-based speaker representations are epitomized by the x-vector [37]. Demonstrating superior performance, the x-vector won accolades at the NIST speaker recognition challenge 2018 [47] and the first DIHARD challenge [4]. Its architecture, distinct from the d-vector, incorporates a time-delay neural network (TDNN) and a statistics pooling layer. This layer aggregates frame-level outputs, computing their mean and standard deviation, and can handle variable

length inputs. This flexibility is particularly beneficial for speaker diarization, accommodating segments of varying lengths, including those truncated at the end of utterances.

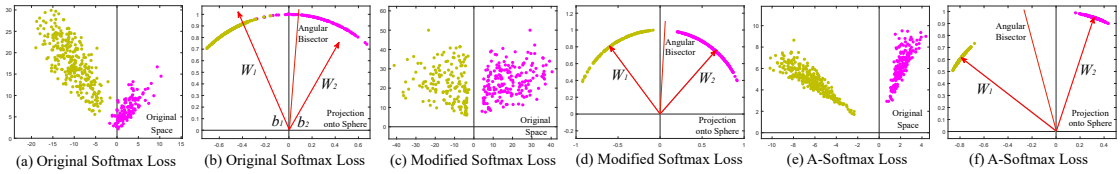
Building on the success of the x-vector architecture, further innovations have emerged by modifying its underlying network structure. Specifically, variants of the x-vector have been introduced by replacing the TDNN with CNN [48, 49] and Transformer-based networks [50, 51]. While these methods feature distinct architectural changes, they all retain the characteristic pooling-based aggregation strategy central to the x-vector design. As a result, despite their differences in implementation, they can be collectively classified as x-vector variants. These developments highlight the ongoing evolution and adaptation of speaker representation techniques in response to advancements in neural network architectures.

2.2.4 Similarity measurement

In the domain of speaker diarization, evaluating the similarity between speaker representations extracted from speech segments is crucial for effective clustering. Probabilistic Linear Discriminant Analysis (PLDA) and cosine similarity are two predominant techniques utilized for this assessment.

PLDA often demonstrates enhanced performance for i-vector and x-vector when compared to cosine similarity. Conversely, while cosine similarity is simple and does not require training or parameter adjustments, it lacks the capacity to project embedding vectors, limiting its ability to refine similarity measurements. In the realm of x-vector systems, this limitation is addressed by introducing various training objectives. These are designed to modify the softmax loss, encouraging the inherent angular distribution of the learned features [52], as illustrated in Figure 2.3. Currently, the combination of x-vectors with cosine similarity has become increasingly popular in speaker recognition and diarization due to its robustness under various

scenarios.



(a) Original Softmax Loss (b) Original Softmax Loss (c) Modified Softmax Loss (d) Modified Softmax Loss (e) A-Softmax Loss (f) A-Softmax Loss

Figure 2.3: Influence of different training objectives to speaker representations. (From Liu et al. [52])

2.2.5 Clustering

In speaker diarization, a clustering algorithm groups speech segments based on the speaker representation and similarity metrics. The primary clustering techniques include agglomerative hierarchical clustering (AHC) and spectral clustering.

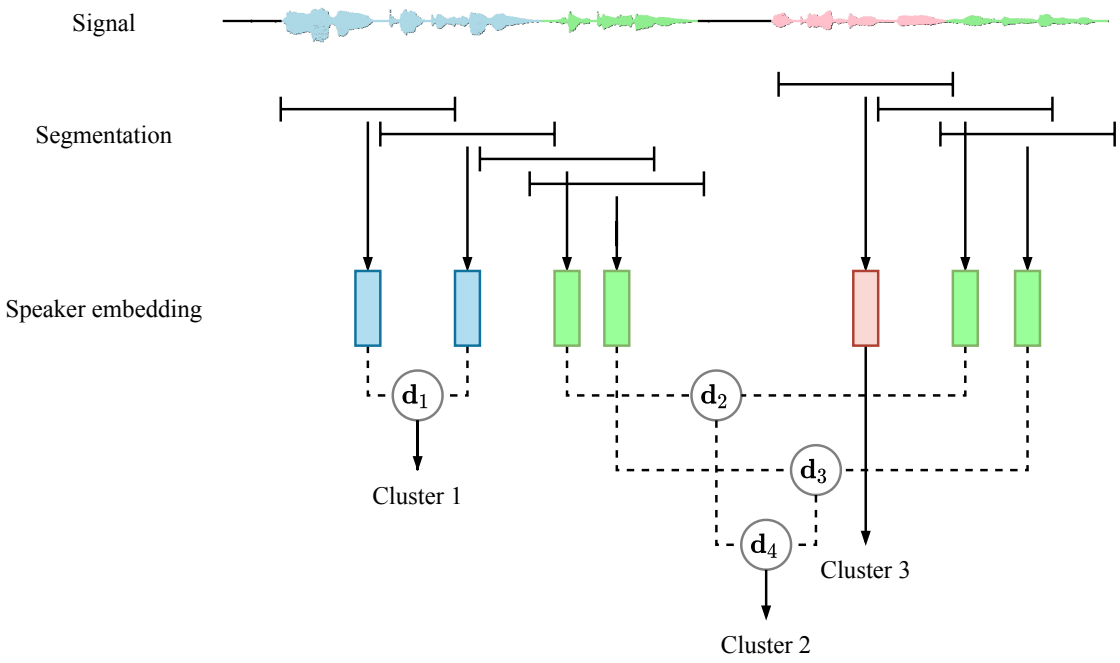


Figure 2.4: An example of agglomerative hierarchical clustering for speaker diarization.

AHC functions by iteratively merging clusters until a pre-set criterion is met. The process commences with determining the similarity among N singleton clusters.

In each iteration, the two clusters with the greatest similarity are merged. A key aspect of AHC is determining when to stop the clustering. In the context of speaker diarization, termination can be based on a pre-defined similarity threshold or the number of clusters. Figure 2.4 illustrates an example of AHC applied to speaker diarization, where $\mathbf{d}_i$ is the distance between different embedding or clustering.

Spectral clustering [53], another prevalent method in speaker diarization, employs eigenvalues and eigenvectors of the data to project data into a lower-dimensional space for clustering. Comprehensive details on spectral clustering will be covered in Chapter 3.

2.2.6 Pre-processing and post-processing

Both pre-processing and post-processing are pivotal stages in speaker diarization. Pre-processing involves converting the raw audio signal into formats that are more conducive to analysis. This includes steps like dereverberation [54], speech enhancement [55, 56], and speech separation [57, 58], etc. On the other hand, post-processing refines the initial diarization results to elevate performance. This refinement encompasses processes such as resegmentation [59], system fusion [60], overlap detection [61, 62, 63], and target speaker voice activity detection (TS-VAD) [64].

2.2.7 Strengths and weaknesses

Over recent decades, the modular speaker diarization system has been predominant, delivering commendable results across numerous competitions [7, 8, 9, 10]. Indeed, when each sub-module is robust—especially the VAD model and speaker representation—the system can effectively handle the data with a large number of speakers in a variety of scenarios. However, this strength can also be a double-edged sword. The modular nature can complicate system-wide optimization. Furthermore, since the speaker embedding is typically trained with the presumption of representing a single

speaker, it struggles with overlapping speech segments containing multiple speakers. While these overlapping regions can be identified during post-processing, it remains a challenge for the modular system.

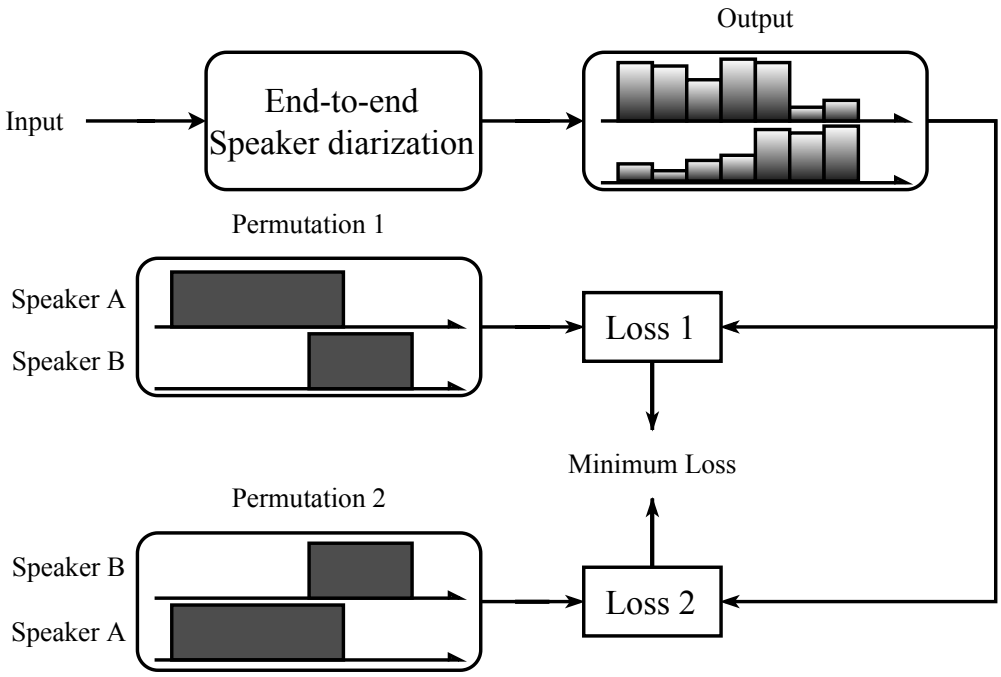


Figure 2.5: The end-to-end speaker diarization model for two speakers.

2.3 End-to-end Speaker Diarization System

2.3.1 Framework

Recently, an end-to-end neural diarization framework, termed EEND, was introduced [65, 66]. This approach conducts all speaker diarization pipelines through a singular neural network. As illustrated in Figure 2.5, when given an audio input comprising two speakers, EEND generates frame-level probabilities for each speaker over time axis. These predicted probabilities are subsequently matched with permutations of the actual labels to calculate respective losses. The minimum loss is then leveraged for optimization. Contrasting the modular speaker diarization approach, EEND can

directly optimize the diarization objective using the binary cross-entropy (BCE) loss [67]:

$$\mathcal{L}_{\text{diar}} = \frac{1}{TS} \min_{\phi \in \Phi(S)} \sum_{t=1}^{T} \text{BCE}(\mathbf{y}_t^{\phi}, \mathbf{p}_t), \tag{2.4}$$

where S is the number of speakers, T is the number of frames, $\Phi(S)$ encompasses all possible speaker permutations, ϕ is a specific permutation within $\Phi(S)$, $\mathbf{y}_t^{\phi}$ indicates the permuted actual label, and $\mathbf{p}_t$ denotes the EEND system's output. The BCE loss is given by:

$$\text{BCE}(\mathbf{y}_t, \mathbf{p}_t) = \sum_{s=1}^{S} \{-y_{s,t} \log(p_{s,t}) - (1 - y_{s,t}) \log(1 - p_{s,t})\}. \tag{2.5}$$

2.3.2 Strengths and weaknesses

EEND offers a distinct advantage over modular speaker diarization systems. While modular systems rely on cascade sub-modules to perform VAD, clustering and overlap detection, EEND detects each speaker's activity independently, obviating the need for additional speech activity and overlap detection modules. As depicted in Figure 2.5, the model directly yields frame-level probability values for each speaker. In scenarios where multiple speakers appear within a frame simultaneously (e.g., the 5th frame in Figure 2.5), EEND can predict high probabilities for all involved speakers. This inherent capability enables EEND to effectively handle overlapping speech.

Limitations of the end-to-end speaker diarization model also exist. A primary challenge is managing speaker permutations, wherein labels of differing sequences are all deemed correct, as visualized in Figure 2.5. To minimize loss and optimize the model, it's imperative to determine losses between outputs and all label permutations, adopting only the minimum loss for optimization. This process is computationally intensive, particularly when handling a large number of speakers. Moreover,

due to GPU memory constraints, the end-to-end model faces difficulties processing long-duration signals.

Chapter 3

A Two-stage Speaker Diarization Framework with Target Speaker Tracking

3.1 Introduction

As delineated in Chapter 2, traditional speaker diarization systems often presume a segment has only one speaker. This makes it challenging to manage overlapping speech involving multiple speakers. To address this, various pre- and post-processing techniques have been introduced to enhance system performance.

In the realm of pre-processing, speech separation techniques have demonstrated noteworthy results for both multi-channel [68, 69] and single-channel [70] diarization tasks, where the overlapping speech will be separated into two channels and each channels only contains one speaker. Although single-channel blind source separation might lead to speech "leakage", such challenges can be addressed through leakage filtering methods [70]. In terms of post-processing, the introduction of overlap detection [71] provides marginal improvements by associating overlapping regions with the two nearest speakers.

Recent academic explorations have concentrated on multi-speaker speech processing. These approaches aim to identify specific speaker information based on their

acoustic characteristics. Such methodologies have been adopted across various tasks, including target speech recognition [72, 73], target speech extraction [74, 75], and target speech detection [76]. Building on this foundation, Medennikov et al. [64] introduced target-speaker voice activity detection (TS-VAD), a method to determine frame-level posterior probabilities using the target speaker's i-vector. This approach has been particularly effective for highly overlapping speech. Moreover, Wang et al. [77] have recently integrated this i-vector-based TS-VAD technique on a multi-domain dataset, achieving impressive results. Notably, models for this method are trained individually for each domain, characterizing it as domain-specific.

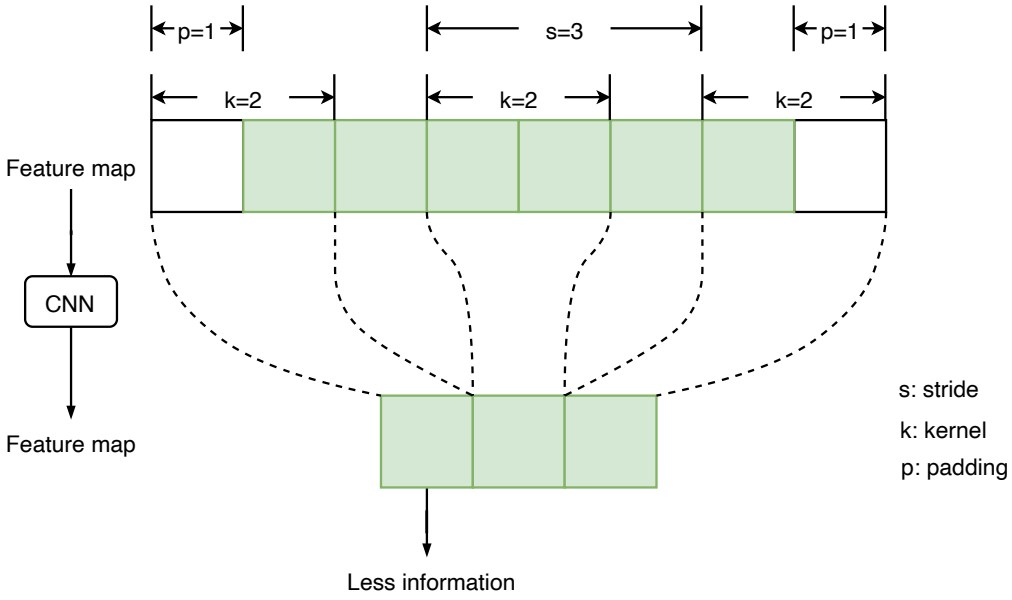


Figure 3.1: Illustration of receptive field of convolutional neural networks.

While each module described is crucial for the diarization system's effective performance, speaker embedding extraction is still the most important sub-module. In recent years, many clustering-based speaker diarization systems have harnessed speaker embeddings derived from uniformly segmented utterances. However, there are inherent challenges with this approach:

1. Most clustering methodologies focus primarily on local similarities between

pairs of embeddings, which are typically extracted from short segments. This approach neglects the sequential context information, as shown in Figure 2.4. Given that conversation contents are often highly structured, this omission is significant.

2. Current trends favor CNN-based architectures like ResNet for embedding extraction. However, issues arise with zero-padding in diarization tasks, which is frequently overlooked. Specifically, frame-level speaker data near segment borders is less informative than central frame data due to the non-informative zero-padding at these edges, as depicted in Figure 3.1. This challenge becomes more pronounced with shorter segments, which are common in speaker diarization tasks.

3. While longer speech segments yield more informative speaker embeddings, these embeddings aren't always speaker-homogeneous, complicating speaker identity clustering. Conversely, shorter segments offer a finer time resolution for the affinity matrix but capture less speaker information, challenging similarity measurement and clustering. Thus, striking a balance in segment length is crucial in a modular speaker diarization system.

In this chapter, a two-stage diarization system is introduced to address the previously discussed limitations. The initial stage utilizes a clustering-based approach, while the second stage employs the TS-VAD method to refine its outcomes. Figure 3.2 shows a complete process for the proposed two-stage diarization system.

For the first stage, we employ a neural network approach, replacing the traditional PLDA or cosine similarity, to measure the similarity between speaker embeddings in the conventional clustering-based method. Contrasting the majority of AHC methods that ignore temporal data, our method leverages BiLSTM or self-attention to

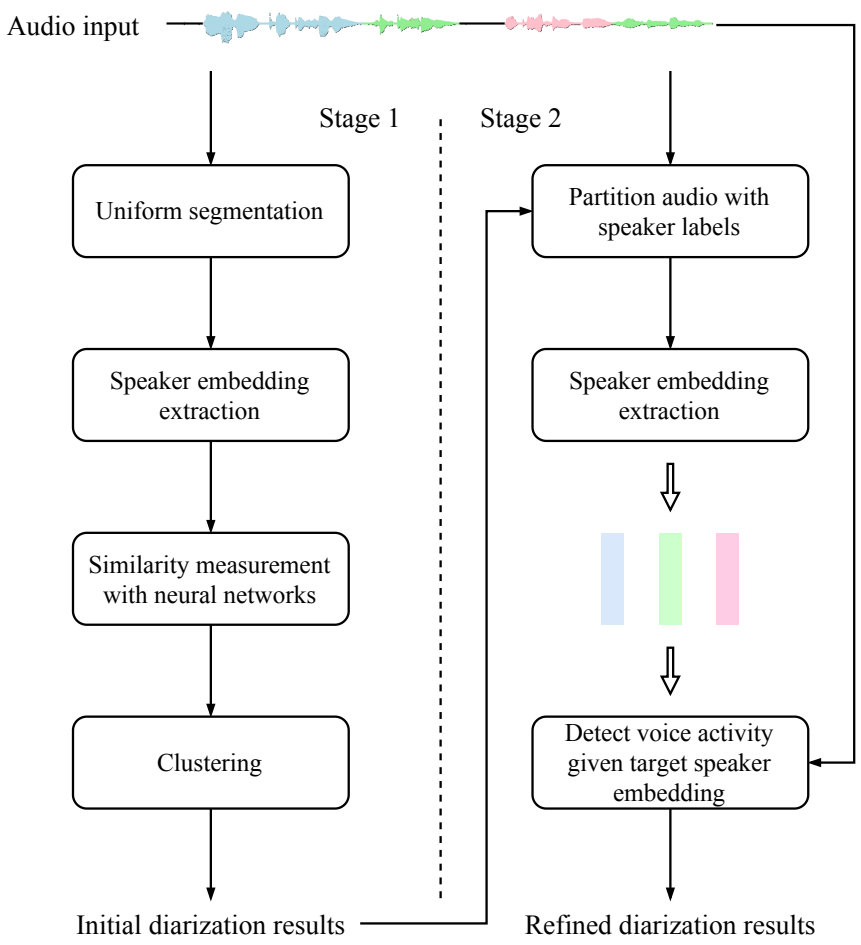


Figure 3.2: The process of the proposed two-stage diarization system.

derive the correlation across a sequence of speaker embedding pairs. Moreover, our approach consistently requires only a single threshold as a hyperparameter for spectral clustering, ensuring consistent diarization results. This is in contrast to AHC, which necessitates tailored threshold adjustments based on varying datasets and speaker embedding extractors. We further enhance performance by training both the embedding extraction and similarity assessment models concurrently. Within this stage, we uphold the conventional modular speaker diarization system's guidelines but introduce optimization to certain modules, including speaker embedding extraction and similarity measurement with temporal information integrated.

In the second stage, we present an x-vector-based TS-VAD technique to pinpoint overlapping speech, integrating outcomes from the first phase. Here, we generate an x-vector for every speaker derived from the initial stage's clustering. This enables voice activity detection for each speaker based on its corresponding x-vector, termed as target-speaker embedding. Given TS-VAD's ability to discern voice activities even in overlapping segments, it consistently outperforms clustering techniques provided the number of speakers is accurately identified in the first stage. During this phase, we combine the target speaker embedding with segment-level embedding to gauge their similarity, eliminating the need for clustering. With neural networks assessing similarity instead of basic metrics like cosine distance, the time resolution of segment-level embedding can be set exceptionally high for a better result.

The proposed framework makes the following contributions:

- We propose a robust two-stage framework for speaker diarization that solves the problem of overlapping speech detection
- We propose a joint training framework of the speaker embedding network and the similarity measurement network using a speaker diarization objective.
- We successfully apply the x-vector to the TS-VAD task, reformulating the TS-VAD as an embedding measurement method, and introducing the pooling strategy to TS-VAD at the segment level for high-resolution similarity measurement.
- We apply a two-speaker TS-VAD method for overlap detection.

3.2 Segment-level Similarity Measurement

In general, a conventional modular speaker diarization system contains VAD, speaker embedding extraction, similarity measurement, and clustering. In this section, we

mainly focus on CNN-based speaker embedding extraction and similarity measurement under the conventional modular speaker diarization framework.

3.2.1 Speaker embedding extraction with segmental pooling

For speaker embedding, we follow the ResNet-style x-vector introduced in [48, 78]. As shown in Figure 3.3, the front-end extractor takes the acoustic features as the input and produces a CNN feature map. Later, a pooling layer aggregates the temporal information in this CNN feature map over time and generates a fixed-length representation. Finally, the penultimate layer produces the utterance-level representation.This entire network is trained for speaker classification, where the classes correspond to the training speakers.

As the pooling layer aggregates the temporal information into a fixed-length

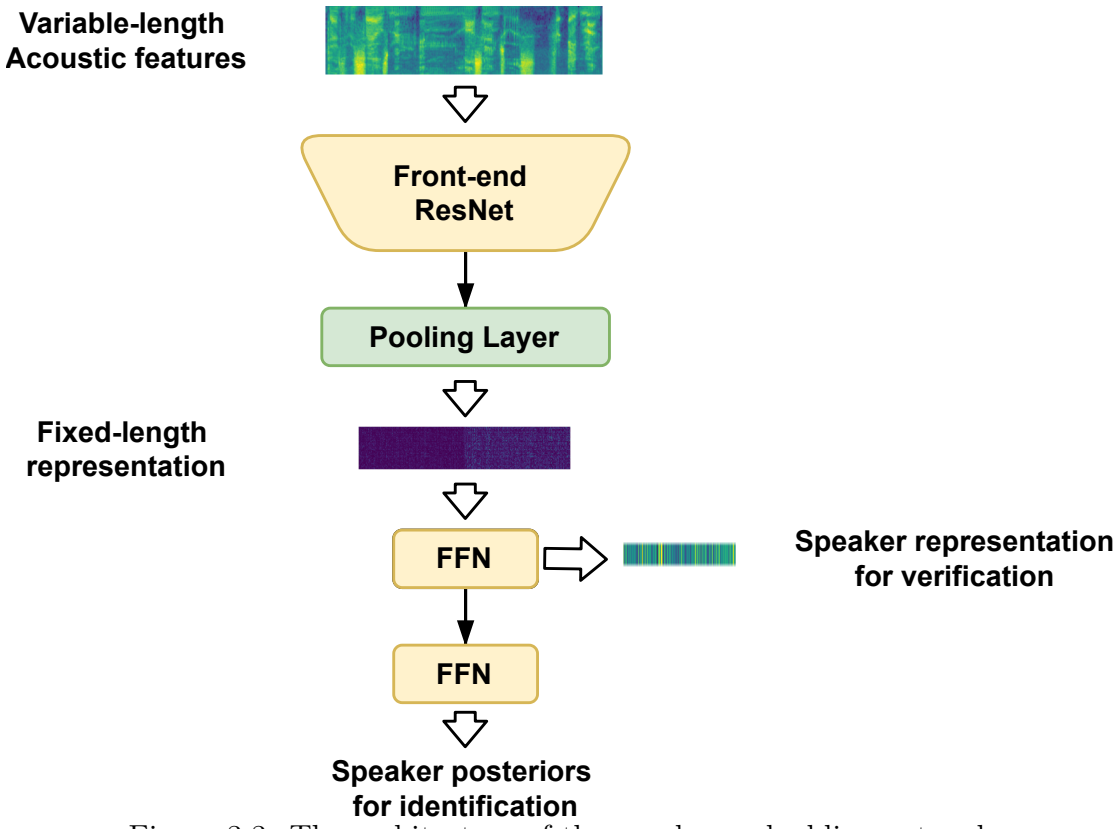


Figure 3.3: The architecture of the speaker embedding network

representation, the design of the pooling layer can directly influence the network performance. Several pooling layers have been explored under this ResNet-based framework and proved to be effective, including temporal average pooling (TAP), temporal statistical pooling (TSP), self-attentive pooling (SAP), and learnable dictionary encoding (LDE) layers [79, 80, 48].

In the speaker diarization task, the input signals are commonly broken into several short segments by uniform segmentation with a fixed window length and window shift. Next, the speaker embeddings are extracted from these segments to represent the identity of the speaker, as Figure 3.4 shows. However, the receptive field of most speaker recognition networks is longer than the length of the segments, e.g., the receptive field of the ResNet34 is over 2 seconds for each frame [81]. This means that the CNN feature map produced by the front-end extractor contains less information when the frames are close to the borders for short segments, whereas the central frames contain more information, as illustrated in Figure 3.1. This is caused by the zero-padding and ignoring the actual context.

Therefore, we propose segmental pooling (SP) to perform the uniform segmentation and pooling operation in a single step, where the segmentation is directly performed on the CNN feature map rather than the audio waveform. First, the audio signal is split into different speech segments based on the oracle VAD label, and we extract the CNN feature map for each speech segment with the front-end ResNet34. Therefore, the zero-padding is applied only at the boundaries of these (often relatively long) segments defined by VAD. As Figure 3.5 shows, we then uniformly split the CNN feature map with a fixed window length and window shift, and we feed these segmented feature maps into the pooling layer to obtain a fixed-length representation for each segment. Finally, the feed-forward network generates the speaker embedding. Note that the parameters of the front-end extractor and the feed-forward network are not changed. We only perform segmentation on the feature

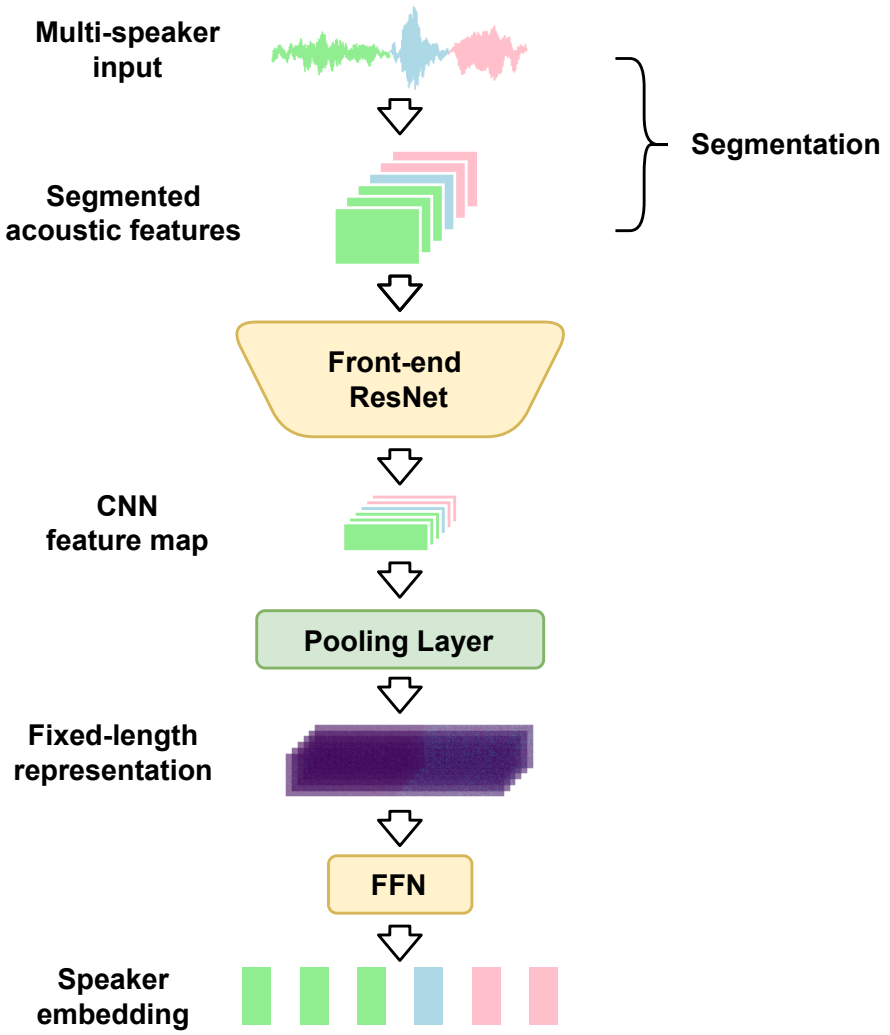


Figure 3.4: The uniform segmentation with speaker embedding extraction in the conventional speaker diarization system

maps before the pooling layer rather than on the input signals. Note that the speaker embeddings extracted this way are not affected by any zero-padding unless they are close to the boundary of the original VAD-based segment. As a consequence, each such embedding effectively "sees" a larger temporal context through the ResNet34 receptive field (3.72s in our experiments) as compared to the embeddings extracted from segmented speech signal (1.28s). In fact, except for the VAD-based segment boundaries where the zero-padding matters, the segmental pooling could be simulated by switching off the zero-padding over time and extracting embeddings from

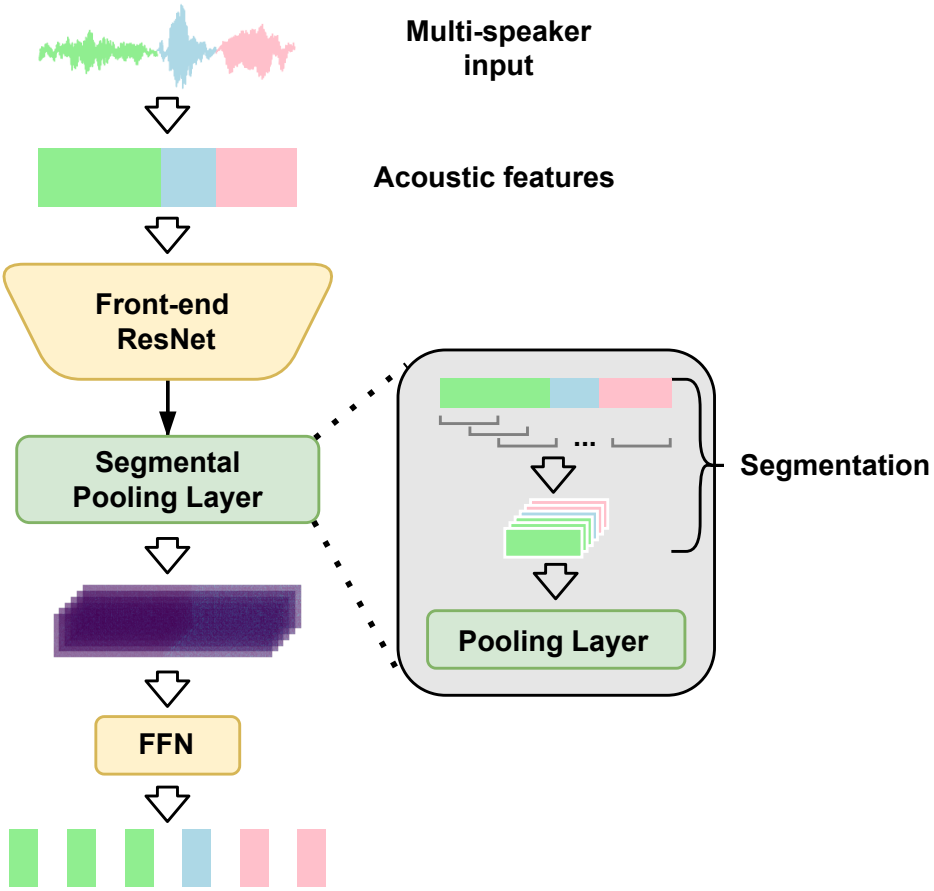


Figure 3.5: The segmental pooling strategy, where the segmentation is performed after the front-end feature extractor

3.72s speech signal segments.

We employ a statistical SP layer after the segmentation [79]. Consider an acoustic feature matrix $\mathbf{A} \in \mathbb{R}^{F \times L}$. The front-end extractor f_{fe} can extract a CNN feature map $\mathbf{M} \in \mathbb{R}^{C \times H \times T}$, where F and L are the dimensions and length of the acoustic feature, respectively; C is the number of channels, and H and T are the height and width of the CNN feature map, respectively. Usually, $H = \frac{F}{8}$ and $T = \frac{L}{8}$ in our ResNet34-based front-end extractor; thus, the CNN feature map can be considered as a downsampling from the acoustic feature. Next, we uniformly split the CNN feature map into short segments with a length T'=16 frames and a shift 8 frames. The dimensions of each segment feature map $\mathbf{M}_i$ are $C \times H \times T^{'}$, where $T^{'}$ is the segment

Table 3.1: The network architecture for embedding extraction, where $\mathbf{C}$(kernel size, stride) denotes the convolutional layer, $[\cdot]$ denotes the residual block; L relates to the duration of the speech and L relates to the number of frequency bins of the Mel spectrogram.

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

length. Finally, we perform statistical pooling on each channel of the segmented CNN feature map. Let $\mathbf{M}_i^c \in \mathbb{R}^{H \times T'}$ denote the feature map of the c-th channel; the SP layer aggregates the two-dimensional feature map as follows:

$$\mu_i^c = \frac{1}{HT'} \sum_{h=1}^{H} \sum_{t=1}^{T'} \mathbf{M}_{i,h,t}^c, \boldsymbol{\mu}_i = [\mu_i^1, ..., \mu_i^C]^\intercal \tag{3.1}$$

$$\sigma_i^c = \sqrt{\frac{1}{HT'} \sum_{h=1}^{H} \sum_{t=1}^{T'} (\mathbf{M}_{i,h,t}^c - \boldsymbol{\mu}_i)^2}, \boldsymbol{\sigma}_i = [\sigma_i^1, ..., \sigma_i^C]^\intercal, \tag{3.2}$$

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

3.2.2 Similarity measurement

In the conventional modular speaker diarization system, the similarities between different segments are computed in a pairwise manner [71]. However, conversations between different speakers are usually highly structured. To take the sequential information into consideration, we propose a similarity measurement method using a recurrent neural network, where both previous and following segments are considered to improve the performance. In this section, we introduce our work using BiLSTM and self-attention for the supervised similarity measurement, which is the baseline in our experiments.

BiLSTM

The LSTM network architecture was originally developed in [82], but the early version of LSTM can only capture the previous context. Later, Schuster et al. [83] proposed BiLSTM, which utilizes the information from both the previous and future context. BiLSTM has shown success in Automatic Speech Recognition (ASR) [84] and speech synthesis [85]. We also employ the BiLSTM to measure the similarity between two speaker embeddings with the previous and future context included.

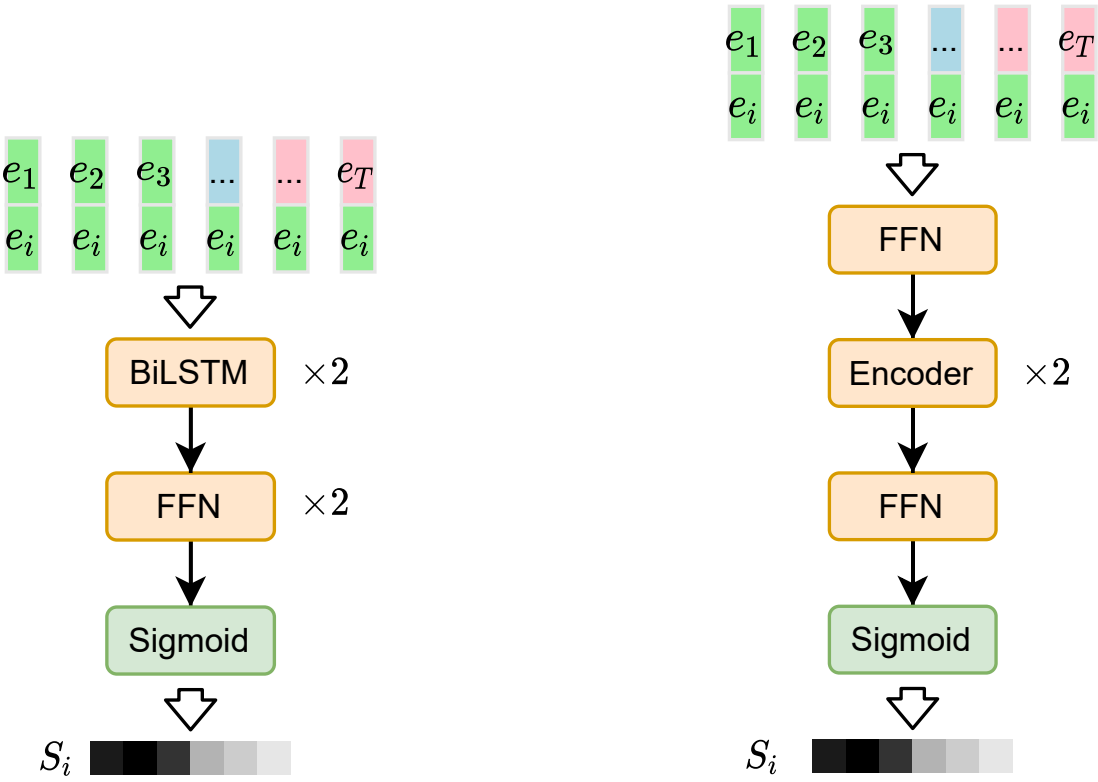
Considering an affinity matrix $\mathbf{S} \in \{0,1\}^{T\times T}$ for an embedding sequence $\mathbf{E} = [\mathbf{e}_1^\intercal, \mathbf{e}_2^\intercal, ..., \mathbf{e}_T^\intercal]$, where $\mathbf{S}_{i,j}$ is 1 when $\mathbf{e}_i$ and $\mathbf{e}_j$ are from the same speaker and 0 otherwise, our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

$$\mathbf{S}_i = [\mathbf{S}_{i,1}, \mathbf{S}_{i,2}, ..., \mathbf{S}_{i,T}] = f_l(\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_i\end{bmatrix}, \begin{bmatrix}\mathbf{e}_2\\ \mathbf{e}_i\end{bmatrix}, ..., \begin{bmatrix}\mathbf{e}_T\\ \mathbf{e}_i\end{bmatrix}), \tag{3.3}$$

where f_l is the BiLSTM network. The complete affinity matrix $\mathbf{S}$ can be obtained

by stacking all rows $\mathbf{S}_i$ together.

The architecture of the BiLSTM network is the same as the network in [26]. As shown in Figure 3.6a, a two-layer BiLSTM takes a sequence of speaker embeddings as input, and two fully-connected layers with a sigmoid function predict the probability that two embeddings correspond to the same speaker, which is one row of the affinity matrix. As will be pointed out in Section 3.3, this process is similar to the TS-VAD method [64], where the target speaker embedding is the average of many "target segment embeddings" $\mathbf{e}_i$ from the same speaker.



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

Figure 3.6: The architectures of the networks for similarity measurement

Self-attention

As an alternative to the BiLSTM-based architecture, we also use the Transformer-based model [86] for the similarity measurement. The inputs and training objectives are the same as for the BiLSTM-based model. BiLSTM layers are replaced with one fully connected layer followed by two Transformer self-attention encoder layers. Unlike in [86], we do not use the positional encoding as it results in performance degradation. Finally, fully connected layers with a sigmoid function predict the similarities. Figure 3.6b shows the architecture of the self-attention-based model. The encoder contains 2 layers, and each layer contains 2 heads with 1024 attention units for each head.

3.2.3 Post-processing for predicted affinity matrix

Affinity matrix partitioning

In practice, the embedding sequence can be long, which requires a large amount of memory. With the LSTM model, we cannot handle long sequences because of memory limitations. Therefore, we split the embedding sequences into several fixed-length overlapping short sub-sequences, and the affinity matrix is also broken into several blocks. Specifically, if we have N sub-sequences, there should be N^2 blocks in the affinity matrix, where the i^{th} and j^{th} sub-sequences can form the block in the corresponding position. After we obtain all blocks, we merge all blocks by placing them in the corresponding location, where the overlapping regions are averaged to produce the complete affinity matrix. This way, we can process each affinity block as a mini-batch and then combine these blocks to form the complete affinity matrix, as shown in Figure 3.7.

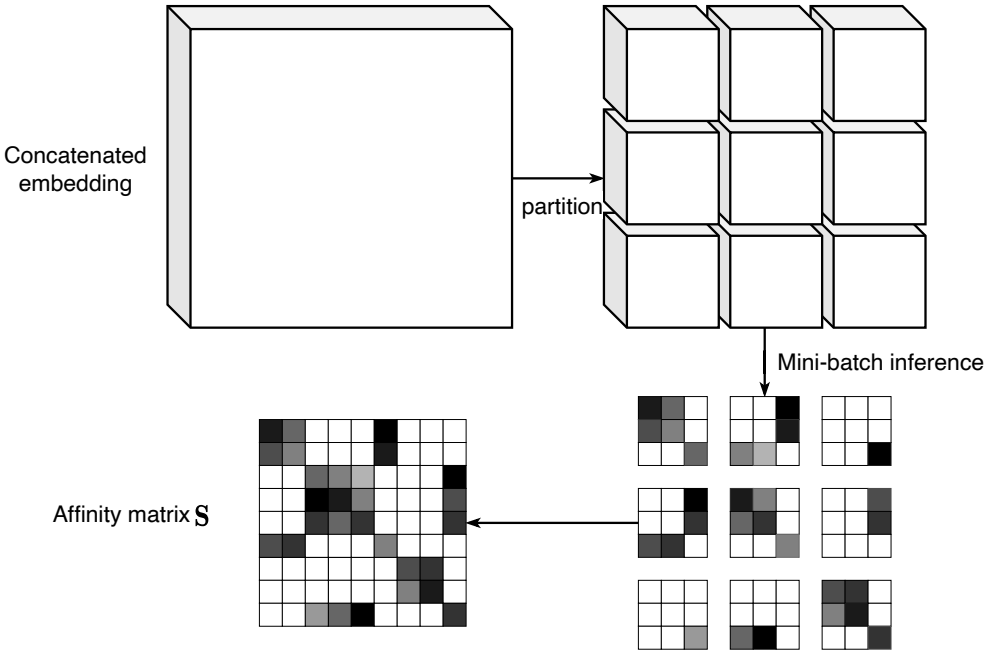


Figure 3.7: The process of the affinity matrix partitioning.

Affinity matrix refinement

As the affinity matrix is processed row by row, it is not symmetric and contains a lot of noise. Hence, the affinity matrix can be smoothed, symmetrized, and enhanced as follows [25]:

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

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

Spectral clustering

After we obtain the refined affinity matrix $\mathbf{S}'$, we employ spectral clustering to obtain the diarization results as mentioned in [53] as follows:

1. Construct the affinity matrix $\mathbf{S}^{'} \in \mathbb{R}^{n \times n}$ and set all diagonal entries to 0.

2. Generate the normalized Laplacian matrix $\mathbf{L}_{\text{norm}}$:

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

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

   where $\mathbf{L}$ is the Laplacian matrix, $\mathbf{D}$ is a diagonal matrix and and $\mathbf{D}_{i,i} = \sum_{j=1}^{n} \mathbf{S}^{'}_{i,j}$.

3. Compute the eigenvalues λ and corresponding eigenvectors $\mathbf{u}$ of $\mathbf{L}_{\text{norm}}$.

4. Compute the number of clusters k. In experiments, we employ a threshold β and find the number of eigenvalues lower than β as k.

5. Find the largest k eigenvalues $\lambda_1, ..., \lambda_k$ and corresponding eigenvectors $\mathbf{u}_1, ..., \mathbf{u}_k$.

6. Let $\mathbf{U} \in \mathbb{R}^{n \times k}$ containing $\mathbf{u}_1, ..., \mathbf{u}_k$ as columns.

7. Cluster the row vectors of $\mathbf{U}$ by the k-means algorithm.

3.2.4 Data augmentation

If only given limited data, our similarity measurement model is easily overfitted to some particular region in the speaker embedding space. One common solution for speech data augmentation is to add background noise and reverberation to the training data, which can increase the model robustness for noisy data. However, this method cannot generate data for new unseen speakers, and the model will lack generalization ability with only hundreds of training speakers. For this purpose, we employ an embedding-level data augmentation strategy, which can rotate all the speaker embeddings to another region of the embedding space without changing the inter-class and intra-class distance. This augmentation method was first proposed in [87].

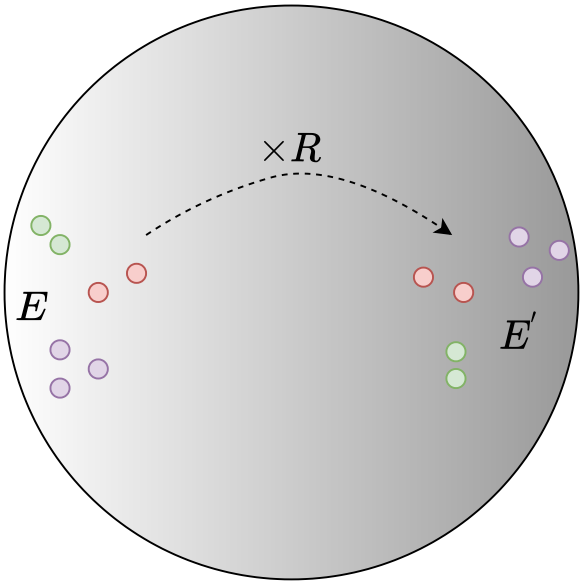


Figure 3.8: An example of the random orthonormal transformation on the speaker embedding sphere.

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

$$\mathbf{E}' = \mathbf{RE} = [\mathbf{Re}_1^\intercal, \mathbf{Re}_2^\intercal, ..., \mathbf{Re}_T^\intercal], \tag{3.4}$$

where $\mathbf{R} \in \mathbb{R}^{D \times D}$ are random orthonormal basis. Figure 3.8 shows an example of the random orthonormal transformation.

After performing the on-the-fly data augmentation on the speaker embeddings, we can generate large-scale samples that have the potential to span the whole speaker embedding space with only limited real data. The model can focus on learning the difference between speaker embeddings instead of remembering the speaker information. Therefore, we can reduce the overfitting risk by applying this augmentation method. We only perform the data augmentation when training the similarity measurement model on the training set instead of finetuning to the development dataset as described in Sec. 3.4.1. The reason is that the model cannot adapt to some spe-

cific region of the development dataset if we randomly rotate the speaker embedding space when finetuning.

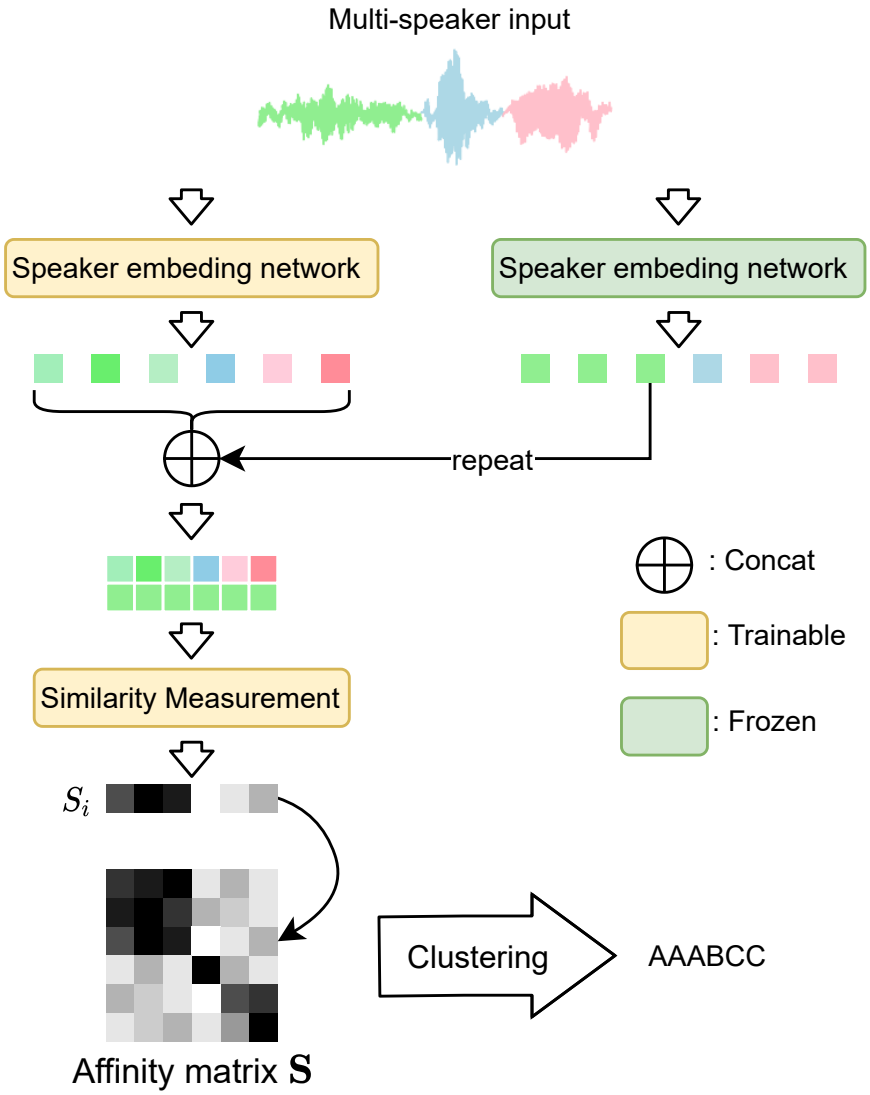


Figure 3.9: The framework of joint training

3.2.5 Joint training

Although embedding-level data augmentation can improve the generalization ability of the similarity measurement model and avoid overfitting, we still like to finetune the model to a specific domain by training with a lower learning rate on domain-

specific data. In our previous work [26, 27], we only finetuned the model for the similarity measurement but kept the speaker embedding network unchanged.

As Figure 3.9 shows, the entire network consists of two sub-models: the front-end speaker embedding network and the back-end similarity measurement model, where the speaker embedding network is the same as that in Figure 3.5, and the similarity measurement model is the same as that in Figure 3.6a.

The joint training framework is only employed in the finetuning stage. Therefore, we do not perform the data augmentation for the embeddings. During the finetuning stage, the networks in the yellow block in Figure 3.9 are jointly trained and updated, whereas the parameters of the green block in Figure 3.9 are frozen.

Unlike the conventional modular speaker diarization system, in which each module is optimized individually, the speaker embedding model in our framework is optimized with the objective of similarity measurement, which connects two modules and improves the system's performance. Under this joint training framework, the right speaker embedding network is a pre-trained segment-level speaker embedding extractor, and the left is a trainable segment-level speaker embedding extractor. If we employ the right network to extract an embedding of a known speaker, this model becomes a target-speaker voice activity detector, as Figure 3.10 shows.

3.3 Segment-Level TS-VAD

In the original TS-VAD [64], the model outputs a sequence of probabilities of the target speaker presence for each time frame, which is a frame-level method. Although the frame-level TS-VAD system has shown good performance on many different datasets [64], sometimes we do not need such a high resolution in time. The information about a single speaker can be aggregated at the segment level and thereby reduce the computational complexity. It is difficult for a back-end network to learn

from a long sequence, and frame-level prediction also contains a lot of noise.

3.3.1 Segment-level TS-VAD

In Eq. 3.3, $\mathbf{e}_i$ can also be considered as a segment-level target speaker embedding, where the speaker identity is unknown. Therefore, we need to perform clustering on the affinity matrix in the first stage to obtain the initial diarization results. In the second stage, we extract "target speaker embedding" $\mathbf{e}_t$ for each speaker identified in

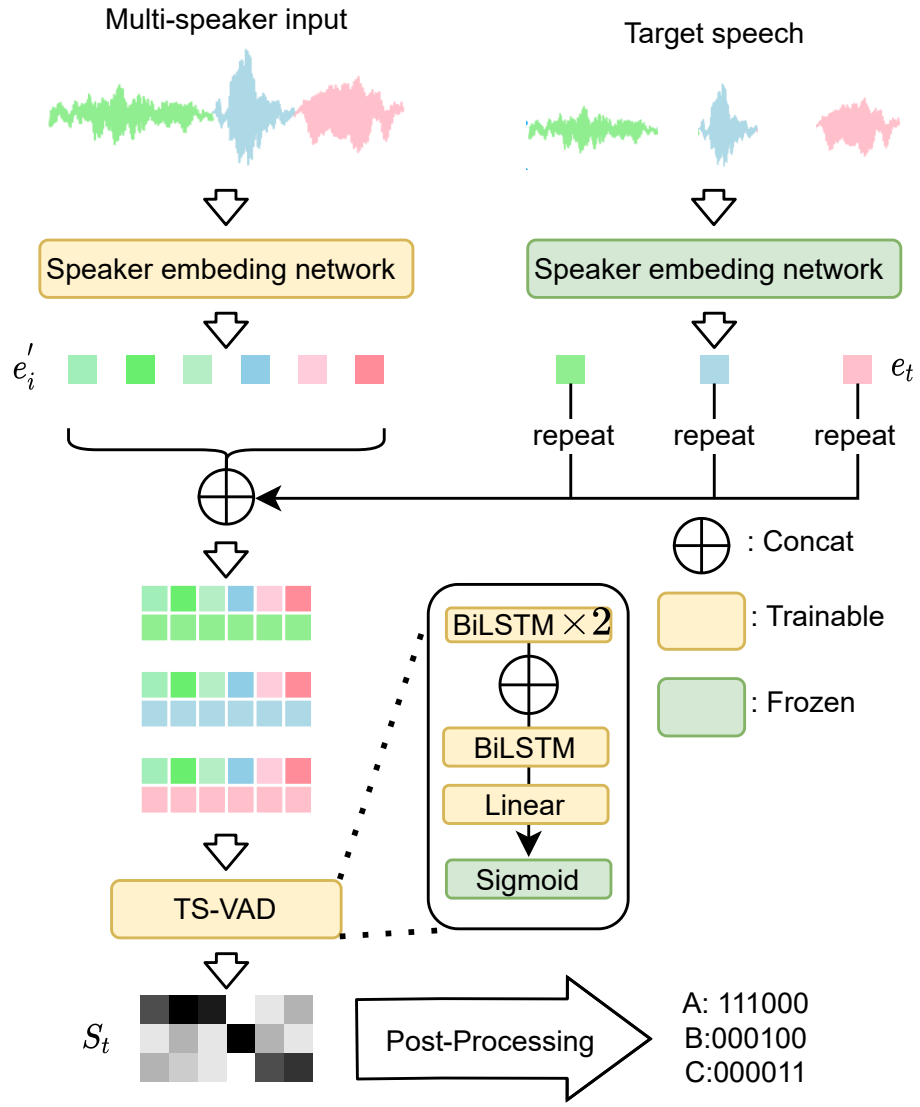


Figure 3.10: The framework of TS-VAD

the first stage. Similarly to Eq. 3.3, we evaluate the similarity between each speech segment and each target speaker, which can be interpreted as TS-VAD:

$$\mathbf{Q}_t = [\mathbf{Q}_{t,1}, \mathbf{Q}_{t,2}, ..., \mathbf{Q}_{t,T}] = f_l(\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_t\end{bmatrix}, \begin{bmatrix}\mathbf{e}_2\\ \mathbf{e}_t\end{bmatrix}, ..., \begin{bmatrix}\mathbf{e}_T\\ \mathbf{e}_t\end{bmatrix}), \tag{3.5}$$

where $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_T$ are the segment embeddings extracted from the pre-trained speaker embedding model. The output $\mathbf{Q}_t$ can represent the features that contain the information of the specific speaker t. Later, these decision states from different speakers are combined for the subsequent layers to further extract the binary decisions for all speakers.

Although Eq. 3.5 defines the general formulation of TS-VAD, we cannot directly perform TS-VAD with the segment embeddings from a pre-trained model $\mathbf{e}_1, ..., \mathbf{e}_T$ for several reasons:

1. The speaker embedding model is trained using single-speaker utterances (i.e., without any overlapping speech), and therefore, the TS-VAD model may not perform well if the target speaker is in an overlapping speech region.

2. While the pre-trained speaker embedding model can extract good quality embeddings from the whole utterances, it may not be able to perform well for short-duration segments.

To improve the multi-speaker information extraction, we keep the target speaker embedding $\mathbf{e}_t$ and replace the segment embedding $\mathbf{e}_i$ with $\mathbf{e}_i^{'}$, where $\mathbf{e}_i^{'}$ comes from the jointly optimized speaker embedding network, as Figure 3.10 shows. The networks for embedding extraction are the same as those of the joint training framework, and the only difference is the architecture of the similarity measurement model.

Figure 3.10 shows the architecture of our SP-based TS-VAD system, which is similar to the joint training framework in Section 3.2.5. First, a pre-trained ResNet

extracts the target speaker embedding $\mathbf{e}_t$, and another trainable ResNet extracts the segment embeddings by SP. For training, the target speaker embedding is extracted from all non-overlapping speech of the target speaker. Next, the target speaker embedding and the segment embeddings are concatenated as Eq. 3.5 shows. We follow the method introduced in [64], where N target speaker embeddings are extracted simultaneously and concatenated with the segment embeddings separately, as Figure 3.10 shows. Finally, two BiLSTM blocks followed by a fully connected layer predict the N speaker presence probabilities. The first block containing 2 BiLSTM layers separately processes the concatenated embeddings for each target speaker, and the second combines the output from the first BiLSTM block. The hidden size is 128 for all BiLSTM layers. The fully-connected layer with sigmoid function maps the output of the second BiLSTM layer to the presence probability for each of the N speakers. We obtain the final diarization results by thresholding on the outputs of the TS-VAD model, which is the post-processing shown in Figure 3.10.

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

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

tried extracting only one embedding for each speaker for inference, and the results were slightly worse.

3.3.2 Segment-level TS-VAD as overlap detection

Although TS-VAD can detect overlapping speech and reduce the missed speaker error, false alarms may increase. Therefore, DER improvement from the overlap detection is moderate. To tackle this problem, we can update the clustering-based results only in the overlapping speech regions. First, we extract the target speaker embedding for each speaker according to the clustering-based results. Next, we select two target speaker embeddings and use a TS-VAD model to find the speech regions for each of these two speakers. Finally, we only update the overlapping speech region between these two speakers. We iteratively select all combinations of two speakers and update the overlapping regions in the clustering-based results. The architecture of this TS-VAD model is the same as that shown in Figure 3.10, except that the output dimension is 2, in that it only predicts the binary decisions for two speakers. During the inference stage, we only select several of the most talkative speakers and discard other speakers for two reasons: (1) Speakers with a long speech time are more likely to have overlapping speech; (2) We reduce the time of inference, as we only run the inference $\binom{\#\text{selected spk}}{2}$ times for each recording.

Unlike previous overlap detection methods that first find the overlapping regions and then assign the closest two speakers to these regions [71], the TS-VAD-based overlap detection can get the overlapping regions between any two speakers, which is more accurate. In addition, we do not need to preset the number of speakers based on the maximum number of speakers in the dataset as the original TS-VAD does. Therefore, this TS-VAD overlap detector is more flexible. However, a limitation also exists. This method is not so efficient, as it needs to iteratively evaluate the TS-VAD many times depending on the number of speaker pairs. That is why we only select

several of the most talkative speakers for the inference.

3.4 Experimental Setup

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

3.4.1 Dataset and evaluation metric

Data Simulation

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

We generate the training data using the ground truth labels of the DIHARD development dataset. We first extract the label matrix $\mathbf{L} \in \{0, 1\}^{N \times T}$ for each recording in the DIHARD development dataset, where N is the number of speakers and T is the length of an utterance with the non-speech regions removed. Next, according to the labels of each speaker, we fill the active regions with a speech of a single speaker from the LibriSpeech or DIHARD development dataset. Finally, we sum all the speech segments to produce one simulated recording. This simulation strategy can generate a huge amount of data similar to the DIHARD development dataset, but the speakers are different. Figure 3.11 shows the process of data simulation.

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

from different domains are used for training the TS-VAD model.

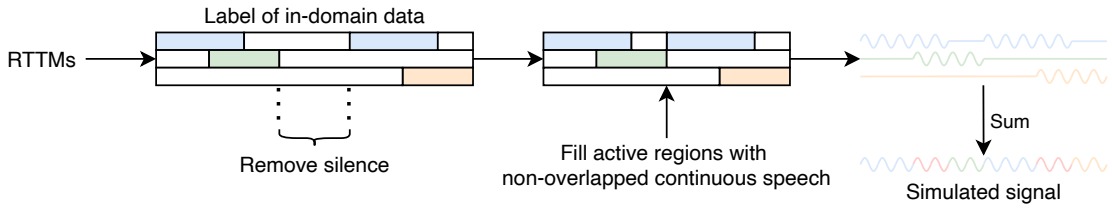


Figure 3.11: The data simulation process

Dataset for Training

- Voxceleb 1 & 2 [88]: For speaker embedding model pre-training.
- AMI [89], ICSI [90], ISL (LDC2004S05), NIST (LDC2004S09), SPINE1&2 (LDC2000S87, LDC2000S96, LDC2001S04, LDC2001S06, LDC2001S08): For similarity measurement pre-training in Section 3.2.2 and 3.2.5.
- LibriSpeech [91]: For 16kHz data simulation and TS-VAD pre-training.
- Switchboard [92] & NIST SRE 2004, 2005, 2006, 2008: For 8kHz telephone data simulation and TS-VAD pre-training.
- MUSAN & RIRs [93]: For waveform data augmentation.

Dataset for Finetuning / Evaluation

We employ the VoxConverse [94], DIHARD II [5], and DIHARD III dataset [95] for evaluation. Both the DIHARD II and DIHARD III datasets contain 11 domains , and VoxConverse is a single-domain dataset. Almost all domains contain 16kHz data, except the conversational telephone speech (CTS) data in DIHARD III, which is originally in 8 kHz. For each dataset, the corresponding development dataset is used as the finetuning dataset, and the details will be discussed in subsequent sections for each task.

Evaluation Metric

We use the diarization error rate (DER) and Jaccard error rate (JER) as the evaluation metric. For the DIHARD dataset, we follow the evaluation protocol in the DIHARD challenge, where no forgiveness collar is applied to the reference segments prior to scoring, and overlapping speech is evaluated. For DIHARD III, only a full set is considered for evaluation. For VoxConverse, we follow the evaluation protocol in the VoxSRC challenge, where a forgiveness collar of 0.25 is employed.

3.4.2 Speaker embedding extraction

The front-end extractor is ResNet34, where the widths (number of channels) of the residual blocks are {32, 64, 128, 256}. The temporal statistic pooling (TSP) layer computes the mean and standard deviation of the output feature maps and projects the variable length input to a fixed-length vector. Next, a fully connected layer predicts the 128-dimensional speaker embedding with a dropout of 0.5. Finally, we use the ArcFace [96] (s=32,m=0.2) as the objective for the training.

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

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

in Table 3.2.

3.4.3 Similarity measurement

Baseline

For the baseline, we extract the speaker embedding from the uniform segmented signal, where the window length is 1.28s and the window shift is 0.64s. The training input is an embedding sequence containing 64 continuous speaker embeddings. The overlapping segments are also involved in training, where the labels of these segments are the most talkative speakers. Therefore, for an overlapping segment, if a speaker talks more than other speakers, we label the segment with that speaker. If they talk for the same duration, we label the segment with a random speaker. For the same pair of speakers, we always label their segments with the same speaker.

The training process for the baseline contains two steps. First, we train the BiLSTM/self-attention-based network on the pre-training dataset introduced in Section 3.4.1 for 100 epochs, where the development set is used for validation. Next, we finetuned the model on the development set for 100 epochs. We use the Adam [97] optimizer with the binary cross-entropy (BCE) function and a learning rate of 0.001 for training and 0.0001 for finetuning. The average DER of the last ten epochs is reported as the final result.

In the inference stage, the embedding sequence is first broken into several subsequences with a length of 64 and a shift of 32. Next, we process each pair of subsequences and generate a 64×64 sub-matrix and merge all sub-matrices to form the complete affinity matrix, as mentioned in Section 3.2.3. Finally, spectral clustering

Table 3.2: The EER of the speaker embedding model on Voxceleb-1 original test set

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

is employed to get the final diarization results.

Data Augmentation

The data augmentation introduced in Section 3.2.4 is only employed in the training stage, where the dataset mentioned in Section 3.4.1 is the training set. During the finetuning stage, we do not employ this data augmentation. The probability of performing data augmentation for each data sample is set to 0.4.

Segmental Pooling

With the speaker embedding network parameters unchanged, we replace the TSP with the SP to extract the speaker embeddings from the uniformly segmented CNN feature map. The temporal resolution of the CNN feature map before the pooling is eight times less than for the input features, as shown in Table 3.1. Therefore, the frame rate of the feature map is 80 ms. At the CNN feature map level, we set the segment length to 16 and the segment shift to 8 so that the window length is 1.28s and the window shift is 0.64s, which is the same as the configuration of the baseline system. The training and the inference processes are the same as those of the baseline, except that the embedding with SP can see more context than the baseline system does. "More context" means that the segments with SP have seen the information in 3.72s as compared to 1.28s for the baseline. That is why the system with SP can show better performance than the baseline system.

Joint Training

The purpose of the joint training is not only to adapt the back-end classifier to the dev set but also to transfer the front-end extractor to the dev set. Therefore, we only perform joint training on the development set of DIHARD II, DIHARD III and VoxConverse datasets. The front-end extractor is the pre-trained speaker embedding

model mentioned in Section 3.4.2, with the pooling layer replaced by SP. The back-end classifier is the pre-trained model mentioned in Section 3.4.3. Then, we jointly finetune these models using three steps:

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

Table 3.3: DER and JER (%, ±STD) of different similarity measurement models on DIHARD II dataset. Embd-Aug is the data augmentation performed on the speaker embedding, SP is segmental pooling, and JT denotes joint training.

Model	**DIHARD II**					
	Dev		**Eval**		**Eval (+dev adapt)**	
	DER	**JER**	**DER**	**JER**	**DER**	**JER**
BiLSTM	24.15±0.41	47.89±0.15	25.59±0.34	49.43±0.16	19.92±0.01	46.70±0.01
+ embd aug	17.63±0.09	43.72±0.32	18.26±0.18	43.36±0.30	18.12±0.01	43.43±0.03
+ SP	17.54±0.13	42.60±0.76	17.81±0.17	42.17±0.30	17.80±0.00	**42.77±0.01**
+ JT	-	-	-	-	**17.76±0.03**	42.83±0.01
+ embd aug (3.72s)	-	-	-	-	18.99±0.03	44.99±0.05
Self-att	20.97±0.67	46.69±0.23	22.48±0.59	47.47±0.20	19.99±0.04	46.13±0.08
+ embd aug	18.17±0.10	44.03±0.57	18.71±0.23	43.89±0.39	18.42±0.01	43.12±0.01
+ SP	18.28±0.26	43.29±0.46	18.76±0.22	43.28±0.40	**18.00±0.01**	**42.12±0.02**
+ embd aug (3.72s)	-	-	-	-	20.25±0.08	46.30±0.17
Official baseline [5]	-	-	-	-	25.99	59.51
winning system (Clustering) [71]	-	-	-	-	18.21	-

The learning rate of the back-end BiLSTM is set to 0.001 for each training step, and the learning rate of the front-end extractor is set to 0.00001 in step 2. The optimizer is Adam, and the loss function is BCE loss. In step 3, the training process is the same as that in Section 3.4.3, where we extract the speaker embedding with the adapted front-end ResNet and finetune the back-end network for 100 epochs.

Table 3.4: DER and JER (%, ±STD) of different similarity measurement models on DIHARD III dataset. Embd-Aug is the data augmentation performed on the speaker embedding, SP is segmental pooling, and JT denotes joint training.

Model	**DIHARD III**					
	Dev		**Eval**		**Eval (+dev adapt)**	
	DER	**JER**	**DER**	**JER**	**DER**	**JER**
BiLSTM	21.03±0.26	40.92±0.14	20.11±0.26	39.62±0.17	17.03±0.01	37.45±0.03
+ embd aug	16.30±0.08	36.17±0.22	15.85±0.13	34.63±0.32	15.62±0.01	34.45±0.01
+ SP	16.29±0.15	35.50±0.50	15.61±0.09	33.98±0.34	15.45±0.02	**33.91±0.03**
+ JT	-	-	-	-	**15.18±0.02**	34.04±0.03
+ embd aug (3.72s)	-	-	-	-	17.48±0.01	36.67±0.06
Self-att	19.99±0.52	40.35±0.20	19.20±0.40	38.26±0.14	16.84±0.07	36.81±0.06
+ embd aug	16.86±0.11	37.42±0.48	16.04±0.16	35.34±0.45	15.82±0.01	34.74±0.06
+ SP	16.69±0.19	36.45±0.35	16.00±0.18	34.57±0.27	**15.65±0.01**	**34.06±0.04**
+ embd aug (3.72s)	-	-	-	-	17.76±0.03	37.96±0.19
Official baseline [95]	-	-	-	-	19.25	42.45
winning system (Clustering) [98]	-	-	-	-	15.77	-

Table 3.5: DER and JER (%, ±STD) of different similarity measurement models on VoxConverse dataset. Embd-Aug is the data augmentation performed on the speaker embedding, SP is segmental pooling, and JT denotes joint training.

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

3.4.4 Target-speaker voice activity detection

Training process

In Section 3.2, we can only slightly change the speaker embedding model with a low learning rate, as a large learning rate will destroy the generalization ability of this well-trained network. However, for TS-VAD, the speaker embedding model needs to be able to learn about overlapping speech. Therefore, we train the model in four stages. The optimizer is Adam, and the loss function is BCE loss in all stages; the

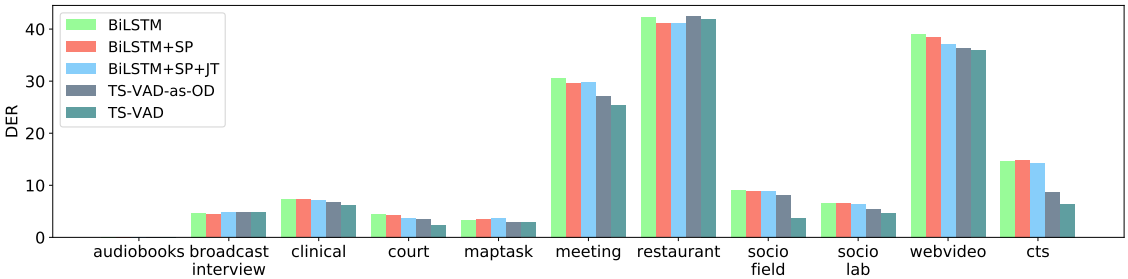


Figure 3.12: DER (%) on different domains in the DIHARD III dataset. SP and JT denote segmental pooling and joint training, respectively. The random rotation augmentation on the speaker embedding is performed.

only difference is the training dataset and learning rate.

- Stage 1: First, we copy the parameters of the pre-trained speaker embedding model to the front-end extractor and keep the front-end extractor frozen. Next, we train the TS-VAD model for ten epochs with a learning rate of 0.0001. For non-CTS data, we use the simulated Librispeech dataset. For CTS data, we use the simulated telephone speech data.

- Stage 2: The configuration is the same as stage 1, except the front-end extractor is unfrozen.

- Stage 3: We continue to train the model for ten epochs, with a learning rate of 0.0001 on the simulated DIHARD dataset.

- Stage 4: We train the model with a learning rate of 0.00001 on the real development set for 100 epochs. We take two recordings from each domain for validation.

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

the clustering-based results is less than N, we use zero-vectors as the invalid speaker embeddings.

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

Ablation Study of the Segmental Pooling for TS-VAD

To determine how segmental pooling can influence the TS-VAD performance, we explore several different pooling sizes. We test different pooling sizes at the CNN feature map level from 1 frame to 8 frames (80 ms $\sim$ 640ms) to determine which pooling size produces the best performance. A small pooling size may not contain enough information, while a large pooling size may include multiple speakers in a single embedding and result in performance degradation.

3.5 Experimental Results and Discussion

3.5.1 Similarity measurement

Table 3.3, 3.4, 3.5 shows the DERs of the similarity measurement models on the DIHARD II, DIHARD III, and VoxConverse datasets, respectively. For the evaluation set, both the results before and after adaptation are reported. The columns of Dev and Eval show the results before finetuning, and the columns of Eval (+dev adapt) show the results after finetuning on the development set. The first row is the result of the baseline without embedding rotation augmentation (Embd-Aug) and SP. The DER is calculated from the average results of the models from the last ten epochs to reduce the variance. Results of the BiLSTM model show that the joint training (JT) with segmental pooling achieves the lowest DER, 17.76 % on DIHARD II and 15.18 % on DIHARD III. Generally, embedding augmentation can

always improve the performance for both BiLSTM and self-attention-based models, and SP can further improve the performance. Compared with BiLSTM, the self-attention-based network shows better performance if we directly train the model without augmentation, which is similar to the findings in our previous work [27]. However, when the augmentation is employed, BiLSTM shows better performance than the self-attention-based network. In addition, we also show the adapted DERs of the system with embedding augmentation for the 3.72-second segment with a shift of 0.64 second in Table 3.3 and 3.4, as the embedding with SP can see more context (3.72s) than its length (1.28s) through the ResNet34 receptive field. When using a long segment with embedding augmentation, it shows degraded performance as the resolution of the long segments is lower than that of the short segments.

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

Compared with the official baseline and the previous winning systems, our method shows better performance. As we do not perform any post-processing and the output is directly generated by spectral clustering, we only compare our results with the clustering-based results in the DIHARD III challenge. Both winning systems employ AHC-based clustering methods, and VB or VBx resegmentation is performed [71,

Table 3.6: DER (%) of the segment-level TS-VAD models on evaluation dataset (N=8, fully assigned)

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

77, 59]. We do not show the winning system of the VoxConverse dataset, as we use the oracle VAD in our experiments, but the winning system used the system VAD in the challenge.

3.5.2 Target-speaker voice activity detection

Table 3.6 shows the DERs of our TS-VAD system on the DIHARD II, DIHARD III, and VoxConverse datasets, and Table 3.7 shows the DERs of TS-VAD as overlap detection. For DIHARD II and VoxConverse, as all recordings are 16kHz, we only train one TS-VAD model with the speaker embedding model trained on the 16kHz Voxceleb dataset. For DIHARD III, we use the same speaker embedding model and the TS-VAD model for 16kHz data. But for 8kHz CTS data, we use another speaker embedding model and TS-VAD model trained on the 8kHz data for inference, and the results are finally merged together.

Table 3.7: DER (%) of the segment-level TS-VAD as overlap detection on the evaluation dataset (N=2, partially assigned overlapping region)

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

During the inference stage, we directly use the clustering-based results from the joint training BiLSTM model as the initialization of the TS-VAD system. As some speakers talk for a long time, and the speech signal is too long for the model to extract the target speaker embedding, the regions that contain a single speaker are first selected and uniformly broken into 16-second segments. Then, we extract the speaker embedding for all segments and take the mean of all embeddings as the target speaker embedding. The outputs of the TS-VAD model are the segment-level probabilities for each speaker. Next, we apply a threshold of 0.5 on the probabilities to get the binary decision. Since we use the oracle VAD, if the probability of a speech frame is lower than the threshold for all speakers, we will assign the speaker with the largest probability to this frame.

Results in Table 3.6 show that our domain-independent TS-VAD method achieves

a DER of 16.48% on DIHARD II and 11.62% on DIHARD III when we directly use the output from the original TS-VAD model as the results. For the DIHARD datasets, both results are better than the clustering-based results. However, for the VoxConverse dataset, the result becomes worse. The reason is that the Voxconverse contains fewer overlapping speech regions than the DIHARD dataset, and the clustering-based method already has a good performance on the VoxConverse dataset. As Table 3.6 shows, the reduction in MISS cannot compensate for the increase in FA and SpkErr.

Table 3.7 shows the results of two-speaker TS-VAD as overlap detection. As we only assign the detected overlapping speech region to the clustering-based results, it shows better performance than the original TS-VAD method on the VoxConverse dataset. However, it becomes worse on the DIHARD dataset.

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

We also show DERs on different domains of the DIHARD III dataset in Figure 3.12. It can be found that the significant improvement in DIHARD III mainly comes

from the CTS dataset, as it only contains two speakers, and almost half of the errors are overlapping speech. We finally reduced the DER on CTS data in the DIHARD III eval set from 14.67% to 6.47%. As the DIHARD challenge is a multi-domain task, our domain-independent method cannot be optimized on each domain without using in-domain data, especially on the extremely noisy and overlapping domain, e.g., restaurant, as shown in Figure 3.12. Note that we train our model on the multi-domain data, and we do not use any evaluation dataset for finetuning. Still, we achieve better performance than the previous domain-dependent i-vector-based TS-VAD model, in which the evaluation data is used for finetuning [77].

3.6 Conclusion

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

In the future, we will explore different pooling strategies, including weighted pooling, attentive pooling, and learnable dictionary encoding. In addition, we will extend this framework to more tasks like end-to-end speaker diarization.

Chapter 4

Target-speaker Voice Activity Detection: From Offline to Online

4.1 Introduction

Currently, most of the modularized and end-to-end speaker diarization systems show promising performance on many datasets, but they are both designed to process pre-recorded audio and operate offline, meaning that they can analyze the entire audio file at once to identify the individual speakers. However, there are several practical applications, such as meeting transcription systems or smart agents, where the demand for low latency in speaker assignment is paramount [2]. Despite ongoing efforts to develop online speaker diarization systems, both within the realm of clustering-based approaches [12, 13] and neural network-based diarization frameworks [14, 15, 16], this remains a challenge that has yet to be fully overcome.

In an online modularized system, each module needs to operate in an online manner, and the key point is how to replace the offline clustering algorithm with an online one to process the speaker representations in chronological order [41, 13]. Under this consideration, most of the online modularized systems only focus on the clustering algorithm and ignore other modules, e.g., overlap speech detection.

For the online EEND framework, the main challenge is the inconsistency of the

speaker order across different timestamps, as the order of the speaker can be one of the permutations of all existing speakers. During the training stage, this inconsistency can be solved using a permutation-free objective [65, 57], but it limits the EEND to be extended for online purposes as the order of speakers are always changing as a new signal comes. One solution is to employ a buffer to store both prior inputs and results to align the current results with the previous one [15, 99, 100], but this usually requires a large buffer for a satisfying performance.

As a post-processing method for offline modularized diarization systems, target-speaker voice activity detection (TS-VAD) is usually employed to refine the diarization results from the clustering algorithm [64]. Given the target speaker representations and audio signals, TS-VAD can recognize the overlapping speech in a block-wise manner. This inherent property implies that TS-VAD can naturally be adapted to an online module. However, this characteristic is usually ignored as TS-VAD serves as a sub-module for the modularized diarization system. If we can obtain the target speaker representations in an online manner, TS-VAD can be easily executed in real time. Furthermore, if the target speaker representations are generated by TS-VAD itself, it can be considered a fully end-to-end online module.

In this chapter, we propose a fully end-to-end online framework for speaker diarization tasks by adapting the conventional TS-VAD for real-time operation, termed online TS-VAD (OTS-VAD). Similar to offline TS-VAD presented in Chapter 3, the OTS-VAD can not only retrieve the speaker activities for each speaker but also detect overlapping speech. Unlike the EEND framework, our proposed system determines speaker activities using target speaker embeddings. These embeddings are self-generated by the OTS-VAD model and are continually updated with incoming audio signals. As the order of these target speaker embeddings remains constant, inconsistencies during the inference phase are eliminated. Therefore, our approach can employ a small block size during the inference stage, ensuring reduced memory

consumption.

The proposed framework makes the following contributions:

- We propose a innovative framework for speaker diarization that can process data in both online and end-to-end manner, achieving state-of-the-art (SOTA) performance through refined training and inference methodologies.
- We systematically review the mechanism behind TS-VAD, so that the proposed online TS-VAD has the similar performance compared with its offline version, e.g., a capacity for detecting overlapping speech.
- We extend the model to accommodate multi-channel data, ensuring improved and consistent performance.

4.2 Related Works

4.2.1 Modularized speaker diarization system

Offline method

The modularized speaker diarization system, often referred to as the traditional speaker diarization system, is composed of several sub-modules. The process begins with voice activity detection (VAD) that filters out non-speech components like background noise from the speech signal [101, 31, 32]. Following this, the audio stream is segmented into speaker-consistent sections [42, 102]. From these segments, speaker embeddings are derived [36, 103]. Leveraging a suitable similarity measurement technique, clustering algorithms such as agglomerative hierarchical clustering (AHC) [104] or spectral clustering [53] are utilized to group these embeddings. Sometimes the pre- and post-processing modules are also included for better performance, such as speech enhancement [105], speech separation [70], overlapping speech detection [106] and resegmentation [107], etc. In addition, two or more of the modules can

be combined by using a single neural network, e.g., the region proposal network is employed to perform VAD, segmentation and embedding extraction at a time [108].

Online method

While VAD, segmentation, and speaker embedding can be executed in real time, the crux of transitioning the modularized system to an online format lies in substituting the offline clustering technique with an online version. Notable approaches include the adapted i-vector using a transformation matrix [41] and modified clustering methods [12]. However, these approaches can be inefficient for extended audio clips due to their time complexities scaling linearly with the number of speaker segments. Other online clustering methods, like incremental clustering [109] and online spectral clustering [110], were also proposed for online diarization tasks with low latency. In addition, several supervised methods such as UIS-RNN [14] and UIS-RNN-SML [111] were proposed, where the segmentation and clustering methods are replaced with a trainable model.

4.2.2 End-to-end speaker diarization system

Offline method

End-to-end neural diarization (EEND) is an innovative framework that performs the complete speaker diarization process using a single deep neural network [65, 66]. Given a speech signal, it outputs the diarization results directly, as well as recognizing overlapping speech. To enhance its versatility and tackle scenarios with an undefined number of speakers, an improved version known as EEND with Encoder-Decoder-based Attractor (EDA) was introduced. This advancement not only bolstered performance but also widened the framework's utility. Further refinements include integrating front-end components like Conformer [112] and introducing EEND with Global and Local Attractors (GLA) [113], both of which have enhanced the system's

efficacy on real datasets. In addition to the fully end-to-end framework, EEND can also be employed in modularized speaker diarization systems, such as embedding extraction [114, 115, 116], overlap-aware resegmentation [117] and post-processing [118].

Online method

The EEND framework has also been widely explored in online speaker diarization tasks. Han et al. [16] proposed a block-wise EEND with EDA (BW-EDA-EEND) which uses hidden states in previous blocks to generate attractors in a sequential manner. Another approach to tackle the issue of speaker permutation ambiguity involves maintaining acoustic features within a speaker-tracing buffer [15, 99]. On top of this approach, Horiguchi et al. [100] enhanced EEND-GLA for online usage by integrating a speaker-tracing buffer. Remarkably, this method achieved SOTA performance across various datasets, outperforming numerous offline modularized speaker diarization systems.

4.2.3 Target speaker voice activity detection

Target speaker tracking is employed in many speech-related tasks to retrieve the information of a specific speaker, including target speaker automatic speech recognition (TS-ASR) [73], target speaker speech separation [119], and TS-VAD [64]. The similarity of these tasks is that they all use a speaker profile to focus on the speech of interest, which consequently refines results corresponding to that particular speaker. Impressively, this method remains effective even when the audio encompasses overlapping speech. Typically, the speaker profile constitutes a pre-enrolled speaker embedding that contains the speaker identity information like i-vector [36] and x-vector [103].

For speaker diarization, TS-VAD has been introduced to refine the results de-

rived from clustering-based diarization systems. It is particularly useful in situations where there are many overlapping speeches in a recording, and the focus is on detecting the presence or absence of a specific speaker. The background of TS-VAD can be traced back to personal VAD [76], which adapts the VAD process to the unique vocal characteristics of an individual speaker. However, the personal VAD method usually does not have good performance in the diarization task as it only considers a single speaker and ignores the relationship between speakers in the speech signal. TS-VAD, in contrast, is designed to simultaneously recognize the speech activities of multiple speakers present at any given segment. Crucially, it incorporates the constraints and relationships among different speakers, ensuring a more comprehensive understanding. As a result, TS-VAD offers an enhanced performance and delivers more reliable diarization outcomes.

The early version of TS-VAD employs a pre-enrolled i-vector as acoustic footprint [64]. It takes a short speech segment and several i-vectors as input and evaluates if one of the frames belongs to one or more speakers, as shown in Figure 4.1. Figure 4.1 shows an i-vector-based TS-VAD model. First, a CNN takes the acoustic features as input and produces a temporal feature map. Next, each i-vector is concatenated with each frame of the feature map, and they are separately fed to the 2-layer BiLSTM, producing a speaker detection (SD) feature for each i-vector. These SD features are concatenated again as the input of the final BiLSTM layer and converted to the posterior presence probabilities of each speaker. Remarkably, the i-vector-based TS-VAD approach demonstrates a notable advantage in terms of diarization error rate (DER) when pitted against traditional clustering-based methods, which brings forth a robust solution for the diarization challenge.

The deployment of i-vectors in TS-VAD, while successful to some extent, showed limitations when applied to multi-scenario data [98]. These findings pave the way for the exploration of x-vectors as an alternative, but a simple swap of i-vectors

for x-vectors did not yield an immediate boost in performance [64]. The speculated cause of this unexpected behavior is that the shallow CNN employed in the front-end of the TS-VAD system might not be adept at efficiently processing the deep learning-driven x-vector. To address this conjecture, we replaced the front-end of the TS-VAD with a pre-trained front-end module tailored for x-vectors, as mentioned in Chapter 3. This modified architecture demonstrated a more competent handling of x-vectors, consequently leading to superior performance when benchmarked against the i-vector-based system. Additionally, the intrinsic robustness and generalization capability of x-vectors empowered this model to perform commendably on challenging datasets like DIHARD III, which represents diverse real-world scenarios.

Usually, TS-VAD can only handle a fixed number of speakers, as the final layer is a linear projection in general. A common strategy is to define the output size based on the maximum anticipated number of speakers. This presents a challenge because, in real-world scenarios, the actual number of speakers is indeterminate. Furthermore, when the input audio features fewer speakers than the maximum set, the TS-VAD model performs redundant computations, especially when certain speaker embeddings are initialized to zero or arbitrary vectors. To address this inefficiency,

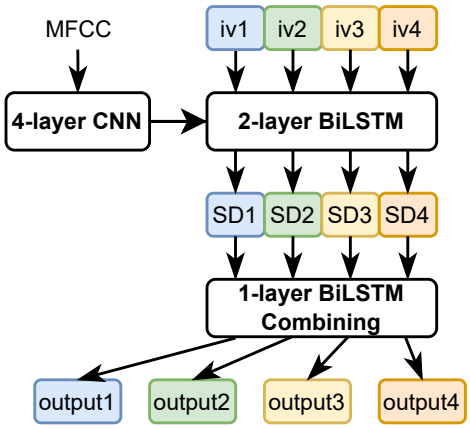


Figure 4.1: The architecture of the i-vector-based TS-VAD [64]

.

the terminal linear layer can be substituted with a recurrent neural network capable of managing variable-length data. For instance, both LSTM [120] and Transformer [121] have been implemented along the speaker dimension rather than the temporal one. Recently, Cheng et al. [1] proposed a sequence-to-sequence (Seq2Seq) architecture for TS-VAD, termed as Seq2Seq TS-VAD. In this approach, frame-level representations and speaker embeddings are input independently into the encoder and decoder, obviating the need for concatenation. Notably, the decoder creates an embedding of consistent dimensionality for each speaker's voice activity, independent of the duration of the input features. Additionally, the final linear layer offers enhanced flexibility, facilitating voice activity predictions at finer temporal resolutions with modest computational demands.

Compared to EEND, TS-VAD serves as a post-processing module within the modularized speaker diarization system. While EEND offers an all-encompassing solution, learning speaker embeddings directly from audio waveforms to produce diarization outcomes, TS-VAD refines the outputs of clustering-based methods. Owing to its reliance on prior knowledge derived from clustering-based diarization results, TS-VAD typically exhibits superior performance to EEND. In the subsequent section, we introduce our TS-VAD-based end-to-end online framework, wherein the speaker embedding is self-generated by the TS-VAD model.

4.3 Online Target Speaker Voice Activity Detection

This section elucidates the workings of the x-vector-based TS-VAD and details its transformation from an offline to an online approach, encompassing both the training and inference procedures.

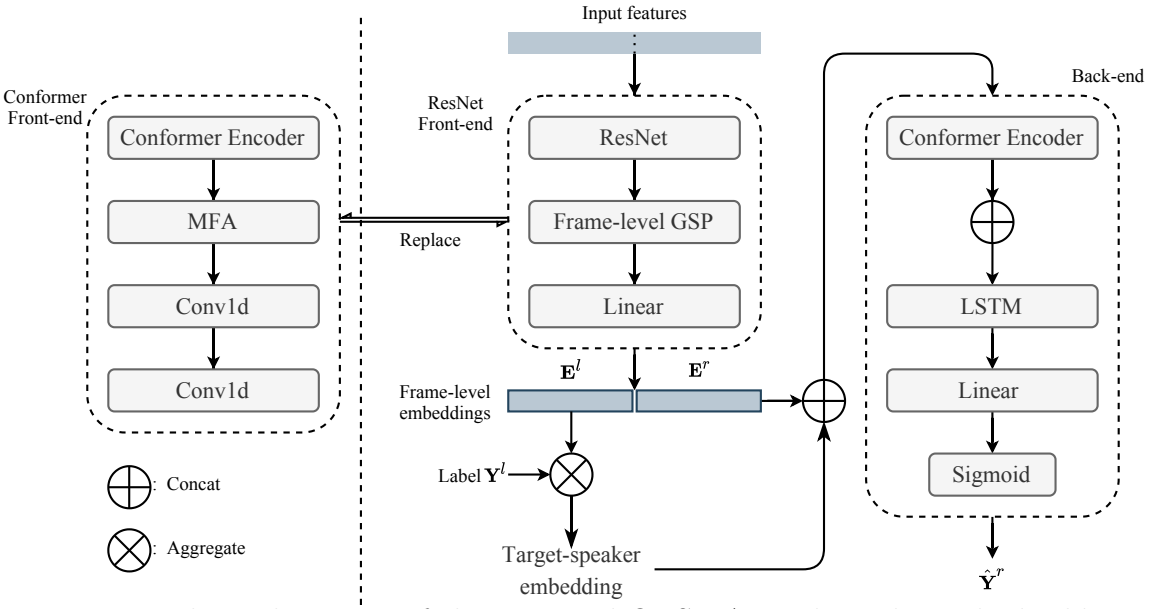


Figure 4.2: The architecture of the proposed OTS-VAD. The right is the backbone of the ResNet-based front-end, and the left is another Conformer-based front-end that can replace the ResNet-based front-end.

4.3.1 X-vector-based TS-VAD

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

$$\mathbf{S}'_n = [\mathbf{S}'_{n,1}, \mathbf{S}'_{n,2}, ..., \mathbf{S}'_{n,T}] = f_d(\begin{bmatrix}\mathbf{e}_n \\ \hat{\mathbf{e}}_1\end{bmatrix}, \begin{bmatrix}\mathbf{e}_n \\ \hat{\mathbf{e}}_2\end{bmatrix}, ..., \begin{bmatrix}\mathbf{e}_n \\ \hat{\mathbf{e}}_T\end{bmatrix}), \tag{4.1}$$

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

$$\mathbf{S} = f_j(\begin{bmatrix} \mathbf{S}'_{1,1} & ... & \mathbf{S}'_{1,T} \\ ... & \mathbf{S}'_{n,t} & ... \\ \mathbf{S}'_{N,1} & ... & \mathbf{S}'_{N,T} \end{bmatrix}), \tag{4.2}$$

where f_j acts as the joint detection function that produces final speech activities. As depicted in Figure 4.1, f_d operates as the 2-layer BiLSTM, while f_j functions as the 1-layer BiLSTM. The front-end model designed for frame-level embedding extraction is constituted by a 4-layer CNN, and this is the typical i-vector TS-VAD framework.

However, this framework does not work well with the x-vector, even though the performance of the x-vector is better than the i-vector [64]. As mentioned in Chapter 3, we can replace the front-end module with a pre-trained network that is used for target speaker embedding extraction, which makes the frame-level embedding learn faster to capture the identity information in target speaker embedding. This x-vector-based TS-VAD method achieved state-of-the-art performance on the DIHARD III dataset and was further improved by the x-vector-based sequence-to-sequence TS-VAD framework [1].

4.3.2 Online TS-VAD

Architecture

Figure 4.2 shows the architecture of the ResNet-based online TS-VAD (OTS-VAD). The model contains two sub-modules: the front-end for embedding extraction and the back-end for target speaker detection. The front-end consists of a ResNet and a fully-connected layer following a frame-level global statistics pooling (GSP) that summarizes the statistics of each frame of the ResNet output. The back-end contains a conformer-based encoder, a BiLSTM layer and a linear layer.

Given an acoustic feature $\mathbf{X} \in \mathbb{R}^{H \times L}$, the ResNet transforms it into a temporal feature map $\mathbf{M} \in \mathbb{R}^{C \times \frac{H}{8} \times \frac{L}{8}}$. Here, C represents the number of channels, H is the feature's dimensionality, and L indicates the total number of frames. It is worth

noting that the temporal resolution of this feature map is reduced by a factor of eight compared to the original input, attributed to ResNet's capability of down-sampling by this magnitude. Subsequently, the frame-level GSP layer computes the mean and standard deviation over the feature dimension, thus generating a vector of dimension $2C$ for each frame. The succeeding fully-connected layer ingests the GSP's output, $\mathbf{P} \in \mathbb{R}^{T \times 2C}$, producing frame-level embeddings, represented as $\hat{\mathbf{E}} = [\hat{\mathbf{e}}_1, ..., \hat{\mathbf{e}}_T] \in \mathbb{R}^{T \times D}$. Here, $T = \frac{L}{8}$ and D is the dimensionality of frame-level embeddings. Furthermore, D matches the dimension of the target speaker embeddings. The usage of frame-level GSP is to imitate the utterance-level GSP in the speaker recognition task [122]. This method interprets each window as a distinct utterance for which embeddings are extracted. By adopting this approach, the front-end module can effectively discern the speaker's identity for each frame, which invariably leads to enhanced performance.

Upon obtaining the frame-level embeddings $\hat{\mathbf{E}}$ and the target speaker embeddings, represented by $\mathbf{E} = [\mathbf{e}_1, ..., \mathbf{e}_N] \in \mathbb{R}^{N \times D}$, the back-end module predicts speech activities in a manner akin to the offline TS-VAD model. To be more specific, each target speaker embedding undergoes replication and is subsequently concatenated with the frame-level embeddings. The initial Encoder separately manages the concatenated embeddings corresponding to each target speaker, yielding a primary decision level. It's noteworthy that these decisions may not strictly indicate probabilities. Following this, decisions pertaining to all speakers are concatenated and fed to the BiLSTM for refinement. This stage facilitates the exchange of inter-speaker information. Finally, a sigmoid-activated fully-connected layer estimates the probabilities of speaker existence.

To make the offline TS-VAD operate in an online manner, the key point is to obtain the target speaker embedding without an additional speaker embedding extractor in both the training and inferring process. In addition, maintaining the

consistency of the target speaker embedding is also important for the back-end module to detect the speech activities. In the following sections, we will describe how to obtain the consistent target speaker embedding in the training and inferring process.

Training process

As shown in Figure 4.2, given an input $\mathbf{X}$ with a length of $2L$, the initial step involves partitioning the input features uniformly into left and right segments $\mathbf{X}^l$ and $\mathbf{X}^r$, each having a length of L. After extracting the frame-level embeddings $\hat{\mathbf{E}}^l, \hat{\mathbf{E}}^r \in \mathbb{R}^{T\times D}$ for the left and right segments, the target speaker embedding for the n-th speaker can be obtained by:

$$\mathbf{e}_n = \frac{\bar{\mathbf{y}}_n^l \hat{\mathbf{E}}^l}{\sum_{i=1}^{T} \bar{\mathbf{y}}_{n,i}^l}, \tag{4.3}$$

where $n \in \{1, 2, ...N\}$, i is the frame index, $T = \frac{L}{8}$ by downsampling from ResNet, $\bar{\mathbf{y}}_n^l \in \{0, 1\}^T$ is the corresponding label of n-th speaker in the left segment with the overlapping regions set to zero. Eq. 4.3 describes that target-speaker embedding can be obtained by averaging all non-overlapping frame-level embeddings for each respective speaker, as shown in Figure 4.3. In addition, due to the limit of the GPU memory, we cannot train the model with long-duration inputs. However, this will lead to a problem that $\mathbf{X}^l$ only contains limited speakers with a short duration, and the back-end module only accepts insufficient target speaker embeddings during the training stage. Therefore, the model cannot deal with the input with lots of speakers. To cope with this, we replace the $\mathbf{X}^l$ with a randomly simulated signal that contains multiple speakers, and the probability of replacement is set to 0.5.

After obtaining the target speaker embedding, the training process of the back-end is the same as the offline TS-VAD, where we repeat and concatenate the target speaker embeddings to the frame-level embeddings as the input of the back-end.

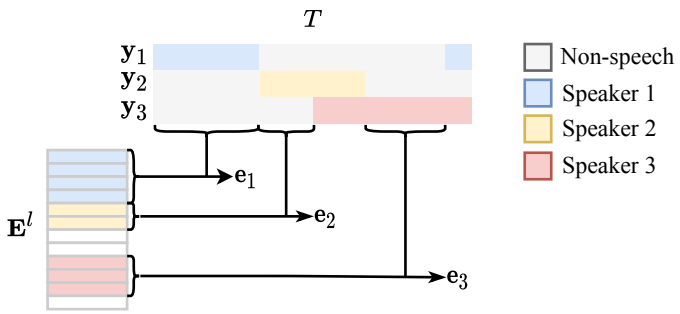


Figure 4.3: Illustration of the target speaker embedding extraction.

Since we already know the order of the speakers in the ground truth label $\mathbf{Y} \in \{0,1\}^{T\times N}$, we will keep the order of all speakers in the target speaker embeddings. Therefore, the speaker order in the output $\hat{\mathbf{Y}} \in (0,1)^{T\times N}$ is the same as the labels, and we do not need to solve the permutation ambiguity as presented in the EEND framework. Then, we optimized the whole model by the binary cross entropy (BCE) loss.

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

Inference process

The inference process is quite different from the training process as the target speaker embeddings are accumulated during the inference process. Specifically, during inference, target speaker embeddings are aggregated incrementally as the OTS-VAD system systematically processes each successive block of audio. Here, we also assume that the silence regions are removed in advance. For clarity, we assume that the features and output have the same time resolution, and we will explain the actual time resolution for all features and output in Section 4.4.

As shown in Figure 4.4, the length and shift for each coming block are l and m,

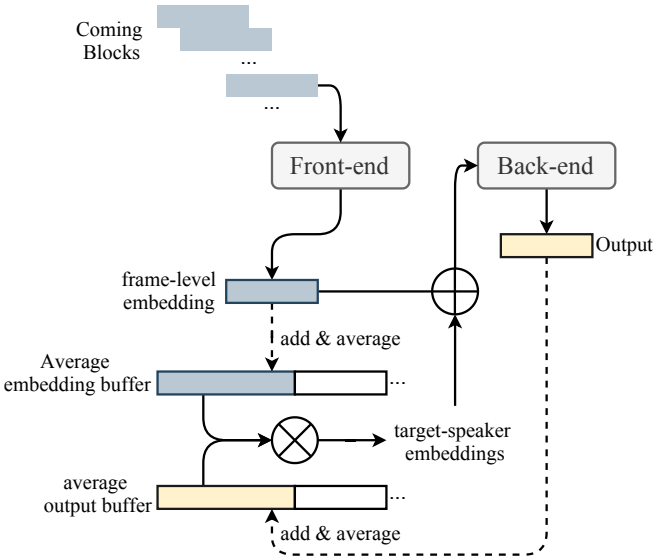


Figure 4.4: Inference process of OTS-VAD with embedding buffer and output buffer. Dash line means updating the buffer after obtaining the output from the current block.

respectively. In addition, we need a frame-level embedding buffer $\hat{\mathbf{E}} \in \mathbb{R}^{T \times D}$and an output buffer $\hat{\mathbf{Y}} \in (0,1)^{T \times N}$ to store the frame-level embedding and corresponding output over time, where T is the total length of output, D is the embedding size and N is the preset number of speakers. These two buffers are both initialized with zeros. Initially, the block size begins at m and expands until there is a sufficient signal to compose a full block of size l. Following this, the block size remains constant at l, advancing with a consistent shift of m. Notably, m also designates the latency of the OTS-VAD system.

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

$$\hat{\mathbf{y}}^1_{n,t} = \begin{cases} 1 & , \quad n = 1 \\ 0 & , \quad n = 2, 3, ..., N \end{cases} \tag{4.4}$$

where $\hat{\mathbf{y}}^1$ is the output of the first block, n and t are the index of speaker and frame, respectively. Finally, both the frame-level embeddings and the output are added to both buffers at the corresponding position and averaged at the overlapping region between the blocks, as shown in Figure 4.4.

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

$$\mathbf{e}_n = \frac{\bar{\mathbf{Y}}_n \hat{\mathbf{E}}}{\sum_{t=1}^{t^{'}} \bar{\mathbf{Y}}_{n,t}}, \tag{4.5}$$

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

An important thing for an online application is the latency. If there is an unlimited number of blocks, it is impossible to store all frame-level embeddings in the buffers, and the computation cost also becomes unacceptable for calculating the

target-speaker embedding. In that case, we can only store those frames-level embeddings with the higher probabilities for each speaker in the buffer and discard others, which significantly reduces the computation cost when the audio signal is extremely long.

Another solution for reducing the latency is to accumulate the frame-level embedding for each speaker and record the sum of all frame-level embeddings, referred to as accumulated total embedding $\tilde{\mathbf{e}}_n$, as shown in Figure 4.5. At the same time, the number of frames F is also recorded. Then the target speaker embedding $\mathbf{e}_n$ can be directly computed by:

$$\mathbf{e}_n^{k+1} = \frac{\tilde{\mathbf{e}}_n^k}{F^k}, \tag{4.6}$$

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

$$\begin{aligned} \tilde{\mathbf{e}}_n^k &= \tilde{\mathbf{e}}_n^{k-1} + \bar{\mathbf{Y}}_n^k \hat{\mathbf{E}}^k, \\ F^k &= F^{k-1} + \sum_{i=0}^{T} \bar{\mathbf{Y}}_{n,t}^k. \end{aligned} \tag{4.7}$$

Similar to the buffer-based inference process mentioned in Figure 4.4, only the frame with probabilities greater than the upper threshold will be used for updating. If the probabilities in the last M frames are lower than the lower threshold, we notify a new speaker coming by setting his/her probabilities to the upper threshold.

The first strategy utilizes a buffer to continuously store average frame-level embeddings and corresponding outputs. This method, by retaining a comprehensive record of embeddings, ensures greater precision in target speaker inference, thereby

potentially enhancing system performance. However, this method's primary limitation emerges when processing extensive input features, resulting in increased computational time and memory requirements. On the other hand, the second approach streamlines the inference process by only calculating the accumulated embeddings and their corresponding frame counts. While this method offers significant computational efficiencies, especially in scenarios with prolonged input sequences, there is a potential slight degradation in performance accuracy. Consequently, the optimal choice between these strategies would depend on the specific application requirements, balancing between accuracy and computational resource constraints.

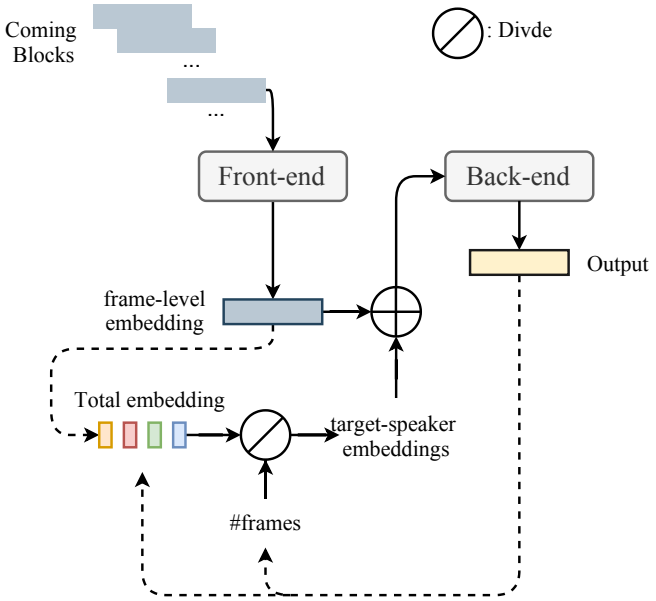


Figure 4.5: Inference process of OTS-VAD with accumulated embedding and the total number of frames. Dash line means updating the accumulated embedding and the number of frames after obtaining the output from the current block.

4.3.3 Incorporation of MFA-Conformer as front-end

Multi-scale feature aggregation (MFA) is a method that joins together output feature maps from every frame-level module within a speaker embedding network before pooling at the utterance level. Research indicates that when used with TDNN-based

networks, this method enhances performance, which suggests that features from the lower levels offer valuable speaker-related information [123].

To incorporate the Conformer encoder into speaker verification tasks, the MFA-Conformer introduces an MFA module within the Conformer encoder [50]. This module combines all the frame-level outputs from the Conformer blocks before moving them to the pooling layer. Once this combined frame-level feature map is obtained, attentive statistics pooling is used to generate an utterance-level representation [124]. Subsequently, the speaker's embedding is derived by employing batch normalization and a connected layer on this utterance-level representation. To replace the ResNet with the MFA Conformer in the OTS-VAD framework, the combined frame-level features before pooling are considered as the frame-level embeddings, which is the output of the MFA layer shown in Figure 4.2. However, the time resolution of the outputs from the MFA layer is too high to be adopted as the frame-level embedding as processing this feature will consume large GPU memory in the back-end module. We then add two more 1D convolutional layers to reduce the time resolution, each of which reduces the time resolution by 2. Finally, the output of the last 1d convolutional layer has the same time resolution as the ResNet-based front-end does.

4.3.4 Multi-channel extension

Cross-channel attention mechanism has achieved significant success in multi-channel speech signal processing domains, including speech enhancement [125], speech separation [126, 127], speech recognition [128], and speaker diarization [129]. Their efficacy lies in their ability to comprehend non-linear contextual relationships across channels both temporally and contextually.

Given C channels of Fbank sequence denoted by $\mathcal{X} = (\mathbf{X}^1, ..., \mathbf{X}^C)$, the target-speaker embedding from N target speakers represented as $\mathcal{E} = (\mathbf{E}^1, ..., \mathbf{E}^C)$, where $\mathbf{E}^c = [\mathbf{e}_1^c, \mathbf{e}_2^c, ..., \mathbf{e}_N^c]$ are the target speaker embeddings extracted from the c-th chan-

nel, the corresponding target speaker decision is denoted by $\mathbf{Y} = (\mathbf{y}_1, ..., \mathbf{y}_T)$. Here, $\mathbf{X}_i \in \mathbb{R}^{L \times H}$ has L frames, $\mathbf{E}^c \in \mathbb{R}^{N \times D}$ signifies the D-dimensional speaker embedding from N target speakers in the c-th channel, and $\mathbf{y}_t \in \{0, 1\}^N$ encapsulates the target speaker decision at time step t with a dimension of N. The front-end ResNet takes the C-channels of Fbank sequence as input, producing the frame-level speaker embedding sequence, denoted by $\hat{\mathcal{E}} = (\hat{\mathbf{E}}^1, ..., \hat{\mathbf{E}}^C)$, where $\hat{\mathbf{E}}^c = [\hat{\mathbf{e}}_1^c, \hat{\mathbf{e}}_2^c, ..., \hat{\mathbf{e}}_T^c]$. Subsequent operations entail repeating the target-speaker embedding T times and the frame-level speaker embedding N times. Post these operations, the two speaker embeddings are concatenated in the embedding dimension, forming the matrix $\mathcal{G} = (\mathbf{G}^1, ..., \mathbf{G}^C)$, where $\mathbf{G}^c \in \mathbb{R}^{T \times N \times 2D}$.

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

$$\mathbf{G}_{\text{in}} = \text{concat}(\mathcal{G}) \in \mathbb{R}^{T \times N \times C \times 2D} \tag{4.8}$$

$$\mathbf{Q}^i = \mathbf{W}_Q^i \mathbf{G}_{\text{in}} + \mathbf{b}_Q^i \tag{4.9}$$

$$\mathbf{K}^i = \mathbf{W}_K^i \mathbf{G}_{\text{in}} + \mathbf{b}_K^i \tag{4.10}$$

$$\mathbf{V}^i = \mathbf{W}_V^i \mathbf{G}_{\text{in}} + \mathbf{b}_V^i, \tag{4.11}$$

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

$$\text{Attention}(\mathbf{Q}^i, \mathbf{K}^i, \mathbf{V}^i) = \text{softmax}\frac{\mathbf{Q}^i(\mathbf{K}^i)^\intercal}{\sqrt{n}}\mathbf{V}^i, \tag{4.12}$$

where $n = M$. Subsequently, a position-wise feed-forward layer complemented by a ReLU activation function is employed, incorporating layer normalization and residual

connections at each layer. The resultant output is denoted by $\mathbf{G}_{\text{out}} \in \mathbb{R}^{T\times N\times C\times 2D}$. This output is averaged across the channel dimension via a global average pooling layer:

$$\mathbf{G}' = \frac{1}{C}\sum_{i=1}^{C}\mathbf{G}_{\text{out},i}, \tag{4.13}$$

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

4.4 Experimental Setup

4.4.1 Dataset

The experiments are conducted on the DIHARD III dataset [95] and AliMeeting dataset [130]. DIHARD III dataset is a multi-domain dataset that contains 34.15 hours of training data and 33.01 hours of evaluation data, including 11 different domains. The number of speakers in each recording ranges from 1 to 9. The average speech overlap ratio is 10.75% for the development set and 9.37% for the evaluation set, respectively. The Alimeeting dataset is a far-field 8-channel dataset that contains 118.75 hours of speech data, and it is divided into 104.75 hours for training, 4 hours for evaluation, and 10 hours for testing. The number of participants within one meeting session ranges from 2 to 4. The duration of each session is about 30 minutes. The average speech overlap ratio is 42.27% for the training set and 34.76% for the evaluation set, respectively.

The existing datasets for diarization tasks generally lack sufficient complexity for effective multi-speaker learning. Therefore, data simulation becomes imperative to train a model with robust generalization capabilities. For instance, to create our

training data for the DIHARD III dataset, we utilize the ground truth labels from the DIHARD III development dataset. First, we derive a label matrix $\mathbf{Y} \in \{0,1\}^{N\times T}$ for each recording within the DIHARD III development dataset, where N denotes the number of speakers and T signifies the duration of an utterance after excluding non-speech regions. The simulated data is then generated on-the-fly in an online manner, where we randomly choose a segment of the label and fill the active regions with the continuous non-overlapped speech segments from the Voxceleb dataset. Similarly, for the alimeeting dataset, the data is directly simulated from the Alimeeting Train set.

We perform data augmentation with MUSAN [93] and RIRs corpus [131]. For the MUSAN corpus, ambient noise, music, television, and babble noise are used for the background additive noise. For the RIRs corpus, only impulse responses from small and medium rooms are employed to perform convolution with training data.

4.4.2 Network configuration

For the ResNet-based OTS-VAD model, the front-end architecture is the same as the ResNet34 in [132]. After obtaining the temporal feature map, we apply the global statistics pooling over each frame of the feature map to produce the frame-level representation. Finally, the linear layer with 256 units projects the representation to frame-level embedding. For the back-end, it contains a 6-layer 8-head Conformer Encoder with a 512-dimensional feed-forward layer, and the dropout is set to 0.1. The hidden size of the single BiLSTM layer is 256. The fully-connected layer projects the output from BiLSTM to a N-dimensional vector, where N is the number of target speakers. In our experiment, N is set to 8 for the DIHARD III dataset and 4 for the Alimeeting dataset. The total number of parameters in the ResNet-based front-end is 20.54M.

For the Conformer-based OTS-VAD model, we employ the Conformer imple-

mented by the NEMO toolkit [133], and we follow the speaker verification training protocol in [134]. The NEMO Conformer is available in three variants: small, medium, and large. For our research, we opted for the 'small' variant[1], characterized by a convolution subsampling rate of 14 and a uniform kernel size of 31 within its convolution modules. This version encompasses 16 Conformer layers with an encoder dimension of 176, accommodates four attention heads, and comprises 704 linear hidden units. We only use the first eight layers for the speaker verification pretraining. The features processed by the MFA layer are then fed to two 1d convolution layers, each of which has a kernel size of 3, stride of 2 and padding of 1. The back-end is the same as the ResNet-based OTS-VAD model. The total number of parameters in the Conformer-based front-end is 9.38M.

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

4.4.3 Training and inferring details

Front-end pretraining

Both the ResNet and the Conformer front-end models are pretrained on the VoxCeleb 2 dataset [88]. For the ResNet model, we follow the training protocol mentioned in [132] and achieve 0.814% EER on the Vox-O trial. For the Conformer model, we follow the training protocol mentioned in [134], which has 0.65% EER on the Vox-O

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

trial.

Training process

In the beginning, the front-end module is initialized from a pretrained ResNet34- or Conformer-based speaker embedding model and other parts are randomly initialized. The training process contains three different stages:

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

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

During training, the L is set to 32s as mentioned in Section 4.3.2, which means that the input data is a 32-second audio signal and will be split into two 16-second blocks, as shown in Figure 4.2. The model is optimized by Adam optimizer with a max learning rate of 10^{-4} at stage 1, 10^{-5} at stage 2, and 5×10^{-6} at stage 3. The learning rate was increased from zero linearly over the first 2000 updates, and then it was annealed to 0 using a cosine schedule. The acoustic features are 80-dimensional log Mel-filterbank energies with a frame length of 25ms and a frame shift of 10 ms. Therefore, the time resolution of acoustic features is $0.01s$. Then, the features are downsampled by the front-end, and the time resolution of the frame-level

embeddings is $0.08s$ for the ResNet front-end and $0.02s$ for the Conformer front-end. For the Conformer front-end, we then employ two 1D convolution layers to reduce the time resolution by four times. Since Conformer and LSTM in the back-end do not change the sample rate of input, the output time resolution is also $0.08s$ for both the ResNet and Conformer front-end modules, respectively.

Inference process

For inference, we tried different block length $l \in \{2s, 4s, 8s, 16s\}$ and block shift $m \in \{0.4s, 0.8s, 1.6s\}$. Note that the block shift is also the latency of the OTS-VAD model. The upper and lower thresholds are tuned by grid search with a fixed block length and shift, e.g., 16s and 0.8s, respectively. After finding the upper and lower thresholds with the lowest DER, we apply these two thresholds for inference with other different block lengths and shifts. Finally, the output of each block is averaged and binarized with a threshold of 0.5 for scoring. During the inference process, we need an additional VAD module to remove the silence regions, and oracle VAD is employed for simplicity in our experiments.

Both two different inference processes mentioned in Figure 4.4 and 4.5 are evaluated. For the buffer-based inference process, we only evaluate it on the DIHARD III dataset in Figure 4.4. The reason is that the average duration of the Alimeeting dataset is about 30 minutes, which requires large memory to save the buffer and a long time for inference. For the accumulation-based inference process, we provide a comprehensive evaluation of both datasets.

Evaluation metric

The OTS-VAD model is evaluated on the evaluation set of both datasets, where we use the diarization error rate (DER) and Jaccard error rate (JER) as the evaluation metric. For the DIHARD III dataset, we follow the evaluation protocol in the

DIHARD III challenge [95], where no forgiveness collar will be applied to the reference segments prior to scoring. For Alimeeting, we follow the evaluation protocol in the Alimeeting challenge [130], where a forgiveness collar of 0.25 is employed. All overlapping speech will be evaluated.

Table 4.1: ResNet front-end DER (%) and JER (%) on different datasets with the embedding buffer-based inference process in Figure 4.4.

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

4.5 Experimental Results and Discussion

4.5.1 Detailed performance with different block length and shift

Table 4.1 and 4.2 shows the results of the OTS-VAD model with different front-end on DIHARD III dataset and Alimeeting dataset with different inference strategy as mentioned in Figure 4.4 and Figure 4.5, where we evaluated the model with different block length and shift.

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

Table 4.2: ResNet front-end DER (%) and JER (%) on different datasets with the accumulation-based inference process in Figure 4.5. SC refers to single-channel and MC refers to multi-channel.

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

For different front-end modules, as Table 4.1 and 4.2 show, ResNet front-end usually achieves better DER and JER scores compared to the Conformer front-end across most block lengths and block shifts. ResNet front-end achieves its best DER score at a block length of 8s or 16s.

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

a tiny performance degradation, e.g., 13.65% compared with 13.31%.

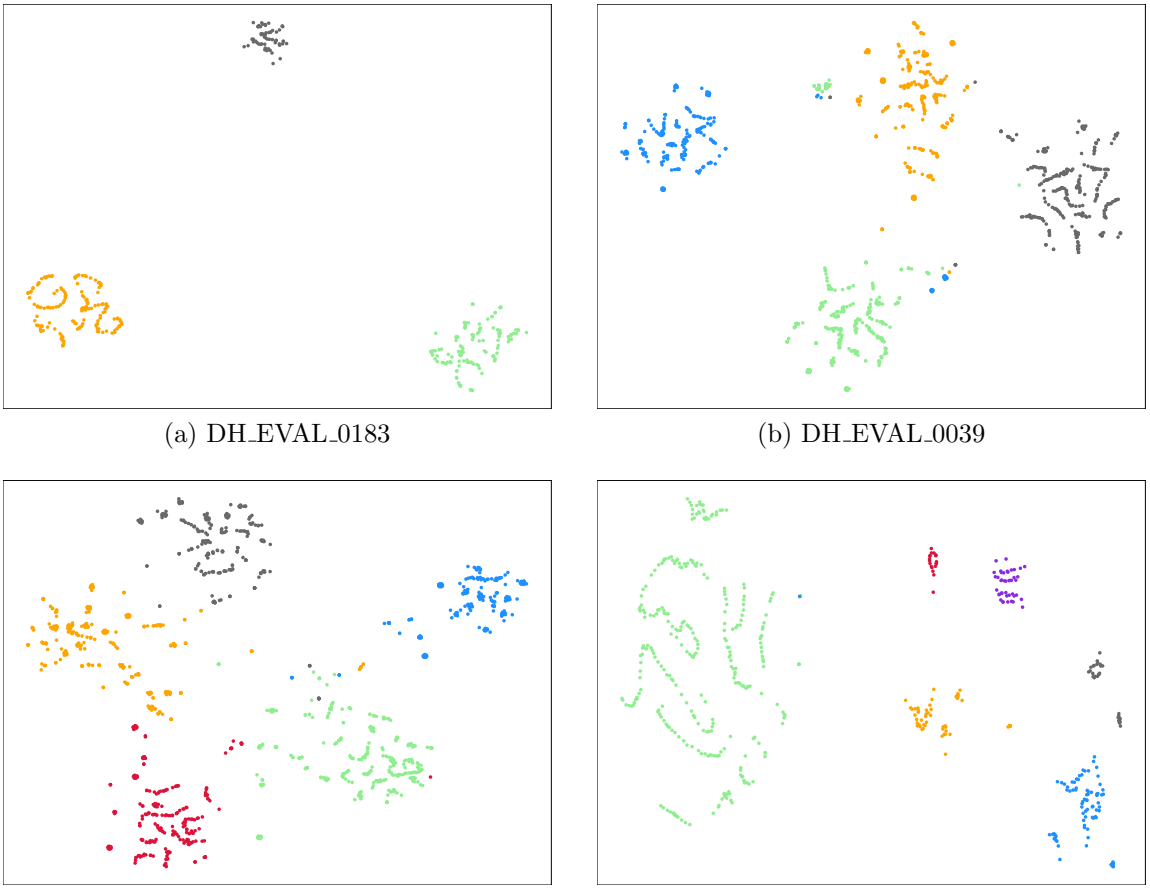
Block length and shift also matter for a better performance. For DIHARD III, the best performance in Table 4.1 and 4.2 are typically achieved with an 8s or 16s block length. Conversely, for Alimeeting, longer block lengths consistently yield improved and stable performance. This contrast may be attributed to the heterogeneous domains of the DIHARD III dataset, making hyper-parameter tuning more difficult and resulting in less stable outcomes. Furthermore, inference with lower block shifts exhibits enhanced stability, while longer block shifts introduce greater performance variability.

We also evaluate the multi-channel data on the cross-channel-based model, which improves the performance on the Eval and Test set by 14% and 9%, respectively. In addition, it can provide a more stable performance under different conditions, e.g., it can still show a satisfying performance with a block length of 2s on the Test set. Actually, compared with the offline TS-VAD [135], the improvement of multi-channel extension in online VAD is moderate; the reason is that we reduce the Encoder size to ensure enough GPU memory for training (three layers four heads v.s. six layers eight heads). We believe the performance will be much better if the same Encoder is employed for multi-channel extension.

4.5.2 Comparison with other offline and online systems

Table 4.3 shows the comparative analysis of various speaker diarization systems' performances on the DIHARD III dataset using the metric of DER, where both offline or online systems with oracle VAD are included. For offline systems, the Seq2seq TS-VAD is the SOTA system, which achieves a DER of 10.77%. For online systems, ResNet-based OTS-VAD shows the best performance with a DER of 13.31%.

Table 4.4 shows the comparison with other systems on the Alimeeting dataset. For the offline system, we show the official baseline [130] and the winner's system



(a) DH_EVAL_0183 (b) DH_EVAL_0039

(c) DH_EVAL_0044 (d) DH_EVAL_0196

Figure 4.6: t-SNE visualization of the target speaker embedding in each 16-second block during the inference process, where the cosine is employed as the distance metric. (a), (b), (c) and (d) are the samples from the DIHARD III dataset, each of which contains $3 \sim 6$ speakers, respectively.

[135]. For the online system, we do not find other online systems evaluated on the Alimeeting dataset, but we can directly compare it with the offline TS-VAD system. For the Alimeeting Eval set, both the single-channel and multi-channel OTS-VAD show a similar performance to the single-channel TS-VAD. For the Alimeeting Test set, the single-channel TS-VAD seems to be overfitted, but the multi-channel OTS-VAD still achieves a good performance compared with the offline multi-channel TS-VAD system.

Table 4.3: DERs (%) of different systems on DIHARD III dataset, where oracle VAD is employed.

Methods	Latency	# speakers $\leqslant 4$	# speakers $\geqslant 5$	Total
Offline				
VBx + resegmentation [117]	-	N/A	N/A	16.2
Seq2Seq TS-VAD [1]	-	6.65	25.90	10.77
Online				
Zhang et al. [13]	0.5s	N/A	N/A	19.57
Core samples selection [136]	2s	N/A	N/A	19.3
EEND-GLA-Large [100]	1s	8.85	38.86	14.70
ResNet-based OTS-VAD	0.8s	**8.78**	**32.03**	**13.31**

Table 4.4: DERs (%) of different systems on Alimeeting dataset, where oracle VAD is employed. SC refers to single channel, and MC refers to multi channel.

		Alimeeting Eval set				**Alimeeting Test set**			
Methods	Latency	2 spk	3 spk	4 spk	All	2 spk	3 spk	4 spk	All
Offline									
Official baseline [130]	-	4.84	10.90	23.18	15.24	3.00	6.36	30.59	15.60
SC TS-VAD [135]	-	0.89	6.63	5.47	4.11	-	-	-	-
MC TS-VAD [135]	-	0.61	3.16	3.05	**2.25**	0.15	5.19	4.37	**2.98**
Online									
ResNet-based SC OTS-VAD	0.8s	0.85	3.98	9.27	5.67	0.33	6.18	8.39	4.96
ResNet-based MC OTS-VAD	0.8s	0.77	4.04	7.45	**4.71**	0.43	11.60	4.73	**4.50**

4.5.3 Importance of pretraining

We also evaluate if the speaker verification pretraining of the front-end module is necessary. As Table 4.1 and 4.2 show, the model trained from scratch has worse performance, especially for very short block lengths. This means that the front-end without speaker verification pretraining cannot generalize well for the frame-level embeddings.

4.5.4 Visualization of target speaker embedding

To evaluate how the model learns the frame-level embeddings to form the target-speaker embedding, we also show the t-SNE of the ResNet-based target speaker embedding in each of the blocks with different numbers of speakers in Figure 4.6, where Figure 4.6a∼4.6d are from DIHARD III dataset with 3∼6 speakers. Each point in the figure is the mean embedding of the frame-level embedding in the blocks during the inference stage. As Figure 4.6 shows, the target speaker embedding can be well separated even if we do not specify an objective for training the embedding.

4.5.5 Training/inference time and real-time factor

Our experiments are conducted on NVIDIA Geforce RTX 3090 GPU. The training time is about two days with two GPUs for each single-channel dataset and four days with 8 GPUs for each multi-channel dataset. For the accumulation-based inference process, we use one NVIDIA Geforce RTX 3090 GPU. For the single-channel ResNet-based OTS-VAD, the inference time for a 16s block is about 0.064s. For the multi-channel ResNet-based OTS-VAD, the inference time for a 16s block is about 0.186s. Given that latency is the block shift, the real-time factors (RTFs) can be easily calculated as $\frac{t}{m}$, where t is the average inference time for a block and m is the block shift. The RTF for single-channel and multi-channel OTS-VAD with 0.4s block shift are 0.16 and 0.465, respectively. This demonstrates that the proposed method is possible for online inference.

4.6 Conclusion

In this chapter, we introduce an innovative end-to-end framework, OTS-VAD, designed for online speaker diarization using a target speaker profile as a guide. Adapted from the offline TS-VAD framework, this online version utilizes self-generated target

speaker embeddings for detecting speaker activity. Evaluated on multiple datasets, including DIHARD III and Alimeeting, OTS-VAD consistently exhibited strong performance across varied scenarios, attesting to its reliability. In addition, we expanded the model's capability to handle multi-channel data, enhancing its resilience to different block lengths and shifts. We also transitioned from the ResNet front-end to the more efficient Conformer Encoder, which reduces the model size with little performance degradation. Our results show that OTS-VAD not only meets but, in some cases, surpasses state-of-the-art benchmarks, drawing close to certain offline speaker diarization systems. Its real-time factor suggests potential suitability for online applications. Given its ability to self-generate target speaker embeddings and produce the diarization results, OTS-VAD can be considered as a fully end-to-end speaker diarization system.

However, the model also has its limitations. First, it currently lacks an integrated VAD component, which could limit its applicability to real-world scenarios. Additionally, its relatively large size and extended CPU inference time present challenges, though techniques like quantization could mitigate these issues. Looking ahead, our future work will focus on incorporating a VAD module and refining the model, both in size and inference strategy, to better suit real-world data applications.

Chapter 5

Target-speaker Voice Activity Detection: From Unimodal to Multimodal

5.1 Introduction

Speaker diarization identifies both speaker identities and the timing of each speaker's presence. This task demands robust speaker representations coupled with contextual information to discriminate speaker identities at different timestamps. However, in many traditional clustering-based methods, this contextual information is often overlooked. In these approaches, speaker representations are typically evaluated using specific metrics and then clustered without any contextual cues.

Several studies have incorporated both speaker representations and contextual information into speaker diarization tasks. Landini et al. [59] introduced a diarization algorithm based on Bayesian Hidden Markov Models (HMM) to refine initial diarization results. Lin et al. [26] proposed an LSTM-based model to extract a similarity matrix, effectively incorporating contextual information. Subsequently, a target-speaker voice activity detection approach also founded on an LSTM framework, was presented to further refine diarization results and recognize overlapping speech. Furthermore, the End-to-end neural diarization (EEND) [65, 66] approach,

which has gained considerable popularity, uses LSTM or Transformers to directly compute speaker diarization results. Within this approach, contextual information can be implicitly learned during the training phase. However, these methodologies often overlook spoken content, which could facilitate a more contextual approach to speaker diarization.

Automatic speech recognition (ASR), which translates spoken language into text, has been effectively developed to enhance the performance of both speaker verification and speaker diarization. ASR's primary focus is on recognizing the linguistic content of speech, paying particular attention to frame-level details. For instance, frame-level phoneme modeling in ASR can enhance speaker verification by identifying distinctive, speaker-specific speech patterns. Previous research provides support for this collaboration, revealing that phoneme modeling improves speaker verification performance in speaker embedding networks [137, 138] as well as in the i-vector statistical model [139, 140].

Speaker diarization systems have also increasingly integrated with ASR techniques. For example, word alignment has been used to refine speech activity detection (SAD) [141], while others have leveraged it to detect change points [110]. Subsequent methods incorporated lexical information for diarization, such as the text-based role recognizer [142] and segmentation using ASR outputs [143]. Furthermore, some studies have jointly optimized ASR and speaker diarization systems. In such configurations, either one system benefits from the other [144], or both systems can be improved from each other [145].

In this chapter, we propose a joint inference method of speaker diarization and ASR, where we directly build a speaker diarization system from a pretrained Conformer-based ASR model to improve the diarization performance. Conformer encoders, initially designed for ASR, exhibit a natural versatility due to their layered structure, which enables them to grasp various aspects of speech. The lower layers of the ASR

Conformer capture a range of speech characteristics, including speaker traits, language patterns, emotions, and phonetic nuances. In contrast, the upper layers focus more on phonetic and contextual details, aligning with ASR goals. Although their primary training focus is ASR, even the initial layers exhibit remarkable proficiency in speaker recognition [146]. This suggests that ASR-trained features can be used for speaker verification effectively. However, the upper layers, which convey contextual information, are ignored in transfer learning because the lack of speaker-related details may degrade the performance of speaker representation. Therefore, we follow the training protocol in [134]. Moreover, we further incorporate the output of the fixed upper layers of the ASR conformer as auxiliary features for speaker diarization.

This chapter introduces the speaker diarization integrated with ASR under both offline and online scenarios. For the offline speaker diarization system as mentioned in Chapter 3, we adapt a sequence-to-sequence target-speaker voice activity detection (Seq2Seq TS-VAD) module [1] from a pretrained ASR Conformer and explore how the ASR pretraining model can improve the performance. For the online speaker diarization system, as mentioned in Chapter 4, it is further improved to become an end-to-end joint model that can perform speaker embedding extraction, speaker diarization, and ASR simultaneously.

5.2 Related Works

5.2.1 Unified ASR and speaker verification

In Figure 5.1, the architectures of the unified ASR and speaker adaptor are consistent with the design presented in [134]. On the left side, there is a frozen, pretrained Conformer-based ASR model, while on the right lies a trainable speaker adaptor that interacts with the ASR model's intermediate representations. This adaptor contains three components: L layer adaptors, K trainable Conformer layers, and a fully

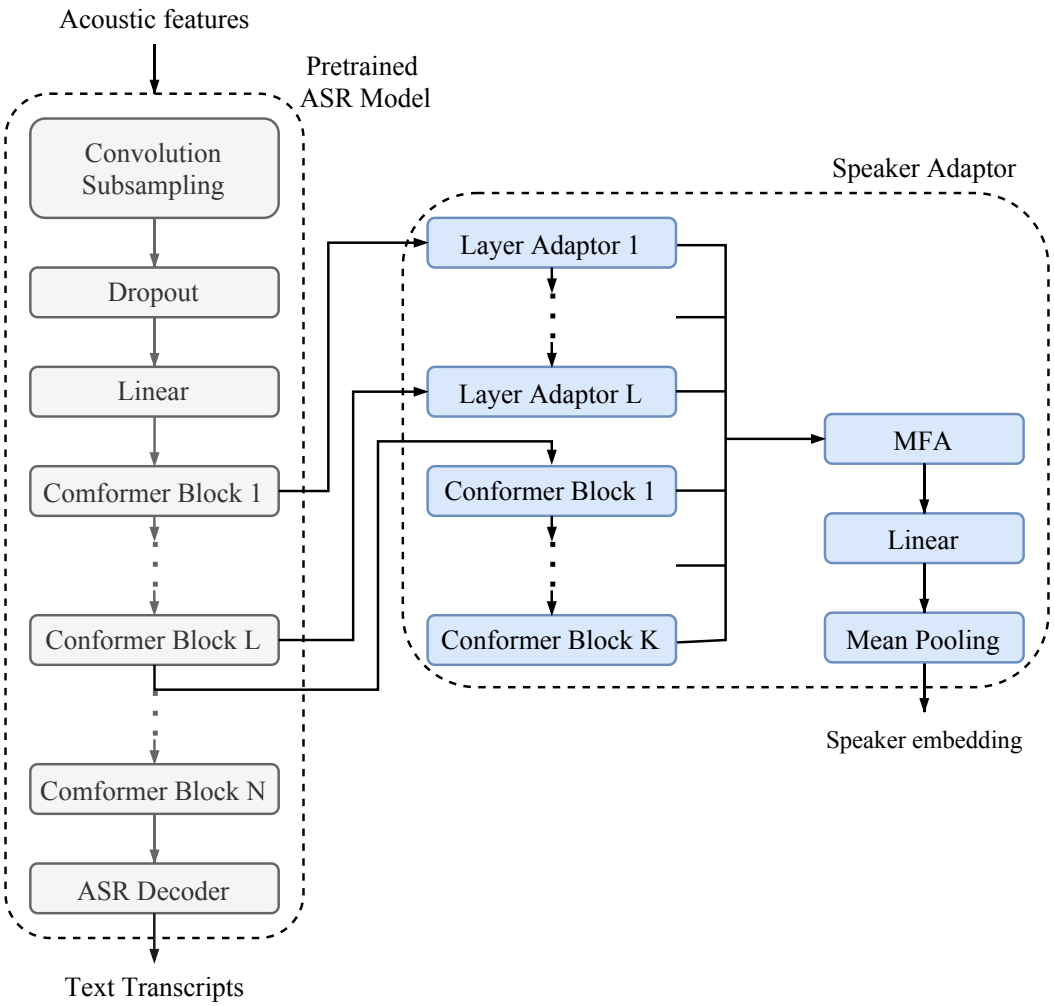


Figure 5.1: The architecture of the speaker verification model with speaker adaptor, where the pretrained ASR model is frozen, and other parts are trainable. In addition, we modify the last layer with mean pooling for speaker diarization usage.

connected layer preceded by a pooling layer for extracting speaker embeddings. Each layer adaptor consists of two linear layers, interspersed with Layer Normalization and an activation function.

Outputs from the first L layers of the pretrained ASR Conformer encoder undergo dimensional reduction through the L layer adaptors and K trainable Conformer layers. Subsequently, the outputs from these adaptors and the trainable Conformer layers are concatenated using a multi-scale feature aggregation (MFA) module [123].

A linear layer transforms these multi-scale feature maps into a lower dimension, and then they are averaged by mean pooling to generate the utterance-level speaker embedding. Notably, given that the parameters of the pretrained ASR remain frozen, the model can concurrently produce the text transcript.

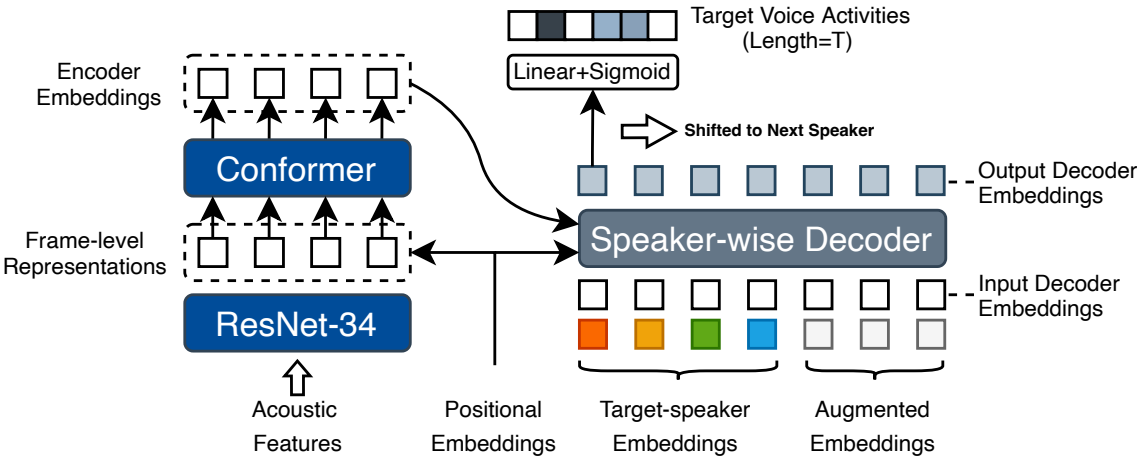


Figure 5.2: The architecture of the Seq2Seq TS-VAD. (From Cheng et al. [1])

5.2.2 Seq2Seq TS-VAD

In conventional TS-VAD systems [64], each target-speaker embedding is concatenated with the frame-level representation to determine the probability of the speaker's presence. However, this process can be resource-intensive, particularly in terms of GPU memory. As the number of target-speaker embeddings and frame sequences (T and N, respectively) increase, there's a significant surge in memory consumption. This limits the model's capability to handle longer feature sequences and accommodate a larger number of speakers simultaneously. Another limitation is that the output length of models relying solely on encoders must match the input length, constraining the temporal resolution of output and making it inflexible.

To address these challenges, we introduced a sequence-to-sequence framework for target-speaker voice activity detection (Seq2Seq TS-VAD) [1]. Within this frame-

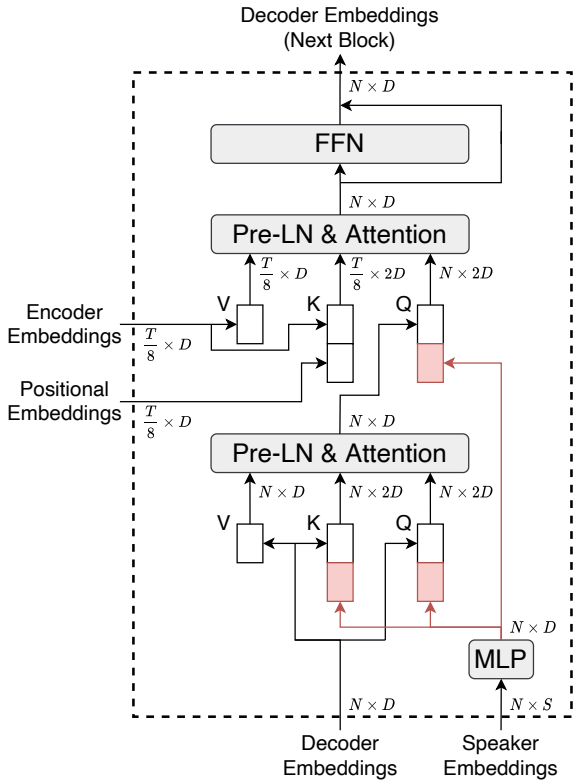


Figure 5.3: The details of the speaker-wise decoder. (From Cheng et al. [1])

work, frame-level representations and speaker embeddings are channeled separately into the encoder and decoder, eliminating the need for concatenation. A key advantage is that the decoder consolidates each speaker's voice activity data into an embedding with a fixed dimension, regardless of the input features' length. Another advantage is the flexibility offered by the final linear layer, which permits voice activity predictions at a higher temporal resolution, achieved with minimal computational overhead. Figure 5.2 shows the architecture of Seq2Seq TS-VAD and Figure 5.3 shows the architecture of speaker-wise decoder.

5.3 Unified ASR and Offline Speaker Diarization

5.3.1 Architecture

Given the observed benefits of speaker verification using the pretrained Conformer-based ASR model, we believe that this model can also enhance speaker diarization performance. Both the speaker identity information from the lower layers and the contextual details from the upper layers play significant roles in speaker diarization. Initially, we utilize only the lower layers to assess the efficacy of the pretrained Conformer-based ASR model in this context.

As depicted in Figure 5.4, the MFA module extracts frame-level speaker representations from the acoustic features. These outputs are subsequently fed to a standard Conformer Encoder to further model the long-term dependencies among frame-level representations. The speaker-wise decoder (SW-D) then discerns the voice activities of the target speaker, taking cross-speaker correlations into consideration. The SW-D inputs encompass both decoder embeddings and auxiliary queries. Notably, these decoder embeddings are initialized to zeros, while the target-speaker embeddings serve as the auxiliary queries, as shown in Figure 5.2. Finally, a linear layer, followed by a sigmoid activation, projects the decoder output into posterior probabilities, indicating the voice activities of each respective speaker.

5.3.2 Integration of contextual details into diarization

To enhance the speaker diarization performance, we also leverage the upper layers of the pretrained Conformer-based ASR model. More specifically, we utilize the output from the final ASR Conformer layer as the ASR features. Subsequently, this output is combined with the intermediate features of the Seq2Seq TS-VAD model and undergoes Layer Normalization. To integrate the contextual details into speaker diarization, we combine the ASR features with the input of the Conformer Encoder

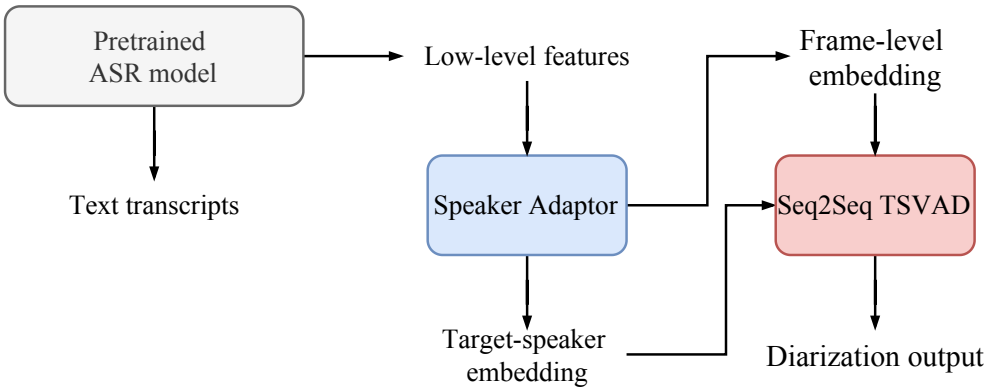


Figure 5.4: The process of offline unified ASR and speaker diarization. The detailed architecture of the pretrained ASR model and speaker adaptor can be found in Figure 5.1. The architecture of the Seq2Seq TS-VAD is shown in Figure 5.2.

or Decoder to let the model learn ASR details itself.

With the integration of ASR features, the performance of speaker diarization can be further improved as the contextual details are included.

5.4 Unified ASR and Online Speaker Diarization

As elaborated in Chapter 4, the offline TS-VAD model can be transitioned to an online version. This transition can similarly be applied to our proposed offline speaker diarization system, which incorporates ASR Integration. Figure 5.5 depicts the architecture of the suggested online speaker diarization system with ASR integration. In contrast to the ResNet-based online system detailed in Chapter 4, which uses frame-level embedding for target-speaker aggregation, our model introduces two distinct speaker adaptors for target-speaker embedding and frame-level embedding extraction. This differentiation optimizes both the target-speaker embedding and frame-level embedding, yielding a more discerning speaker representation.

As illustrated in Figure 5.5, the t-adaptor is for target-speaker embedding extraction, while the f-adaptor is for frame-level embedding extraction. The training and inference processes are the same as those delineated in Chapter 4. This model

uniquely addresses three tasks using a singular model: ASR, speaker verification, and speaker diarization.

It's important to highlight that the Seq2Seq TS-VAD often outperforms the standard TS-VAD when the target speaker embedding is well-trained and discriminative. However, during the online TS-VAD inference, the initial accumulated speaker embedding might be imprecise. In such scenarios, the standard TS-VAD is typically more effective than the Seq2Seq TS-VAD. This underlines our rationale for not adopting the Seq2Seq TS-VAD for online applications.

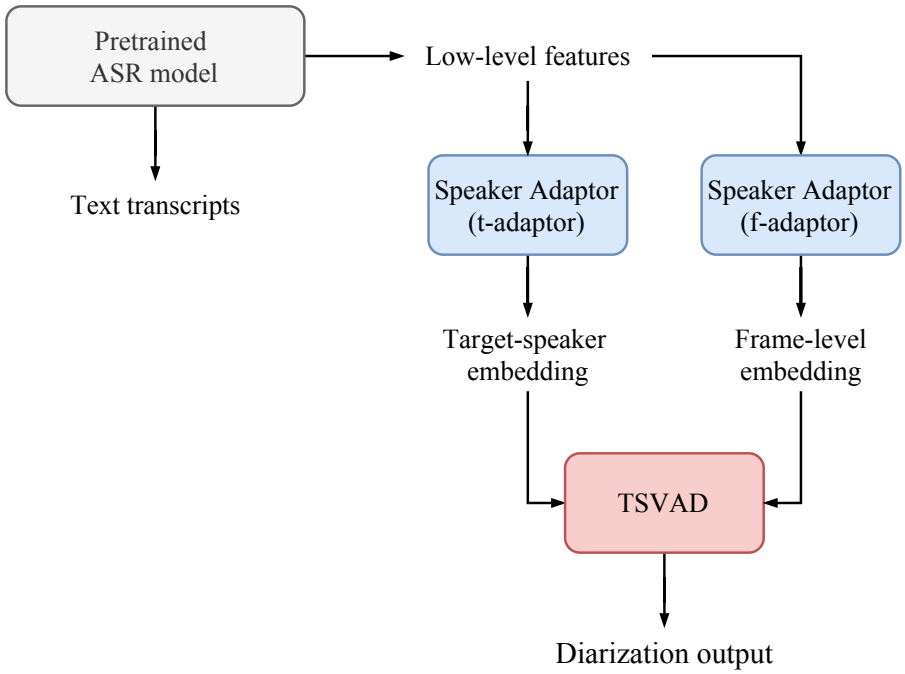


Figure 5.5: The process of online unified ASR and speaker diarization. The detailed architecture of the pretrained ASR model and speaker adaptor can be found in Figure 5.1.

5.5 Experimental Setup

5.5.1 Pretrained Conformer-based ASR

We utilized the pretrained ASR Conformer-based models available in the NEMO toolkit [133]. Our preference for this model was guided by its commendable performance and versatility across a range of benchmark datasets. The NEMO ASR Conformer is available in three variants: small, medium, and large. For our research, we opted for the 'small' variant[1], characterized by a convolution subsampling rate of 14 and a uniform kernel size of 31 within its convolution modules. This version encompasses 16 Conformer layers with an encoder dimension of 176, accommodates four attention heads, and comprises 704 linear hidden units.

5.5.2 Unified ASR and speaker verification

This model was trained using the development set from VoxCeleb 2, comprising 1,092,009 audio recordings spanning 5,994 distinct speakers.

During the training phase, the entire pretrained ASR Conformer was frozen without updates, and we only focused on training the speaker adaptor. This component contains four layer adaptors with an output dimension of 128, 2 streamlined Conformer layers with an output dimension of 176, a pooling layer, and subsequent linear layers, all of which are optimized with respect to the speaker verification objective. Consequently, only the outputs from the initial 4 Conformer layers were utilized as inputs for the speaker adaptor. For this configuration, $L = 4$ and $K = 2$. Ultimately, the outputs from both the adaptors and trainable Conformer layers are concatenated, forming a feature sequence with a dimension of $128 \times 4 + 176 \times 2 = 864$. Following layer normalization and pooling, this sequence is transformed into an utterance-level speaker embedding with a size of 256. Comprehensive training details and hyper-

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

parameters can be referred to in [134].

5.5.3 Unified ASR and speaker diarization

For offline speaker diarization, we utilize the pretrained ASR model and speaker adaptor as the front-end module, with the speaker adaptor pretrained on VoxCeleb 2. We adopt the feature sequence preceding the pooling layer as the frame-level speaker representation, which possesses a dimension of 864 for each frame. Subsequently, a Conformer Encoder refines this feature sequence, and the outputs with target-speaker embeddings and zero-initialized embeddings are fed to the Decoder. The Decoder's output is then converted into posterior probabilities representing voice activities for all speakers. All encoder-decoder components consist of 6 layers and maintain identical configurations: 512-dimensional attentions with eight heads and 1024-dimensional feed-forward layers with a dropout rate set at 0.1.

The training audio signals are segmented into 8-second chunks. These segments serve as input to the model, which utilizes 80-dimensional log Mel-filterbank energies with a frame length of 25 ms and a frame shift of 10 ms for acoustic features. Additionally, data augmentation is performed using background noise from Musan [93] and reverberation from RIRs [131].

During the training phase, we employ the BCE loss and the Adam optimizer with a linear learning rate warm-up strategy. Initially, the model with a frozen ASR model and frozen speaker adaptor is trained using simulated data created from the Voxceleb 2 dataset [88] until convergence. Subsequently, the parameters of the speaker adaptor are unfrozen for training. Real data from the DIHARD III dataset [95] is then incorporated with the simulated data at a ratio of 0.2. Ultimately, the model undergoes fine-tuning exclusively with real data without any simulation. The initial two training stages encompass roughly 200 epochs with a learning rate set at 1e-4, while the final stage adjusts the learning rate to 5e-6.

During the inference stage, we utilize spectral clustering [26] to obtain the initial clustering-based results. Note that this step can be replaced by adopting an EEND model for initial clustering [147]. For evaluation purposes, each test recording is divided into 8-second segments. These are then input into the Seq2Seq TS-VAD model, accompanied by the target-speaker embeddings. We cap the number of speaker profiles at 10, padding with zero values for any additional speaker embeddings as required. To enhance the results, oracle VAD is employed during post-processing to retrieve missing frames. Comprehensive details on training and inference can be referenced in [1].

For online speaker diarization, the training and inference are the same as that mentioned in Chapter 4. In the first stage, both the t-adaptor and f-adaptor are initialized by a pre-trained speaker adaptor, and the parameters of these two adaptors are frozen. In the second and third stages, the f-adaptor is unfrozen to learn the pattern of the development set. The model is trained on the simulated data created from the Voxceleb 2 dataset [88], finetuned on the AMI train set and evaluated on the AMI dev/eval set [89].

5.5.4 Integration of ASR features

We leverage the output from the final ASR Conformer layers to acquire contextual information. This output corresponds to the feature sequence preceding the ultimate linear layer responsible for token conversion, with a feature dimension of 176. The backbone architecture of the Seq2Seq TS-VAD model remains unchanged, and the ASR feature is only utilized to assist the model in learning enhanced representations. This ASR feature sequence can be integrated either with the Encoder or the Decoder input. The combined input will be layer-normalized before sending it to the Encoder or Decoder.

Note that we only integrate the ASR features into offline speaker diarization as

the speaker embedding estimated in the online process may not be very stable.

5.6 Experimental Results and Discussion

For simplicity and efficiency, we utilize the small version of the unified ASR and speaker verification ($L = 4, K = 2$). This version achieves an EER of 0.98 on VoxCeleb1-O with a model size of 6.6M [134].

5.6.1 Offline speaker diarization

The unified ASR and speaker diarization models are evaluated on Track 1 of the DIHARD III dataset [95] under the offline scenario. The initialization for clustering, which is used for target speaker embedding extraction, is based on the LSTM-SC method [26]. This method has a DER of 15.4% with Oracle VAD on the evaluation set.

Table 5.1 presents the results of the offline unified ASR and speaker diarization in comparison with other approaches. Without the aid of the ASR features, it achieves a DER of 10.52%, which is competitive with other SOTA systems. It also offers the capability to produce text transcripts based on diarization results for each speaker. Moreover, by integrating the ASR features with either Encoder or Decoder input, the DER can be further lowered to 10.39% and 10.42%, respectively.

Actually, as Table 5.1 demonstrates, there isn't a significant overall improvement when utilizing ASR features from the last Conformer layer. We hypothesize that this is because certain domains are too complex for the model to effectively learn the contextual details. For instance, audio might contain excessive overlapping speech or the signal-to-noise ratio may be too low. As indicated by Table 5.2, we assess performance across different domains. We can find out that there is performance improvement (sys2 vs. sys1) in all domains except the Restaurant and Socio field domains, which might be because of the inaccurate ASR predictions.

Table 5.1: DERs (%) on the DIHARD III Dataset (Offline).

Method	**DER (%)**	**JER (%)**
VBx [59]	16.54	37.82
Hitachi-JHU [148]	12.74	34.08
USTC-NELSLP [77]	12.41	-
ANSD-MA-MSE [149]	11.12	-
Seq2Seq TS-VAD [1]	10.77	28.46
LSTM-SC	15.40	33.27
+ Unified ASR & SD	10.52	28.12
+ ASR feat before SD Encoder	**10.39**	27.97
+ ASR feat before SD Decoder	10.42	**27.85**

It is worth noting that in some specific domains (e.g., Clinical), where the predicted text has more role information, the ASR feature can bring in more improvement.

Table 5.2: DERs (%) over 11 domains on the DIHARD III dataset (Offline). Sys 1, 2 and 3 refer to the last three rows in Table 5.1, respectively.

Domain	**LSTM-SC**	**Sys 1**	**Sys 2**	**Sys 3**
Audiobook	0.00	0.00	0.00	0.00
Broadcast	5.06	3.82	3.78	3.85
Clinical	7.59	5.01	4.63	4.61
Courtroom	4.10	2.06	2.04	2.18
CTS	14.46	6.01	5.89	5.85
Maptask	3.85	1.21	1.12	1.22
Meeting	28.70	22.95	22.54	23.16
Restaurant	42.30	38.97	39.36	39.25
Socio field	8.42	5.12	5.45	5.39
Socio lab	7.08	2.82	2.79	2.71
Webvideo	38.13	34.38	33.31	33.38
All	15.40	10.52	10.39	10.42

Table 5.3 shows the number of the trainable parameters in the front-end model of Seq2Seq TS-VAD. The unified ASR and SD system with a small front-end, with a mere tenth of the parameters of the one in the original Seq2Seq TS-VAD system, can

produce competitive diarization results. This efficiency demonstrates the potential of the Speaker Adaptor in harnessing vital speaker information with fewer trainable parameters.

Table 5.3: The number of parameters for the speaker embedding extractor front-end in the Seq2Seq TS-VAD and unified ASR and speaker diarization systems (Offline). The difference is that the trainable front-end of these two systems are ResNet34 and speaker adaptor, respectively. The number of the parameters of the first four layers of the ASR model is 3.92M, which is borrowed from ASR and remains unchanged.

System	**Front-end**	**#Parameters**
Seq2Seq TS-VAD	ResNet34	20.56M
Unified ASR & SD	Speaker Adaptor	2.68M (+3.92M)

5.6.2 Online speaker diarization

For the online scenario, we evaluate the model on the AMI dataset [89]. Since the proposed method is a fully end-to-end model, we do not need another model for initialization. Since there are currently no online methods employed for this dataset, we compare our system with the popular offline methods. Table 5.4 shows the results of the online unified ASR and speaker diarization. From the table, we can find that our online model is even much better than those systems executed in an offline manner.

Table 5.4: DERs (%) on the AMI Dataset (Online).

		Dev		**Test**	
Method	**Immediacy**	**DER (%)**	**JER (%)**	**DER (%)**	**JER (%)**
VBx [59]	Offline	22.98	-	22.85	-
Transcribe-to-Diarize [150]	Offline	15.98	-	16.58	-
MSDD [151]	Offline	22.20	-	21.19	-
Unified ASR & SD	Online	**9.04**	13.67	**14.56**	24.11

5.7 Conclusion

In this chapter, we introduced a unified ASR and speaker diarization framework that can concurrently perform ASR and speaker diarization inference. Within the proposed framework, features from the lower layers of a pretrained ASR model are utilized to extract speaker-related information, while the upper layers provide ASR-related features that can enhance diarization performance. On the DIHARD III dataset, this framework demonstrates competitive performance when compared to other state-of-the-art systems. However, the DIHARD dataset, being a challenging compilation that contains substantial background noise and overlapping speech, can severely ruin the contextual information. As such, the benefits of integrating ASR features might be moderate on this dataset. Future research will involve evaluating our model on some domain-specific datasets.

It's also noteworthy that the parameters of the ASR model remain frozen, complicating the joint training of ASR and speaker diarization. In the future, we plan to unfreeze the ASR model, leveraging diarization results to refine ASR performance, such as in multi-talker ASR or target speaker ASR scenarios.

Chapter 6

Conclusion

This dissertation presents a systematic investigation into speaker diarization, advancing techniques to address three pivotal challenges: overlapping speech detection, online extension, and contextual data integration.

Chapter 3 tackles the intricacies of overlapping speech segments, a persisting issue for conventional modular diarization systems. We introduce a two-stage framework, first employing neural networks for enhanced speaker embedding extraction and similarity measurement. Subsequently, a TS-VAD module leverages these target embeddings to precisely identify overlapping regions. This approach effectively integrates sequential data and demonstrates significant improvements.

Chapter 4 focuses on the pressing need for online, low-latency speaker diarization. We adapt the target-speaker voice activity detection technique into a fully end-to-end system capable of real-time inference. It continually updates speaker embeddings and diarization outputs using incoming audio signals while maintaining consistent speaker order across blocks. This online diarization approach exhibits robust performance across datasets.

Chapter 5 explores the integration of ASR into speaker diarization. By leveraging both low-level acoustic patterns and high-level textual context from ASR systems, speaker discrimination and overall diarization accuracy are enhanced. We propose

techniques to unify ASR and diarization in both offline and online settings.

Collectively, this dissertation offers substantial contributions, systematically addressing limitations in handling overlapping speech, enabling online use cases, and harnessing multimodal data. The approaches presented not only push the boundaries of academic research but also demonstrate practical, real-world potential. Looking ahead, further work can build upon these advancements to develop more versatile, efficient and accurate speaker diarization systems. With speech technologies becoming integral in diverse environments, enhancing speaker diarization remains an impactful pursuit within machine learning.

Appendix A

Publications during Ph.D.

Journal Articles

Weiqing Wang, Ming Li, "End-to-end Online Speaker Diarization with Target Speaker Tracking", submitted to IEEE/ACM Transactions on Audio, Speech, and Language Processing.

Weiqing Wang, Qingjian Lin, Danwei Cai, Ming Li, "Similarity measurement of segment-level speaker embeddings in speaker diarization", published in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 30, pp. 2645-2658, 2022.

Danwei Cai, **Weiqing Wang**, Ming Li, "Incorporating visual information in audio based self-supervised speaker recognition", published in in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 30, pp. 1422-1435, 2022.

Conference Proceedings

Weiqing Wang, Danwei Cai, Ming Cheng, Ming Li, "Joint inference of speaker diarization and ASR with multi-stage information sharing", submitted to IEEE In-

ternational Conference on Acoustics, Speech and Signal Processing (ICASSP) 2024.

Zexin Cai, **Weiqing Wang**[1] Ming Li, "Waveform boundary detection for partially spoofed audio", published in ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1-5, 2023.

Danwei Cai, **Weiqing Wang**, Ming Li, Rui Xia, Chuanzeng Huang, "Pretraining Conformer with ASR for Speaker Verification", published in ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1-5, 2023.

Weiqing Wang, Qingjian Lin, Ming Li, "Online target speaker voice activity detection for speaker diarization", published in Conference of the International Speech Communication Association (INTERSPEECH), pp. 1441–1445, 2022.

Weiqing Wang, Ming Li, "Incorporating end-to-end framework into target-speaker voice activity detection", published in ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8362-8366, 2022.

Weiqing Wang, Xiaoyi Qin, Ming Li, "Cross-channel attention-based target speaker voice activity detection: Experimental results for the M2MeT challenge", published in ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 9171-9175, 2022.

Danwei Cai, **Weiqing Wang**, Ming Li, "An iterative framework for self-supervised deep speaker representation learning", published in ICASSP 2021 - 2021 IEEE In-

[1] Authors contributed equally

ternational Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6728-6732, 2021.

Bibliography

[1] Ming Cheng, Weiqing Wang, Yucong Zhang, Xiaoyi Qin, and Ming Li, "Target-speaker voice activity detection via sequence-to-sequence prediction,", in *Proc. of ICASSP*, 2023, pp. 1–5.

[2] Tae Jin Park, Naoyuki Kanda, Dimitrios Dimitriadis, Kyu J Han, Shinji Watanabe, and Shrikanth Narayanan, "A review of speaker diarization: Recent advances with deep learning,", *Computer Speech & Language*, vol. 72, pp. 101317, 2022.

[3] John S Garofolo, Jonathan G Fiscus, Alvin F Martin, David S Pallett, and Mark A Przybocki, "Nist rich transcription 2002 evaluation: A preview.,", in *LREC*, 2002.

[4] Neville Ryant, Kenneth Church, Christopher Cieri, Alejandrina Cristia, Jun Du, Sriram Ganapathy, and Mark Liberman, "First dihard challenge evaluation plan,", *tech. Rep.*, 2018.

[5] Neville Ryant, Kenneth Church, Christopher Cieri, Alejandrina Cristia, Jun Du, Sriram Ganapathy, and Mark Liberman, "The Second DIHARD Diarization Challenge: Dataset, Task, and Baselines,", in *Proc. of Interspeech*, 2019, pp. 978–982.

[6] Neville Ryant, Prachi Singh, Venkat Krishnamohan, Rajat Varma, Kenneth Church, Christopher Cieri, Jun Du, Sriram Ganapathy, and Mark Liberman, "The third dihard diarization challenge,", *arXiv preprint arXiv:2012.01477*, 2020.

[7] Joon Son Chung, Arsha Nagrani, Ernesto Coto, Weidi Xie, Mitchell McLaren, Douglas A Reynolds, and Andrew Zisserman, "Voxsrc 2019: The first voxceleb speaker recognition challenge,", *arXiv preprint arXiv:1912.02522*, 2019.

[8] Arsha Nagrani, Joon Son Chung, Jaesung Huh, Andrew Brown, Ernesto Coto, Weidi Xie, Mitchell McLaren, Douglas A Reynolds, and Andrew Zisserman,

"Voxsrc 2020: The second voxceleb speaker recognition challenge,", *arXiv preprint arXiv:2012.06867*, 2020.

[9] Andrew Brown, Jaesung Huh, Joon Son Chung, Arsha Nagrani, Daniel Garcia-Romero, and Andrew Zisserman, "Voxsrc 2021: The third voxceleb speaker recognition challenge,", *arXiv preprint arXiv:2201.04583*, 2022.

[10] Jaesung Huh, Andrew Brown, Jee-weon Jung, Joon Son Chung, Arsha Nagrani, Daniel Garcia-Romero, and Andrew Zisserman, "Voxsrc 2022: The fourth voxceleb speaker recognition challenge,", *arXiv preprint arXiv:2302.10248*, 2023.

[11] Ming Cheng, Weiqing Wang, Xiaoyi Qin, Yuke Lin, Ning Jiang, Guoqing Zhao, and Ming Li, "The dku-msxf diarization system for the voxceleb speaker recognition challenge 2023,", *arXiv preprint arXiv:2308.07595*, 2023.

[12] Dimitrios Dimitriadis and Petr Fousek, "Developing on-line speaker diarization system.,", in *Proc. of Interspeech*, 2017, pp. 2739–2743.

[13] Yucong Zhang, Qinjian Lin, Weiqing Wang, Lin Yang, Xuyang Wang, Junjie Wang, and Ming Li, "Low-Latency Online Speaker Diarization with Graph-Based Label Generation,", in *Proc. of Odyssey*, 2022, pp. 162–169.

[14] Aonan Zhang, Quan Wang, Zhenyao Zhu, John Paisley, and Chong Wang, "Fully supervised speaker diarization,", in *Proc. of ICASSP*, 2019, pp. 6301–6305.

[15] Yawen Xue, Shota Horiguchi, Yusuke Fujita, Shinji Watanabe, Paola García, and Kenji Nagamatsu, "Online end-to-end neural diarization with speaker-tracing buffer,", in *Proc. of SLT*, 2021, pp. 841–848.

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

[17] Jonathan G Fiscus, Jerome Ajot, Martial Michel, and John S Garofolo, "The rich transcription 2006 spring meeting recognition evaluation,", in *Machine Learning for Multimodal Interaction: Third International Workshop, MLMI 2006, Bethesda, MD, USA, May 1-4, 2006, Revised Selected Papers 3*. Springer, 2006, pp. 309–322.

[18] Chuck Wooters, James Fung, Barbara Peskin, and Xavier Anguera, "Towards Robust Speaker Segmentation: The ICSI-SRI Fall 2004 Diarization System,",

in *Proceedings of Fall 2004 Rich Transcription Workshop (RT-04F)*, 2004, p. 23.

[19] Ruben Zazo, Tara N. Sainath, Gabor Simko, and Carolina Parada, "Feature Learning with Raw-Waveform CLDNNs for Voice Activity Detection,", in *Proc. of Interspeech*, 2016, pp. 3668–3672.

[20] Florian Eyben, Felix Weninger, Stefano Squartini, and Björn Schuller, "Real-life Voice Activity Detection with LSTM Recurrent Neural Networks and an Application to Hollywood Movies,", in *Proc. of ICASSP*, 2013, pp. 483–487.

[21] Shuo-Yiin Chang, Bo Li, Gabor Simko, Tara N Sainath, Anshuman Tripathi, Aäron van den Oord, and Oriol Vinyals, "Temporal Modeling Using Dilated Convolution and Gating for Voice-activity-detection,", in *Proc. of ICASSP*, 2018, pp. 5549–5553.

[22] Marek Hrúz and Zbyněk Zajíc, "Convolutional Neural Network for Speaker Change Detection in Telephone Speaker Diarization System,", in *Proc. of ICASSP*, 2017, pp. 4945–4949.

[23] Ruiqing Yin, Hervé Bredin, and Claude Barras, "Speaker Change Detection in Broadcast TV Using Bidirectional Long Short-Term Memory Networks,", in *Proc. of Interspeech*, 2017, pp. 3827–3831.

[24] Gregory Sell and Daniel Garcia-Romero, "Speaker Diarization with PLDA i-vector Scoring and Unsupervised Calibration,", in *Proc. of of SLT*, 2014, pp. 413–417.

[25] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopz Moreno, "Speaker Diarization with LSTM,", in *Proc. of ICASSP*, 2018, pp. 5239–5243.

[26] Qingjian Lin, Ruiqing Yin, Ming Li, Hervé Bredin, and Claude Barras, "LSTM Based Similarity Measurement with Spectral Clustering for Speaker Diarization,", in *Proc. of Interspeech*, 2019, pp. 366–370.

[27] Qingjian Lin, Yu Hou, and Ming Li, "Self-Attentive Similarity Measurement Strategies in Speaker Diarization,", in *Proc. of Interspeech*, 2020, pp. 284–288.

[28] Tim Ng, Bing Zhang, Long Nguyen, Spyros Matsoukas, Xinhui Zhou, Nima Mesgarani, Karel Veselý, and Pavel Matějka, "Developing a speech activity detection system for the DARPA RATS program,", in *Proc. of Interspeech*, 2012, pp. 1969–1972.

[29] Thilo Pfau, Daniel PW Ellis, and Andreas Stolcke, "Multispeaker speech activity detection for the icsi meeting recorder,", in *Proc. of ASRU*, 2001, pp. 107–110.

[30] Ruhi Sarikaya and John HL Hansen, "Robust detection of speech activity in the presence of noise,", in *Proc. of ICSLP*, 1998, pp. 1455–8.

[31] Samuel Thomas, Sriram Ganapathy, George Saon, and Hagen Soltau, "Analyzing convolutional neural networks for speech activity detection in mismatched acoustic conditions,", in *Proc. of ICASSP*, 2014, pp. 2519–2523.

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

[33] David Haws, Dimitrios Dimitriadis, George Saon, Samuel Thomas, and Michael Picheny, "On the importance of event detection for asr,", in *Proc. of ICASSP*, 2016, pp. 5705–5709.

[34] Rashmi Gangadharaiah, Balakrishnan Narayanaswamy, and Narayanaswamy Balakrishnan, "A novel method for two-speaker segmentation,", in *Proc. of ICSLP*, 2004.

[35] Alain Tritschler and Ramesh A Gopinath, "Improved speaker segmentation and segments clustering using the bayesian information criterion,", in *Proc. of Eurospeech*, 1999.

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

[37] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition,", in *Proc. of ICASSP*, 2018, pp. 5329–5333.

[38] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn, "Speaker verification using adapted gaussian mixture models,", *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[39] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel, "Speaker and session variability in gmm-based speaker verification,", *IEEE*

Transactions on Audio, Speech, and Language Processing, vol. 15, no. 4, pp. 1448–1460, 2007.

[40] Patrick Kenny, Pierre Ouellet, Najim Dehak, Vishwa Gupta, and Pierre Dumouchel, "A study of interspeaker variability in speaker verification,", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 980–988, 2008.

[41] Weizhong Zhu and Jason Pelecanos, "Online speaker diarization using adapted i-vector transforms,", in *Proc. of ICASSP*, 2016, pp. 5045–5049.

[42] Mohammed Senoussaoui, Patrick Kenny, Themos Stafylakis, and Pierre Dumouchel, "A study of the cosine distance-based mean shift for telephone speech diarization,", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 217–227, 2013.

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

[44] Yi Sun, Xiaogang Wang, and Xiaoou Tang, "Deep learning face representation from predicting 10,000 classes,", in *Proc. of CVPR*, 2014, pp. 1891–1898.

[45] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, "Deepface: Closing the gap to human-level performance in face verification,", in *Proc. of CVPR*, 2014, pp. 1701–1708.

[46] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer, "End-to-end text-dependent speaker verification,", in *Proc. of ICASSP*, 2016, pp. 5115–5119.

[47] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Fred Richardson, Suwon Shon, François Grondin, et al., "State-of-the-art speaker recognition for telephone and video speech: The jhu-mit submission for nist sre18.,", in *Proc. of INTERSPEECH*, 2019, pp. 1488–1492.

[48] Weicheng Cai, Jinkun Chen, and Ming Li, "Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System,", in *Proc. of Odyssey 2018*, 2018, pp. 74–81.

[49] Tianyan Zhou, Yong Zhao, and Jian Wu, "Resnext and res2net structures for speaker verification,", in *Proc. of SLT*, 2021, pp. 301–307.

[50] Yang Zhang, Zhiqiang Lv, Haibin Wu, Shanshan Zhang, Pengfei Hu, Zhiyong Wu, Hung yi Lee, and Helen Meng, "MFA-Conformer: Multi-scale Feature Aggregation Conformer for Automatic Speaker Verification,", in *Proc. of Interspeech*, 2022, pp. 306–310.

[51] Rui Wang, Junyi Ao, Long Zhou, Shujie Liu, Zhihua Wei, Tom Ko, Qing Li, and Yu Zhang, "Multi-view self-attention based transformer for speaker recognition,", in *Proc. of ICASSP*, 2022, pp. 6732–6736.

[52] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition,", in *Proc. of CVPR*, 2017, pp. 212–220.

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

[54] Takuya Yoshioka and Tomohiro Nakatani, "Generalization of multi-channel linear prediction methods for blind mimo impulse response shortening,", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 10, pp. 2707–2720, 2012.

[55] Tian Gao, Jun Du, Li-Rong Dai, and Chin-Hui Lee, "Densely connected progressive learning for lstm-based speech enhancement,", in *Proc. of ICASSP*, 2018, pp. 5054–5058.

[56] Philipos C Loizou, *Speech enhancement: theory and practice*, CRC press, 2007.

[57] Morten Kolbæk, Dong Yu, Zheng-Hua Tan, and Jesper 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.

[58] Yi Luo and Nima Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation,", *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[59] Federico Landini, Ján Profant, Mireia Diez, and Lukáš Burget, "Bayesian HMM Clustering of x-vector Sequences (VBx) in Speaker Diarization: Theory, Implementation and Analysis on Standard Tasks,", *Computer Speech & Language*, p. 101254, 2021.

[60] Desh Raj, Leibny Paola Garcia-Perera, Zili Huang, Shinji Watanabe, Daniel Povey, Andreas Stolcke, and Sanjeev Khudanpur, "Dover-lap: A method for

combining overlap-aware diarization outputs,", in *Proc. of SLT*, 2021, pp. 881–888.

[61] Jürgen T. Geiger, Florian Eyben, Björn Schuller, and Gerhard Rigoll, "Detecting overlapping speech with long short-term memory recurrent neural networks,", in *Proc. of Interspeech*, 2013, pp. 1668–1672.

[62] Valentin Andrei, Horia Cucu, and Corneliu Burileanu, "Detecting overlapped speech on short timeframes using deep learning.,", in *Proc. of Interspeech*, 2017, pp. 1198–1202.

[63] Latané Bullock, Hervé Bredin, and Leibny Paola Garcia-Perera, "Overlap-aware diarization: Resegmentation using neural end-to-end overlapped speech detection,", in *Proc. of ICASSP*, 2020, pp. 7114–7118.

[64] Ivan Medennikov, Maxim Korenevsky, Tatiana Prisyach, Yuri Khokhlov, Mariya Korenevskaya, Ivan Sorokin, Tatiana Timofeeva, Anton Mitrofanov, Andrei Andrusenko, Ivan Podluzhny, Aleksandr Laptev, and Aleksei Romanenko, "Target-Speaker Voice Activity Detection: A Novel Approach for Multi-Speaker Diarization in a Dinner Party Scenario,", in *Proc. of Interspeech*, 2020, pp. 274–278.

[65] Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Kenji Nagamatsu, and Shinji Watanabe, "End-to-End Neural Speaker Diarization with Permutation-Free Objectives,", in *Proc. of Interspeech*, 2019, pp. 4300–4304.

[66] Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Yawen Xue, Kenji Nagamatsu, and Shinji Watanabe, "End-to-end neural speaker diarization with self-attention,", in *Proc. of ASRU*, 2019, pp. 296–303.

[67] Shota Horiguchi, Yusuke Fujita, Shinji Watanabe, Yawen Xue, and Paola Garcia, "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.

[68] Christoph Boeddecker, Jens Heitkaemper, Joerg Schmalenstroeer, Lukas Drude, Jahn Heymann, and Reinhold Haeb-Umbach, "Front-end Processing for the CHiME-5 Dinner Party Scenario,", in *Proc. CHiME 2018 Workshop on Speech Processing in Everyday Environments*, 2018, pp. 35–40.

[69] Takuya Yoshioka, Hakan Erdogan, Zhuo Chen, Xiong Xiao, and Fil Alleva, "Recognizing Overlapped Speech in Meetings: A Multichannel Separation Approach Using Neural Networks,", in *Proc. of Interspeech*, 2018, pp. 3038–3042.

[70] Xiong Xiao, Naoyuki Kanda, Zhuo Chen, Tianyan Zhou, Takuya Yoshioka, Sanyuan Chen, Yong Zhao, Gang Liu, Yu Wu, Jian Wu, et al., "Microsoft speaker diarization system for the voxceleb speaker recognition challenge 2020,", in *Proc. of ICASSP*, 2021, pp. 5824–5828.

[71] Federico Landini, Shuai Wang, Mireia Diez, Lukáš Burget, Pavel Matějka, Kateřina Žmolíková, Ladislav Mošner, Anna Silnova, Oldřich Plchot, Ondřej Novotnỳ, et al., "But System for the Second Dihard Speech Diarization Challenge,", in *Proc. of ICASSP*, 2020, pp. 6529–6533.

[72] Naoyuki Kanda, Shota Horiguchi, Ryoichi Takashima, Yusuke Fujita, Kenji Nagamatsu, and Shinji Watanabe, "Auxiliary Interference Speaker Loss for Target-Speaker Speech Recognition,", in *Proc. of Interspeech*, 2019, pp. 236–240.

[73] Takafumi Moriya, Hiroshi Sato, Tsubasa Ochiai, Marc Delcroix, and Takahiro Shinozaki, "Streaming Target-Speaker ASR with Neural Transducer,", in *Proc. of Interspeech*, 2022, pp. 2673–2677.

[74] Kateřina Žmolíková, Marc Delcroix, Keisuke Kinoshita, Takuya Higuchi, Atsunori Ogawa, and Tomohiro Nakatani, "Speaker-Aware Neural Network Based Beamformer for Speaker Extraction in Speech Mixtures,", in *Proc. of Interspeech*, 2017, pp. 2655–2659.

[75] Marc Delcroix, Katerina Zmolikova, Keisuke Kinoshita, Atsunori Ogawa, and Tomohiro Nakatani, "Single channel target speaker extraction and recognition with speaker beam,", in *Proc. of ICASSP*, 2018, pp. 5554–5558.

[76] Shaojin Ding, Quan Wang, Shuo-Yiin Chang, Li Wan, and Ignacio Lopez Moreno, "Personal VAD: Speaker-Conditioned Voice Activity Detection,", in *Proc. of Odyssey*, 2020, pp. 433–439.

[77] Yuxuan Wang, Maokui He, Shutong Niu, Lei Sun, Tian Gao, Xin Fang, Jia Pan, Jun Du, and Chin-Hui Lee, "USTC-NELSLIP system description for DIHARD-III challenge,", *arXiv preprint arXiv:2103.10661*, 2021.

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

[79] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur, "Deep Neural Network-based

Speaker Embeddings for End-to-end Speaker Verification,", in *Proc. of SLT*, 2016, pp. 165–170.

[80] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu, "Deep Speaker: an End-to-end Neural Speaker Embedding System,", *arXiv preprint arXiv:1705.02304*, 2017.

[81] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel, "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks,", in *Proc. of NeurIPS*, 2016, pp. 4905–4913.

[82] Sepp Hochreiter and Jürgen Schmidhuber, "Long Short-term Memory,", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[83] Mike Schuster and Kuldip K Paliwal, "Bidirectional Recurrent Neural Networks,", *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[84] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed, "Hybrid Speech Recognition with Deep Bidirectional LSTM,", in *Proc. of ASRU*, 2013, pp. 273–278.

[85] Yuchen Fan, Yao Qian, Feng-Long Xie, and Frank K Soong, "TTS Synthesis with Bidirectional LSTM Based Recurrent Neural Networks,", in *Proc. of Interspeech*, 2014, pp. 1964–1968.

[86] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is All You Need,", in *Proc. of NeurIPS*, 2017, pp. 5998–6008.

[87] Qiujia Li, Florian L Kreyssig, Chao Zhang, and Philip C Woodland, "Discriminative Neural Clustering for Speaker Diarisation,", in *Proc. of SLT*, 2021, pp. 574–581.

[88] Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman, "Voxceleb: Large-scale Speaker Verification in the Wild,", *Computer Speech & Language*, vol. 60, pp. 101027, 2020.

[89] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al., "The AMI Meeting Corpus: A Pre-announcement,", in *Proc. of MLMI*, 2005, pp. 28–39.

[90] Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al., "The ICSI Meeting Corpus,", in *Proc. of ICASSP*, 2003, pp. I–I.

[91] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: an ASR Corpus Based on Public Domain Audio Books,", in *Proc. of ICASSP*, 2015, pp. 5206–5210.

[92] John J Godfrey, Edward C Holliman, and Jane McDaniel, "SWITCHBOARD: Telephone Speech Corpus for Research and Development,", in *Proc. of ICASSP*, 1992, pp. 517–520.

[93] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus,", *arXiv:1510.08484*, 2015.

[94] Joon Son Chung, Jaesung Huh, Arsha Nagrani, Triantafyllos Afouras, and Andrew Zisserman, "Spot the Conversation: Speaker Diarisation in the Wild,", in *Proc. of Interspeech*, 2020, pp. 299–303.

[95] Neville Ryant, Prachi Singh, Venkat Krishnamohan, Rajat Varma, Kenneth Church, Christopher Cieri, Jun Du, Sriram Ganapathy, and Mark Liberman, "The third dihard diarization challenge,", *arXiv preprint arXiv:2012.01477*, 2020.

[96] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "Arcface: Additive Angular Margin Loss for Deep Face Recognition,", in *Proc. of CVPR*, 2019, pp. 4690–4699.

[97] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization,", in *Proc. of ICLR*, 2015.

[98] Yu-Xuan Wang, Jun Du, Maokui He, Shu-Tong Niu, Lei Sun, and Chin-Hui Lee, "Scenario-Dependent Speaker Diarization for DIHARD-III Challenge,", in *Proc. of Interspeech*, 2021, pp. 3106–3110.

[99] Yawen Xue, Shota Horiguchi, Yusuke Fujita, Yuki Takashima, Shinji Watanabe, Leibny Paola García Perera, and Kenji Nagamatsu, "Online Streaming End-to-End Neural Diarization Handling Overlapping Speech and Flexible Numbers of Speakers,", in *Proc. of Interspeech 2021*, 2021, pp. 3116–3120.

[100] Shota Horiguchi, Shinji Watanabe, Paola García, Yuki Takashima, and Yohei 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.

[101] Neville Ryant, Mark Liberman, and Jiahong Yuan, "Speech activity detection on youtube using deep neural networks.,", in *Proc. of Interspeech*. Lyon, France, 2013, pp. 728–731.

[102] Gregory Sell, David Snyder, Alan McCree, Daniel Garcia-Romero, Jesús Villalba, Matthew Maciejewski, Vimal Manohar, Najim Dehak, Daniel Povey, Shinji Watanabe, et al., "Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge.,", in *Proc. of Interspeech*, 2018, pp. 2808–2812.

[103] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, "Deep Neural Network Embeddings for Text-Independent Speaker Verification,", in *Proc. of Interspeech*, 2017, pp. 999–1003.

[104] Mireia Diez, Lukáš Burget, Federico Landini, Shuai Wang, and Honza Černockỳ, "Optimizing bayesian hmm based x-vector clustering for the second dihard speech diarization challenge,", in *Proc. of ICASSP*, 2020, pp. 6519–6523.

[105] Hakan Erdogan, John R Hershey, Shinji Watanabe, and Jonathan Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks,", in *Proc. of ICASSP*, 2015, pp. 708–712.

[106] Kofi Boakye, Beatriz Trueba-Hornero, Oriol Vinyals, and Gerald Friedland, "Overlapped speech detection for improved speaker diarization in multiparty meetings,", in *Proc. of ICASSP*, 2008, pp. 4353–4356.

[107] Mireia Diez, Lukáš Burget, Federico Landini, and Jan Černockỳ, "Analysis of speaker diarization based on bayesian hmm with eigenvoice priors,", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 355–368, 2019.

[108] Zili Huang, Shinji Watanabe, Yusuke Fujita, Paola García, Yiwen Shao, Daniel Povey, and Sanjeev Khudanpur, "Speaker diarization with region proposal network,", in *Proc. of ICASSP*, 2020, pp. 6514–6518.

[109] Juan M Coria, Hervé Bredin, Sahar Ghannay, and Sophie Rosset, "Overlap-aware low-latency online speaker diarization based on end-to-end local segmentation,", in *Proc. of ASRU*, 2021, pp. 1139–1146.

[110] Wei Xia, Han Lu, Quan Wang, Anshuman Tripathi, Yiling Huang, Ignacio Lopez Moreno, and Hasim Sak, "Turn-to-diarize: Online speaker diarization constrained by transformer transducer speaker turn detection,", in *Proc. of ICASSP*, 2022, pp. 8077–8081.

[111] Enrico Fini and Alessio Brutti, "Supervised online diarization with sample mean loss for multi-domain data,", in *Proc. of ICASSP*. IEEE, 2020, pp. 7134–7138.

[112] Yi Chieh Liu, Eunjung Han, Chul Lee, and Andreas Stolcke, "End-to-End Neural Diarization: From Transformer to Conformer,", in *Proc. of Interspeech*, 2021, pp. 3081–3085.

[113] Shota Horiguchi, Shinji Watanabe, Paola Garcia, Yawen Xue, Yuki Takashima, and Yohei Kawaguchi, "Towards neural diarization for unlimited numbers of speakers using global and local attractors,", in *Proc. of ASRU*, 2021, pp. 98–105.

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

[115] Keisuke Kinoshita, Marc Delcroix, and Naohiro Tawara, "Advances in Integration of End-to-End Neural and Clustering-Based Diarization for Real Conversational Speech,", in *Proc. of Interspeech*, 2021, pp. 3565–3569.

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

[117] Hervé Bredin and Antoine Laurent, "End-To-End Speaker Segmentation for Overlap-Aware Resegmentation,", in *Proc. of Interspeech*, 2021, pp. 3111–3115.

[118] Shota Horiguchi, Paola García, Yusuke Fujita, Shinji Watanabe, and Kenji Nagamatsu, "End-to-end Speaker Diarization as Post-processing,", in *Proc. of ICASSP*, 2021, pp. 7188–7192.

[119] Tingle Li, Qingjian Lin, Yuanyuan Bao, and Ming Li, "Atss-Net: Target Speaker Separation via Attention-Based Neural Network,", in *Proc. of Interspeech*, 2020, pp. 1411–1415.

[120] Chin-Yi Cheng, Hung-Shin Lee, Yu Tsao, and Hsin-Min Wang, "Multi-target extractor and detector for unknown-number speaker diarization,", *IEEE Signal Processing Letters*, 2023.

[121] Dongmei Wang, Xiong Xiao, Naoyuki Kanda, Takuya Yoshioka, and Jian Wu, "Target speaker voice activity detection with transformers and its integration with end-to-end neural diarization,", in *Proc. of ICASSP*, 2023, pp. 1–5.

[122] Danwei Cai, Weicheng Cai, and Ming Li, "Within-sample variability-invariant loss for robust speaker recognition under noisy environments,", in *Proc. of ICASSP*, 2020, pp. 6469–6473.

[123] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification,", in *Proc. of Interspeech*, 2020, pp. 3830–3834.

[124] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda, "Attentive Statistics Pooling for Deep Speaker Embedding,", in *Proc. of Interspeech*, 2018, pp. 2252–2256.

[125] Minh Tri Ho, Jinyoung Lee, Bong-Ki Lee, Dong Hoon Yi, and Hong-Goo Kang, "A Cross-Channel Attention-Based Wave-U-Net for Multi-Channel Speech Enhancement,", in *Proc. of Interspeech*, 2020, pp. 4049–4053.

[126] Dongmei Wang, Zhuo Chen, and Takuya Yoshioka, "Neural Speech Separation Using Spatially Distributed Microphones,", in *Proc. of Interspeech*, 2020, pp. 339–343.

[127] Dongmei Wang, Takuya Yoshioka, Zhuo Chen, Xiaofei Wang, Tianyan Zhou, and Zhong Meng, "Continuous speech separation with ad hoc microphone arrays,", in *Proc. of EUSIPCO*, 2021, pp. 1100–1104.

[128] Feng-Ju Chang, Martin Radfar, Athanasios Mouchtaris, and Maurizio Omologo, "Multi-Channel Transformer Transducer for Speech Recognition,", in *Proc. of Interspeech*, 2021, pp. 296–300.

[129] Shota Horiguchi, Yuki Takashima, Paola Garcia, Shinji Watanabe, and Yohei Kawaguchi, "Multi-channel end-to-end neural diarization with distributed microphones,", in *Proc. of ICASSP*, 2022, pp. 7332–7336.

[130] Fan Yu, Shiliang Zhang, Yihui Fu, Lei Xie, Siqi Zheng, Zhihao Du, Weilong Huang, Pengcheng Guo, Zhijie Yan, Bin Ma, et al., "M2met: The icassp 2022 multi-channel multi-party meeting transcription challenge,", in *Proc. of ICASSP*, 2022, pp. 6167–6171.

[131] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition,", in *Proc. of ICASSP*, 2017, pp. 5220–5224.

[132] Weiqing Wang, Danwei Cai, Xiaoyi Qin, and Ming Li, "The dku-dukeece systems for voxceleb speaker recognition challenge 2020,", *arXiv preprint arXiv:2010.12731*, 2020.

[133] Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, et al., "Nemo: a toolkit for building ai applications using neural modules,", *arXiv preprint arXiv:1909.09577*, 2019.

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

[135] Weiqing Wang, Xiaoyi Qin, and Ming Li, "Cross-channel attention-based target speaker voice activity detection: Experimental results for the m2met challenge,", in *Proc. of ICASSP*. IEEE, 2022, pp. 9171–9175.

[136] Yanyan Yue, Jun Du, Mao-Kui He, YuTing Yeung, and Renyu Wang, "Online Speaker Diarization with Core Samples Selection,", in *Proc. of Interspeech*, 2022, pp. 1466–1470.

[137] Tianyan Zhou, Yong Zhao, Jinyu Li, Yifan Gong, and Jian Wu, "Cnn with phonetic attention for text-independent speaker verification,", in *Proc. of ASRU*, 2019, pp. 718–725.

[138] Shuai Wang, Johan Rohdin, Lukáš Burget, Oldřich Plchot, Yanmin Qian, Kai Yu, and Jan Černocký, "On the Usage of Phonetic Information for Text-Independent Speaker Embedding Extraction,", in *Proc. of Interspeech*, 2019, pp. 1148–1152.

[139] Ming Li, Lun Liu, Weicheng Cai, and Wenbo Liu, "Generalized i-vector representation with phonetic tokenizations and tandem features for both text independent and text dependent speaker verification,", *Journal of Signal Processing Systems*, vol. 82, pp. 207–215, 2016.

[140] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network,", in *Proc. of ICASSP*, 2014, pp. 1695–1699.

[141] Jing Huang, Etienne Marcheret, Karthik Visweswariah, and Gerasimos Potamianos, "The ibm rt07 evaluation systems for speaker diarization on lecture meetings,", in *International Evaluation Workshop on Rich Transcription*, 2007, pp. 497–508.

[142] Nikolaos Flemotomos, Panayiotis Georgiou, and Shrikanth Narayanan, "Linguistically aided speaker diarization using speaker role information,", in *Proc. of Odyssey*, 2020, pp. 117–124.

[143] Tae Jin Park and Panayiotis Georgiou, "Multimodal speaker segmentation and diarization using lexical and acoustic cues via sequence to sequence neural networks,", in *Proc. of Interspeech*, 2018, pp. 1373–1377.

[144] Huanru Henry Mao, Shuyang Li, Julian McAuley, and Garrison W. Cottrell, "Speech Recognition and Multi-Speaker Diarization of Long Conversations,", in *Proc. of Interspeech*, 2020, pp. 691–695.

[145] Naoyuki Kanda, Zhong Meng, Liang Lu, Yashesh Gaur, Xiaofei Wang, Zhuo Chen, and Takuya Yoshioka, "Minimum bayes risk training for end-to-end speaker-attributed asr,", in *Proc. of ICASSP*, 2021, pp. 6503–6507.

[146] Danwei Cai, Weiqing Wang, Ming Li, Rui Xia, and Chuanzeng Huang, "Pretraining conformer with asr for speaker verification,", in *Proc. of ICASSP*, 2023, pp. 1–5.

[147] Weiqing Wang and Ming Li, "Incorporating end-to-end framework into target-speaker voice activity detection,", in *Proc. of ICASSP*, 2022, pp. 8362–8366.

[148] Shota Horiguchi, Nelson Yalta, Paola Garcia, Yuki Takashima, Yawen Xue, Desh Raj, Zili Huang, Yusuke Fujita, Shinji Watanabe, and Sanjeev 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.

[149] Mao-Kui He, Jun Du, Qing-Feng Liu, and Chin-Hui Lee, "Ansd-ma-mse: Adaptive neural speaker diarization using memory-aware multi-speaker embedding,", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[150] Naoyuki Kanda, Xiong Xiao, Yashesh Gaur, Xiaofei Wang, Zhong Meng, Zhuo Chen, and Takuya Yoshioka, "Transcribe-to-diarize: Neural speaker diarization for unlimited number of speakers using end-to-end speaker-attributed asr,", in *Proc. of ICASSP*, 2022, pp. 8082–8086.

[151] Tae Jin Park, Nithin Rao Koluguri, Jagadeesh Balam, and Boris Ginsburg, "Multi-scale Speaker Diarization with Dynamic Scale Weighting,", in *Proc. of Interspeech*, 2022, pp. 5080–5084.

Biography

Weiqing Wang received a B.S.(2018) in Computer Science from Sun Yat-sen University, and he is currently a Ph.D. candidate in Electrical and Computer Engineering at Duke University, advised by Prof. Ming Li and Prof. Xin Li. His research focuses on speaker verification, speech recognition and speaker diarization. Before joining Duke, he was a research assistant at Duke Kunshan University (2018-2019), and his research mainly focused on automatic piano transcription.

Weiqing Wang has published multiple papers at IEEE/ACM TASLP, ICASSP, and InterSpeech. In addition, he has achieved the 1st place in many international competitions on speaker diarization tasks, including VoxSRC 2021 [9], VoxSRC 2022 [10], VoxSRC 2023 [11] and M2MeT challenge [130].