Study on overlap-aware speaker diarization and its applications

March 2023

Shota Horiguchi

Study on overlap-aware speaker diarization and its applications

Graduate School of Science and Technology
Degree Programs in Systems and Information Engineering
University of Tsukuba

March 2023

Shota Horiguchi

Abstract

Speech is a fundamental communication tool for human beings forming a society. Automatic speech recognition (ASR) is the most representative and practical application for speech, but at present, it is mainly used in situations where there is a single speaker or where each speaker has a separate channel like video meeting applications. However, considering that speech is a communication tool, there is a high demand for ASR in situations where speech from multiple speakers is mixed and observed. In the presence of multiple speakers, it is important to determine not only what was spoken, but also who spoke it and when. The task of estimating who spoke and when from speech, which is the focus of this thesis, is called speaker diarization.

In multi-speaker conversations, several speakers inevitably speak at the same time. It is known that the ratio of speaker overlaps of speech in most meetings is 10–20 %, and at dinner parties, it can be as high as 50 %. Nevertheless, speaker diarization studies have long adopted cascaded approaches which assign each time frame for one of the speakers and thus ignore speaker overlaps. On the other hand, end-to-end approaches that can capture speaker overlaps have emerged in recent years, but many challenges remain from the aspect of practicality. This thesis aims to develop overlap-aware speaker diarization methods that can be applied in such realistic multi-speaker conditions and their applications for multi-speaker ASR.

The first part of this thesis addresses the practicality of overlap-aware speaker diarization based on end-to-end approaches. In Chapter 3, we propose overlap-aware speaker diarization for unknown numbers of speakers, whereas conventional methods require that the number of speakers is given in advance. First, we propose a method to estimate the number of speakers present in input and their corresponding representations; then, we achieve speaker diarization of unknown numbers of speakers based on the representations. We also propose a buffering method that enables online speaker diarization for unknown numbers of speakers. In Chapter 4, we propose more accurate overlap-aware speaker diarization by utilizing spatial information obtained from multi-channel signals, whereas the con-

ventional method is based on single-channel signals. We also propose a method to improve the performance of both single- and multi-channel models by mutually training them. In Chapter 5, we propose to use an end-to-end speaker diarization model to detect speaker overlaps and assign appropriate speakers by post-processing the result of diarization ignoring speaker overlaps. The effectiveness of these methods was verified using both large-scale simulated mixtures and real multi-speaker conversational datasets in various domains.

In the second part, we propose applications that use overlap-aware speaker diarization and demonstrate that it is also an important step for later procedures. In Chapter 6, we propose a system for transcribing meetings from audio recorded by distributed microphones, using a chain of speaker diarization, speech separation, and ASR. We validate the usefulness of the system by using a newly recorded meeting dataset and demonstrate the importance of highly accurate overlap-aware speaker diarization. In Chapter 7, we extend the speech separation algorithm based on the results of speaker diarization, which is also used in the system above, to enable online processing. We experimentally show that the proposed method significantly speeds up the conventional method sufficiently to enable online operation.

Acknowledgments

I would like to express my deepest gratitude to all those who supported and advised me during the research activities related to this thesis.

First, I would like to express my sincere gratitude to my advisor, Prof. Takeshi Yamada at the University of Tsukuba. He kindly accepted my request to supervise me, a complete stranger. Although it was only one year, he gave me a lot of support while taking into consideration the fact that I was a working student to the maximum extent.

I am also grateful to Prof. Shoji Makino at Waseda University and my doctoral committee members at the University of Tsukuba, Prof. Kazuhiro Fukui, Prof. Keisuke Kameyama, Prof. Hiroyuki Kudo, Prof. Naoto Wakatsuki, and Prof. Hotaka Takizawa for their valuable advice in the completion of my doctoral thesis.

I would also like to express our gratitude to the Hitachi and ex-Hitachi members who worked with me in our research activities. I especially thank Dr. Naoyuki Kanda (currently at Microsoft Research) for inviting me to the world of speech processing when I joined Hitachi. It was a great opportunity for me that I could have started my career as a speech researcher with a distinguished researcher like him. I would also like to thank Mr. Yusuke Fujita (currently at LINE corporation) for our collaboration. All of the contents in this thesis are inspired by his papers on speaker diarization, and I could not have achieved so much without him. I would also like to express my appreciation to my successive managers, Dr. Kenji Nagamatsu and Dr. Yohei Kawaguchi. They took special care of my plans to pursue a Ph.D. degree right after I joined Hitachi and encouraged me to attend university.

I would sincerely like to thank my research collaborators, Prof. Shinji Watanabe at Carnegie Mellon University and Dr. Paola Garcia at Johns Hopkins University for their fruitful advice on regular meetings and support when submitting papers. Most of the papers on which this thesis is based were co-authored with them and would not have been possible without them. Many papers have been produced and presented at various international conferences during our journey together.

Unfortunately, we could not meet face-to-face most of the time, since many of them were held online due to COVID-19. I hope that I will have many more opportunities to meet them in person as I continue my research activity.

I would like to express my gratitude to Prof. Kiyoharu Aizawa at the University of Tokyo for supervising me even after I graduated from the master's program. Two journal papers published under his supervision really helped me for continuing my career as a researcher.

I also thank all the members of the multimedia laboratory at the University of Tsukuba. At the company, I had many non-research-related duties, so there were many periods when I was unable to conduct research activities, but I found solace in listening to their research progress at the weekly speech team meetings.

Finally, I want to thank my family for their support and understanding.

Contents

Abstract i

Acknowledgements iii

Contents v

List of Figures ix

List of Tables xi

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Research History . . . 3
- 1.3 Approaches . . . 4
  - 1.3.1 Speaker Diarization Methods . . . 4
  - 1.3.2 Applications of Speaker Diarization . . . 5
- 1.4 Thesis Overview . . . 6

2 Review of Speaker Diarization 7
- 2.1 Problem Formulation . . . 7
- 2.2 Cascaded Approach . . . 8
  - 2.2.1 Problem Formulation . . . 8
  - 2.2.2 Implementation . . . 8
  - 2.2.3 Variants . . . 10
- 2.3 End-to-End Approach . . . 11
  - 2.3.1 Overview . . . 11
  - 2.3.2 End-to-End Neural Diarization . . . 12
  - 2.3.3 Online Speaker Diarization . . . 15
- 2.4 Speech Separation Based on Speaker Diarization Results . . . 18
- 2.5 Evaluation Protocols . . . 21

3 End-to-End Speaker Diarization for Unknown Numbers of Speakers 23
- 3.1 Introduction . . . 23
- 3.2 End-to-End Speaker Diarization for Flexible Numbers of Speakers . . . 25

3.2.1 Related Work . . . 25
3.2.2 Attractor-Based End-to-End Neural Diarization . . . 26
3.2.3 Inference Methodology . . . 30
3.2.4 Experimental Settings . . . 34
3.2.5 Results of Fixed-Numbers-of-Speakers Experiments . . . 35
3.2.6 Results of Unknown-Numbers-of-Speakers Experiments . . . 40
3.2.7 Analysis . . . 45
3.3 End-to-End Speaker Diarization for Unlimited Numbers of Speakers 47
3.3.1 Related Work . . . 48
3.3.2 End-to-End Neural Diarization with Global and Local Attractors . . . 48
3.3.3 Experimental Settings . . . 53
3.3.4 Results . . . 54
3.4 Block-Online Speaker Diarization for Unlimited Numbers of Speakers 57
3.4.1 Block-Wise Speaker-Tracing Buffer . . . 58
3.4.2 Speaker-Balanced Sampling Probabilities . . . 60
3.4.3 Variable Chunk-Size Training via Minibatch Reshaping . . . 61
3.4.4 Experimental Settings . . . 61
3.4.5 Evaluation of the Variations of Speaker-Tracing Buffer . . . 62
3.4.6 Main Results . . . 63
3.4.7 Real-Time Factor . . . 66
3.5 Conclusion . . . 67

4 Multi-Channel End-to-End Speaker Diarization 69
4.1 Introduction . . . 69
4.2 Multi-Channel End-to-End Speaker Diarization . . . 70
4.2.1 Related Work . . . 70
4.2.2 Conventional Single-Channel EEND . . . 71
4.2.3 Multi-Channel EEND . . . 71
4.2.4 Experimental Settings . . . 74
4.2.5 Results . . . 75
4.3 Mutual Learning of Single-Channel and Multi-Channel End-to-End Speaker Diarization . . . 77
4.3.1 Introduction . . . 77
4.3.2 Related Work . . . 79
4.3.3 Proposed Method . . . 79
4.3.4 Experimental Settings . . . 83
4.3.5 Results . . . 85
4.4 Conclusion . . . 87

5 End-to-End Speaker Diarization as Post-Processing 89
5.1 Introduction . . . 89

5.2 EEND as Post-Processing . . . 90
5.2.1 Overview . . . 90
5.2.2 Algorithm . . . 92
5.2.3 Training Strategy of the EEND-EDA Model . . . 94
5.3 Experimental Settings . . . 94
5.4 Results . . . 95
5.4.1 Preliminary Evaluation of the Training Using Frame Selection 95
5.4.2 Evaluation of EENDasP . . . 96
5.5 Conclusion . . . 97
5.6 Acknowledgment . . . 98

6 Meeting Transcription with Distributed Microphones 99
6.1 Introduction . . . 99
6.2 Method . . . 100
6.2.1 Blind Synchronization . . . 102
6.2.2 Speaker Diarization . . . 102
6.2.3 Speech Separation . . . 104
6.2.4 Speech Recognition . . . 104
6.2.5 Duplication Reduction . . . 105
6.3 Results . . . 105
6.4 Conclusions . . . 107

7 Block-Online Guided Source Separation 109
7.1 Introduction . . . 109
7.2 Related Work . . . 110
7.3 Block-Online Guided Source Separation . . . 111
7.3.1 Overview . . . 111
7.3.2 Proposed Algorithm . . . 111
7.4 Experimental Settings . . . 115
7.5 Results . . . 116
7.6 Conclusion . . . 119

8 Conclusions 121
8.1 Contributions . . . 121
8.2 Future Directions . . . 122
8.2.1 Simulated Datasets for the Training of Diarization Models . . 122
8.2.2 Speaker Diarization in Wilder Conditions . . . 123
8.2.3 Tight Integration of Diarization and Subsequent Processes . . 123

Bibliography 125

A Datasets 143
A.1 Speaker Diarization Datasets . . . 143

A.1.1 Single-Channel Datasets 143
A.1.2 Multi-Channel Datasets 146
A.2 Speech Recognition Datasets 148
A.2.1 CHiME-6 Corpus 148
A.2.2 Meeting Corpus 149

B Publications **151**

List of Figures

1.1 Thesis overview 6

2.1 Transformer encoder. Each yellow area is skipped via a residual connection. $\otimes$ denotes matrix multiplication. 14

2.2 Online diarization using speaker-tracing buffer proposed in [1, 2]. . . 16

3.1 EEND with encoder-decoder-based attractor calculation (EEND-EDA). 27

3.2 Iterative inference in the case of $S_{\text{max}} = 3$. 33

3.3 Visualization of embedding and attractors within each recording. For conventional EEND, weights of last fully connected layer W_{cls} were visualized instead of attractors. 38

3.4 Visualization of attractors across recordings. Selected speakers' attractors are marked by dots, and their interference speakers' attractors are marked by crosses. Colors of crosses correspond to speaker identities within each figure. Each pair of attractors from same mixture are connected with gray line. 39

3.5 t-SNE visualization of frame-wise embeddings extracted from simulated 5-speaker mixtures (top) and 6-speaker mixtures (bottom). The EEND-EDA used for extraction was trained using {1,2,3,4}-speaker mixtures. Single-speaker frames are denoted by the dots with colors corresponding to the speaker identities and overlapped frames are denoted by the crosses in light gray. Frames of silence were excluded from the visualization. 47

3.6 End-to-end neural diarization with global and local attractors (EEND-GLA). The attractor existence losses are omitted from the illustration. 50

3.7 Online diarization using speaker-tracing buffer with block-wise update. 58

3.8 Batch creation in the VCT. 61

3.9 Frame-wise breakdown of diarization error on CALLHOME. 63

3.10 Real time factor of EEND-GLA-Small with BW-STB calculated using Sim5spk. The filled areas represent the standard deviations. The DERs are 16.95 % and 18.18 % with 10 frames/s and 20 frames/s conditions, respectively. 68

4.1 Multi-channel encoders. Each yellow area is skipped via residual connection. 72
4.2 VRAM usage during training with $T = 500$ and batch size of 64. . . . 77
4.3 Mutual learning of single- and multi-channel EEND. 78
4.4 The architectures of encoders used in this chapter. D: the dimensionality of embeddings, T: sequence length, C: the number of channels. 81

5.1 Diagram of EEND as post-processing when the number of speakers is three. It refines the diarization results of each pair of speakers iteratively. Given initial diarization results (top left), the proposed method (i) first determines the processing order on the basis of the number of frames and (ii) then refines the corresponding results of each pair using two-speaker EEND. 91

6.1 Overview of our meeting transcription system using asynchronous distributed microphones. 101

7.1 Case when almost same cACGMMs are obtained by conventional utterance-wise offline GSS algorithm. 113
7.2 WERs (%) on CHiME-6 development set with various block size L and pre-context size C. Decay strategy with $\eta = 0.9$ was used for proposed algorithm. 117

A.1 Recording environment of the meeting corpus. 11 smartphones, each equipped with a monaural microphone, were distributed on the table. 149

List of Tables

1.1 Overlap ratios of real multi-speaker datasets. 2

3.1 DERs (%) for two-speaker evaluations. 0.25 s of collar tolerance was allowed. 36

3.2 DERs (%) for three-speaker evaluations. 0.25 s of collar tolerance was allowed. 36

3.3 DERs on Sim2spk (overlap ratio: 34.4 %) using various types of sequences. 37

3.4 DERs (%) of cross evaluations of two- and three-speaker EEND-EDA. 0.25 s of collar tolerance was allowed. 40

3.5 Step-by-step improvement on simulated datasets. For Sim2spk and Sim3spk, we used $\beta = 2$ and $\beta = 5$, respectively. In $\mathcal{L}_{\text{exist}}$ column, we show which parameters were updated using $\mathcal{L}_{\text{exist}}$ during training. 40

3.6 DERs (%) of CALLHOME. 0.25 s of collar tolerance was allowed. TDNN-based x-vector results were obtained with Kaldi recipe. DERs of single-speaker regions are reported in brackets. AHC: agglomerative hierarchical clustering, VB: Variational Bayes resegmentation [3], VBx: Variational Bayes HMM clustering [4]. 42

3.7 Confusion matrices for speaker counting on CALLHOME Part 2. X-vector-based results were obtained with oracle SAD, while EEND-based results were obtained without external SAD. 43

3.8 DERs and JERs (%) for AMI headset mix. No collar tolerance was allowed. 44

3.9 DERs and JERs (%) for DIHARD II eval. No collar tolerance was allowed. 45

3.10 DERs and JERs (%) for DIHARD III eval. No collar tolerance was allowed. 46

3.11 Breakdown results of DIHARD III eval for each number of speakers with oracle speech segments. 46

3.12 DERs (%) on the simulated datasets with 0.25 s collar tolerance. . . . 55

3.13 Offline DERs (%) of EEND-GLA-Small with various training and inference strategies. Loss: the training objective used for training. Inference: attractors used during inference. 55
3.14 Confusion matrices for speaker counting on the simulated datasets. . 56
3.15 DERs (%) of various offline diarization methods on CALLHOME with 0.25 s collar tolerance. 56
3.16 DERs (%) on DIHARD II with no collar tolerance. 57
3.17 DERs (%) on DIHARD III with no collar tolerance. 57
3.18 Example of sampling weights determined by (2.36) and (3.31). 60
3.19 Step-by-step improvement in the online inference of EEND-EDA on the CALLHOME dataset. VCT: Variable chunk-size training. 62
3.20 DERs (%) on the simulated datasets with 0.25 s collar tolerance. Unless otherwise specified, each online system had an algorithmic latency of 1 s. 64
3.21 Confusion matrices for speaker counting on the simulated datasets. . 65
3.22 DERs (%) of various online diarization methods on CALLHOME with 0.25 s collar tolerance. Unless otherwise specified, each online system had an algorithmic latency of 1 s. 65
3.23 DERs (%) of vairous online diarization methods on DIHARD II with no collar tolerance. Each online system had an algorithmic latency of 1 s. 66
3.24 DERs (%) of various online diarization methods on DIHARD III with no collar tolerance. Unless otherwise specified, each online system had an algorithmic latency of 1 s. 67

4.1 DERs on Sim2spk-multi-eval and Sim2spk-multi-hybrid. 75
4.2 DERs on CSJ-multi-eval and CSJ-multi-dialog. 76
4.3 DERs (%) improvement on Sim2spk-multi-eval with the proposed method. The values in gray indicate the mismatched condition in the number of channels. The values in the parentheses are with median filtering. 86
4.4 DERs (%) on single-channel real conversational datasets. 86
4.5 DERs (%) on multi-channel real conversational datasets. 87

5.1 Ratio (%) of region where at least n speakers are active over all speech region of various real multi-speaker datasets. 92
5.2 DERs (%) on CALLHOME-2spk. Collar tolerance of 0.25 s is allowed. 95
5.3 DERs (%) on CALLHOME. All the results include overlapped regions and are NOT based on oracle SAD. Collar tolerance of 0.25 s is allowed. 96

5.4 DERs and JERs (%) on AMI eval. VB: Variational Bayes resegmentation, OVL: Overlap detection and speaker assignment [5]. All the results include overlapped regions and are NOT based on oracle SAD. No collar tolerance is allowed. 97
5.5 DERs and JERs (%) on DIHARD II eval. All the results include overlapped regions and are NOT based on oracle SAD. No collar torelance is allowed. 98

6.1 CERs (%) obtained using various microphone combinations. 106
6.2 CERs (%) obtained with various scaling factors λ. 107
6.3 Ablation study using 11 microphones. 107

7.1 Parameters used in online experiments. 116
7.2 WERs (%) on CHiME-6 development set. 117
7.3 CERs (%) on the meeting corpus recorded using asynchronous distributed microphones. The block size L and context size C were set to 150 for proposed algorithm. 118
7.4 Execution times on CHiME-6 development set. Mean and standard deviation among 10 trials are shown. The block size L and the precontext size C were set to 150, which corresponds to 2.4 s. 119

A.1 Statistics of single-channel simulated speaker diarization corpora. . . 145
A.2 Statistics of single-channel real-recorded speaker diarization datasets. 146
A.3 Recording environment of CSJ-multi-train, CSJ-multi-eval, and CSJ-multi-dialog. 147
A.4 Statistics of multi-channel speaker diarization corpora. 148
A.5 Statistics of speech recognition datasets. 148

Chapter 1

Introduction

1.1 Background

Speech communication is an essential tool for social activities such as information exchange, decision-making, and consensus building. Therefore, automatic speech recognition (ASR) is an important technology for assisting such social activities. For example, when combined with speech translation, it can realize communication that surpasses language barriers. When combined with dialogue systems or text mining, it can replace human operators, thereby solving social problems such as labor shortages. ASR can also support decision-making and consensus-building processes through automatic minuting and documentation and can improve well-being by freeing people from such simple labor.

In recent years, ASR under somewhat controlled conditions has improved dramatically and reached a practical level. Single-speaker ASR, such as voice input in smartphone/tablet devices and smart speakers, has already been used by a wide range of customers. Even when multiple speakers exist, video meeting applications (such as Zoom and Microsoft Teams) can perform highly accurate ASR because each speaker has a separate channel.

The remaining challenge is the cocktail party problem, ASR in situations where multiple speakers may be speaking simultaneously. The aforementioned use cases of ASR in social activities fall into these situations, which means the importance of multi-speaker ASR. For multi-speaker ASR, it is necessary to determine not only what was spoken but also when and by whom it was spoken. The latter task, determining who spoke when is called *speaker diarization*. With speaker diarization, ASR results can be treated as a transcription of a conversation, rather than a mere transcribed speech.

Table 1.1: Overlap ratios of real multi-speaker datasets.

Dataset	# of speakers	Overlap ratio
CALLHOME [6]	2–7	16.9 %
AMI [7]	3–5	19.4 %
ICSI [8]	6 (ave.)	18.6 %
CHiME-6 [9]	4	33.9 %
Internal meeting [10]	N/A	14.7 %
Internal meeting [11]	4/6	16.3 %/16.0 %
Internal meeting [12]	5–8	13.2 %

Since distinguishing the mixed speech of multiple speakers is essential for solving the cocktail party problem, it is important to identify speaker overlap in speech. Table 1.1 summarizes the overlap ratio of various real multi-speaker datasets. Various multi-talker meeting corpora show over 10 % of overlap ratio [13], and it is raised to over 40 % in the dinner party scenarios in CHiME-5/6 [14, 15]. Assuming the ASR system that transcribes each speech segment extracted in the speaker diarization step. If a speaker diarization method has no way to deal with speaker overlap, at least one of the overlapped segments will not appear in the transcription; therefore, the overlap ratio directly becomes an error in ASR. If speaker overlaps can be correctly captured, it is possible to accurately remove the interference speaker's speech from each utterance in the subsequence speech separation step [16, 17], thereby reducing the effect of speaker overlaps in the last ASR step.

Since speech separation makes the speech of each speaker not include the speech of other speakers, the separated speech itself is also useful for solving speaker diarization. There are also approaches such as speech-separation-guided speaker diarization [18] and joint modeling of speaker diarization and speech separation [19]. However, machine-learning-based speech separation, which has been studied widely in recent years, mostly uses the oracle-separated results or a time-frequency mask calculated from them as the target of training, making it difficult to use real multi-speaker recordings for training. In addition, despite the difficulty of using real data for training, it has also been reported that speech separation models are more sensitive to domain mismatch than diarization models [18]. For these reasons, the diarization approach alone is important.

1.2 Research History

The conventional approach for speaker diarization is based on a stack of multiple modules:

1. Speech activity detection: detect the interval in which at least one speaker is speaking,

2. Segmentation: divide each detected interval into short segments by sliding window or speaker change detection,

3. Speaker embedding extraction: extract a feature vector that represents speaker characteristics from each segment,

4. Clustering: group the extracted embeddings based on their similarity, and determine that the same speaker is speaking if the embeddings belong to the same cluster, and different speakers are speaking if they belong to different clusters,

5. (Optional) overlap handling: detect the interval in which two or more speakers are speaking and assign the appropriate second speaker for each interval.

Again, to solve the cocktail party problem, it is important to identify the speaker overlap in speech (i.e., the last step above). Despite this, notably, it has been often neglected in studies of speaker diarization for a long time from their methodologies and even evaluations [20, 3, 4, 21, 22] (The most popular diarization evaluation script `mdeval.pl` can easily realize such forgiveness by using the option "-1"). This is because it treats speaker diarization as a set-partitioning problem with the adoption of the clustering step, and treats overlap detection and speaker assignment as optional.

In contrast, the end-to-end approach for speaker diarization that emerged in 2019 treats speaker diarization as a multi-label classification problem [23]. It classifies each time frame into speech or silence for each speaker; thus, it naturally handles speaker overlap. It brought a paradigm shift to the evaluation of speaker diarization; now it becomes popular to include speaker overlaps in evaluation to see the real performance of diarization systems. In addition, the end-to-end approach has several advantages such as 1) easy implementation and 2) easy optimization as an entire diarization system (including domain adaptation).

1.3 Approaches

1.3.1 Speaker Diarization Methods

As explained in the previous section, the conventional cascaded approach has difficulty with overlap handling and the end-to-end method is a promising solution for the problem. However, the early type of end-to-end [23, 24] or hybrid methods [25] have some practical issues and room for improvement. We first focus on improving the practicality of end-to-end speaker diarization in the following perspectives.

Numbers of speakers

In most cases where we need speaker diarization, the number of speakers is not known a priori, *e.g.*, discussions in meetings or congress, doctor-patient conversations at a hospital, trials at a court, and talk on TV. However, the early types fixed the number of speakers by their network architectures, which clearly limits the range of applications of the diarization methods. To solve this limitation, we tackle end-to-end speaker diarization for unknown numbers of speakers. We first propose encoder-decoder-based attractor calculation (EDA), with which flexible numbers of speaker-wise representative vectors, or *attractors*, are calculated from frame-wise embeddings. Then, we use it with end-to-end speaker diarization to deal with flexible numbers of speakers, namely, EEND-EDA.

While EEND-EDA can handle flexible numbers of speakers, the maximum number is still capped by the dataset used for the model training. To remedy this problem, we further extend the method for unlimited numbers of speakers by introducing an unsupervised clustering step. In addition to attractors calculated from an entire embedding sequence (*i.e.*, glocal attractors), those calculated from each chunked short sequence (*i.e.*, local attractors) were utilized; then, the local attractors were clustered to find the optimal inter-chunk correspondence. This EEND with global and local attractors, or EEND-GLA, does not have limitations in the output number of speakers because of the unsupervised clustering step.

Online processing

One of the advantages of speech communication over text communication is the speed of information transfer. Therefore, some speech applications require real-time transcription, *e.g.*, spoken dialogue systems and simultaneous translation, and thus speaker diarization also has to be operated in an online manner. We propose

a method to enable online inference of EEND-GLA based on the buffer that is updated by block-wise sampling and first-in-first-out manners.

Multi-channel processing

While end-to-end speaker diarization is mainly tackled as a single-channel problem in the literature, utilizing spatial information from multi-channel inputs in addition to spectral information is a promising way of solving the cocktail party problem. Actually, the diarization-guided speech separation [16] takes multi-channel signals as input, and the grand challenges of multi-speaker ASR are mostly based on multi-channel [15, 26]. Since multi-channel processing based on distributed microphones is attracting attention in recent years [27, 28, 15, 29], we follow this direction; we propose a multi-channel end-to-end speaker diarization method that is based on distributed microphones. We also propose a method to further improve the multi-channel model along with a single-channel model by mutually training them using knowledge distillation and finetuning.

End-to-end speaker diarization for overlap detection and speaker assignment

Indeed end-to-end speaker diarization is a promising approach for overlap-aware speaker diarization, cascaded approaches are still competitive. Practically, multiple diarization systems may be ensembled to improve accuracy, and since it is important to gather diverse varieties of systems in an ensemble, a combination of end-to-end and cascaded approaches is effective [90, 67]. However, cascaded approaches require an extra module for overlap detection and speaker assignment. We, therefore, propose a method to utilize an end-to-end speaker diarization model for this purpose, namely, EEND as post-processing.

1.3.2 Applications of Speaker Diarization

Since speaker diarization is often placed as a preprocess of multi-speaker ASR, it is important to demonstrate how an accurate speaker diarization method can benefit it. In this thesis, we propose two diarization-related applications. The first one is a speaker-diarization-driven meeting transcription system based on distributed microphones, in which speech separation is conducted using speaker diarization results [16]. However, the speech separation algorithm requires a large amount of execution time, which inhibits the system from online processing. As the second one, we propose a block-online algorithm of the speech separation method to improve its inference speed.

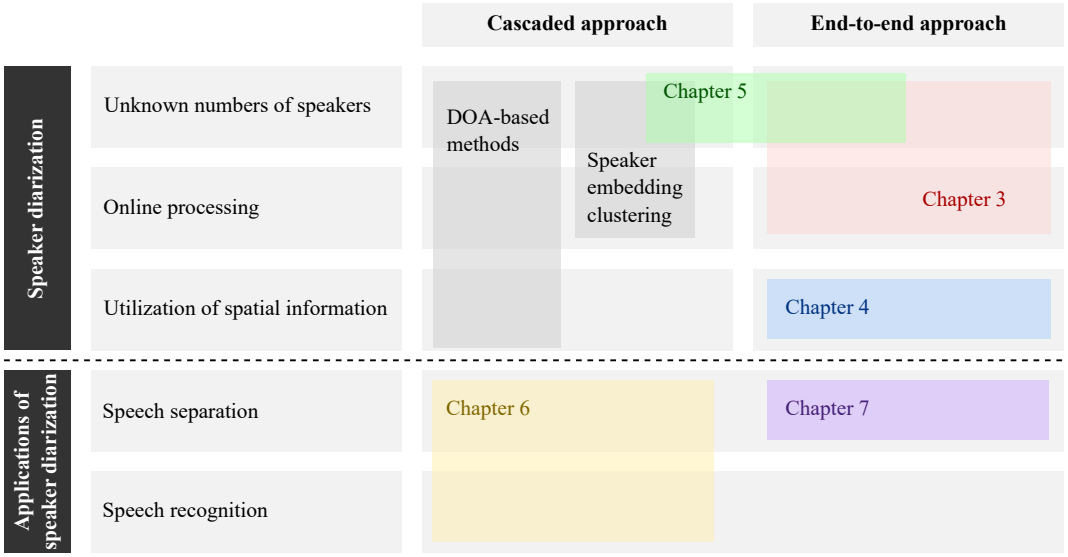


Figure 1.1: Thesis overview

1.4 Thesis Overview

The organization of this thesis is as follows, which is also illustrated in Figure 1.1. Chapter 2 gives a review of speaker diarization, including conventional cascaded approaches and emerging end-to-end approaches. Chapter 3 addresses end-to-end speaker diarization for unknown numbers of speakers and its extension for online processing. Chapter 4 addresses multi-channel end-to-end speaker diarization. Chapter 5 addresses a method to utilize an end-to-end speaker diarization model for overlap detection and speaker assignment to update a result from a cascaded speaker diarization method. Chapter 6 addresses a meeting transcription based on diarization-separation-recognition chain. Chapter 7 addresses block-online speech separation conditioned on speaker diarization results. Chapter 8 concludes this thesis with some future directions of end-to-end speaker diarization.

Chapter 2

Review of Speaker Diarization

This chapter reviews speaker diarization research in the literature. Section 2.1 provides a strict formulation of speaker diarization. Section 2.2 summarizes cascaded approaches for speaker diarization Section 2.3 summarizes end-to-end approaches for speaker diarization. Section 2.4 introduces a speech separation method that utilizes speaker diarization results. Section 2.5 defines the metrics to evaluate speaker diarization systems and their applications.

2.1 Problem Formulation

Speaker diarization is a task to estimate speech activities for each speaker appearing in an input recording. Different from speaker recognition or speaker identification, it does not require estimating the identity of the speakers, but only requires distinguishing all the speakers in the input. Given F-dimensional T-length acoustic features $X = (\boldsymbol{x}_t)_{t=1}^{T} \in \mathbb{R}^{F \times T}$, where $t \in \{1, \dots, T\}$ is a frame index, speaker diarization is formulated as the process of determining the most plausible speaker activities as

$$\hat{Y} = \arg\max_{Y} P\left(Y \mid X\right), \tag{2.1}$$

where $Y = (\boldsymbol{y}_t)_{t=1}^{T} \in \{0, 1\}^{S \times T}$ is S speakers' speech activities defined as follows:

$$y_{s,t} = \begin{cases} 0 & \text{(Speaker } s \text{ is inactive at } t\text{)} \\ 1 & \text{(Speaker } s \text{ is active at } t\text{)} \end{cases}. \tag{2.2}$$

Note that the number of speakers S can be an estimated value or known in advance.

2.2 Cascaded Approach

This section provides an overview of conventional cascaded approaches (or modular-based approaches / pipeline approaches / clustering-based approaches). Section 2.2.1 introduces a typical formulation of speaker diarization used in most cascaded approaches. Section 2.2.2 describes the typical implementation of cascaded approaches for speaker diarization. Section 2.2.3 describes some variants of cascaded approaches.

2.2.1 Problem Formulation

Most cascaded approaches assume that only one speaker is active in each frame. The simplified formulation below is used instead of (2.1):

$$[\hat{y}_1, \ldots, \hat{y}_T] = \underset{y_1,\ldots,y_T}{\arg\max}\, P\left(y_1, \ldots, y_T \mid X\right), \tag{2.3}$$

where $y_t \in \{0, \ldots, S\}$ denotes silence $(y_t = 0)$ or the index of the active speaker $(1 \leq y_t \leq S)$ at t.

2.2.2 Implementation

A cascaded approach typically consists of a stack of the following four modules: 1) speech activity detection, 2) segmentation, 3) speaker embedding extraction, and 4) embedding clustering. To estimate the most possible label sequence $y_1 \ldots, y_T$ with the stacked modules, the probability $P\left(y_1, \ldots, y_t \mid X\right)$ is rewritten as the product of conditional probabilities by using chain rule as

$$P\left(y_1, \ldots, y_T \mid X\right) = P_{\text{SAD}}\left(z_1, \ldots, z_T \mid X\right) P_{\text{clst}}\left(y_1, \ldots, y_T \mid X, z_1, \ldots, z_T\right), \tag{2.4}$$

where $z_t \in \{0, 1\}$ is one if at least one speaker is active at t and zero otherwise.

Speech activity detection

In speech activity detection (SAD), or voice activity detection (VAD), intervals in which at least one speaker is speaking are estimated by finding a sequence $z_1, \ldots, z_T$ that maximizes the probability $P_{\text{SAD}}\left(z_1, \ldots, z_T \mid X\right)$ in (2.4). Only the acoustic features of the frames in which speech activity is detected are passed to the following steps.

Segmentation

In this step, each interval obtained with SAD is divided into short segments. A commonly used method is to divide every interval into fixed-length segments using a sliding window (*e.g.*, window width of 1.5 seconds, shift length of 0.75 seconds, etc.). There is a trade-off in the window length; if we increase the window length, we can obtain high-fidelity speaker embeddings in the next *speaker embedding extraction* step, but it results in low temporal resolution. To ease this problem, some methods investigate extracting speaker embeddings from multi-scale windows [30, 31] or improving the quality of speaker embeddings extracted from short segments [32]. Another method is to detect speaker changes and divide each interval at the point of detection [33]. This makes it possible to extract speaker features from as long a segment of speech as possible, depending on the situation, but it is cumbersome in that it requires one more module.

Speaker embedding extraction

In this step, an embedding that represents speaker characteristics is extracted from each segment obtained in the previous *segmentation* step. The earliest successful speaker embedding is i-vector [34], which is obtained as a Gaussian mixture model supervector, and showed a remarkable performance on speaker diarization [35, 36, 37, 38]. Recent methods are mostly based on speaker embeddings extracted from deep neural networks such as x-vectors [39, 40, 41] and d-vectors [20, 21]. The extractors of these embeddings differ slightly in the treatment of input features (variable-length vs. fixed-length) and the use of statistics pooling, but are basically the same in that they are both trained as multi-speaker classifiers and use features near the last layer as speaker embeddings.

Embedding clustering

In this step, the extracted speaker embeddings are grouped by using a clustering algorithm to get $y_1 \dots y_T$ that maximize the probability $P_{\text{clst}}(y_1, \dots, y_T \mid X, z_1, \dots, z_T)$ in (2.4). It is assumed that the same speaker is speaking if a pair of embeddings belong to the same cluster, and different speakers are speaking if they belong to different clusters. Earlier studies used traditional clustering algorithms, *e.g.*, K-means clustering [42, 43], agglomerative hierarchical clustering (AHC) [36, 44, 45], mean-shift clustering [37], and spectral clustering [20, 46]. Recently, better clustering methods have been proposed, such as variational Bayes hidden Markov model clustering (VBx) [3, 4], auto-tuning spectral clustering [47], or fully supervised clus-

tering [21, 22]. Note that they are usually used for hard clustering, so most cascaded methods (with some exceptions [48]) cannot deal with speaker overlap.

Researchers mainly focus on better speaker embedding extractors and better clustering methods; thus, speech activity detection has often been ignored in evaluations of cascaded approaches by using oracle speech activities instead [38, 20, 21, 22, 47]. Moreover, obvious from the problem formulation in (2.4), this approach generally does not consider speaker overlaps. A further problem is that, although overlap ratios of conversations tend to exceed 10 % [13], speaker overlaps tend to be excluded from the evaluation of these approaches. This prevents measuring the actual performance of the methods and gives them an unfair advantage over other approaches.

2.2.3 Variants

There are also various investigations on cascaded approaches to improve their performance or bring additional functions. We list three major variants below.

Overlap handling

Since the commonly used problem formulation is as in Section 2.2.1, it ignores speaker overlaps. To handle speaker overlap in this framework, there are two approaches. One is to treat overlap handling as a post-process. In this approach, overlapped frames are firstly detected, and then assign the second speaker for the detected frames based on heuristics [49, 50] or the results of VB resegmentation [40]. The other is based on the clustering of overlapped segments [48]. It first extracts overlapped segments using a region proposal network and then applies clustering for embeddings extracted from each of them.

Online processing

To enable online inference of cascaded approaches, all modules have to work in an online manner. The most crucial part is the clustering of speaker embeddings, and many methods have been proposed for that in the literature, *e.g.*, UIS-RNN [21], UIS-RNN-SML [51], constraint incremental clustering in overlap-aware online speaker diarization [52], and turn-to-diarize [53]. Generally, online clustering is not as good as offline clustering. In particular, VBx [4], the current state-of-the-art offline clustering method for diarization, relies on two-stage clustering to refine the results and thus is difficult to be used for online inference. In fact, even if the rest

of the modules are similar between offline and online methods, replacing VBx with online clustering reportedly causes a significant drop in performance [52].

Multi-channel processing

In cascaded approaches that utilize multi-channel inputs, spatial features are used instead of, or altogether with, speaker embeddings. For example, time difference of arrival (TDOA) [54], direction of arrival (DOA) [55], or generalized cross correlation with phase transform (GCC-PHAT) [56] were used in the conventional studies.

2.3 End-to-End Approach

End-to-end approaches are emerging that directly produce diarization results from input audio or acoustic features using neural networks. The major advantages of end-to-end approaches over cascaded approaches are that 1) they naturally handle speaker overlap and 2) are simpler from an engineering perspective because there is only one module that makes up a diarization system. This section first reviews representative methods based on end-to-end neural networks in Section 2.3.1, and then details the methods we focus on in this thesis in Section 2.3.2 and Section 2.3.3.

2.3.1 Overview

Some methods such as personal VAD [57], VoiceFilter-Lite [58] take a target speaker's embedding and extract speech activities of the corresponding speakers. If we do inference for each speaker in the speech using these methods, we can obtain diarization results. Target-speaker voice activity detection (TS-VAD) further improved the methods by accepting multiple speakers' embeddings to predict their speech activities simultaneously [25, 59]. Here, it is a restrictive assumption that the speakers' embeddings are obtained in advance; in fact, TS-VAD obtains the embeddings for each speaker from the diarization results from a separately prepared speaker diarization system based on a cascaded approach.

For some methods, models have been trained for speech separation, and diarization results have been obtained as byproducts. For example, recurrent selective attention network (RSAN) estimates each speaker's time-frequency mask one by one in an autoregressive manner [60, 19]. To train such models, clean signals or ideal time-frequency masks computed from them are required as the target values. Therefore, it is difficult to make use of real multi-speaker recordings for the train-

ing, and we are unavoidably compelled to rely only on simulated multi-speaker mixtures, which can cause a performance drop due to their domain mismatch.

One promising approach is end-to-end neural diarization (EEND) [23, 24], which does not require target speakers' embeddings for inference or clean signals for training. EEND is firstly implemented on the basis of bi-directional long short-term memories [23], and replacing them with Transformer encoders [61] has brought a significant performance improvement. For better modeling of both global and local contexts, methods based on Conformers [62], time-dilated convolutional neural networks [63], residually-connected Transformers [64] were later proposed. This thesis focuses on EEND based on Transformer encoders, detailed in the following subsection.

2.3.2 End-to-End Neural Diarization

EEND [23, 24] is a method for estimating multiple speakers' speech activities simultaneously from an input recording. Given frame-wise F-dimensional acoustic features $X = (\boldsymbol{x}_t)_{t=1}^{T} \in \mathbb{R}^{F \times T}$, where $t \in \{1, \ldots, T\}$ is a frame index, EEND estimates speech activities $Y = (\boldsymbol{y}_t)_{t=1}^{T} \in \{0, 1\}^{S \times T}$. Here, $\boldsymbol{y}_t \coloneqq [y_{1,t}, \ldots, y_{s,t}, \ldots, y_{S,t}]^{\mathsf{T}}$ denotes speech activities of S speakers at t defined as

$$y_{s,t} = \begin{cases} 0 & \text{(Speaker } s \text{ is inactive at } t\text{)} \\ 1 & \text{(Speaker } s \text{ is active at } t\text{)} \end{cases}. \tag{2.5}$$

EEND assumes that $y_{s,t}$ is conditionally independent given the acoustic features, namely,

$$P(Y \mid X) = \prod_{t=1}^{T} \prod_{s=1}^{S} P(y_{s,t} \mid X). \tag{2.6}$$

With this assumption, speaker diarization can be regarded as a multi-label classification problem and can thus be easily modeled using a neural network f_{EEND} as

$$(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_T) = f_{\mathsf{EEND}}(X), \tag{2.7}$$

where $\boldsymbol{p}_t \coloneqq [p_{1,t}, \ldots, p_{S,t}]^{\mathsf{T}} \in (0, 1)^S$ is the posterior probabilities of S speakers' speech activities at frame index t. The estimation of speech activities $(\hat{\boldsymbol{y}}_t)_{t=1}^{T}$ is

$$\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_T = \underset{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_T}{\arg\max} P(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T \mid X), \tag{2.8}$$

$$= (\mathbb{1}(p_{s,t} > 0.5))_{\substack{1 \le s \le S \\ 1 \le t \le T}}, \tag{2.9}$$

where $\mathbb{1}(\text{cond})$ is an indicator function that returns 1 if cond is satisfied and 0 otherwise. Note that the threshold value in (2.9) is always set to 0.5 in this thesis for simplicity.

The conventional EEND is implemented as a composition of an embedding part $g : \mathbb{R}^{F\times T} \to \mathbb{R}^{D\times T}$ and a classification part $h : \mathbb{R}^{D\times T} \to (0,1)^{S\times T}$, *i.e.*,

$$f_{\mathsf{EEND}} = h \circ g. \tag{2.10}$$

The first embedding part g converts input acoustic features into D-dimensional frame-wise embeddings. The acoustic features are first converted using a position-wise fully connected layer parameterized by $W_0 \in \mathbb{R}^{D\times F}$ and $\boldsymbol{b}_0^D$ and layer normalization LN as

$$\boldsymbol{e}_1^{(0)}, \ldots, \boldsymbol{e}_T^{(0)} = \mathsf{LN}\left(W_0 X + \boldsymbol{b}_0 \mathbf{1}_D^\mathsf{T}\right), \tag{2.11}$$

where $\mathbf{1}_D$ is D-dimensional all-one vector. The resulting frame-wise embeddings are further converted using N-stacked encoders, each of which converts a flexible length of embedding sequence $(\boldsymbol{e}_t^{(n-1)})_{t=1}^T$ into the same length of embedding sequence $(\boldsymbol{e}_t^{(n)})_{t=1}^T$ as

$$\boldsymbol{e}_1^{(n)}, \ldots, \boldsymbol{e}_T^{(n)} = g^{(n)}\left(\boldsymbol{e}_1^{(n-1)}, \ldots, \boldsymbol{e}_T^{(n-1)}\right), \tag{2.12}$$

where $g^{(n)}$ is the n-th encoder layer, which is implemented as a Transformer encoder but without positional encodings to prevent the outputs from being affected by the absolute position of the frames. Hereafter, for simplicity, we use $\boldsymbol{e}_t$ to denote the embeddings from the last encoder, *i.e.*, $\boldsymbol{e}_t \coloneqq \boldsymbol{e}_t^{(N)}$ for $t \in \{1, \ldots, T\}$.

Then, the classification part h in (2.10) converts the embeddings $(\boldsymbol{e}_t)_{t=1}^T$ to posteriors of speech activities $(\boldsymbol{p}_t)_{t=1}^T$ in (2.7). It is implemented by using a fully connected layer and an element-wise sigmoid function $\sigma(\cdot)$ that takes a tensor as an argument:

$$[\boldsymbol{p}_1, \ldots, \boldsymbol{p}_T] = h(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_T; W_{\text{cls}}, \boldsymbol{b}_{\text{cls}}) \tag{2.13}$$

$$= \sigma\left(W_{\text{cls}}^\mathsf{T}\left[\boldsymbol{e}_1, \ldots, \boldsymbol{e}_T\right] + \boldsymbol{b}_{\text{cls}} \mathbf{1}_D^\mathsf{T}\right) \in (0,1)^{S\times T}, \tag{2.14}$$

where $(\cdot)^\mathsf{T}$ denotes the matrix transpose and $W_{\text{cls}} \in \mathbb{R}^{D\times S}$ and $\boldsymbol{b}_{\text{cls}} \in \mathbb{R}^S$ are the weight and bias of the fully connected layer, respectively.

EEND outputs posteriors of multiple speakers simultaneously but without any conditions to decide the order of the speakers. Such a network is optimized by using a permutation-free objective [65, 66], which was originally proposed for multi-talker speech separation. It computes the loss for all possible speaker assignments between predictions $(\boldsymbol{p}_t)_{t=1}^T$, as introduced in (2.7), and groundtruth labels $(\boldsymbol{y}_t)_{t=1}^T$, and it picks the minimum one for backpropagation as follows.

$$\mathcal{L}_{\text{diar}} = \frac{1}{TS} \min_{\boldsymbol{\phi}\in\Phi(S)} \sum_{t=1}^{T} H\left(\boldsymbol{y}_t^{\phi}, \boldsymbol{p}_t\right), \tag{2.15}$$

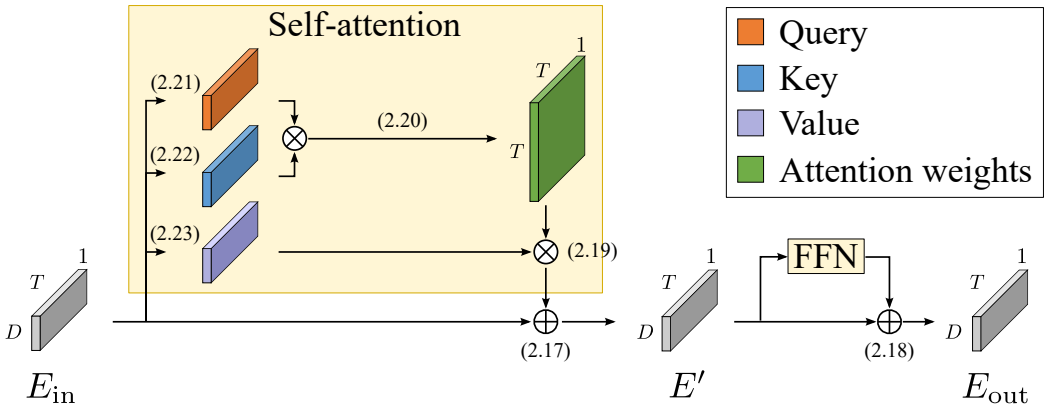


Figure 2.1: Transformer encoder. Each yellow area is skipped via a residual connection. $\otimes$ denotes matrix multiplication.

where $\Phi(S)$ is a set of all possible permutations of the sequence $(1, \ldots, S)$, $\boldsymbol{\phi} \coloneqq (\phi_1, \ldots, \phi_S)$ is the permuted sequence, $\boldsymbol{y}_t^{\phi} \coloneqq \left[y_{\phi_1,t}, \ldots, y_{\phi_S,t}\right]^{\mathsf{T}} \in \{0,1\}^S$ is the permuted groundtruth labels using $\boldsymbol{\phi}$, and $H(\cdot,\cdot)$ is the binary cross entropy defined as

$$H(\boldsymbol{y}_t, \boldsymbol{p}_t) \coloneqq \sum_{s=1}^{S} \left\{-y_{s,t} \log p_{s,t} - (1 - y_{s,t}) \log (1 - p_{s,t})\right\}. \tag{2.16}$$

Compared with cascaded approaches, EEND has two significant strengths. One is that the cascaded approaches conduct diarization by dividing frame-wise speaker embeddings, so they require SAD as pre-processing and overlap detection and assignment as post-processing. In contrast, EEND estimates each speaker's speech activities independently, so no extra modules for speech activity detection and overlap detection are needed. The other strength is that the EEND model can be adapted to the desired domain's dataset, while cascaded approaches typically tune only probabilistic linear discriminant analysis (PLDA) parameters to optimize intra- and inter-speaker similarity between speaker embeddings [36, 49, 67].

Transformer encoder

EEND-EDA uses a Transformer encoder [61] without positional encodings (Figure 2.1) for $g^{(n)}$ in (2.12). Given $E_{\text{in}} \in \mathbb{R}^{D \times T}$, the encoder converts it into $E_{\text{out}} \in \mathbb{R}^{D \times T}$ as follows:

$$E' = \mathsf{LN}\left(E_{\text{in}} + \mathsf{MA}\left(E_{\text{in}}, E_{\text{in}}, E_{\text{in}}\right); \boldsymbol{\Theta}, \boldsymbol{\Phi}\right), \tag{2.17}$$

$$E_{\text{out}} = \mathsf{LN}\left(E' + \mathsf{FFN}\left(E'; \Psi\right)\right), \tag{2.18}$$

where $\mathbf{\Theta}$, $\mathbf{\Phi}$, and Ψ are sets of parameters, and MA and FFN denote multi-head scaled dot-product attention and a feed-forward network, respectively.

Given d_k-dimensional query $Q \in \mathbb{R}^{d_k \times T}$, key $K \in \mathbb{R}^{d_k \times T}$, and d_v-dimensional value $V \in \mathbb{R}^{d_v \times T}$ inputs, multi-head scaled dot-product attention MA is calculated as

$$\mathsf{MA}\left(Q, K, V; \mathbf{\Theta}, \mathbf{\Phi}\right) = W_O \begin{bmatrix} V^{(1)} A^{(1)\mathsf{T}} \\ \vdots \\ V^{(h)} A^{(h)\mathsf{T}} \end{bmatrix} + \boldsymbol{b}_O \mathbf{1}^\mathsf{T} \in \mathbb{R}^{d_v \times T}, \tag{2.19}$$

$$A^{(i)} = \mathsf{softmax}\left(\frac{Q^{(i)\mathsf{T}} K^{(i)}}{\sqrt{d_k / h}}\right) \in (0,1)^{T \times T}, \tag{2.20}$$

$$Q^{(i)} = W_Q^{(i)} Q + \boldsymbol{b}_Q^{(i)} \mathbf{1}^\mathsf{T} \in \mathbb{R}^{\frac{d_k}{h} \times T}, \tag{2.21}$$

$$K^{(i)} = W_K^{(i)} K + \boldsymbol{b}_K^{(i)} \mathbf{1}^\mathsf{T} \in \mathbb{R}^{\frac{d_k}{h} \times T}, \tag{2.22}$$

$$V^{(i)} = W_V^{(i)} V + \boldsymbol{b}_V^{(i)} \mathbf{1}^\mathsf{T} \in \mathbb{R}^{\frac{d_v}{h} \times T}, \tag{2.23}$$

where h is the number of heads, $i \in \{1, \ldots, h\}$ is the head index, and $\mathsf{softmax}\left(\cdot\right)$ is the column-wise softmax function. The set of parameters $\mathbf{\Theta}$ and $\mathbf{\Phi}$ are defined as

$$\mathbf{\Theta} \coloneqq \bigcup_{1 \le i \le h} \left\{ W_Q^{(i)}, \boldsymbol{b}_Q^{(i)}, W_K^{(i)}, \boldsymbol{b}_K^{(i)} \right\}, \tag{2.24}$$

$$\mathbf{\Phi} \coloneqq \left\{ W_O, \boldsymbol{b}_O \right\} \cup \bigcup_{1 \le i \le h} \left\{ W_V^{(i)}, \boldsymbol{b}_V^{(i)} \right\}. \tag{2.25}$$

The feed-forward network FFN consists of two fully connected layers:

$$\mathsf{FFN}\left(E'; \mathbf{\Psi}\right) = \left(W_2 \left[W_1 E' + \boldsymbol{b}_1 \mathbf{1}^\mathsf{T} \right]_+ + \boldsymbol{b}_2 \mathbf{1}^\mathsf{T} \right), \tag{2.26}$$

$$\mathbf{\Psi} \coloneqq \left\{ W_1, \boldsymbol{b}_1, W_2, \boldsymbol{b}_2 \right\}, \tag{2.27}$$

where $W_1 \in \mathbb{R}^{d_f \times D}$ and $W_2 \in \mathbb{R}^{D \times d_f}$ are projection matrices, $\boldsymbol{b}_1 \in \mathbb{R}^{d_f}$ and $\boldsymbol{b}_2 \in \mathbb{R}^D$ are biases, and $\left[\cdot\right]_+$ is the ramp function.

2.3.3 Online Speaker Diarization

End-to-end approaches have also been explored in online diarization. Online diarization with end-to-end models has two directions. One is to train a model with frame-wise or block-wise inputs separately from the offline model. For example, Online RSAN [19, 68] is trained with block-wise inputs to extend the original RSAN [60] for an online manner. This method uses speaker embeddings to convey information between blocks to make the order of output speakers consistent. BW-EDA-EEND [69] replaced the Transformer encoders in EEND with Transformer-XL [70]

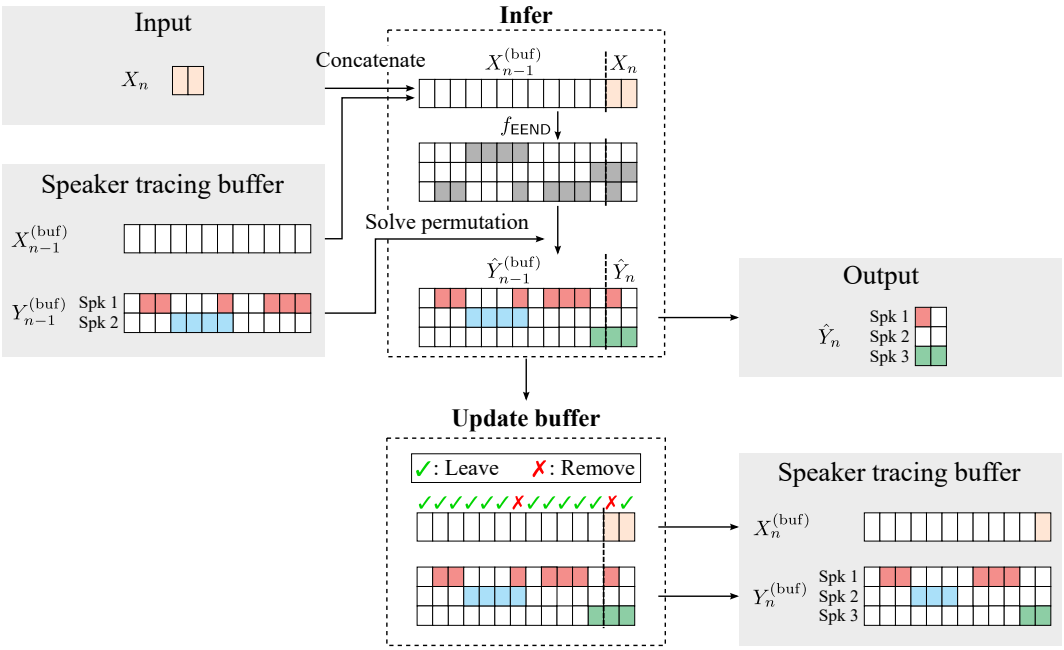


Figure 2.2: Online diarization using speaker-tracing buffer proposed in [1, 2].

to extend EEND-EDA [71, 72] to deal with block-wise inputs. In this method, the hidden state embeddings obtained during the processing of the previous blocks are used to process the current block, thereby solving the speaker permutation ambiguity between blocks. This direction is beneficial to optimize online diarization itself, but the training cost is doubled if we need to prepare diarization systems for both offline and online inference independently. The other possibility is to divert an offline diarization model for online inference. For this purpose, speaker-tracing buffer [1, 2] has been proposed to implement online inference of EEND with no modification of the network architecture. It stores acoustic features and their corresponding diarization results of the selected past frames to solve the speaker permutation ambiguity. Because it was reported that EEND-EDA with speaker-tracing buffer outperformed BW-EDA-EEND [2], we focused on this direction in this study.

Online Speaker Diarization with Speaker-Tracing Buffer

A speaker-tracing buffer has been proposed to enable online inference of EEND without additional training [1, 2]. The speaker-tracing buffer stores the past acoustic features and the corresponding estimation to solve the speaker permutation ambiguity. The schematic diagram of online diarization using a speaker-tracing buffer is shown in Figure 2.2.

In the situation of online diarization, chunked acoustic features sequentially ar-

rive, and the length of each chunk is ν. Suppose $X_{n-1}^{(\mathrm{buf})} \in \mathbb{R}^{F \times T_{n-1}^{(\mathrm{buf})}}$ and $Y_{n-1}^{(\mathrm{buf})} \in \mathbb{R}^{\hat{S}_{n-1} \times T_{n-1}^{(\mathrm{buf})}}$ are features and the corresponding estimations stored in the buffer just before the n-th input, respectively, where $T_{n-1}^{(\mathrm{buf})}$ is their length and $\hat{S}_{n-1}$ is the previously estimated number of speakers. Given n-th input $X_n \in \mathbb{R}^{F \times \nu}$, it is concatenated with the features in the buffer and processed by EEND f_{EEND} as

$$\begin{bmatrix} \hat{Y}_{n-1}^{(\mathrm{buf})} & \hat{Y}_n \end{bmatrix} = f_{\mathsf{EEND}}\left(\begin{bmatrix} X_{n-1}^{(\mathrm{buf})} & X_n \end{bmatrix} \right) \in (0,1)^{\hat{S}'_n \times \left(T_{n-1}^{(\mathrm{buf})} + \nu\right)}, \tag{2.28}$$

where $\hat{S}'_n$ is the newly estimated number of speakers. Here, $\hat{Y}_{n-1}^{(\mathrm{buf})} \in (0,1)^{\hat{S}'_n \times T_{n-1}^{(\mathrm{buf})}}$ and $\hat{Y}_n \in (0,1)^{\hat{S}'_n \times \nu}$ are the estimated results that correspond to $X_{n-1}^{(\mathrm{buf})}$ and X_n, respectively. Then, the number of speakers of $Y_{n-1}^{(\mathrm{buf})}$, $\hat{Y}_{n-1}^{(\mathrm{buf})}$, and $\hat{Y}_n$ is aligned to $\hat{S}_n = \max\left(\hat{S}_{n-1}, \hat{S}'_n\right)$ by

$$Y_{n-1}^{(\mathrm{buf})} \leftarrow \begin{bmatrix} Y_{n-1}^{(\mathrm{buf})} \\ O_{\hat{S}_n - \hat{S}_{n-1}, T_{n-1}^{(\mathrm{buf})}} \end{bmatrix}, \tag{2.29}$$

$$\begin{bmatrix} \hat{Y}_{n-1}^{(\mathrm{buf})} & \hat{Y}_n \end{bmatrix} \leftarrow \begin{bmatrix} \hat{Y}_{n-1}^{(\mathrm{buf})} & \hat{Y}_n \\ O_{\hat{S}_n - \hat{S}'_n, T_{n-1}^{(\mathrm{buf})}} & O_{\hat{S}_n - \hat{S}'_n, \nu} \end{bmatrix}, \tag{2.30}$$

where $O_{a,b}$ is the $a \times b$ zero matrix and a can be zero. Ideally, $\hat{S}'_n$ is not less than $\hat{S}_{n-1}$. The order of speakers is then permuted to be aligned to that of $\hat{Y}_{n-1}^{(\mathrm{buf})}$ as

$$\begin{bmatrix} \hat{Y}_{n-1}^{(\mathrm{buf})} & \hat{Y}_n \end{bmatrix} \leftarrow P_\psi \begin{bmatrix} \hat{Y}_{n-1}^{(\mathrm{buf})} & \hat{Y}_n \end{bmatrix}, \tag{2.31}$$

$$\psi = \underset{\phi \in \Phi\left(\hat{S}_n\right)}{\arg\max} \left\langle Y_{n-1}^{(\mathrm{buf})}, P_\phi \hat{Y}_{n-1}^{(\mathrm{buf})} \right\rangle_{\mathrm{F}}, \tag{2.32}$$

where $\langle A, B \rangle_{\mathrm{F}}$ denotes the Frobenius inner product between real-valued two matrices $A = \begin{bmatrix} a_{ij} \end{bmatrix}$ and $B = \begin{bmatrix} b_{ij} \end{bmatrix}$ defined as[1]

$$\langle A, B \rangle_{\mathrm{F}} := \sum_{i,j} a_{ij} b_{ij}. \tag{2.33}$$

Note that (2.32) is executable in polynomial time by using the well-known Hungarian algorithm. Finally, the permuted $\hat{Y}_n$ is output as the estimated result for X_n.

For the next (*i.e.*, $(n+1)$-th) input, the buffer is updated with the current input features and the corresponding results. If the buffer length M is large enough to store all the features and results, *i.e.*, $T_{n-1}^{(\mathrm{buf})} + \nu \leq M$, we update the buffer using

$$X_n^{(\mathrm{buf})} \leftarrow \begin{bmatrix} X_{n-1}^{(\mathrm{buf})} & X_n \end{bmatrix}, \tag{2.34}$$

$$Y_n^{(\mathrm{buf})} \leftarrow \begin{bmatrix} \hat{Y}_{n-1}^{(\mathrm{buf})} & \hat{Y}_n \end{bmatrix}. \tag{2.35}$$

[1] In the original STB paper, mean normalization is applied for each of A and B before the calculation of the Frobenius inner product, but it does not affect the result so we omit it here.

If $T_{n-1}^{(\mathrm{buf})} + \nu > M$, only M frames among them are selected to be stored. The original speaker-tracing buffer mainly utilized the following two update strategies.

1. **First-in-first-out (FIFO)**: acoustic features and results of the latest M frames are always stored in the buffer. Speakers who do not appear in the last M frames are not tracked with this strategy; thus, this strategy alone is not preferable.

2. **Sampling**: the features and results of informative M frames to solve speaker permutation ambiguity are selected among $T_{n-1}^{(\mathrm{buf})} + \nu$ and stored. In the previous studies [1, 2], sampling probabilities based on Kullback-Leibler (KL) divergence were used. The KL divergence at the t-th frame ω_t is calculated from the speaker-normalized posteriors $\bar{y}_{s,t}$ and the discrete uniform distribution with the posterior probability of $\frac{1}{S_n}$ as

$$\omega_t = \sum_{s=1}^{S_n} \bar{y}_{s,t} \log\left(\bar{y}_{s,t} S_n\right), \tag{2.36}$$

$$\bar{y}_{s,t} = \frac{y_{s,t}}{\sum_{s'=1}^{S_n} y_{s',t}}. \tag{2.37}$$

   The sampling probabilities $\tilde{\omega}_t$ are defined as the normalized KL divergence so that the sum is one:

$$\tilde{\omega}_t = \frac{\omega_t}{\sum_{t'} \omega_{t'}}. \tag{2.38}$$

With the aforementioned speaker-tracing buffer, a trained EEND model can be used for online inference as it is. However, EEND is generally trained with a fixed length of chunks, *e.g.*, 500 frames, so the diarization performance decreases at the very beginning of the inference where the number of frames is low. To cope with this problem, variable chunk-size training (VCT) was proposed [1]. For VCT, the length of each chunk is varied by masking the input minibatch. It has been evaluated in two-speaker conditions [1] but has not been evaluated in the flexible-number-of-speaker conditions [2]. Even the two-speaker experiments have a limited analysis of how VCT improved the diarization error rates (DERs).

2.4 Speech Separation Based on Speaker Diarization Results

As described in Section 1.1, the most possible use of diarization is a prior step to speech separation. In this section, we detail a speech separation algorithm in which diarization results are utilized, namely, guided source separation (GSS) [16]. As in

previous studies [16, 73, 74, 15], we assume that the following procedure is performed for each utterance.

Before getting to the explanation of GSS, we first introduce a basic speech separation that uses a complex angular Gaussian mixture model (cACGMM) [75] as a generative model. A probability density function of a cACGMM at a frequency index f is determined as

$$p\left(\hat{\boldsymbol{x}}_{t,f}; \left\{\alpha_f^{(k)}, B_f^{(k)}\right\}_k\right) = \sum_k \alpha_f^{(k)} \mathcal{A}\left(\hat{\boldsymbol{x}}_{t,f}; B_f^{(k)}\right), \tag{2.39}$$

$$\hat{\boldsymbol{x}}_{t,f} = \frac{\boldsymbol{x}_{t,f}}{\left\|\boldsymbol{x}_{t,f}\right\|}, \tag{2.40}$$

where $\boldsymbol{x}_{t,f} \in \mathbb{C}^M$ is the M-channel observed signal in a short-time Fourier transform (STFT) domain, t is the time index, and k is the source index. The observation can be dereverberated beforehand by using, *e.g.*, the weighted prediction error (WPE) [76]. The $\alpha_f^{(k)}$ is the mixture weight of the k-th source at f, and $\mathcal{A}(\boldsymbol{x}; B)$ is a complex angular central Gaussian distribution [77] parameterized by $B \in \mathbb{C}^{M \times M}$ as

$$\mathcal{A}(\boldsymbol{x}; B) \coloneqq \frac{(M-1)!}{2\pi^M \det(B)} \frac{1}{\left(\boldsymbol{x}^{\mathsf{H}} B^{-1} \boldsymbol{x}\right)^M}, \tag{2.41}$$

where $(\cdot)^{\mathsf{H}}$ denotes the Hermitian transpose. The optimization of the cACGMM is done using the EM algorithm. At the E-step, the posterior for each source at a time-frequency bin is calculated as

$$\gamma_{t,f}^{(k)} \leftarrow \frac{\alpha_f^{(k)} \frac{1}{\det\left(B_f^{(k)}\right)} \frac{1}{\left[\hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}} \left(B_f^{(k)}\right)^{-1} \hat{\boldsymbol{x}}_{t,f}\right]^M}}{\sum_{k'} \alpha_f^{(k')} \frac{1}{\det\left(B_f^{(k')}\right)} \frac{1}{\left[\hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}} \left(B_f^{(k')}\right)^{-1} \hat{\boldsymbol{x}}_{t,f}\right]^M}}. \tag{2.42}$$

At the M-step the parameters $\alpha_f^{(k)}$ and $B_f^{(k)}$ are updated as follows:

$$\alpha_f^{(k)} \leftarrow \frac{1}{T} \sum_t \gamma_{t,f}^{(k)}, \tag{2.43}$$

$$B_f^{(k)} \leftarrow M \frac{\sum_t \gamma_{t,f}^{(k)} \frac{\hat{\boldsymbol{x}}_{t,f} \hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}}}{\hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}} \left(B_f^{(k)}\right)^{-1} \hat{\boldsymbol{x}}_{t,f}}}{\sum_t \gamma_{t,f}^{(k)}}. \tag{2.44}$$

With GSS, the activities of each speaker are assumed known a priori and used for parameter updates. Given $d_t^{(k)} \in \{0, 1\}$, that is, an activity of source k that takes

1 if the source k is active at t and 0 otherwise, the E-step is replaced with

$$\gamma_{t,f}^{(k)} \leftarrow \frac{\alpha_f^{(k)} d_t^{(k)} \frac{1}{\det\left(B_f^{(k)}\right)} \frac{1}{\left[\hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}} \left(B_f^{(k)}\right)^{-1} \hat{\boldsymbol{x}}_{t,f}\right]^M}}{\sum_{k'} \alpha_f^{(k')} d_t^{(k')} \frac{1}{\det\left(B_f^{(k')}\right)} \frac{1}{\left[\hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}} \left(B_f^{(k')}\right)^{-1} \hat{\boldsymbol{x}}_{t,f}\right]^M}} \tag{2.45}$$

to force the posteriors of inactive sources to be zero. The diarization information helps to make the model free from the frequency permutation problem because it is frequency-independent. However, it is still affected by the permutation between the target utterance and noise because their activities are always one during the utterance. To solve this, GSS also uses preceding and subsequent signals of the utterance, which are called "context," for parameter update. In this thesis, we refer to the preceding context as pre-context and the subsequent context as post-context.

In the first iteration of the EM updates, $\alpha_f^{(k)}$ and $B_f^{(k)}$ are unknown, so the following (2.46) and (2.47) are used for the E-step and the update of $B_f^{(k)}$ in the M-step instead, respectively:

$$\gamma_{t,f}^{(k)} \leftarrow \frac{d_t^{(k)}}{\sum_{k'} d_t^{(k')}}, \tag{2.46}$$

$$B_f^{(k)} \leftarrow M \frac{\sum_t \gamma_{t,f}^{(k)} \hat{\boldsymbol{x}}_{t,f} \hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}}}{\sum_t \gamma_{t,f}^{(k)}}. \tag{2.47}$$

After convergence, spatial covariance matrices for speech and noise are calculated using the posteriors $\gamma_{t,f}^{(k)}$ as follows:

$$R_f^{\text{speech}} = \frac{1}{T} \sum_t \gamma_{t,f}^{(k_{\text{target}})} \boldsymbol{x}_{t,f} \boldsymbol{x}_{t,f}^{\mathsf{H}} \in \mathbb{C}^{M \times M}, \tag{2.48}$$

$$R_f^{\text{noise}} = \frac{1}{T} \sum_t \left(1 - \gamma_{t,f}^{(k_{\text{target}})}\right) \boldsymbol{x}_{t,f} \boldsymbol{x}_{t,f}^{\mathsf{H}} \in \mathbb{C}^{M \times M}. \tag{2.49}$$

Here we assume that the target source is $k_{\text{target}} \in \{1, \ldots, K\}$. The minimum variance distortionless response (MVDR) beamformer $\boldsymbol{w}_f \in \mathbb{C}^M$ is calculated using the spatial covariance matrices as

$$\boldsymbol{w}_f = \frac{{R_f^{\text{noise}}}^{-1} R_f^{\text{speech}} \boldsymbol{r}}{\text{tr}\left({R_f^{\text{noise}}}^{-1} R_f^{\text{speech}}\right)}, \tag{2.50}$$

where $\boldsymbol{r} \in \{0, 1\}^M$ is a one-hot vector that corresponds to the reference microphone, which is selected to maximize the signal-to-noise ratio. Finally, blind analytic normalization [78] is applied for $\boldsymbol{w}_f$ to obtain the final beamformer, which is used for speech separation. The enhanced signal in the STFT domain is calculated as

$$z_{t,f} = \boldsymbol{w}_f^{\mathsf{H}} \boldsymbol{x}_{t,f}. \tag{2.51}$$

2.5 Evaluation Protocols

To evaluate diarization performance directly, two common metrics are used in this thesis: diarization error rates (DERs) and Jaccard error rates (JERs) [95]. Most evaluations report DERs, but JERs are sometimes DER is defined as

$$\mathrm{DER} = \frac{T_{\mathrm{MI}} + T_{\mathrm{FA}} + T_{\mathrm{CF}}}{T_{\mathrm{Speech}}}, \tag{2.52}$$

where T_{Speech}, T_{MI}, T_{FA}, and T_{CF} denote the duration of total speech, missed speech, false-alarmed speech, and speaker confusion, respectively. Following the literature, we used collar tolerance at each speech boundary in some evaluations, *i.e.*, a certain duration at the start and end points are omitted from evaluation. We emphasize that speaker overlaps were NOT excluded from the evaluations.

To calculate JER, first, the optimal assignment between reference and system speakers is calculated. JER is the average score of each reference speaker defined as

$$\mathrm{JER} = \frac{1}{S_{\mathrm{ref}}} \sum_{s=1}^{S_{\mathrm{ref}}} \frac{T_{\mathrm{FA}}^{(s)} + T_{\mathrm{MI}}^{(s)}}{T_{\mathrm{Union}}^{(s)}}, \tag{2.53}$$

where S_{ref} is the number of reference speakers, and $T_{\mathrm{MI}}^{(s)}$ and $T_{\mathrm{FA}}^{(s)}$ are the duration of the missed and false-alarmed speech calculated between speech activities of the s-th reference speaker and the paired system speaker, respectively. $T_{\mathrm{Union}}^{(s)}$ is the time duration in which at least one of the s-th reference speakers of a paired system speaker is active.

The core difference betweeen DER and JER is their weighting of each speaker. In the calculation of DER, speakers are weighted according to the length of their utterances, whereas in the calculation of JER, each speaker is treated equally. This means that JER treats speakers who speak only a little equally with other speakers, and such speakers' speech segments tend to hard to be estimated, so in most cases JER is higher than DER.

Speaker diarization is used as post-process of ASR in Chapters 6 and 7 of this thesis. Of course the accuracy of speaker diarization affect the performance of ASR, it is not always true that speaker diarization systems better in DER or JER are also better from ASR perspectives. For example, missed segments in diarization cannot be transcribed by ASR, it is more beneficial to reduce missed speech than to reduce false alarms. Therefore, evaluation of speaker diarization applications should be done solely from the perspective of ASR. For English corpora (*i.e.*, CHiME-6 corpus in this thesis), we used word error rates (WERs) defined as

$$\mathrm{WER} = \frac{N_{\mathrm{sub}}^{(\mathrm{w})} + N_{\mathrm{del}}^{(\mathrm{w})} + N_{\mathrm{ins}}^{(\mathrm{w})}}{N_{\mathrm{ref}}^{(\mathrm{w})}}, \tag{2.54}$$

where $N_{\text{sub}}^{(\text{w})}$, $N_{\text{del}}^{(\text{w})}$, $N_{\text{ins}}^{(\text{w})}$, $N_{\text{ref}}^{(\text{w})}$ denote the number of substituted words, deleted words, inserted words, and words in the reference.

For some languages that are not space-separated, *e.g.*, Japanese and Mandarin, it is hard to calculate WERs because the boundaries of words are not obvious. We use instead character error rates (CERs) for the corpora in such languages (*i.e.*, the meeting corpus in this thesis) defined as

$$\text{CER} = \frac{N_{\text{sub}}^{(\text{c})} + N_{\text{del}}^{(\text{c})} + N_{\text{ins}}^{(\text{c})}}{N_{\text{ref}}^{(\text{c})}}, \tag{2.55}$$

where $N_{\text{sub}}^{(\text{c})}$, $N_{\text{del}}^{(\text{c})}$, $N_{\text{ins}}^{(\text{c})}$, $N_{\text{ref}}^{(\text{c})}$ denote the number of substituted characters, deleted characters, inserted characters, and characters in the reference.

Note that all the metrics are "the lower, the better" indicators with the lowest value of zero. However, they are not upper-bounded by one because false alarm or insertion can increase these error rates by any amount.

Chapter 3

End-to-End Speaker Diarization for Unknown Numbers of Speakers

3.1 Introduction

While EEND has the advantage in the way of handling speaker overlaps, it has the limitation in the number of output speakers; the number of output speakers S is fixed by the fully connected layer in the classification part h as in (2.14). In some applications, the number of speakers is obvious (*e.g.*, most conversations in a call center consist of two speakers: a customer and an operator), but in meetings, for example, the number of participants may vary from case to case. The practicality of EEND would be significantly improved by making it handle situations where the number of speakers is unknown. One possible way to treat a unknown number of speakers with this fixed-output architecture is to set the number of outputs to be large enough. However, it requires knowing the maximum number of speakers in advance, and it has been already verified that such a strategy results in poor performance. It is also a problem that the calculation cost of the permutation-free loss increases if we set a large number of speakers to be output. Therefore, a significant research question is how to output diarization results for a unknown number of speakers. This chapter addresses the diarization methods that handle unknown numbers of speakers on the basis of EEND.

Section 3.2 proposes attractor-based EEND (EEND-EDA) that provides end-to-end speaker diarization of flexible numbers of speakers. We introduce encoder-decoder-based attractor calculation module (EDA) to EEND. Once frame-wise em-

beddings are obtained, EDA sequentially generates speaker-wise attractors on the basis of a sequence-to-sequence method using an LSTM encoder-decoder. The attractor generation continues until a stopping condition is satisfied; thus, the number of attractors can be flexible. Diarization results are then estimated as dot products of the attractors and embeddings. The embeddings from speaker overlaps result in larger dot product values with multiple attractors; thus, this method can deal with speaker overlaps. Because the maximum number of output speakers is still limited by the training set, we also propose an iterative inference method to remove this restriction. Further, we propose a method that aligns the estimated diarization results with the results of an external speech activity detector, which enables fair comparison against cascaded approaches.

Section 3.3 further extends EEND-EDA to make the method deal with unlimited numbers of speakers, namely, EEND-GLA. While EEND-EDA can deal with flexible numbers of speakers, the maximum number of speakers is still empirically known to be bounded by their training datasets. In the proposed method, in addition to the attractors calculated from the entire recording (*i.e.*, *global attractors*) in the same manner as in EEND-EDA, we also utilize attractors calculated from each short block (*i.e.*, *local attractors*) to obtain block-wise diarization results. Because the set of speakers and their output order may be different among the blocks, we use clustering to find the appropriate speaker correspondence between the blocks on the basis of the similarities between the local attractors. Here, we assume that the number of speakers appearing in a short period is low, and so the number of speakers within each block can be limited and fixed with a maximum number. However, the total number of speakers is estimated as the result of clustering; it is no longer limited by the network architecture or training datasets.

Section 3.4 proposes a method to use EEND-GLA in an online manner on the basis of STB described in Section 2.3.3. Because EEND-GLA adopt block-wise processing to utilize the nature that the number of speakers speaks within a short interval is small, it is not compatible with the original STB that includes the frame-wise selection step. To enable online inference of EEND-GLA, we propose a block-wise speaker-tracing buffer, which extends the original speaker-tracing buffer [1, 2] to update the buffer elements in a block-wise manner. With this modification, we can assume that the number of speakers within each block is limited in the buffer as well because each block stores time-consecutive elements.

3.2 End-to-End Speaker Diarization for Flexible Numbers of Speakers

In this section, we extend the conventional EEND to handle a flexible number of speakers in Section 3.2.2. We also provide novel inference techniques in Section 3.2.3.

3.2.1 Related Work

Speech processing based on neural networks for unknown numbers of speakers

While some methods have achieved promising results with a fixed number of output speakers in diarization [23, 24, 25] and speech separation [66, 79, 80, 81] contexts, it is challenging to make them able to deal with unknown numbers of speakers. The difficulty of neural-network-based speech processing for unknown numbers of speakers is that we cannot fix the output dimension.

One possible approach is to determine the maximum number of speakers to decode. In this case, the number of outputs is set to a sufficiently large value. Some methods treat a flexible number of speakers by outputting null speech activities if the number of outputs is smaller than the network capacity [82]. However, this approach did not work well with EEND (see [83]). In other methods, the number-of-speaker-wise output branches are trained independently, and the most probable is used during inference [84]. In this case, we have to know the maximum number of speakers. One of the strengths of EEND is that it can be finetuned using a target domain dataset from a pretrained model, but we usually cannot access the maximum number of speakers of the target domain beforehand. Therefore, a method that does not require that the maximum number of speakers be defined would be preferable.

Another approach is to decode speakers one by one until a stopping condition is satisfied, like SC-EEND [83]. For speech separation, RSAN [60, 19] and one-and-rest permutation invariant training (OR-PIT) [85] can be used. The key difference between speech separation and diarization is whether or not the residual output can be defined. RSAN uses a mask-based approach, in which each time-frequency bin is softly assigned to each speaker so that the process finishes when all the elements of the residual mask become zero. OR-PIT is time-domain speech separation by which residual output is determined as a mixture that contains other speakers rather than the target speaker. Both require clean recordings to determine oracle masks or signals. However, they are not always accessible in the diarization con-

text, in which only multi-talker recordings and speech segments are provided.

For EEND-EDA, we adopted an attractor-based approach like deep attractor networks (DANet) [86, 82]. While the number of speakers [86] or maximum number of speakers [82] is fixed for the original DANet, we calculated a flexible number of attractors without defining them.

Neural-network-based representative vector calculation

There have been several efforts to calculate representative vectors from a sequence of embeddings in an end-to-end trainable fashion. For example, Set Transformer [87] enables set-to-set transformation, which can be used to calculate cluster centroids from a set of embeddings. However, the number of outputs has to be known in advance, so it cannot be used for our purpose. Meier *et al*. proposed an end-to-end clustering framework [88], in which clustering for all possible number of clusters $K \in \{1, \dots, K_{\max}\}$ is performed and the result of the most probable number of clusters is used. The framework performs the clustering of a flexible number of clusters in an end-to-end manner, but the maximum number of clusters is limited by $K_{\max}$. EDA, in comparison, determines a flexible number of attractors from an input embedding without prior knowledge of the number of speakers. Thus, we can use datasets of the different maximum number of speakers during pretraining and finetuning.

3.2.2 Attractor-Based End-to-End Neural Diarization

We assume that the embedding part g in (2.10) is implemented in the same manner as the conventional EEND described in Section 2.3.2. Given frame-wise D-dimensional embeddings $\{\boldsymbol{e}_t\}_{t=1}^{T}$, our goal is to produce posteriors for a flexible number of speakers in the classification part h. To achieve this goal, we propose a method to calculate a flexible number of speaker-wise attractors from embeddings and then calculate diarization results on the basis of attractors and embeddings. The proposed method is depicted in Figure 3.1.

EDA: Encoder-decoder-based attractor calculation

EDA converts frame-wise embeddings into speaker-wise attractors using a sequence-to-sequence method with an LSTM encoder-decoder. The LSTM encoder h^{enc} takes the frame-wise embeddings as input and updates its hidden state $\boldsymbol{h}_t^{\mathsf{enc}}$

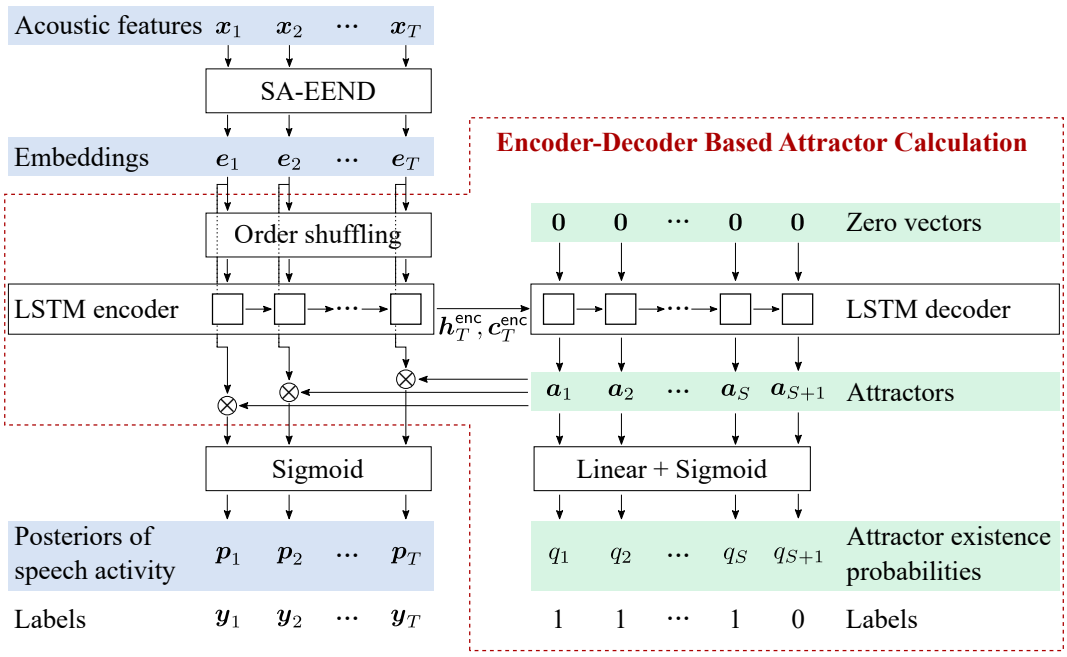


Figure 3.1: EEND with encoder-decoder-based attractor calculation (EEND-EDA).

and cell state $\boldsymbol{c}_t^{\text{enc}}$ as

$$\boldsymbol{h}_t^{\text{enc}}, \boldsymbol{c}_t^{\text{enc}} = h^{\text{enc}}\left(\boldsymbol{e}_t, \boldsymbol{h}_{t-1}^{\text{enc}}, \boldsymbol{c}_{t-1}^{\text{enc}}\right) \qquad (t = 1, \dots, T)\,. \tag{3.1}$$

The hidden and cell states of the encoder are initialized with zero vectors, *i.e.*, $\boldsymbol{h}_0^{\text{enc}} = \boldsymbol{c}_0^{\text{enc}} = \boldsymbol{0}$. The LSTM decoder h^{dec} estimates speaker-wise attractors as

$$\boldsymbol{h}_s^{\text{dec}}, \boldsymbol{c}_s^{\text{dec}} = h^{\text{dec}}\left(\boldsymbol{0}, \boldsymbol{h}_{s-1}^{\text{dec}}, \boldsymbol{c}_{s-1}^{\text{dec}}\right) \qquad (s = 1, 2, \dots)\,. \tag{3.2}$$

We treat the hidden state at each step $\boldsymbol{h}_s^{\text{dec}} =: \boldsymbol{a}_s \in (-1, 1)^D$ as speaker s's attractor, whose dimensionality D is the same as that of the frame-wise embeddings $\boldsymbol{e}_t$. The hidden and cell states of the decoder are initialized by the final hidden and cell states of the encoder as

$$\boldsymbol{h}_0^{\text{dec}} = \boldsymbol{h}_T^{\text{enc}}, \tag{3.3}$$

$$\boldsymbol{c}_0^{\text{dec}} = \boldsymbol{c}_T^{\text{enc}}, \tag{3.4}$$

which is shown as a right arrow from the LSTM encoder to the LSTM decoder in Figure 3.1. In general applications of a sequence-to-sequence method, *e.g.*, speech recognition or machine translation, the output is sentences, *i.e.*, a sequence of words, so the order of output is fixed. However, EDA cannot determine the order of output speakers in advance because this order is determined by minimizing cross entropy as in (2.15). Even if the order could be predetermined, it would not be possible to determine the optimal attractor outputs. Thus, the well-known strategy

of teacher forcing, for which the optimal outputs with their order have to be known in advance, cannot be used. Furthermore, the s-th attractor can correspond to any speaker that is not contained in the first $(s-1)$ attractors. To make this attractor calculation procedure fully order-free, we input a zero vector as input at each step as in (3.2). Using zero vectors as inputs provides flexibility to change the number of output speakers across pretraining and finetuning rather than using, for example, trainable parameters. This is why we chose an LSTM-based encoder-decoder rather than Transformer encoder-decoder, which requires input queries rather than zero vectors.

Here, the input order to the EDA encoder affects the output attractors because EDA is based on a sequence-to-sequence method. To investigate the effect of the input order, we tried two types of input orders: chronological and shuffled orders. In the chronological order setting, embeddings are input in the order of frame indexes as in (3.1). In the shuffled order setting, we use the following instead of (3.1) :

$$\boldsymbol{h}_t^{\mathsf{enc}}, \boldsymbol{c}_t^{\mathsf{enc}} = h^{\mathsf{enc}}\left(\boldsymbol{e}_{\psi_t}, \boldsymbol{h}_{t-1}^{\mathsf{enc}}, \boldsymbol{c}_{t-1}^{\mathsf{enc}}\right) \qquad (t = 1, \dots, T)\,, \tag{3.5}$$

where $(\psi_1, \dots, \psi_T)$ is a randomly chosen permutation of $(1, \dots, T)$.

The diarization results $\boldsymbol{p}_t$ in (2.7) are calculated on the basis of the dot product of the frame-wise embeddings and speaker-wise attractors ($\otimes$ in Figure 3.1):

$$\boldsymbol{p}_t = \sigma\left(A^{\mathsf{T}}\boldsymbol{e}_t\right) \in (0,1)^S\,, \tag{3.6}$$

where $A \coloneqq [\boldsymbol{a}_1, \dots, \boldsymbol{a}_S]$ are the speaker-wise attractors. The posteriors are optimized by using (2.15) in the same manner as the conventional EEND. This posterior calculation no longer depends on the fully connected layer, which determines the output number of speakers as in (2.14); therefore, EDA-based diarization can vary the output number of speakers.

Comparing (2.14) and (3.6), the conventional EEND can also be regarded as using fixed attractors W_{cls} (with bias $\boldsymbol{b}_{\mathrm{cls}}$). In comparison, EDA calculates attractors from an input sequence of embeddings, which makes attractors adaptive to the embeddings. This makes EEND-EDA more accurate even under the fixed-number-of-speakers condition (see Table 3.1).

Attractor existence probability

As in (3.2), we can obtain an infinite number of attractors. To decide when to stop the attractor calculation, we calculate the attractor existence probabilities from the

calculated attractors by using a fully connected layer followed by sigmoid activation:

$$q_s = \sigma\left(\boldsymbol{w}_{\text{exist}}^{\mathsf{T}} \boldsymbol{a}_s + b_{\text{exist}}\right), \tag{3.7}$$

where $\boldsymbol{w}_{\text{exist}} \in \mathbb{R}^D$ and $b_{\text{exist}} \in \mathbb{R}$ are trainable weights and bias parameters of the fully connected layer, respectively.

During training, we know the oracle number of speakers S, so the training objective of the attractor existence probabilities is based on the first $(S+1)$-th attractors using the binary cross entropy defined in (2.16):

$$\mathcal{L}_{\text{exist}} = \frac{1}{S+1} H\left(\boldsymbol{l}, \boldsymbol{q}\right), \tag{3.8}$$

where

$$\boldsymbol{l} \coloneqq [\underbrace{1, \ldots, 1}_{S}, 0]^{\mathsf{T}}, \tag{3.9}$$

$$\boldsymbol{q} \coloneqq [q_1, \ldots, q_{S+1}]^{\mathsf{T}}. \tag{3.10}$$

The total loss is defined as the weighted sum of $\mathcal{L}_{\text{diar}}$ in (2.15) and $\mathcal{L}_{\text{exist}}$ in (3.8) with the weighting parameter $\alpha \in \mathbb{R}_+$ as

$$\mathcal{L} = \mathcal{L}_{\text{diar}} + \alpha \mathcal{L}_{\text{exist}}. \tag{3.11}$$

In this thesis, we use $\alpha = 1$. This multi-task loss aims to optimize frame- and speaker-wise posteriors with $\mathcal{L}_{\text{diar}}$ and attractor existence probabilities with $\mathcal{L}_{\text{exist}}$.

While (3.11) was used for the network optimization in our previous study [71], we found that the optimization of $\mathcal{L}_{\text{exist}}$ inhibits the minimization of $\mathcal{L}_{\text{diar}}$ during the training of a model with a flexible number of speakers, which is more important for improving diarization accuracy. Therefore, when a flexible number of speakers' dataset is used for training, we use $\mathcal{L}_{\text{exist}}$ to update only the fully connected layer parameterized by $\boldsymbol{w}_{\text{exist}}$ and b_{exist} in (3.7). This can be implemented by cutting the graph before the fully connected layer to disable backpropagation to the preceding layers.

During inference, we cannot access the oracle number of speakers; thus, it is estimated using q_s in (3.7) as follows.

$$\hat{S} = \min\left\{s \mid s \in \mathbb{Z}_+ \wedge q_{s+1} < \tau\right\}, \tag{3.12}$$

where $\tau \in (0, 1)$ is a thresholding parameter, which is set to 0.5 in this thesis. We then use the first $\hat{S}$ attractors to calculate posteriors as in (3.6).

3.2.3 Inference Methodology

SAD post-processing

Diarization methods, especially cascaded ones, are sometimes evaluated with oracle speech segments. When evaluated in such a way, the comparison between cascaded methods and EEND-methods becomes hard, mainly because EEND-based methods perform SAD and diarization simultaneously. One reason evaluations of cascaded approaches are mainly based on oracle speech segments is to consider speaker errors and SAD errors separately. It is reasonable to use oracle speech segments to focus on reducing speaker errors. However, such segments are not accessible in real scenarios, and the existence of SAD errors may worsen the clustering performance, which directly affects the diarization accuracy. Thus, we believe that SAD errors should also be considered in the context of cascaded methods. However, it is hard to say how accurate the SAD should be for a fair comparison between cascaded and EEND-based methods. Therefore, to align with the the cascaded methods, we introduce SAD post-processing for evaluating EEND. With this method, we can conduct a fair comparison between cascaded and EEND-based methods with the same SAD. Note that it can be used to improve the diarization performance by eliminating false alarm speech and recovering missed speech when an accurate external SAD system is given.

The SAD post-processing algorithm is described in Algorithm 3.1. Here, we assume that we have SAD results $z_1, \ldots, z_T$ in addition to frame- and speaker-wise posteriors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_T$. We first estimate speech activities as usual by using (2.9) (line 1). However, this estimation is not always consistent with SAD results. Thus, we first filter false alarms (FA) by using SAD results. For each frame (line 2), if it is estimated that some speakers are active while the speech activity should be zero (line 3), we update the estimations with a zero vector (line 4). This procedure will always improve DER if $z_1, \ldots, z_T$ are the oracle speech activities. We also recover missed frames (MI) if no speaker is estimated as active while the speech activity is one (line 5). For each of such frames, we treat the speaker with the highest posterior as an active speaker (line 6–line 7). Including the oracle SAD as input will also improve the DER because missed-frame errors are replaced by correct estimation or at least speaker errors.

Iterative inference

Even if the model is trained to output a flexible number of speakers, the output number of speakers is empirically limited by the maximum number of speakers in a recording observed during pre-training (see Table 3.5). How to output the results

Algorithm 3.1: SAD post-processing.

Input : $(\boldsymbol{p}_1, \dots, \boldsymbol{p}_T) \in (0,1)^{S \times T}$ // Frame-wise posteriors
$(z_1, \dots, z_T) \in \{0,1\}^T$ // SAD results
Output: $(\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_T) \in \{0,1\}^{S \times T}$ // Speech activities

1 Compute $\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_T$ using (2.9) // Initial results
2 **foreach** $t \in \{1, \dots, T\}$ **do**
3 **if** $\|\hat{\boldsymbol{y}}_t\|_1 > 0 \wedge z_t = 0$ **then** // Filter FA
4 $\hat{\boldsymbol{y}}_t \leftarrow [0, \dots, 0]^\mathsf{T}$
5 **else if** $\|\hat{\boldsymbol{y}}_t\|_1 = 0 \wedge z_t = 1$ **then** // Recover MI
6 $s^* \leftarrow \arg\max_{s \in 1, \dots, S} \boldsymbol{p}_t$
7 $\hat{\boldsymbol{y}}_t \leftarrow [0, \dots, 0, \overset{\wedge}{1}, 0, \dots, 0]^\mathsf{T} \in \{0,1\}^S$ ($\wedge$ at s^*)

of more than N speakers even if the model is trained on at most N-speaker mixtures is still an open question. While the method we propose in Section 3.3 tries to solve this problem by enhancing the model itself, here we propose an iterative inference method to produce results for more than N speakers by applying EEND decoding with iterative frame selection.

Preliminarily, we first reveal the characteristics of the EEND models that consist of stacked Transformer encoders and EDA. A Transformer encoder involves neither recurrence nor convolutional calculation, and we do not use positional encoding in this thesis; thus, the embedding part g in (2.10) is an order-free transformation. EDA contains an LSTM encoder-decoder, but if the order of the input sequence to EDA is shuffled, we can say that EDA does not depend on the input order, so the EDA's classification part h in (2.10) is also an order-free function. Therefore, EEND-EDA does not depend on the order of the input features, which makes it possible to process features that are not extracted at equal intervals along the time axis, as in EEND as post-processing [89]. The proposed iterative inference also utilizes this characteristic.

Algorithm 3.2 shows the algorithm of iterative inference. In the algorithm, two processes are iteratively conducted: decoding and silence frame selection. Each process at the n-th iteration is described as follows.

1. **Decoding** (line 3): Acoustic features $\boldsymbol{x}_t$ of the selected frames $\mathcal{T}$ are fed into EEND, and the corresponding posteriors $\boldsymbol{p}_t^{(n)} \in (0,1)^{S^{(n)}}$ are obtained as

$$\left(\boldsymbol{p}_t^{(n)}\right)_{t \in \mathcal{T}} \leftarrow f_{\mathsf{EEND}}\left((\boldsymbol{x}_t)_{t \in \mathcal{T}}\right), \tag{3.13}$$

where $S^{(n)} \in \{0, \dots, S_{\max}\}$ is the number of decoded speakers. The posteriors

Algorithm 3.2: Iterative inference.

Input : $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ // Acoustic features
f_{EEND} // EEND model
$S_{\max} \in \mathbb{N}$ // Max # of speakers that EEND can output
Output: $\hat{Y} \in \{0,1\}^{S \times T}$

1 $\mathcal{T} \leftarrow \{1, \ldots, T\}$ // Frame set
2 **for** $n \leftarrow 1$ **to** ∞ **do**
3 Compute $\hat{Y}^{(n)}$ by (3.13), (3.14), and (2.9) // Decoding
4 Update $\mathcal{T}$ by (3.15) // Silence frame selection
5 **if** $S^{(n)} < S_{\max} \vee |\mathcal{T}| = 0$ **then**
6 **break**
7 $\hat{Y} \leftarrow \begin{bmatrix} \hat{Y}^{(1)} \\ \vdots \\ \hat{Y}^{(n)} \end{bmatrix}$

of the frames that are not in $\mathcal{T}$ are set to zero as

$$\boldsymbol{p}_t^{(n)} \leftarrow [\underbrace{0, \ldots, 0}_{S^{(n)}}]^{\mathsf{T}} \quad (t \in \{1, \ldots, T\} \setminus \mathcal{T}) . \tag{3.14}$$

With the posteriors $\boldsymbol{p}_t^{(n)}$ for $t \in \{1, \ldots, T\}$, diarization results $\hat{Y}^{(n)} = \left(\hat{\boldsymbol{y}}_1^{(n)}, \ldots, \hat{\boldsymbol{y}}_T^{(n)}\right)$ are computed using (2.9). Note that $\hat{Y}^{(n)}$ corresponds to the speech activities of the $((n-1)S_{\max}+1)$-th through $((n-1)S_{\max}+S^{(n)})$-th speakers.

2. **Silence frame selection** (line 4): Given the diarization results decoded at the n-th iteration, we select the frames in which no speaker is active to update $\mathcal{T}$ as

$$\mathcal{T} \leftarrow \left\{ t \;\middle|\; t \in \mathcal{T}, \left\| \hat{\boldsymbol{y}}_t^{(n)} \right\|_1 = 0 \right\} . \tag{3.15}$$

The above processes start with the initial value of $\mathcal{T}$ as the set of all frames $\{1, \ldots, T\}$ (line 1), and last until $\mathcal{T}$ becomes the empty set or when it is assumed that all the speakers are decoded (line 5–line 6). Here, we assume that all the speakers are decoded if the number of output speakers $S^{(n)}$ is smaller than the maximum output of EEND $S_{\max}$.

After the iterative process is finished, the final results $\hat{Y}$ are obtained by concatenating the results calculated at each iteration (line 7). With iterative inference, the number of speakers to be decoded is no longer limited by the training dataset. The iterative inference workflow when $S_{\max} = 3$ is also illustrated in Figure 3.2.

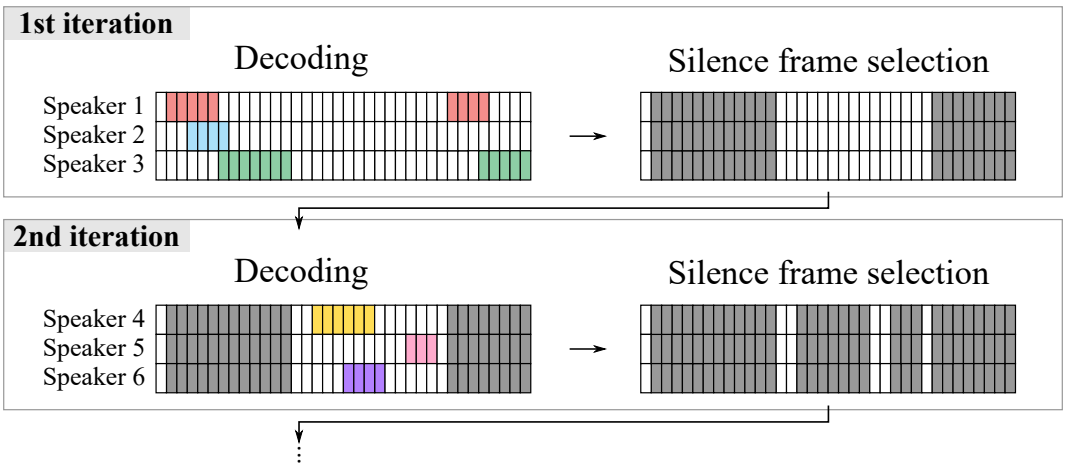


Figure 3.2: Iterative inference in the case of $S_{\max} = 3$.

Iterative inference with DOVER-Lap (or iterative inference+)

Despite iterative inference being able to produce more than $S_{\max}$ speakers' speech activities, it has a potential problem in that the speech activities of two speakers decoded at different iterations never overlap. For example, the $(S_{\max}+1)$-th speaker's speech activities never overlap with those of the first $S_{\max}$ speakers. This is because the frames in which the first $S_{\max}$ speakers are active will not be processed in the second iteration. To ease this problem, we introduce DOVER-Lap [90], which is the extension of DOVER [91]. Both of them are methods for combining multiple diarization results on the basis of majority voting, but unlike DOVER, DOVER-Lap take speaker overlap into account. We used a modified version of DOVER-Lap presented in [67], in which the speaker assignment strategy when multiple speakers were ranked equally was slightly different from the original DOVER-Lap [90]. Note that we did not use a hypothesis-wise weighting of DOVER-Lap, which is also introduced in [67].

The algorithm of iterative inference incorporated with DOVER-Lap is shown in Algorithm 3.3. In this thesis, we refer to this inference as iterative inference+. The difference from the iterative inference in Algorithm 3.2 is that we limit the number of speakers to decode at the first iteration with $S_{\text{limit}} (\leq S_{\max})$ (line 5–line 6). After the decoding step at the first iteration using (3.13), (3.14), and (2.9), we choose at most the first S_{limit} speakers' speech activities from $\hat{Y}^{(1)} := (\hat{y}_{s,t})_{s,t}$ as

$$\hat{Y}^{(1)} \leftarrow (\hat{y}_{s,t})_{\substack{1 \leq s \leq \min\left(S^{(1)}, S_{\text{limit}}\right) \\ 1 \leq t \leq T}} . \tag{3.16}$$

The other procedures are the same as those in Algorithm 3.2, and finally, we obtain S_{limit}-wise diarization results $Y_{S_{\text{limit}}}$ (line 10).

Algorithm 3.3: Iterative inference with DOVER-Lap (or iterative inference+).

Input : $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ // Acoustic features
f_{EEND} // EEND model
$S_{\max} \in \mathbb{N}$ // Max # of speakers that EEND can output
Output: $\hat{Y} \in \{0,1\}^{S \times T}$

1 **for** $S_{\text{limit}} = 1$ **to** $S_{\max}$ **do**
2 $\quad \mathcal{T} \leftarrow \{1, \ldots, T\}$ // Frame set
3 $\quad$ **for** $n \leftarrow 1$ **to** ∞ **do**
4 $\quad\quad$ Compute $\hat{Y}^{(n)}$ by (3.13), (3.14), (2.9) // Decoding
5 $\quad\quad$ **if** $n = 1$ **then**
6 $\quad\quad\quad$ Limit the number of speakers in $\hat{Y}^{(n)}$ by (3.16)
7 $\quad\quad$ Update $\mathcal{T}$ by (3.15) // Silence frame selection
8 $\quad\quad$ **if** $S^{(n)} < S_{\max} \vee |\mathcal{T}| = 0$ **then**
9 $\quad\quad\quad$ **break**
10 $\quad \hat{Y}_{S_{\text{limit}}} \leftarrow \begin{bmatrix} \hat{Y}^{(1)} \\ \vdots \\ \hat{Y}^{(n)} \end{bmatrix}$
11 $\hat{Y} \leftarrow \text{DOVER-Lap}\left(\hat{Y}_1, \ldots, \hat{Y}_{S_{\max}}\right)$

In iterative inference+, S_{limit} is varied from 1 to $S_{\max}$ (line 1), which results in $S_{\max}$ diarization results for each recording. We then combine them by using DOVER-Lap to obtain the final result $\hat{Y}$ (line 11). With this procedure, the k-th speaker's speech activities can be overlapped with those of the $\max(1, (k - S_{\max} + 1))$-th to $(k + S_{\max} - 1)$-th speakers.

3.2.4 Experimental Settings

For the embedding part g in (2.10) of the proposed EEND-EDA, we used four-stacked Transformer encoders with four attention heads without positional encodings, each of which outputs 256-dimensional frame-wise embeddings. The inputs for the model were log-scaled Mel-filterbank-based features. We first extracted 23-dimensional log-scaled Mel-filterbanks with a frame length of 25 ms and frame shift of 10 ms. Each of them was then concatenated with those of the preceding and following seven frames, followed by subsampling with a factor of 10. As a result, a 345 $(= 23 \times 15)$ dimensional acoustic feature was extracted for each 100 ms.

In this thesis, we evaluated EEND-EDA for both fixed-numbers-of-speakers

and unknown-numbers-of-speakers conditions; thus, a model was trained for each purpose. For the fixed-number-of-speakers evaluation, the model was first trained on the Sim*k*spk training set for 100 epochs and evaluated on the Sim*k*spk test set. We also adapted the model to CALLHOME-*k*spk for another 100 epochs to evaluate the model on real recordings. We used $k \in \{2, 3\}$ in this thesis. For the unknown-number-of-speakers evaluation, the model that was trained on Sim2spk was fine-tuned by using the concatenation of Sim{1,2,3,4}spk or Sim{1,2,3,4,5}spk for 50 epochs. The model was also adapted to each target dataset for another 500 epochs.

For network training using simulated mixtures, we used the Adam optimizer [92] with the Noam scheduler [61] with 100,000 warm-up steps. For adaptation, we also used the Adam optimizer but with a fixed learning rate of 1×10^{-5}. For efficient batch processing during training, we split each recording into 500 frames when using Sim*k*spk and 2000 frames when using the adaptation sets. The batch size for training was set to 64. Note that an entire recording is fed into the network without splitting during inference.

3.2.5 Results of Fixed-Numbers-of-Speakers Experiments

Two-speaker experiment

First, we evaluated our method under the two-speaker condition. In this case, the model was first trained on Sim2spk and then adapted to CALLHOME-2spk Part 1. For the EEND-based methods, we used the model trained on Sim2spk to evaluate the simulated datasets and the one adapted to CALLHOME-2spk Part 1 to evaluate CALLHOME-2spk Part 2 and CSJ. For EEND-EDA, we used the first two output attractors for speech activity calculation.

Table 3.1 shows the results of the two-speaker evaluation. We observed that the proposed method with the shuffled order setting achieved the best DERs. Despite EEND-EDA being designed to deal with flexible numbers of speakers, it outperformed the conventional EENDs, *i.e.*, BLSTM-EEND and SA-EEND, which output diarization results for fixed numbers of speakers. This is because the conventional EEND can be regarded as a fixed-attractor-based method, while EEND-EDA is an adaptive-attractor-based method as described in the last paragraph of Section 3.2.2. This flexibility of attractors makes the proposed method more accurate even in fixed-number-of-speakers evaluations. In terms of the order of the input to EDA, shuffled sequences always performed better than chronologically ordered sequences. It indicates that the global context is more important than the temporal context to calculate attractors.

Table 3.1: DERs (%) for two-speaker evaluations. 0.25 s of collar tolerance was allowed.

	Simulated			Real	
Method	$\beta = 2$	$\beta = 3$	$\beta = 5$	CALLHOME-2spk	CSJ
i-vector + AHC	33.74	30.93	25.96	12.10	27.99
x-vector (TDNN) + AHC	28.77	24.46	19.78	11.53	22.96
BLSTM-EEND [23]	12.28	14.36	19.69	26.03	39.33
SA-EEND [24]	4.56	4.50	3.85	9.54	20.48
EEND-EDA (Chronol.)	3.07	2.74	3.04	8.24	18.89
EEND-EDA (Shuffled)	**2.69**	**2.44**	**2.60**	**8.07**	**16.27**

Table 3.2: DERs (%) for three-speaker evaluations. 0.25 s of collar tolerance was allowed.

	Simulated			Real
Method	$\beta = 2$	$\beta = 3$	$\beta = 5$	CALLHOME-3spk
x-vector (TDNN) + AHC	31.78	26.06	19.55	19.01
SA-EEND [24]	8.69	7.64	6.92	14.00
EEND-EDA (Chronol.)	13.02	11.65	10.41	15.86
EEND-EDA (Shuffled)	**8.38**	**7.06**	**6.21**	**13.92**

Three-speaker experiment

We also evaluated the method under the three-speaker condition. We first trained the model on Sim3spk and then adapted it to CALLHOME-3spk Part 1. We validated the performance on Sim3spk using the model trained on Sim3spk and that on CALLHOME-3spk Part 2 using the model adapted to CALLHOME-3spk Part 1. We used the first three attractors to evaluate EEND-EDA's performance. As shown in Table 3.2, EEND-EDA with sequence shuffling performed best on both simulated and real datasets.

Effect of input order

For a better understanding of EDA, we tried various types of sequences as inputs to the models, each of which was trained on chronologically ordered sequences and shuffled sequences. We evaluated matched and unmatched conditions of orders, and we also evaluated the effect of reducing the sequence length by subsampling or using the last $1/N$ part of the sequences. Table 3.3 shows the results on Sim2spk

Table 3.3: DERs on Sim2spk (overlap ratio: 34.4 %) using various types of sequences.

(a) Using whole sequence

Method	Test: Chronol.	Test: Shuffled
EEND-EDA (Train: Chronol.)	3.07	30.04
EEND-EDA (Train: Shuffled)	2.69	2.69

(b) Subsample $1/N$

Method	$N = 2$	$N = 4$	$N = 8$	$N = 16$	$N = 32$
EEND-EDA (Train: Chronol.)	3.54	7.32	14.48	21.13	27.18
EEND-EDA (Train: Shuffled)	2.70	2.68	2.79	3.09	5.08

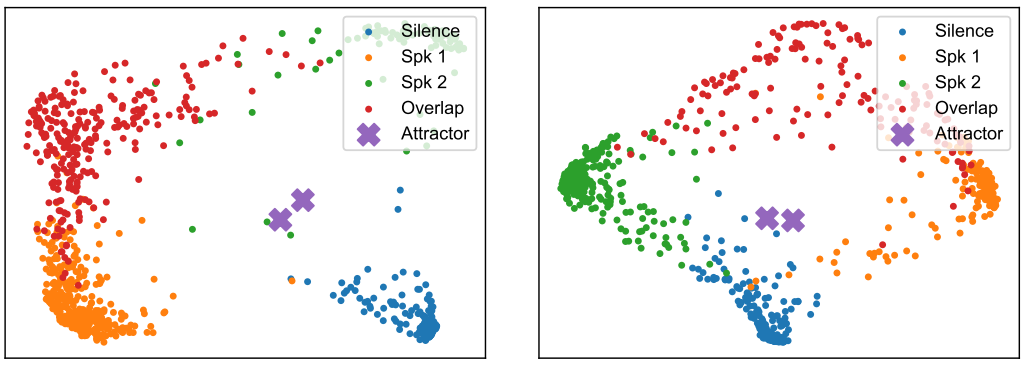
(c) Use the last $1/N$

Method	$N = 2$	$N = 4$	$N = 8$	$N = 16$	$N = 32$
EEND-EDA (Train: Chronol.)	3.67	4.97	5.40	6.11	7.68
EEND-EDA (Train: Shuffled)	3.36	5.92	7.46	8.59	10.65

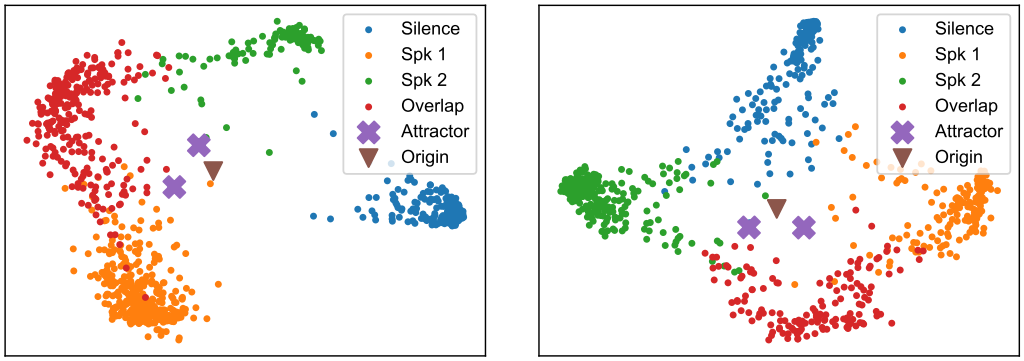
($\beta = 2$). The EEND-EDA that was trained on chronologically ordered sequences performed well on chronologically ordered sequences but did poorly on shuffled sequences. It was also affected by subsampling, while it was slightly influenced by using the last $1/N$ part. These results indicate that the length of each utterance is an important factor to decide the output attractors for the model trained on chronologically ordered sequences. On the other hand, when the model was trained on shuffled sequences, it was not that affected by the order of sequences nor subsampling. However, when the last $1/N$ of the sequences were used, its performance degradation was worse than the model trained on chronologically ordered sequences. These results indicate that EDA trained on shuffled sequences captured the distribution of embeddings; thus, subsampling did not affect the performance that much, while using the last $1/N$, *i.e.*, biased sampling, degraded the DERs.

Embedding visualization

For intuitive understanding of the behavior of EDA, we visualized the embeddings $\boldsymbol{e}_t$ and attractors $\boldsymbol{a}_s$ within a two-speaker mixture from Sim2spk ($\beta = 2$) in Figure 3.3(b). They were projected to two-dimensional space by using principal com-



(a) Conventional EEND [24]



(b) EEND-EDA

Figure 3.3: Visualization of embedding and attractors within each recording. For conventional EEND, weights of last fully connected layer W_{cls} were visualized instead of attractors.

ponent analysis (PCA). We observed that the embeddings of two speakers were well distinguished from those of silence frames, and those of overlapped frames were distributed between the areas of the two speakers. For EEND-EDA, two attractors were calculated for each of the two speakers successfully as in Figure 3.3(b); each hyperplane orthogonal to the attractor divides the embeddings into two parts, depending on whether the corresponding speaker is active or not. In Figure 3.3(a), in comparison, the fixed attractors W_{cls} of the conventional EEND were not well separated compared with the attractors calculated using EDA.

To understand the characteristics of attractors from EDA, we also visualized the inter-mixture relationship of attractors. For visualization, we first chose an anchor speaker and then selected mixtures that contained the anchor speaker. We calculated two attractors from each mixture by using EEND-EDA and mapped them

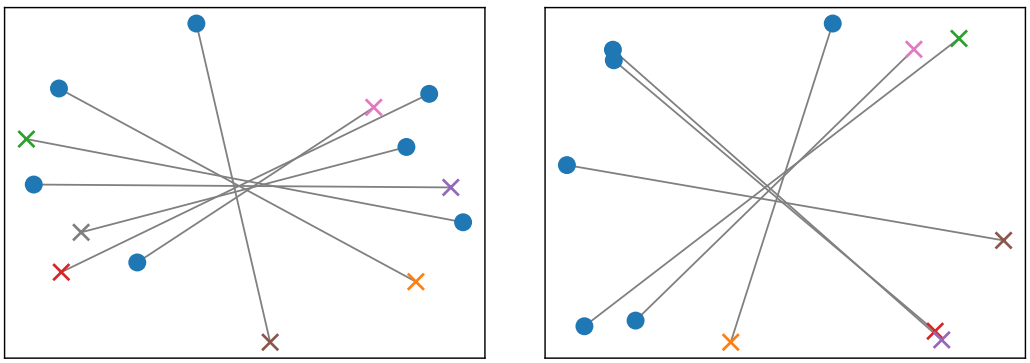


Figure 3.4: Visualization of attractors across recordings. Selected speakers' attractors are marked by dots, and their interference speakers' attractors are marked by crosses. Colors of crosses correspond to speaker identities within each figure. Each pair of attractors from same mixture are connected with gray line.

onto a two-dimensional space using PCA. The speaker assignment from the calculated attractors to speaker identifiers was based on the groundtruth labels. Figure 3.4 shows the attractors of two-speaker mixtures that contain the same anchor speaker. It clearly shows that the each anchor speaker's attractors were not distributed near each other.

From these results, the embeddings and attractors were calculated only to separate speakers in each mixture. We can also say that the attractors were not suited for speaker identification. This also supports the idea that attractors are adaptively calculated from input embeddings. A similar observation on attractors from DANet [86] in speech separation was provided in Section 5 of [93] that attractors cannot be used for speaker identification or tracing.

Evaluation on the mismatched number of speakers

We also evaluated two-speaker EEND-EDA on three-speaker datasets, and three-speaker EEND-EDA on two-speaker datasets. We used the model trained on Sim2spk or Sim3spk for the evaluation on the simulated datasets, and used the model adapted to CALLHOME-2spk or CALLHOME-3spk for the evaluation on the real datasets. The order of the embeddings is shuffled before being fed into EDA. The results are shown in Table 3.4. It is clearly observed that the DERs degraded when the number of speakers during training and inference was different. It is worth mentioning that three-speaker EEND-EDA did not work well on the two-speaker datasets; this indicates that the larger number of speakers during training does not serve the smaller number of speakers during inference.

Table 3.4: DERs (%) of cross evaluations of two- and three-speaker EEND-EDA. 0.25 s of collar tolerance was allowed.

	Two-speaker datasets		Three-speaker datasets	
Model	Sim2spk ($\beta = 2$)	CALLHOME -2spk	Sim3spk ($\beta = 5$)	CALLHOME -3spk
Two-speaker EEND-EDA	2.69	8.07	28.79	20.80
Three-speaker EEND-EDA	15.12	9.95	8.38	13.92

Table 3.5: Step-by-step improvement on simulated datasets. For Sim2spk and Sim3spk, we used $\beta = 2$ and $\beta = 5$, respectively. In $\mathcal{L}_{\text{exist}}$ column, we show which parameters were updated using $\mathcal{L}_{\text{exist}}$ during training.

				Sim$k$spk				
Model	Training data	#Epochs	$\mathcal{L}_{\text{exist}}$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
EEND-EDA	$k \in \{1, \ldots, 4\}$	25	Update all the parameters in f_{EEND}	0.39	4.33	8.94	13.76	N/A
	$k \in \{1, \ldots, 4\}$	25	Update only $\boldsymbol{w}_{\text{exist}}$ and b_{exist}	0.25	4.06	7.68	10.12	23.08
	$k \in \{1, \ldots, 5\}$	25	Update only $\boldsymbol{w}_{\text{exist}}$ and b_{exist}	0.21	4.22	8.25	10.75	13.70
	$k \in \{1, \ldots, 5\}$	50	Update only $\boldsymbol{w}_{\text{exist}}$ and b_{exist}	0.36	3.65	7.70	9.97	11.95
SA-EEND	$k \in \{1, \ldots, 4\}$	50	N/A	0.60	4.39	9.40	13.56	25.22
	$k \in \{1, \ldots, 5\}$	50	N/A	0.50	3.95	9.18	12.24	17.42

3.2.6 Results of Unknown-Numbers-of-Speakers Experiments

Simulated mixtures

To train EEND-EDA to output flexible numbers of speakers' results, we finetuned the model from the two-speaker model for at most 50 epochs using Sim1spk to Sim4spk or Sim1spk to Sim5spk. Table 3.5 shows the step-by-step improvement of the model. Note that the results on the top row correspond to our previous paper [71]. First, disabling backpropagation from the attractor existence loss $\mathcal{L}_{\text{exist}}$ to update only $\boldsymbol{w}_{\text{exist}}$ and b_{exist} improved the DERs for Sim1spk to Sim4spk. However, we observed that the model still did not perform well on Sim5spk, which was not included in the training set. Adding Sim5spk to the training set solved the problem as shown in the third row, which shows DERs that improved for Sim5spk from 23.08 % to 13.70 %. This indicates that EEND-EDA's number of output speakers was empirically limited by its training datasets, even though it does not limit the number of output speakers with its network architecture. Increasing the number of training epochs further improved the DERs as shown in the last row. We also showed the DERs computed by SA-EEND [24] trained on a flexible number of speakers' dataset in the last two rows. In each case, the model's output number of speakers was set to the maximum number of speakers in the dataset, *i.e.*, four or

five, and the model was trained to output null speech activities if a recording of a fewer number of speakers was input. EEND-EDA outperformed SA-EEND in all datasets. Hereafter, we use the EEND-EDA model of the fourth row ($k \in \{1, \ldots, 5\}$, 50 epochs, using $\mathcal{L}_{\text{exist}}$ to update only $\boldsymbol{w}_{\text{exist}}$ and b_{exist} during training) and the SA-EEND model of the sixth row ($k \in \{1, \ldots, 5\}$, 50 epochs).

CALLHOME

Since the CALLHOME dataset does not include an official dev/eval split, we used the split provided in the Kaldi recipe and performed cross-validation. For comparison with the prior work on EEND, we also report the results obtained for Part 2 of the dataset using the model adapted to Part 1. For SAD post-processing described in Section 3.2.3, we used the TDNN-based SAD provided in the Kaldi ASpIRE recipe[1] and oracle speech segments.

We show the number-of-speakers-wise results of cross-validation in Table 3.6(a). We also show the results for only evaluated single speaker regions in brackets. For this purpose, we chose up the most probable speakers from each time frame of the EEND-EDA results for fair comparison with x-vector-based methods. EEND-EDA outperformed the state-of-the-art x-vector-based methods in total DERs. One reason is that EEND-EDA can handle speaker overlap, but it showed a competitive DER (5.29 %) even when speaker overlaps were excluded from the evaluation. Considering the number of speakers in a mixture, EEND-EDA did especially better than the x-vector-based methods with VBx clustering when the number of speakers was small (# of speakers=2,3,4), while it was worse or on par when the number of speakers was large (# of speakers=5,6,7). One reason is that the pretraining was based on mixtures with at most five speakers, and another reason is that mixtures of a larger number of speakers are rare in the CALLHOME dataset. Compared to SA-EEND, EEND-EDA achieved better DERs on all the cases. Table 3.6(b) shows the results on CALLHOME Part2. It clearly shows that EEND-EDA outperformed the other EEND-based methods [83, 94] by over two percent of absolute DER.

Table 3.7 shows confusion matrices for the speaker counting of x-vector (TDNN) + AHC, x-vector (ResNet101) + AHC + VBx [4], SC-EEND [83], and EEND-EDA on CALLHOME Part 2. Our method achieved a higher speaker counting accuracy than the other methods by a large margin.

[1] https://github.com/kaldi-asr/kaldi/tree/master/egs/aspire/s5

Table 3.6: DERs (%) of CALLHOME. 0.25 s of collar tolerance was allowed. TDNN-based x-vector results were obtained with Kaldi recipe. DERs of single-speaker regions are reported in brackets. AHC: agglomerative hierarchical clustering, VB: Variational Bayes resegmentation [3], VBx: Variational Bayes HMM clustering [4].

(a) Results of cross-validation.

		# of speakers						
Method	SAD	2	3	4	5	6	7	Total
SA-EEND	-	8.51	19.84	26.16	36.82	48.52	38.24	19.82 (13.38)
EEND-EDA	-	8.18	15.05	16.54	27.29	31.40	37.23	14.81 (8.68)
X-vector (TDNN) + AHC	TDNN	14.66	18.42	20.46	31.40	32.62	46.43	19.48 (10.25)
X-vector (TDNN) + AHC + VB	TDNN	11.68	17.22	19.71	30.24	32.07	46.49	17.80 (8.29)
SA-EEND	TDNN	7.42	18.10	21.80	31.69	44.61	35.09	17.41 (10.66)
EEND-EDA	TDNN	**6.79**	**13.74**	**15.53**	**25.25**	**27.65**	**34.49**	**13.36** (**7.12**)
X-vector (TDNN) + AHC	Oracle	13.68	17.04	17.89	29.96	32.55	45.20	18.04 (8.54)
X-vector (TDNN) + AHC + VB	Oracle	10.94	15.85	17.40	29.23	33.97	42.69	16.57 (6.63)
X-vector (ResNet101) + AHC + VBx [4]	Oracle	9.83	15.23	14.29	**19.24**	**25.76**	36.25	14.21 (**4.42**)
SA-EEND	Oracle	6.02	16.28	20.26	30.42	43.51	35.09	15.90 (8.99)
EEND-EDA	Oracle	**5.50**	**12.17**	**12.86**	23.17	27.96	**34.08**	**11.72** (5.29)

(b) Results on CALLHOME Part 2.

Method	SAD	DER
SA-EEND	-	21.19
SC-EEND [83]	-	15.75
SAD-OD-fiert SC-EEND [94]	-	15.32
EEND-EDA (From [71])	-	15.29
EEND-EDA	-	**12.88**
X-vector (TDNN) + AHC	TDNN	19.43
X-vector (TDNN) + AHC + VB	TDNN	17.61
SA-EEND	TDNN	19.85
EEND-EDA	TDNN	**13.84**
X-vector (TDNN) + AHC	Oracle	17.02
X-vector (TDNN) + AHC + VB	Oracle	15.57
X-vector (ResNet101) + AHC + VBx [4]	Oracle	13.33
SA-EEND	Oracle	16.79
EEND-EDA	Oracle	**10.46**

Table 3.7: Confusion matrices for speaker counting on CALLHOME Part 2. X-vector-based results were obtained with oracle SAD, while EEND-based results were obtained without external SAD.

(a) X-vector (TDNN) + AHC (Accuracy=56.4 %)

		Ref. # of speakers					
		1	2	3	4	5	6
Pred. # of speakers	1	**0**	2	1	0	0	0
	2	0	**87**	19	3	0	0
	3	0	59	**51**	14	3	2
	4	0	2	4	**3**	2	1
	5	0	0	0	0	**0**	0
	6	0	0	0	0	0	**0**

(b) X-vector (ResNet101) + AHC + VBx [4] (Accuracy=72.0 %)

		Ref. # of speakers					
		1	2	3	4	5	6
Pred. # of speakers	1	**0**	21	3	0	0	0
	2	0	**122**	22	2	0	0
	3	0	3	**44**	7	0	0
	4	0	2	5	**10**	2	1
	5	0	0	0	1	**3**	0
	6	0	0	0	0	0	**1**
	7	0	0	0	0	0	1

(c) SC-EEND [83] (Accuracy=76.4 %)

		Ref. # of speakers					
		1	2	3	4	5	6
Pred. # of speakers	1	**0**	1	0	0	0	0
	2	0	**134**	20	4	0	0
	3	0	13	**51**	10	4	2
	4	0	0	3	**6**	1	1
	5	0	0	0	0	**0**	0
	6	0	0	0	0	0	**0**

(d) EEND-EDA (Accuracy=84.4 %)

		Ref. # of speakers					
		1	2	3	4	5	6
Pred. # of speakers	1	**0**	1	0	0	0	0
	2	0	**142**	7	1	0	0
	3	0	5	**54**	4	0	0
	4	0	0	13	**14**	4	1
	5	0	0	0	1	**1**	2
	6	0	0	0	0	0	**0**

Table 3.8: DERs and JERs (%) for AMI headset mix. No collar tolerance was allowed.

		Dev		Eval	
Method	SAD	DER	JER	DER	JER
SA-EEND	-	31.66	39.20	27.70	37.50
EEND-EDA	-	21.93	25.86	21.56	29.99
X-vector (ResNet101) + AHC	Oracle	19.61	23.90	21.43	25.50
X-vector (ResNet101) + AHC + VBx [4]	Oracle	16.33	**20.57**	18.99	**24.57**
SA-EEND	Oracle	23.95	35.64	20.88	34.38
EEND-EDA	Oracle	**15.69**	22.19	**15.80**	26.68

AMI headset mix

We next evaluated our method on the AMI headset mix, which has a different domain from the pretraining data (telephone conversation vs. meeting). We trained the model on the training set for 500 epochs and evaluated it on the dev and eval sets. The oracle speech segments were also used for SAD post-processing.

The results are shown in Table 3.8. EEND-EDA outperformed the x-vector-based methods on both the dev and eval sets with the oracle SAD. Note that the x-vector-based methods tuned the PLDA parameters on the dev set, so the superiority of EEND-EDA was smaller on the dev set than the eval set. EEND-EDA also outperformed SA-EEND with and without the oracle SAD. We also note that the average duration of the recordings in the AMI headset mix test set is over 30 min. The performance of EEND-EDA showed that EEND-EDA generalized well to such long recordings while using 200 s segments during adaptation.

DIHARD II & III

Finally, we evaluated our method on the DIHARD II and III datasets, which contain recordings from multiple domains. In this evaluation, we used iterative inference with and without DOVER-Lap, each of which are described in Section 3.2.3, to deal with large numbers of speakers. For SAD post-processing, we used oracle segments and the system used in the Hitachi-JHU submission to the DIHARD III challenge [67].

The results are shown in Tables 3.9 and 3.10. We can see that iterative inference with DOVER-Lap (iterative inference+) consistently improved DERs. Compared

Table 3.9: DERs and JERs (%) for DIHARD II eval. No collar tolerance was allowed.

Method	SAD	DER	JER
SA-EEND	-	32.14	54.32
EEND-EDA	-	29.57	51.50
EEND-EDA (Iterative inference)	-	29.41	49.61
EEND-EDA (Iterative inference+)	-	**28.52**	**49.77**
X-vector (TDNN) + AHC + VBx [49]	BUT [49]	**27.11**	**49.07**
SA-EEND	BUT [49]	32.01	54.66
EEND-EDA	BUT [49]	30.48	51.78
EEND-EDA (Iterative inference)	BUT [49]	29.80	49.99
EEND-EDA (Iterative inference+)	BUT [49]	29.09	50.45
DIHARD II baseline [95]	Oracle	28.81	50.12
X-vector (TDNN) + AHC + VBx [49]	Oracle	**18.21**	N/A
X-vector (ResNet101) + AHC [4]	Oracle	23.59	43.93
X-vector (ResNet101) + AHC + VBx [4]	Oracle	18.55	**43.91**
SA-EEND	Oracle	23.25	50.30
EEND-EDA	Oracle	20.54	46.92
EEND-EDA (Iterative inference)	Oracle	21.00	45.30
EEND-EDA (Iterative inference+)	Oracle	20.24	45.62

with the x-vector-based methods, EEND-EDA performed best on DIHARD III full, while the x-vector-based methods were better on DIHARD II and DIHARD III core.

We show the number-of-speakers-wise DERs and JERs on DIHARD III in Table 3.11. Our method performed better when the number of speakers was small and worse when the number of speakers was large. This is why EEND-EDA performed well on DIHARD III full and worse on DIHARD II and DIHARD III eval. We also observed that the proposed iterative inference+ improved the performance, especially in terms of JERs on a large number of speaker cases, but it was still worse than the x-vector method. From these results, handling a large number of speakers with EEND is the remaining challenge.

3.2.7 Analysis

As shown in Table 3.5, the maximum number of speakers to be output was empirically revealed to be limited by the dataset used during training. For example, if EEND-EDA is trained using mixtures, each of which contains at most four speakers, it cannot produce a valid result for the fifth or later speaker even if a mixture

Table 3.10: DERs and JERs (%) for DIHARD III eval. No collar tolerance was allowed.

Method	SAD	Core		Full	
		DER	JER	DER	JER
SA-EEND	-	27.49	49.64	22.64	43.14
EEND-EDA	-	25.94	47.76	21.55	41.15
EEND-EDA (Iterative inference)	-	25.76	45.35	21.40	39.09
EEND-EDA (Iterative inference+)	-	**24.77**	**45.18**	**20.69**	**39.07**
X-vector (TDNN) + AHC + VBx [67]	Hitachi-JHU [67]	**22.99**	42.44	21.48	38.73
X-vector (TDNN) + AHC + VBx + OVL [67]	Hitachi-JHU [67]	24.58	**42.02**	21.47	**37.83**
SA-EEND	Hitachi-JHU [67]	25.79	49.20	21.29	42.68
EEND-EDA	Hitachi-JHU [67]	23.96	46.82	20.03	40.31
EEND-EDA (Iterative inference)	Hitachi-JHU [67]	24.41	44.70	20.30	38.47
EEND-EDA (Iterative inference+)	Hitachi-JHU [67]	23.43	44.93	**19.53**	38.78
DIHARD III baseline [96]	Oracle	20.65	47.74	19.25	42.45
X-vector (TDNN) + AHC + VBx [67]	Oracle	**16.89**	38.49	15.83	34.27
X-vector (TDNN) + AHC + VBx + OVL [67]	Oracle	18.20	**38.42**	15.65	**33.71**
X-vector (ResNet152) + AHC + VBx [97]	Oracle	16.56	38.72	15.79	34.46
SA-EEND	Oracle	20.21	46.17	16.19	39.44
EEND-EDA	Oracle	18.38	43.69	14.91	36.93
EEND-EDA (Iterative inference)	Oracle	18.87	41.58	15.21	35.08
EEND-EDA (Iterative inference+)	Oracle	17.86	41.69	**14.42**	35.30

Table 3.11: Breakdown results of DIHARD III eval for each number of speakers with oracle speech segments.

(a) DER (%)

Method	# of speakers								
	1	2	3	4	5	6	7	8	9
X-vector (TDNN) + AHC + VBx	**1.30**	11.43	16.76	**23.09**	**44.99**	**26.43**	**25.61**	**35.57**	**2.03**
EEND-EDA	2.80	7.52	15.79	25.63	47.66	31.73	35.47	38.19	18.73
EEND-EDA (Iterative inference+)	1.47	**6.98**	**15.55**	26.32	47.48	31.44	34.79	38.26	14.99

(b) JER (%)

Method	# of speakers								
	1	2	3	4	5	6	7	8	9
X-vector (TDNN) + AHC + VBx	**2.40**	16.99	44.68	**44.70**	**66.17**	**53.32**	**56.05**	**56.71**	**8.01**
EEND-EDA	3.37	11.77	**38.70**	48.37	67.40	64.85	67.77	69.00	57.60
EEND-EDA (Iterative inference+)	3.31	**11.34**	39.60	48.76	68.46	62.41	62.65	65.36	41.23

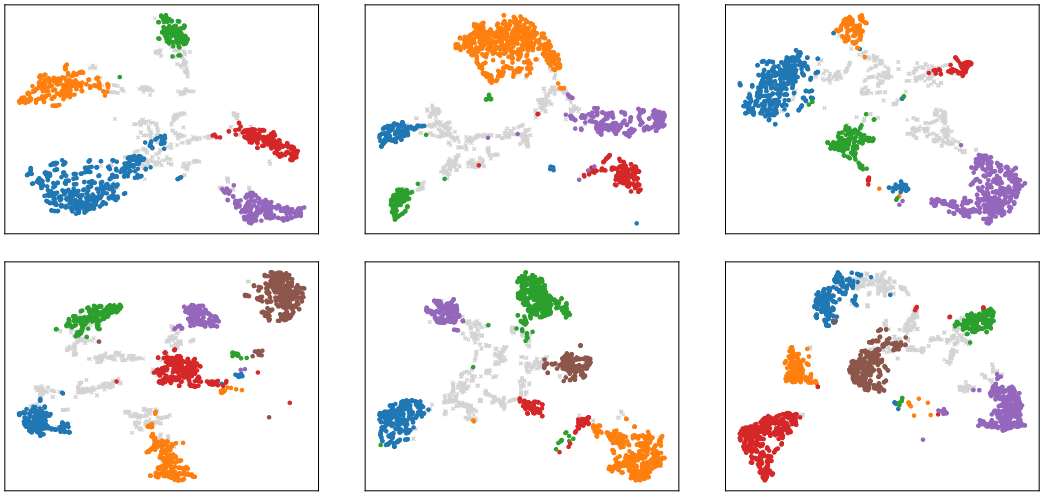


Figure 3.5: t-SNE visualization of frame-wise embeddings extracted from simulated 5-speaker mixtures (top) and 6-speaker mixtures (bottom). The EEND-EDA used for extraction was trained using {1,2,3,4}-speaker mixtures. Single-speaker frames are denoted by the dots with colors corresponding to the speaker identities and overlapped frames are denoted by the crosses in light gray. Frames of silence were excluded from the visualization.

contains more than four speakers.

To reveal which part causes this limitation, we visualized the frame-wise embeddings that were output from the last Transformer encoder using t-SNE [98] in Figure 3.5. Even though EEND-EDA was trained on mixtures, each of which consists of at most four speakers, five or six speakers' speeches were clearly separated in the embedding space. The visualization revealed that the EDA restricted the output number of speakers.

3.3 End-to-End Speaker Diarization for Unlimited Numbers of Speakers

This section proposes a method to overcome the limitation of the EDA to enable overlap-aware speaker diarization of unlimited numbers of speakers.

3.3.1 Related Work

One promising approach for removing the restriction on maximum number of speakers described in Section 3.2.7 is to incorporate clustering into the inference. For example, time-frequency-bin-wise embeddings extracted from a deep clustering model trained on two-speaker mixtures are still discriminative on three-speaker mixtures [65]. In the diarization context, for example, EEND-vector clustering [99, 100], in which a combination of EEND and speaker embedding clustering was investigated, can deal with the mismatched condition. Here, we briefly introduce EEND-vector clustering and how our method proposed in this section is differentiated from it.

EEND-vector clustering uses EEND for shortly divided blocks and then finds the speaker corresponding between them using speaker embeddings. It is relevant to our method in this thesis, but some differences exist between them. One is that EEND-vector clustering requires unique speaker identity labels *over* the recordings in the training set. This means that we must know whether or not a pair of speakers that appeared in different recordings has the same identity. Such information can be easily obtained from simulation data but is not always suitable for real recordings. EEND-GLA only requires the speaker labels *within* for each recording; thus, we can use such real recordings for training. This property is also powerful when conducting, for example, unsupervised or semi-supervised domain adaptation [101]. Another difference is that EEND-vector clustering requires a somewhat high length of blocks (*e.g.*, 30 s) to obtain reliable speaker embeddings to achieve the best performance. However, because the number of output speakers within a block is limited by the network architecture, the length would result in a limited output number of speakers in the final results. Another problem is that the length causes a severe latency if we want to use it for online inference. However, EEND-GLA splits a sequence into short blocks after generating frame-wise embeddings from acoustic features using stacked Transformer encoders. As a result, the frame-wise embeddings can capture the global context, so we can use a lower block length (5 s in this thesis) than EEND-vector clustering.

3.3.2 End-to-End Neural Diarization with Global and Local Attractors

To overcome the limitation of the EDA, we use the nature that the number of speakers speaking in a short period is low. The core idea of the method is that we first conduct attractor-based diarization for each short block and then find inter-block speaker correspondence on the basis of the similarity of the attractors. We call the attractors calculated within each block *local attractors*. Even if the number of speakers within each block is limited owing to EDA, the total number of speakers within

a recording can be higher than the upper bound. Our method also utilizes global-attractor-based diarization just as EEND-EDA does.

Training

Figure 3.6 illustrates the proposed diarization based on global and local attractors, which we call EEND-GLA. The global-attractor-based diarization is identical to EEND-EDA described in Section 3.2; in this section, we introduce local-attractor-based diarization. Given frame-wise embeddings E, we first split them into short blocks, each of which has a length of λ. Here, we assume that the embeddings is split into L blocks, *i.e.*, $E := \begin{bmatrix} E^{(1)} & \cdots & E^{(L)} \end{bmatrix}$, where $E^{(l)} \in \mathbb{R}^{D \times \lambda}$ for $l \in \{1, \dots, L\}$ and $L := \frac{T}{\lambda}$[2]. From the l-th block, local attractors $\boldsymbol{a}_1^{(l)}, \dots, \boldsymbol{a}_{S_l}^{(l)} =: A^{(l)}$ are calculated using (3.5) and (3.2), and the speech activities for the l-th block $\hat{Y}^{(l)} \in (0,1)^{S_l \times \lambda}$ are calculated using (3.6). Here, S_l is the number of speakers that appeared in the l-th block, which satisfies $0 \leq S_l \leq S$. The diarization loss $\mathcal{L}_{\text{diar}}^{(l)}$ and attractor existence loss $\mathcal{L}_{\text{exist}}^{(l)}$ for the l-th block are calculated using (2.15) and (3.8), respectively.

The local attractors are also used for finding inter-block speaker correspondence with clustering. The local attractors themselves are optimized to minimize the diarization error; thus, we convert them by using the following Transformer decoder:

$$B^{(l)} = \mathsf{TransformerDecoder}\left(A^{(l)}, E, E\right) \in \mathbb{R}^{D \times S_l}, \tag{3.17}$$

where the first, second, and third arguments for the Transformer decoder are query, key, and value inputs, respectively. Here, the converted attractors B are expected to be speaker discriminative within each input audio, so we refer to them as relative speaker embeddings. The relative speaker embeddings from all the blocks are gathered $B = [\boldsymbol{b}_i]_i := \begin{bmatrix} B^{(1)}, \dots, B^{(L)} \end{bmatrix} \in \mathbb{R}^{D \times S^*}$ and optimized to minimize the pairwise loss defined as follows:

$$\mathcal{L}_{\text{pair}} = \sum_{i,j \in \{1,\dots,S^*\}} \frac{1}{S^2 c_i c_j} \left(\chi_{ij} \left(1 - \text{sim}\left(\boldsymbol{b}_i, \boldsymbol{b}_j\right)\right) + \left(1 - \chi_{ij}\right) \left[\text{sim}\left(\boldsymbol{b}_i, \boldsymbol{b}_j\right) - \delta\right]_+ \right), \tag{3.18}$$

$$\chi_{ij} = \begin{cases} 1 & \left(\boldsymbol{b}_i \text{ and } \boldsymbol{b}_j \text{ are from the same block}\right) \\ 0 & \left(\text{otherwise}\right) \end{cases}, \tag{3.19}$$

where $S^* := \sum_{l=1}^{L} S_l$ is the total number of local attractors, $\text{sim}\left(\boldsymbol{b}_i, \boldsymbol{b}_j\right) := \frac{\boldsymbol{b}_i^\mathsf{T} \boldsymbol{b}_j}{\|\boldsymbol{b}_i\| \|\boldsymbol{b}_j\|}$ is the cosine similarity between $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, and $[\cdot]_+$ is the hinge function. c_i (c_j) is

[2]For simplicity, we assume that the length of sequence T is divisible by λ, but in practice, the length of the last block can be shorter than λ.

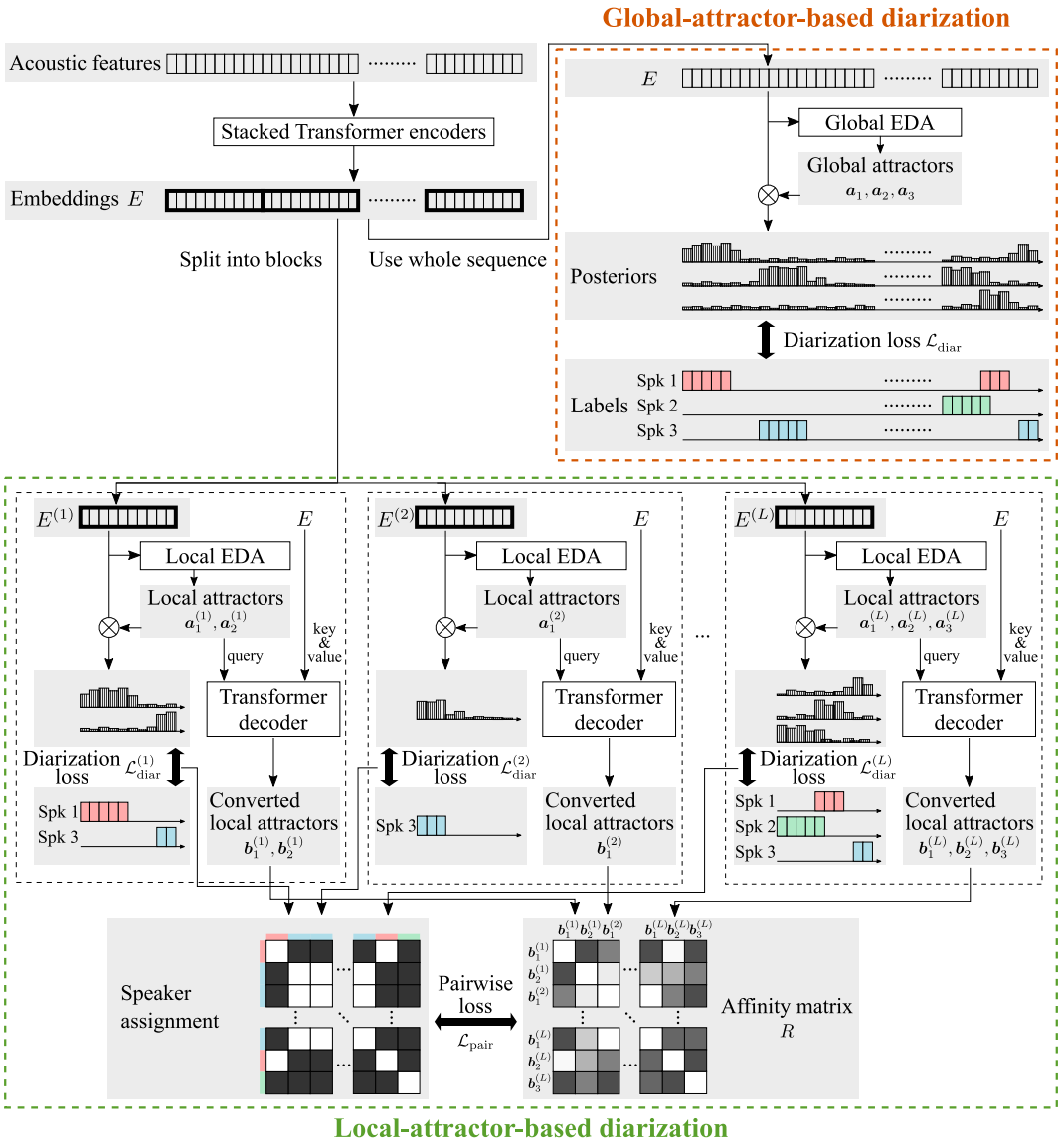


Figure 3.6: End-to-end neural diarization with global and local attractors (EEND-GLA). The attractor existence losses are omitted from the illustration.

the number of local attractors that correspond to the i-th (j-th) attractor's speaker, and this correspondence is obtained by finding the optimal speaker permutation when calculating the diarization loss using (2.15). This pairwise loss aims to make the angle between relative speaker embeddings of the same speaker as small as possible and those of different speakers at least $\arccos \delta$. Note that this loss definition is based on the contrastive loss used for instance segmentation in computer vision [102, 103]. The process of grouping pixel-wise embeddings into instances is very similar to our problem setting of grouping local attractors into speaker identities. While x-vectors or frame-wise embeddings cannot be hardly assigned to one of the speaker identities because of overlaps, the local attractors can be divided by speaker identities because each of them corresponds to one speaker.

As a result, the loss based on local attractors is defined as

$$\mathcal{L}_{\text{local}} = \frac{1}{L} \sum_{l=1}^{L} \left(\mathcal{L}_{\text{diar}}^{(l)} + \alpha \mathcal{L}_{\text{exist}}^{(l)} \right) + \gamma \mathcal{L}_{\text{pair}}, \tag{3.20}$$

where γ is the weighting parameter for which we set $\gamma = 1$ in this study. The total loss of EEND-GLA is defined as a sum of global- and local-attractor-based losses:

$$\mathcal{L}_{\text{both}} = \mathcal{L}_{\text{local}} + \mathcal{L}_{\text{global}}. \tag{3.21}$$

Inference

During inference, the number of speakers within each block $\hat{S}_l \in \mathbb{Z}_{\geq 0}$ is estimated using (3.12), and speech activities of $\hat{S}_l$ speakers are estimated using (3.6). Speaker correspondence between blocks can be found by clustering the relative speaker embeddings B, and the problem here is how to determine the number of clusters.

Some conventional studies [20, 47] were based on the following processes: 1) construct an affinity matrix from frame-wise embeddings, 2) calculate its graph Laplacian, 3) use eigenvalue decomposition, and 4) determine the number of speakers as the value that maximizes the eigengap. Some tricks were used in these studies to reduce the effect of noise in the affinity matrix. In one study, an affinity matrix calculated from frame-wise d-vectors was smoothed by using Gaussian blur [20]. Another study utilized p nearest binarization to the affinity matrix to remove unreliable values [47]. In our case, local attractors are extracted not only for each block but also for each speaker within a block; thus, smoothing cannot be used. In our method, a few local attractors are calculated every five seconds, and hence p nearest neighbor binarization is also not suited because it generally requires dozens of embeddings per cluster.

Therefore, in EEND-GLA, we use the unprocessed affinity matrix to estimate the number of clusters. However, if we estimate it based on the eigengaps of graph

Laplacian, noises cause a lot of tiny clusters because the size of clusters is not considered in this approach. Thus, we use the affinity matrix directly instead of its graph Laplacian to penalize small clusters more. Given the positive-semidefinite affinity matrix $R = \left(r_{ij}\right) \in [-1, 1]^{S^* \times S^*}$, where $r_{ij} = \mathrm{sim}\left(\boldsymbol{b}_i, \boldsymbol{b}_j\right)$, the number of clusters $\hat{S}$ can be estimated using its eigenratios instead of eigengaps as

$$\hat{S} = \min_{1 \leq s \leq S^*-1} \frac{\lambda_{s+1}}{\lambda_s}, \tag{3.22}$$

where $\lambda_1 \geq \cdots \geq \lambda_{S^*}$ are the non-negative eigenvalues of R, which are obtained with matrix decomposition:

$$R = V \,\mathrm{diag}\left(\lambda_1, \ldots, \lambda_{S^*}\right) V^{-1}, \tag{3.23}$$

where each row of $V \in \mathbb{R}^{S^* \times S^*}$ is the eigenvector that corresponds to the eigenvalues. Note that the eigenvalues indicate the number of elements of each cluster where local attractors are softly assigned.

We used the hinge function to calculate the pairwise loss in (3.18), and we also know that attractors from the same block correspond to different speakers. Thus, instead of R, we use the affinity matrix $R' = \left(r'_{ij}\right)$ defined as

$$r'_{ij} = \begin{cases} \mathbb{1}\left(i = j\right) & \left(\boldsymbol{b}_i \text{ and } \boldsymbol{b}_j \text{ are from} \right. \\ & \left. \text{the same block}\right) \\ \frac{1}{1-\delta}\left[\mathrm{sim}\left(\boldsymbol{b}_i, \boldsymbol{b}_j\right) - \delta\right]_+ & (\text{otherwise}) \end{cases}, \tag{3.24}$$

where $\mathbb{1}\left(\mathrm{cond}\right)$ is the indicator function that returns 1 if cond is true and 0 otherwise. Matrix decomposition is then applied to R' to obtain eigenvalues $\lambda'_1 \geq \cdots \geq \lambda'_{S^*}$ as the same manner as in (3.23). Although R' is no longer positive-semidefinite, its eigenvalues are still good indicators of the cluster size. We only use the eigenvalues of not less than one to estimate the number of speakers $\hat{S}$ as follows:

$$\hat{S} = \min_{\substack{1 \leq s \leq S^*-1 \\ \lambda'_s \geq 1}} \frac{\lambda'_{s+1}}{\lambda'_s}. \tag{3.25}$$

Although we set the affinity value between a pair of local attractors from the same block to be zero in (3.24), naive clustering methods cannot force them to be assigned to different clusters. Thus, we utilize a clustering method that can use cannot-link constraints. COP-Kmeans clustering [104], which is used in EEND-vector clustering [100, 99] is one possible choice, but it sometimes cannot find the solution. Thus, we use the CLC-Kmeans algorithm [105], which is the modified version of the COP-Kmeans clustering, for inference of EEND-GLA. To avoid having no solution due to cannot-link constraints, we update the estimated number of

speakers before applying clustering as

$$\hat{S} \leftarrow \max\left(\hat{S}, \max_{1 \leq l \leq L} \hat{S}_l\right). \tag{3.26}$$

EEND-GLA is optimized using both global- and local-attractor-based losses as in (3.21), and we can use not only local attractors but also global attractors for inference. Although local-attractor-based inference can deal with an arbitrary number of speakers, we found that global-attractor-based inference performs better when the number of speakers is low because it is trained in a fully supervised manner. Therefore, we use the results from global and local attractors depending on the estimated number of speakers. Assume that EEND-GLA is trained on mixtures each of which contains at most N speakers. If the estimated number of speakers using global attractors is less than N, we use the inference results based on global attractors. However, if it is equal to or larger than N, we use the inference results based on local attractors.

3.3.3 Experimental Settings

The initial training of each EEND-based model was based on the simulated mixtures shown in Table A.1. We first trained each EEND-based model using Sim2spk from scratch for 100 epochs and then finetuned it using the concatenation of Sim{1,2,3,4}spk for another 50 epochs. The Adam optimizer [92] with Noam scheduler [61] was used during the training using the simulated datasets. For online purposes, the model was adapted using the adaptation set of Sim{1,2,3,4} for an additional 100 epochs using variable chunk-size training (VCT). This time, the Adam optimizer with a fixed learning rate of 1×10^{-5} was used.

We also used the three real datasets shown in Table A.2 for evaluation: CALLHOME, DIHARD II, and DIHARD III. The model pretrained using Sim{1,2,3,4}spk was further adapted to the CALLHOME, DIHARD II, and DIHARD III datasets, respectively. The adaptation was conducted for another 100 epochs using the Adam optimizer with a learning rate of 1×10^{-5}.

For EEND-GLA, we used four- or six-stacked Transformer encoders, each outputting 256-dimensional embeddings. We call each EEND-GLA-Small and EEND-GLA-Large, respectively. For the inputs to the models, 345-dimensional acoustic features extracted for each 100 ms were used, and they were obtained in the following steps: 1) extract 23-dimensional log-mel filterbanks for every 10 ms, 2) apply frame splicing (± 7 frames), and 3) subsample by a factor of 10.

For evaluating offline diarization, we utilized several cascaded methods [4, 106,

49, 107, 67] and end-to-end methods [71, 72, 83, 99] for comparison. For the evaluation protocol, we used DERs. Following the previous studies [72], we forgave 0.25 s of its collar tolerance in the evaluations of the simulated datasets and CALLHOME, while we did not allow such a collar in the evaluations of the DIHARD II and DIHARD III datasets.

3.3.4 Results

Simulated dataset

We first evaluated EEND-GLA on the simulated datasets. The results are shown in Table 3.12.

In the evaluation of offline processing, EEND-based methods outperformed the x-vector clustering baseline in the Kaldi recipe[3]. EEND-EDA and EEND-GLA-Small performed evenly on the datasets of the seen number of speakers, while EEND-GLA-Small significantly outperformed EEND-EDA on the datasets of the unseen number of speakers. It clearly showed that EEND-GLA-Small could deal with a higher number of speakers than that observed during training by introducing clustering. It is worth mentioning that EEND-EDA sometimes outputs more than four attractors, but the results in Table 3.12 in which ignoring the fifth and subsequent attractors improved the DERs indicate that these attractors were not correctly calculated to represent the fifth and subsequent speakers. Using EEND-GLA-Large improved the DERs for the seen number of speakers, but those for the unseen number of speakers were degraded. We considered this to be because the network was overtrained on the seen number of speakers with the larger model. For comparison, we also showed the DERs of EEND-EDA trained using mixtures, each of which contained at most five speakers. It showed a better DER on five-speaker mixtures, but the DER on six-speaker mixtures degraded rapidly. EEND-GLA achieved DERs comparable to EEND-EDA for five-speaker mixtures and significantly outperformed it for six-speaker mixtures.

Table 3.13 shows the DERs of EEND-GLA-Small obtained with various training and inference strategies. Even when only local attractors were used during both training and inference, it achieved better DERs than EEND-EDA for the unseen numbers of speakers but worse ones for the seen numbers of speakers (first row). Using the global attractors jointly for training improved the performance for the seen numbers of speakers, but it was still not as good as EEND-EDA when only the local attractors were used for inference (second row), especially when the number of speakers was low (*i.e.*, one- or two-speaker cases). This is because a small error

[3] `https://github.com/kaldi-asr/kaldi/tree/master/egs/callhome_diarization/v2`

Table 3.12: DERs (%) on the simulated datasets with 0.25 s collar tolerance.

	# of speakers					
	seen				unseen	
	1	2	3	4	5	6
X-vector clustering	37.42	7.74	11.46	22.45	31.00	38.62
EEND-EDA [71, 72]	0.15	3.19	6.60	9.26	23.11	34.97
EEND-EDA [71, 72] †	0.15	3.19	6.60	8.68	22.43	33.28
EEND-GLA-Small	0.25	**3.53**	6.79	8.98	**12.44**	**17.98**
EEND-GLA-Large	**0.09**	3.54	**5.74**	**6.79**	12.51	20.42
EEND-EDA [71, 72] ‡	0.36	3.65	7.70	9.97	11.95	22.59

† Four attractors were used at most.
‡ Trained on Sim{1,2,3,4,5}spk. Five attractors were used at most.

Table 3.13: Offline DERs (%) of EEND-GLA-Small with various training and inference strategies. Loss: the training objective used for training. Inference: attractors used during inference.

		# of speakers					
		seen				unseen	
Loss	Inference	1	2	3	4	5	6
$\mathcal{L}_{\text{local}}$ (3.20)	Local	8.85	12.71	10.31	11.14	14.11	19.36
$\mathcal{L}_{\text{local}} + \mathcal{L}_{\text{global}}$ (3.21)	Local	2.84	10.21	7.54	9.08	**12.40**	18.03
$\mathcal{L}_{\text{local}} + \mathcal{L}_{\text{global}}$ (3.21)	Local & Global	**0.25**	**3.53**	**6.79**	**8.98**	12.44	**17.98**

in the number of speakers (*e.g.*, ± 1) led to a high degradation of DER. Using the results based on global attractors when the number of speakers was low resulted in good DERs for both seen and unseen numbers of speakers (third row).

We also show the confusion matrices for speaker counting on the simulated datasets in Table 3.14. The speaker counting accuracy of EEND-GLA-Small with BW-STB outperformed that of EEND-EDA with FW-STB, and the gaps between them were larger especially when the number of speakers was higher than four. Note that EEND-EDA with FW-STB sometimes produced the results of more than four speakers, but they did not help estimate the speech activities of more than four speakers correctly as we stated in this section.

Table 3.14: Confusion matrices for speaker counting on the simulated datasets.

(a) EEND-EDA

		Ref. # of speakers					
		1	2	3	4	5	6
Pred. # of speakers	1	**500**	0	0	0	0	0
	2	0	**482**	0	0	0	0
	3	0	17	**435**	5	1	0
	4	0	1	65	**447**	224	139
	5	0	0	0	48	**268**	337
	6	0	0	0	0	7	**24**
	7+	0	0	0	0	0	0

(b) EEND-GLA-Small

		Ref. # of speakers					
		1	2	3	4	5	6
Pred. # of speakers	1	**498**	0	0	0	0	0
	2	2	**474**	0	0	0	0
	3	0	25	**451**	17	2	1
	4	0	1	33	**412**	78	30
	5	0	0	10	62	**361**	183
	6	0	0	6	7	47	**229**
	7+	0	0	0	2	12	57

Table 3.15: DERs (%) of various offline diarization methods on CALLHOME with 0.25 s collar tolerance.

	# of speakers					
Method	2	3	4	5	6	All
VBx [4] †	9.44	13.89	16.05	**13.87**	24.73	13.28
MTFAD [106]	N/A	N/A	N/A	N/A	N/A	14.31
SC-EEND [83]	9.57	14.00	21.14	31.07	37.06	15.75
EEND-EDA (Section 3.2)	7.83	12.29	17.59	27.66	37.17	13.65
EEND-vector clust. [99]	7.94	11.93	16.38	21.21	23.10	12.49
EEND-GLA-Small	**6.94**	**11.42**	14.49	29.76	24.09	11.92
EEND-GLA-Large	7.11	11.88	**14.37**	25.95	**21.95**	**11.84**

† The oracle SAD was used.

CALLHOME

Table 3.15 shows the DERs on the CALLHOME dataset. In the evaluation of offline processing, EEND-GLA-Small and EEND-GLA-Large outperformed the conventional methods with 11.92 % and 11.84 % DERs, respectively.

DIHARD II and III

Table 3.16 shows the results on the DIHARD II dataset. In offline diarization, EEND-GLA-Small and EEND-GLA-Large improved the DERs from EEND-EDA, especially when the number of speakers was higher than four. Compared with the

Table 3.16: DERs (%) on DIHARD II with no collar tolerance.

	# of speakers		
Method	≤ 4	≥ 5	All
BUT system [49]	**21.34**	39.85	27.11
VBx + overlap-aware resegmentation [107]	21.41	**36.93**	**26.25**
EEND-EDA [71]	22.09	47.66	30.07
EEND-GLA-Small	22.24	44.92	29.31
EEND-GLA-Large	21.40	43.62	28.33

Table 3.17: DERs (%) on DIHARD III with no collar tolerance.

	# of speakers		
Method	≤ 4	≥ 5	All
VBx + overlap handling [67]	16.38	42.51	21.47
VBx + overlap-aware resegmentation [107]	15.32	**35.87**	**19.33**
EEND-EDA [71, 72]	15.55	48.30	21.94
EEND-GLA-Small	14.39	44.32	20.23
EEND-GLA-Large	**13.64**	43.67	19.49

cascaded method [97] or the cascaded method incorporated with EEND for post-processing [107], EEND-GLA-Large performed on par with them when the number of speakers was low, but not when the number of speakers was high.

We also show the DERs on the DIHARD III dataset in Table 3.17. The results were almost the same as those of the DIHARD II dataset. EEND-GLA-Large achieved 19.49 % DER in offline diarization, which was as accurate as the best performing conventional method [107].

3.4 Block-Online Speaker Diarization for Unlimited Numbers of Speakers

As introduced in Section 2.3.3, the original speaker-tracing buffer includes a frame-wise selection step to meet the requirement of the buffer length. Hereafter, for sake of distinction, we refer to it as FW-STB. When trying to use FW-STB with EEND-GLA to perform online diarization of an unlimited number of speakers, the frame-wise selection can become a problem if the selected frames are not consecutive in the whole buffer. The FIFO strategy ensures that the frames in the buffer are consec-

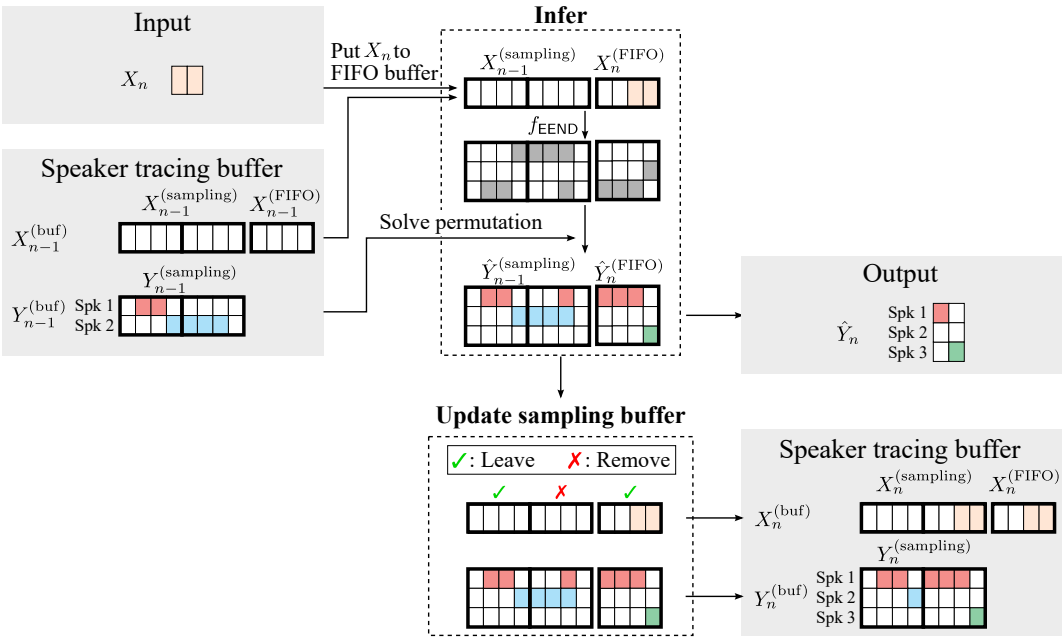


Figure 3.7: Online diarization using speaker-tracing buffer with block-wise update.

utive, but as mentioned in Section 2.3.3, it has difficulty in capturing long context. On the other hand, while the sampling strategy can maintain long-range speaker consistency, the buffer can potentially contain non-consecutive frames of many different speakers; thus, the assumption of a limited number of speakers in a limited sequence of frames in the buffer does not hold. To overcome this dilemma, we propose a block-wise speaker-tracing buffer (BW-STB).

3.4.1 Block-Wise Speaker-Tracing Buffer

The core idea of BW-STB is that it guarantees that the buffer consists of blocks, and each block contains the features and the corresponding results of consecutive frames. If each block in the buffer is short enough that we can assume a limited amount of speakers, EEND-GLA can be used in the same way as the offline inference in which local attractors are obtained from the blocks formed by consecutive frames. However, a naive implementation that waits for block-length features to accumulate and then processes them would result in block-length latency. Thus, we use a frame-wise FIFO buffer and block-wise sampling buffer together to enable a low-latency online inference of EEND-GLA.

Figure 3.7 shows the proposed BW-STB. For simplicity, we assume that the buffer length M is divisible by and longer than the block length λ, and λ is divisible by the online processing unit ν. M-length BW-STB is divided into blocks of

length λ each. The first $\frac{M}{\lambda} - 1$ is updated via block-wise sampling, and the last one is updated in a frame-wise FIFO manner. We call them the sampling buffer and the FIFO buffer, respectively. The features in BW-STB before the n-th input $X_n \in \mathbb{R}^{F \times \nu}$ can be written as

$$X_{n-1}^{(\text{buf})} = \left[X_{n-1}^{(\text{samp})} \quad X_{n-1}^{(\text{FIFO})} \right] \in \mathbb{R}^{F \times M}, \tag{3.27}$$

$$X_{n-1}^{(\text{samp})} = \left[X_{n-1}^{(\text{samp})}[1] \dots X_{n-1}^{(\text{samp})}\left[\tfrac{M}{\lambda} - 1\right] \right] \in \mathbb{R}^{F \times (M-\lambda)}. \tag{3.28}$$

Here, $X_{n-1}^{(\text{samp})}$ are the features in the sampling buffer, where each $X_{n-1}^{(\text{samp})}[k] \in \mathbb{R}^{F \times \lambda}$ $\left(k \in \left\{1, \dots, \frac{M}{\lambda} - 1\right\}\right)$ are the features of consecutive λ frames. $X_{n-1}^{(\text{FIFO})} \in \mathbb{R}^{F \times \lambda}$ is the features in the FIFO buffer, which contains those of the latest consecutive λ frames. In addition, each buffer contains the corresponding diarization results $Y_{n-1}^{(\text{samp})} \in (0,1)^{\hat{S}_{n-1} \times (M-\lambda)}$ and $Y_{n-1}^{(\text{FIFO})} \in (0,1)^{\hat{S}_{n-1} \times \lambda}$.

Given the input X_n, the features in the FIFO buffer are first updated as

$$X_n^{(\text{FIFO})} = X_{n-1}^{(\text{FIFO})} \begin{bmatrix} O_{\nu,\lambda-\nu} & O_{\nu,\nu} \\ I_{\lambda-\nu} & O_{\lambda-\nu,\nu} \end{bmatrix} + X_n \begin{bmatrix} O_{\nu,\lambda-\nu} & I_\nu \end{bmatrix}, \tag{3.29}$$

where I_a is an $a \times a$ identity matrix. Note that the first $\lambda - \nu$ columns of $X_n^{(\text{FIFO})}$ are identical to the last $\lambda - \nu$ columns of $X_{n-1}^{(\text{FIFO})}$, and the last ν columns of $X_n^{(\text{FIFO})}$ are identical to X_n. Then, the diarization results are calculated from the concatenation of the features in the sampling and FIFO buffers as

$$\left[\hat{Y}_{n-1}^{(\text{samp})} \quad \hat{Y}_n^{(\text{FIFO})} \right] = f_{\mathsf{EEND}} \left(\left[X_{n-1}^{(\text{samp})} \quad X_n^{(\text{FIFO})} \right] \right). \tag{3.30}$$

With this estimation, the number of speakers is aligned via zero padding as described in Section 2.3.3, and then the speaker order of $\hat{Y}_{n-1}^{(\text{samp})}$ and $\hat{Y}_n^{(\text{FIFO})}$ is aligned to that of $Y_{n-1}^{(\text{samp})}$ using (2.31)–(2.33). Next, we output the last ν columns of the updated $Y_n^{(\text{FIFO})}$, which correspond to the input X_n.

The sampling buffer is updated every time the FIFO buffer is fully replaced, *i.e.*, after processing the n-th input where $n \equiv 0 \mod \frac{\lambda}{\nu}$. During updates, $\frac{M}{\lambda} - 1$ blocks are selected from $X_{n-1}^{(\text{samp})}[1] \dots X_{n-1}^{(\text{samp})}\left[\frac{M}{\lambda} - 1\right]$ and $X_n^{(\text{FIFO})}$, and they are stored as $X_n^{(\text{samp})}$ in the sampling buffer. The sampling probability of each block is calculated as a sum of $\tilde{\omega}_t$ of the frames in the block calculated using (2.38).

With the aforementioned BW-STB, the online inference having the algorithmic latency of ν ($\ll \lambda$) is enabled. Note that online diarization is performed using the FIFO buffer in the same way as FW-STB from the first to $\frac{\lambda}{\nu}$-th iterations because the sampling buffer is empty.

Table 3.18: Example of sampling weights determined by (2.36) and (3.31).

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
$y_{1,t}$	0.999	0.999	0.999	0.999	0.999	0.001	0.001	0.001
$y_{2,t}$	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.999
$\tilde{\omega}_t$ by (2.38)(2.36)	0.167	0.167	0.167	0.167	0.167	0.000	0.000	0.167
$\tilde{\omega}_t$ by (2.38)(3.31)	0.101	0.101	0.101	0.101	0.101	0.000	0.000	0.497

3.4.2 Speaker-Balanced Sampling Probabilities

The score in (2.36) is designed to weigh more on frames where a single speaker dominates the conversation; as a result, the speaker-tracing buffer becomes informative enough to solve the speaker permutation ambiguity in (2.31)–(2.32). However, in the case where some speakers dominate the conversation, the buffer contents might be biased toward those speakers, and hence the permutation ambiguity cannot be solved correctly. For example, in the two-speaker example shown in Table 3.18, $\tilde{\omega}_t$ is maximized at $t \in \{1,2,3,4,5,8\}$, where $(y_{1,t}, y_{2,t}) \in \{(0.001, 0.999), (0.999, 0.001)\}$. If $t = 8$ is not selected to be stored in the buffer and the third speaker emerges in the next input, we cannot distinguish between the second and third speakers.

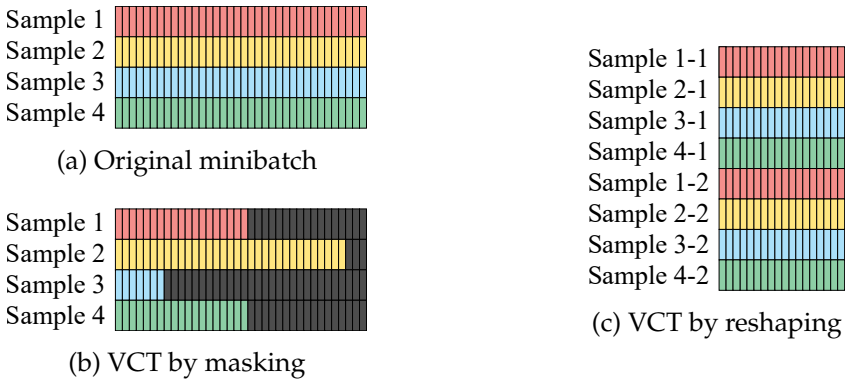
To make the buffer unbiased, we introduce the weighting factor r_t into the sampling probability ω_t to balance the number of frames to be stored for each speaker. We propose the following alternative:

$$\omega_t = r_t \underbrace{\sum_{s=1}^{S_n} \bar{y}_{s,t} \log\left(\bar{y}_{s,t} S_n\right)}_{(2.36)}, \tag{3.31}$$

where r_t is defined as

$$r_t = \sum_{s=1}^{S} \frac{y_{s,t}}{\sum_{t'=1}^{T} y_{s,t'}}. \tag{3.32}$$

By this modification, in Table 3.18, the sampling probability of $t = 8$ becomes about a five times larger value (0.497) than that of $t \in \{1,2,3,4\}$ (0.101); thus, it is more likely to prevent the buffer from storing information that is biased toward the dominant speaker, *i.e.*, the first speaker.



(a) Original minibatch

(b) VCT by masking

(c) VCT by reshaping

Figure 3.8: Batch creation in the VCT.

3.4.3 Variable Chunk-Size Training via Minibatch Reshaping

The VCT described in the last paragraph of Section 2.3.3 varied the length of sequence by masking a part of each sequence (Figure 3.8(b)). However, its calculation efficiency is low because the masked part does not contribute to the network optimization while still consuming GPU memory during training.

Therefore, we consider a method to use inputs of various lengths in the training process by reshaping the minibatch instead of masking. If the minibatch at an iteration has minibatch size B and input length T, we first reshape it to be a new minibatch with the size $B' = \frac{BT}{T'}$ and length T', and then use it for training. For Figure 3.8, the original minibatch has the size of four (Figure 3.8(a)), and the reshaped minibatch has the size of eight by setting $T' = \frac{T}{2}$ (Figure 3.8(c)). In this thesis, we set $B = 64$ and $T = 2000$, and in each iteration, with a probability of 50 %, we set T' to one of $\{50, 100, 200, 500, 1000\}$ to conduct VCT.

3.4.4 Experimental Settings

For online purpose, the models used in Section 3.3 were adapted using variable chunk-size training (VCT). To evaluate the performance on the simulated datasets, the models trained using Sim{1,2,3,4}spk were adapted using the adaptation set of Sim{1,2,3,4}spk for an additional 100 epochs. To evaluate the performance on the real datasets, the models trained on the simulated mixtures were adapted to the CALLHOME, DIHARD II, and DIHARD III datasets, respectively. This time, the Adam optimizer with a fixed learning rate of 1×10^{-5} was used.

Unless otherwise specified, the length of an online processing unit ν was set to 1 s, and the buffer length was set to 100 s. The block length λ of the BW-STB was

Table 3.19: Step-by-step improvement in the online inference of EEND-EDA on the CALLHOME dataset. VCT: Variable chunk-size training.

			Buffer length (s)							
	VCT	ω_t	1	2	5	10	20	50	100	∞
FW-STB [2] †	None	(2.36)	N/A	N/A	N/A	26.6	N/A	20.0	19.5	N/A
FW-STB [2]	None	(2.36)	89.79	76.98	43.87	28.13	21.82	19.69	18.54	18.34
FW-STB	Mask	(2.36)	50.56	44.95	27.48	21.53	18.12	16.82	16.78	15.69
FW-STB	Reshape	(2.36)	44.11	36.61	25.41	20.54	18.11	16.20	15.74	15.00
FW-STB	Reshape	(3.31)	45.60	37.36	24.19	20.10	16.79	15.50	14.93	15.00
BW-STB	Reshape	(3.31)	N/A	N/A	N/A	24.27	16.84	15.03	15.06	15.42

† The values are from the original FW-STB paper [2].

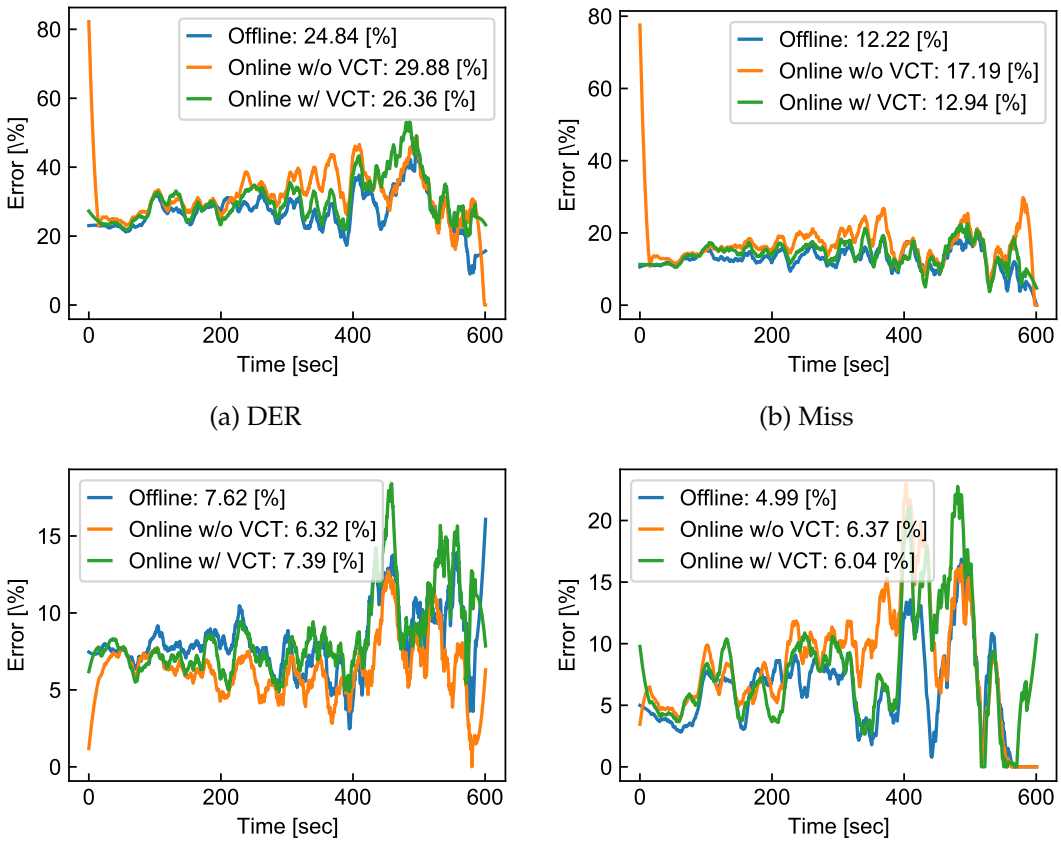
set to 5 s following EEND-GLA [108]; as a result, the length of sampling and FIFO buffers are 95 s and 5 s, respectively.

For evaluating online diarization, we used FW-STB with EEND-EDA based on four-stacked Transformers [2]. In addition, we referred to the results of various conventional online diarization methods [21, 51, 109, 110, 2, 69, 52] on various datasets. Some cascaded comparison methods [21, 51, 110] used the oracle SAD; for a fair comparison, we used SAD post-processing [72] for the results of EEND-based methods to recover missed speech and filter false-alarmed speech.

3.4.5 Evaluation of the Variations of Speaker-Tracing Buffer

Before we dive into the evaluation of EEND-GLA, we evaluated the effects of each modification on the speaker-tracing buffer using EEND-EDA. Step-by-step improvement on the CALLHOME dataset is shown in Table 3.19. The DERs were significantly reduced by using VCT. In a comparison of the results in the third and fourth lines, VCT by reshaping outperformed that by masking in all the conditions. Introducing the speaker-balancing term in the sampling probability as in (3.31) improved the DERs except when the buffer length was too short to store enough information to solve the speaker permutation ambiguity, as in the fifth line. Finally, replacing FW-STB with BW-STB did not affect the diarization performance as shown in the last line, except when the buffer length was 10 s, where the sampling buffer consisted of only one block.

For the detailed error analyses, we show the frame-level breakdown of the diarization error of FW-STB with and without VCT in Figure 3.9. Each graph was smoothed along the time axis using the Savizky-Golay filter [111] for visualization



(a) DER

(b) Miss

(c) False alarm

(d) Speaker confusion

Figure 3.9: Frame-wise breakdown of diarization error on CALLHOME.

purposes. We clearly observed that VCT drastically decreased the error caused by missed speech at the very beginning of recordings with a slight increase in false alarms. Note that the DERs shown in Figure 3.9 are different from those in Table 3.19 because the results are without a collar. The training process took about two weeks with a single NVIDIA® Tesla® V100 GPU.

In the following experiments, we used FW-STB and BW-STB in the last two lines in Table 3.19, *i.e.*, VCT by reshaping and speaker-balanced sampling probabilities were utilized.

3.4.6 Main Results

Table 3.20 shows the DERs of various online diarization method on the simulated mixtures. STB-based methods outperformed BW-EDA-EEND [69] in all but single-speaker data even though the online processing unit was 1 s, which was ten times

Table 3.20: DERs (%) on the simulated datasets with 0.25 s collar tolerance. Unless otherwise specified, each online system had an algorithmic latency of 1 s.

	# of speakers					
	seen				unseen	
Method	1	2	3	4	5	6
BW-EDA-EEND [69][§]	**1.03**	6.10	12.58	19.17	N/A	N/A
EEND-EDA [71, 72] + FW-STB	1.50	5.91	9.79	11.92	26.57	37.31
EEND-EDA [71, 72] + FW-STB [†]	1.50	5.91	9.79	11.85	26.63	37.25
EEND-GLA-Small + BW-STB	1.19	5.18	9.41	13.19	**16.95**	**22.55**
EEND-GLA-Large + BW-STB	1.12	**4.61**	**8.14**	**11.38**	17.27	25.77
EEND-EDA [71, 72] + FW-STB [‡]	1.33	6.01	10.49	12.64	15.28	26.09

[†] Four attractors were used at most.
[‡] Trained on Sim{1,2,3,4,5}spk. Five attractors were used at most.
[§] Algorithmic latency 10 s.

shorter than that of BW-EEND-EDA. Online inference of EEND-GLA-Small using BW-STB significantly improved DERs on five- and six-speaker mixtures, which were not observed during training. EEND-GLA-Large improved the DERs for the seen number of speaker conditions of EEND-GLA-Small but degraded the DERs for the unseen number of speaker conditions, the same as in offline inference.

Simulated datasets

We also show the confusion matrices for speaker counting on the simulated datasets in Table 3.14. The speaker counting accuracy of EEND-GLA-Small with BW-STB outperformed that of EEND-EDA with FW-STB, and the gaps between them were larger especially when the number of speakers was higher than four. Note that EEND-EDA with FW-STB sometimes produced the results of more than four speakers, but they did not help estimate the speech activities of more than four speakers correctly as we stated in this section.

Real datasets

Table 3.22 shows the results on the CALLHOME dataset. Compared with the original FW-STB [2], our updates on VCT and the sampling probabilities improved the DERs from 19.51 % to 14.93 %. EEND-GLA-Small and EEND-GLA-Large with BW-STB further improved DERs to 14.80 % and 14.29 %, respectively. Our method also

Table 3.21: Confusion matrices for speaker counting on the simulated datasets.

(a) EEND-EDA + FW-STB

		Ref. # of speakers					
		1	2	3	4	5	6
Pred. # of speakers	1	**376**	0	0	0	0	0
	2	120	**244**	0	0	0	0
	3	4	249	**252**	1	0	0
	4	0	7	245	**449**	271	172
	5	0	0	3	50	**222**	314
	6	0	0	0	0	7	**14**
	7+	0	0	0	0	0	0

(b) EEND-GLA-Small + BW-STB

		Ref. # of speakers					
		1	2	3	4	5	6
Pred. # of speakers	1	**411**	0	0	0	0	0
	2	84	**343**	0	0	0	0
	3	5	156	**370**	3	0	0
	4	0	1	109	**302**	16	0
	5	0	0	20	181	**364**	38
	6	0	0	1	13	114	**385**
	7+	0	0	0	1	6	77

Table 3.22: DERs (%) of various online diarization methods on CALLHOME with 0.25 s collar tolerance. Unless otherwise specified, each online system had an algorithmic latency of 1 s.

	# of speakers					
Method	2	3	4	5	6	All
BW-EDA-EEND [69] †	11.82	18.30	25.93	N/A	N/A	N/A
EEND-EDA [71, 72] + FW-STB ‡	12.70	18.40	24.30	35.83	42.21	19.51
EEND-EDA [71, 72] + FW-STB	9.08	13.33	19.36	30.09	37.21	14.93
EEND-GLA-Small + BW-STB	**9.01**	12.73	19.45	32.26	36.78	14.80
EEND-GLA-Large + BW-STB	9.20	**12.42**	**18.21**	**29.54**	**35.03**	**14.29**

† Algorithmic latency of 10 s.

‡ The values are from the original FW-STB paper [2].

outperformed BW-EDA-EEND [69] by a large margin.

We also show the results on the DIHARD II dataset in Table 3.23. The DER of EEND-EDA was improved by using the proposed FW-STB from 36.09 % to 33.37 %, and BW-STB further improved the DERs to 31.47 % and 30.24 % with EEND-GLA-Small and EEND-GLA-Large, respectively. If we focus on the comparison methods, overlap-aware speaker embedding [107, 52] had a large gap in the DERs between offline and online inference (26.25 % in Table 3.16 vs. 34.99 % in Table 3.23). This is because its offline performance was highly boosted by using VBx [4], which is not suited for online inference. However, the gap between the DERs of offline and online inference of EEND-GLA was only about two points (29.31 % in Table 3.16 vs. 31.47 % in Table 3.23 with EEND-GLA-Small and 28.33 % in Table 3.16 vs. 30.24 %

Table 3.23: DERs (%) of vairous online diarization methods on DIHARD II with no collar tolerance. Each online system had an algorithmic latency of 1 s.

	# of speakers		
Method	≤ 4	≥ 5	All
Without oracle voice activity detection			
Overlap-aware speaker embeddings [52]	27.00	52.62	34.99
EEND-EDA [71, 72] + FW-STB †	28.14	53.64	36.09
EEND-EDA [71, 72] + FW-STB	25.63	50.45	33.37
EEND-GLA-Small + BW-STB	23.96	48.06	31.47
EEND-GLA-Large + BW-STB	**22.62**	**47.06**	**30.24**
With oracle voice activity detection			
UIS-RNN [21]	N/A	N/A	30.9
UIS-RNN-SML [51]	N/A	N/A	27.3
Core samples selection [109]	N/A	N/A	23.1
EEND-EDA [71, 72] + FW-STB †	17.21	43.58	25.44
EEND-EDA [71, 72] + FW-STB	16.56	42.58	24.67
EEND-GLA-Small + BW-STB	15.29	40.85	23.26
EEND-GLA-Large + BW-STB	**13.55**	**40.39**	**21.92**

† The values are from the original FW-STB paper [2].

in Table 3.23 with EEND-GLA-Large) and outperformed the comparable method in both the cases where the number of speakers was low or high. We also show the DERs with UIS-RNN [21] and UIS-RNN-SML [51], which are based on fully supervised clustering of d-vectors extracted using a sliding window, under the condition that the oracle SAD was used. In this case, too, the EEND-GLA-based methods outperformed these comparable methods.

The results on the DIHARD III dataset show the same trend as in Table 3.24. EEND-GLA-Large achieved 20.73 % DER in online diarization, which was about seven points better than that of the conventional method [52].

3.4.7 Real-Time Factor

To show that our method is applicable for real-time inference, we calculated the real-time factor of EEND-GLA-Small with BW-STB. For the calculation, we used Sim5spk, in which clustering of relative speaker embeddings is always necessary (*c.f.* Table 3.21(b)). The calculation was on an Intel Xeon Gold 6132 CPU @ 2.60 GHz using seven threads without any GPUs. Again, we used the buffer length of

Table 3.24: DERs (%) of various online diarization methods on DIHARD III with no collar tolerance. Unless otherwise specified, each online system had an algorithmic latency of 1 s.

	# of speakers		
Method	≤ 4	≥ 5	All
Without oracle voice activity detection			
Overlap-aware speaker embeddings [52]	21.07	54.28	27.55
EEND-EDA [71, 72] + FW-STB	19.00	50.21	25.09
EEND-GLA-Small + BW-STB	15.87	47.27	22.00
EEND-GLA-Large + BW-STB	**14.81**	**45.17**	**20.73**
With oracle voice activity detection			
System by Zhang *et al*. [110] †	N/A	N/A	19.57
Core samples selection [109]	N/A	N/A	19.3
EEND-EDA [71, 72] + FW-STB	12.80	42.46	18.58
EEND-GLA-Small + BW-STB	9.91	40.21	15.82
EEND-GLA-Large + BW-STB	**8.85**	**38.86**	**14.70**

† Algorithmic latency of 0.5 s.

100 s buffer and the online process unit length of 1 s. Figure 3.10 shows the real-time factor calculated as the processing time for each online process unit. The real-time factor increased approximately linearly until the buffer was filled, and then it became constant. It indicates that, at least for buffer length of 100 s, the inference speed of EEND-GLA is not constrained by clustering of local attractors described in Section 3.3, which has $O(n^3)$ time complexity. The convergence value of the real-time factor was about 0.16 with 10 frames per second and 0.38 with 20 frames per second. These results demonstrate that our method is fast enough for real-time inference.

3.5 Conclusion

In this chapter, we presented i) EEND-EDA to make the output number of speakers flexible, ii) EEND-GLA to remove the upper limit of the output number of speakers of EEND-EDA, and iii) BW-STB to enable online inference of EEND-GLA.

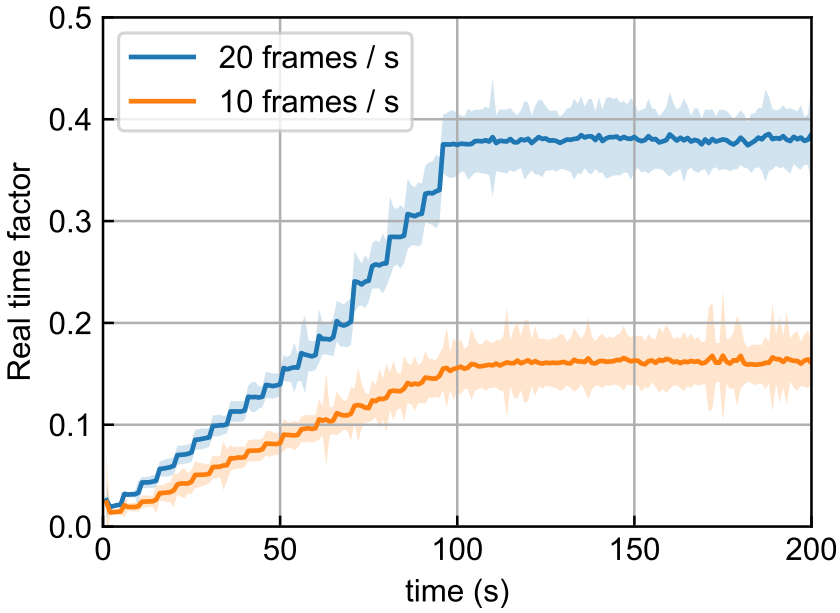


Figure 3.10: Real time factor of EEND-GLA-Small with BW-STB calculated using Sim5spk. The filled areas represent the standard deviations. The DERs are 16.95 % and 18.18 % with 10 frames/s and 20 frames/s conditions, respectively.

Chapter 4

Multi-Channel End-to-End Speaker Diarization

4.1 Introduction

One application of speaker diarization is meeting transcription, which is also tackled in this thesis in Chapter 6. Some meeting transcription systems are based on distributed microphones [27, 112, 28], which enables the flexibility of recording devices and a wide range of sound collection. However, many end-to-end diarization methods have been proposed as single-channel models, where no spatial information is utilized. If we can improve diarization accuracy by extending the diarization methods to distributed microphone settings, it will be compatible with those meeting transcription systems.

Even if multi-channel inputs are given, diarization methods that heavily rely on spatial information are sometimes inoperative. The best examples are direction-of-arrival (DOA) based diarization [56, 55]. Due to COVID-19, meetings are now often held remotely or in a hybrid version of in-person and virtual gatherings. In hybrid meetings, remote attendees' utterances are played via one loudspeaker, and DOA is no longer a clue to distinguish these speakers. To cope with this situation, spatial information needs to be properly incorporated into speaker-characteristic-based speaker diarization. This chapter addresses multi-channel end-to-end speaker diarization (EEND) that is invariant to the number and order of channels for distributed microphone settings.

Section 4.2 proposes multi-channel end-to-end speaker diarization by replacing Transformer encoders in the single-channel models with two types of multi-channel encoders: a spatio-temporal encoder [29, 113] and co-attention encoder.

We further propose to adapt multi-channel EEND only with single-channel real recordings without losing the ability to benefit from spatial information given a multi-channel input during inference.

Section 4.3 proposes a training strategy of single- and multi-channel models to improve the performance of both. We first introduce an end-to-end neural diarization model that can handle both single- and multi-channel inputs. Using this model, we alternately conduct i) knowledge distillation from a multi-channel model to a single-channel model and ii) finetuning from the distilled single-channel model to a multi-channel model.

4.2 Multi-Channel End-to-End Speaker Diarization

In this section, we investigate the effective multi-channel encoders that replace the Transformer encoders in EEND-EDA.

4.2.1 Related Work

Some multi-channel diarization methods are fully based on DOA estimation [56, 55], but assume that different speakers are not in the same direction, thus are not appropriate for hybrid meetings. Therefore, spatial information needs to be incorporated with single-channel-based methods as in *e.g.* [54]. Another possible approach is to combine channel-wise diarization results by using an ensemble method [91, 90], but it does not fully utilize spatial information. Some recent neural-network-based diarization methods utilize spatial information by aggregating multi-channel features. For example, online RSAN [68] uses inter-microphone phase difference features in addition to a single-channel magnitude spectrogram. However, the number of channels is fixed due to the network architecture, making the method less flexible. Moreover, phase-based features are not suited for distributed microphone settings, in which clock drift between channels exists. Multi-channel target-speaker voice activity detection (TS-VAD) [25] combines embeddings extracted from the second from the last layer of single-channel TS-VAD. Although it is flexible in terms of the number of channels because an attention-based combination is used, it requires an external diarization system that gives an initial i-vector estimation for each speaker.

If we broaden our view to speech processing other than diarization, there are several methods for neural-network-based end-to-end multi-channel speech processing that are invariant to the number of channels, *e.g.* speech recognition [114, 115, 116], separation [117, 118, 29, 119, 113], and dereverberation [120]. Many

use attention mechanisms to work with an arbitrary number of channels. Our proposed method also uses attention-based multi-channel processing.

4.2.2 Conventional Single-Channel EEND

While we have already formulated EEND (or EEND-EDA) in the previous chapters, we provide a more simplified formulation here for a better understanding of the following multi-channel EEND.

In the EEND framework, S speakers' speech activities are jointly estimated. Given F-dimensional acoustic features for each T frames $X \in \mathbb{R}^{F \times T}$, we first apply a linear projection parameterized by $W_0 \in \mathbb{R}^{D \times F}$ and $\boldsymbol{b}_0 \in \mathbb{R}^D$ followed by layer normalization [121] LN to obtain D-dimensional frame-wise embeddings

$$E^{(0)} = \mathsf{LN}\left(W_0 X + \boldsymbol{b}_0 \mathbf{1}^\mathsf{T}\right) \in \mathbb{R}^{D \times T}, \tag{4.1}$$

where $\mathbf{1}$ is the T-dimensional all-one vector. It is further converted by N-stacked encoders, where the n-th encoder converts frame-wise embeddings $E^{(n-1)}$ into the same dimensional embeddings $E^{(n)}$:

$$E^{(n)} = \mathsf{Encoder}\left(E^{(n-1)}\right) \in \mathbb{R}^{D \times T}. \tag{4.2}$$

Finally, the frame-wise posteriors of speech activities for S speakers are estimated. In this thesis, we used EEND-EDA [71, 72], with which the speaker-wise attractor B is first calculated using an encoder-decoder based attractor calculation module (EDA) and then the posteriors Y are estimated as

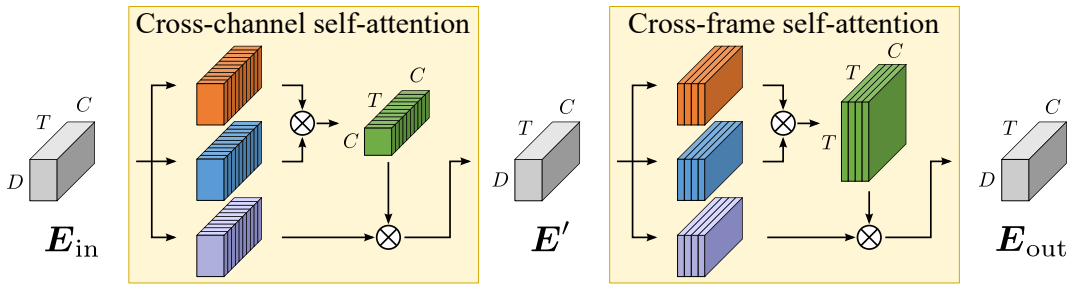
$$B = \mathsf{EDA}\left(E^{(N)}\right) \in \mathbb{R}^{D \times S}, \tag{4.3}$$

$$Y = \sigma\left(B^\mathsf{T} E^{(N)}\right) \in (0,1)^{S \times T}, \tag{4.4}$$

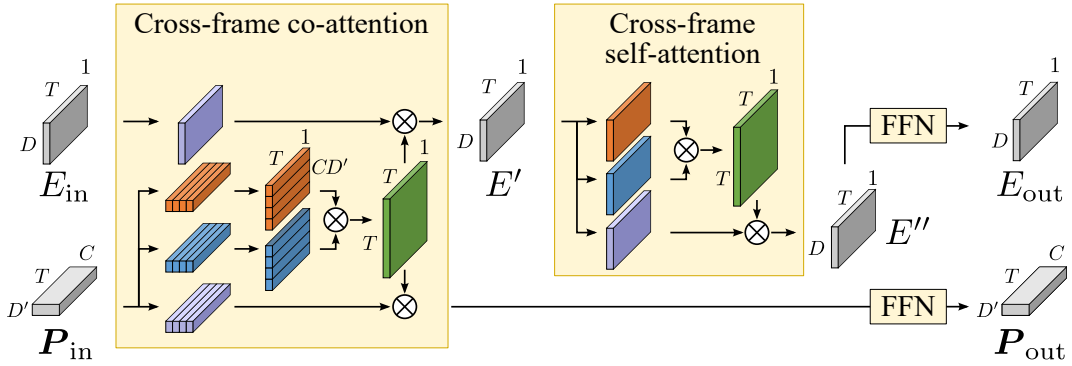
where $\sigma(\cdot)$ is the element-wise sigmoid function. A permutation-free objective is used for optimization as in previous studies [23, 71, 72].

4.2.3 Multi-Channel EEND

To accept multi-channel inputs, we replaced Transformer encoders in EEND-EDA with multi-channel encoders. In this section, we investigated two types of encoders: spatio-temporal encoder and co-attention encoder.



(a) Spatio-temporal encoder



(b) Co-attention encoder

Figure 4.1: Multi-channel encoders. Each yellow area is skipped via residual connection.

Spatio-temporal encoder

The spatio-temporal encoder was originally proposed for speech separation on the basis of distributed microphones [29, 113]. It uses stacked cross-channel and cross-frame self-attentions in one encoder block, as illustrated in Figure 4.1(a). In the encoder, frame-wise C-channel embeddings $\boldsymbol{E}_{\text{in}} = (\boldsymbol{e}_{\text{in},t,c})_{t,c} \in \mathbb{R}^{D\times T\times C}$, where $\boldsymbol{e}_{\text{in},t,c} \in \mathbb{R}^{D}$, are first converted to the same shape of tensor $\boldsymbol{E}' = (\boldsymbol{e}'_{t,c})_{t,c} \in \mathbb{R}^{D\times T\times C}$ using cross-channel self-attention as

$$[\boldsymbol{e}'_{t,1}, \ldots, \boldsymbol{e}'_{t,C}] = \mathsf{LN}\left(E_{\text{in},t} + \mathsf{MA}\left(E_{\text{in},t}, E_{\text{in},t}, E_{\text{in},t}; \boldsymbol{\Theta}, \boldsymbol{\Phi}\right)\right), \tag{4.5}$$

$$E_{\text{in},t} \coloneqq [\boldsymbol{e}_{\text{in},t,1}, \ldots, \boldsymbol{e}_{\text{in},t,C}]. \tag{4.6}$$

The tensor $\boldsymbol{E}'$ is then converted to $\boldsymbol{E}_{\text{out}} = (\boldsymbol{e}_{\text{out},t,c})_{t,c} \in \mathbb{R}^{D\times T\times C}$ by cross-frame self-attention as

$$[\boldsymbol{e}_{\text{out},1,c}, \ldots, \boldsymbol{e}_{\text{out},T,c}] = \mathsf{LN}\left(E'_c + \mathsf{MA}\left(E'_c, E'_c, E'_c; \boldsymbol{\Theta}', \boldsymbol{\Phi}'\right)\right), \tag{4.7}$$

$$E'_c \coloneqq [\boldsymbol{e}'_{1,c}, \ldots, \boldsymbol{e}'_{T,c}]. \tag{4.8}$$

In the final encoder block, cross-frame self-attention is calculated over the embeddings that are averaged across channels to form $E^{(N)}$ in (4.3), *i.e.*, the following are

used instead of (4.7) and (4.8) as

$$E^{(N)} = \mathsf{LN}\left(E' + \mathsf{MA}\left(E', E', E'; \boldsymbol{\Theta}', \boldsymbol{\Phi}'\right)\right), \tag{4.9}$$

$$E' \coloneqq \frac{1}{C}\sum_{c=1}^{C} E'_c \tag{4.10}$$

to calculate speech activities using (4.3) and (4.4). All calculations using (4.5)–(4.10) do not involve a specific number of channels or microphone geometry, which makes this encoder independent of the number and geometry of microphones. Note that we did not include feed-forward networks FFN in this encoder following previous studies [29, 113] because we observed performance degradation.

Co-attention encoder

The spatio-temporal encoder includes cross-channel self-attention, the performance of which highly depends on the number of channels. Therefore, we also propose an encoder based only on cross-frame attention, which is characterized by the use of co-attention. The encoder accepts two inputs: frame-wise embeddings $E_{\text{in}} \in \mathbb{R}^{D\times T}$ and frame-channel-wise embeddings $\boldsymbol{P}_{\text{in}} = [P_{\text{in},1}, \ldots, P_{\text{in},C} \mid P_{\text{in},c} \in \mathbb{R}^{D'\times T}]$. The proposed encoder converts these inputs to $E_{\text{out}} \in \mathbb{R}^{D\times T}$ and $\boldsymbol{P}_{\text{out}} = [P_{\text{out},1}, \ldots, P_{\text{out},C} \mid P_{\text{out},c} \in \mathbb{R}^{D'\times T}]$ as follows:

$$E' = \mathsf{LN}\left(E_{\text{in}} + \mathsf{MCA}\left(\boldsymbol{P}_{\text{in}}, \boldsymbol{P}_{\text{in}}, E_{\text{in}}; \boldsymbol{\Theta}_P, \boldsymbol{\Phi}_E\right)\right), \tag{4.11}$$

$$E'' = \mathsf{LN}\left(E' + \mathsf{MA}\left(E', E', E'; \boldsymbol{\Theta}'_E, \boldsymbol{\Phi}'_E\right)\right) \tag{4.12}$$

$$E_{\text{out}} = \mathsf{LN}\left(E'' + \mathsf{FFN}\left(E''; \boldsymbol{\Psi}_E\right)\right), \tag{4.13}$$

$$P'_c = \mathsf{LN}\left(P_{\text{in},c} + \mathsf{MCA}\left(\boldsymbol{P}_{\text{in}}, \boldsymbol{P}_{\text{in}}, P_{\text{in},c}; \boldsymbol{\Theta}_P, \boldsymbol{\Phi}_P\right)\right), \tag{4.14}$$

$$P_{\text{out,c}} = \mathsf{LN}\left(P'_c + \mathsf{FFN}\left(P'_c; \boldsymbol{\Psi}_P\right)\right), \tag{4.15}$$

where $\boldsymbol{\Theta}_P$, $\boldsymbol{\Phi}_E$, $\boldsymbol{\Theta}'_E$, $\boldsymbol{\Phi}'_E$, $\boldsymbol{\Psi}_E$, $\boldsymbol{\Psi}_P$, $\boldsymbol{\Phi}_P$, and $\boldsymbol{\Psi}_P$ are the sets of parameters. The single-channel input E_{in} is converted by multi-head co-attention MCA in (4.11), multi-head attention MA in (4.12), and feed-forward network FFN in (4.13). Each channel in the multi-channel input $\boldsymbol{P}_{\text{in}}$ is first converted by MCA in (4.14), the attention weights of which are shared with those in (4.11), then processed using FFN in (4.15).

Multi-head co-attention MCA is similar to MA in (2.19), but the attention weights are calculated using multi-channel inputs as

$$\mathsf{MCA}\left(\boldsymbol{Q}, \boldsymbol{K}, V; \boldsymbol{\Theta}, \boldsymbol{\Phi}\right) = W_O \begin{bmatrix} V^{(1)}A^{(1)\mathsf{T}} \\ \vdots \\ V^{(h)}A^{(h)\mathsf{T}} \end{bmatrix} + \boldsymbol{b}_O \mathbf{1}^{\mathsf{T}} \in \mathbb{R}^{d_v\times T}, \tag{4.16}$$

$$A^{(i)} = \mathsf{softmax}\left(\frac{\left[Q_1^{(i)\mathsf{T}}, \ldots, Q_C^{(i)\mathsf{T}}\right]\left[K_1^{(i)\mathsf{T}}, \ldots, K_C^{(i)\mathsf{T}}\right]^{\mathsf{T}}}{\sqrt{CD/h}}\right). \tag{4.17}$$

Here, $Q_c^{(i)}$ and $K_c^{(i)}$ for $c \in \{1, \ldots, C\}$ are calculated using (2.21) and (2.22) for each channel, and $V^{(i)}$ are calculated using (2.23). Note that the parameter sets $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ are shared among channels.

After the final encoder block, two outputs are concatenated as

$$E^{(N)} = \begin{bmatrix} E_{\text{out}} \\ \frac{1}{C}\sum_{c=1}^{C} P_{\text{out},c} \end{bmatrix} \in \mathbb{R}^{(D+D')\times T} \tag{4.18}$$

to calculate speech activities using (4.3) and (4.4).

Domain adaptation

EEND performance can be improved by domain adaptation using real recordings. However, the number of real recordings is usually limited, and even more the case when distributed microphones are used. Therefore, it would be useful if multi-channel EEND can be adapted to the target domain only with single-channel recordings. To ensure that adaptation using only single-channel recordings does not lose the ability to benefit from multi-channel recordings, we propose to update only the channel-invariant part of the model. For the spatio-temporal encoder, we freeze the parameters of cross-channel self-attention $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ in (4.5). For the co-attention encoder, we freeze the parameters related to multi-channel processing: $\boldsymbol{\Theta}_P$ in (4.11) and (4.14), $\boldsymbol{\Phi}_P$ in (4.14), and $\boldsymbol{\Psi}_P$ in (4.15).

4.2.4 Experimental Settings

For the evaluation, we used Sim2spk-multi-train, Sim2spk-multi-eval, and Sim2-spk-multi-dialog as simulated datasets, and CSJ-multi-train, CSJ-multi-eval, CSJ-multi-dialog as real-recorded datasets. Refer to Section A.1.2 for further details of the datasets. Note that we did not consider situations where the speakers are moving around.

As inputs to a single-channel baseline model [71, 72], 23-dimensional log-mel filterbanks were extracted for each 10 ms followed by splicing ($\pm$7 frames) and subsampling by factor of 10, resulting in 345-dimensional features for each 100 ms. For the spatio-temporal model, we extracted features from each channel in the same manner. For the co-attention model, the 345-dimensional features were averaged across channels to be used as the single-channel input. As the multi-channel input, the log-mel filterbanks of $\pm$7 frames were averaged followed by subsampling; thus, a 23-dimensional feature was obtained for each 100 ms. We set the embedding dimensionalities as $D = 256$ and $D' = 64$, *i.e.*, 345-dimensional features were first

Table 4.1: DERs on Sim2spk-multi-eval and Sim2spk-multi-hybrid.

	Sim2spk-multi-eval					Sim2spk-multi-eval-hybrid				
Method	1ch	2ch	4ch	6ch	10ch	1ch	2ch	4ch	6ch	10ch
1ch + posterior avg.	5.13	4.60	4.31	4.19	4.10	6.07	5.68	5.42	5.38	**5.33**
Spatio-temporal	32.86	2.97	**1.49**	**1.19**	**1.03**	34.73	10.60	8.65	8.36	8.21
Spatio-temporal†	6.34	3.02	1.56	1.28	1.07	8.11	8.23	6.98	6.72	6.40
Co-attention	7.23	2.83	1.85	1.59	1.50	9.03	7.53	6.82	6.51	6.65
Co-attention†	**4.68**	**2.52**	1.71	1.40	1.23	**5.73**	**5.34**	**5.05**	**5.18**	5.35

† Channel dropout was used during training.

converted to 256 dimensional via (4.1) and 23-dimensional features were converted to 64 dimensional in the same manner. For each model, the four encoder blocks illustrated in Figure 4.1 were stacked.

Each model was first trained on Sim2spk-multi-train for 500 epochs with the Adam optimizer [92] using Noam scheduler [61] with 100,000 warm-up steps. At each iteration, four of ten channels were randomly selected and used for training. The models were then evaluated on Sim2spk-multi-eval and Sim2spk-multi-eval-hybrid using $\{1, 2, 4, 6, 10\}$-channel inputs. Each model was further adapted to CSJ-multi-train for 100 epochs with the Adam optimizer with a fixed learning rate of 1×10^{-5}. The adapted models were evaluated on CSJ-multi-eval and CSJ-multi-dialog using $\{1, 2, 4, 6, 9\}$-channel inputs. To evaluate the conventional EEND-EDA [71, 72] with multi-channel inputs, we first found the optimal speaker permutation between results from each channel by using correlation coefficients of posteriors and then averaged the posteriors among channels. To prevent the models from being overly dependent on spatial information, we also introduce channel dropout, in which multi-channel inputs are randomly dropped to be a single channel. The ratio of channel dropout was set to 0.1. Each method was evaluated using diarization error rates (DERs) with 0.25 s of collar tolerance.

All the experiments were based on two-speaker mixtures because our scope was investigating multi-channel diarization. Note that EEND-EDA can also be used when the number of speakers is unknown [71, 72].

4.2.5 Results

Table 4.1 shows the DERs on Sim2spk-multi-eval and Sim2spk-multi-eval-hybrid. From the results on Sim2spk-multi-eval, both spatio-temporal and co-attention models outperformed the single-channel model with posterior averaging. Com-

Table 4.2: DERs on CSJ-multi-eval and CSJ-multi-dialog.

		CSJ-multi-eval					CSJ-multi-dialog				
Method	Adapt	1ch	2ch	4ch	6ch	9ch	1ch	2ch	4ch	6ch	9ch
1ch + posterior avg.	None	11.17	9.44	8.94	8.89	8.44	28.15	26.01	25.56	24.74	24.87
1ch + posterior avg.	1ch	3.27	2.31	2.25	2.05	1.75	22.56	20.82	20.34	19.68	20.25
Spatio-temporal	None	10.98	10.20	4.29	3.27	2.63	36.13	45.19	36.48	37.14	37.63
Spatio-temporal	1ch	3.44	1.60	1.34	1.07	1.13	**20.06**	20.02	17.83	16.19	19.74
Spatio-temporal	1ch‡	3.64	1.78	1.64	1.27	1.32	20.57	19.02	17.37	15.49	18.70
Spatio-temporal	4ch	3.82	**1.06**	0.61	0.43	**0.39**	21.01	**15.87**	**14.21**	15.71	14.20
Co-attention	None	9.49	3.36	1.42	1.40	0.94	27.96	22.52	19.37	18.23	17.99
Co-attention	1ch	**2.75**	1.41	0.75	0.63	0.52	23.49	22.83	20.70	17.59	15.77
Co-attention	1ch‡	3.26	1.46	0.68	0.48	0.42	22.45	17.90	15.53	14.34	14.05
Co-attention	4ch	3.31	1.19	**0.57**	**0.40**	**0.39**	21.42	17.51	14.95	**14.21**	**13.87**

‡ Adapted only the channel-invariant part of each model.

paring the two multi-channel models, the spatio-temporal model significantly degraded DER with single-channel inputs. Channel dropout eased the situation, but the co-attention model still outperformed the spatio-temporal model when the number of channels was small. In the evaluation of Sim2spk-multi-eval-hybrid, the co-attention model always achieved the same or better DERs than the single-channel model. This means that the lack of spatial information does not lead to degradation in diarization performance in the co-attention model because it does not rely on cross-channel self-attention.

Table 4.2 shows the DERs on CSJ-multi-eval and CSJ-multi-dialog. The evaluation was based on the models trained using channel dropout. Without adaptation, we can see that the co-attention model generalized well. The performance of all models improved through adaptation, regardless of whether the data used for adaptation were 1ch or 4ch. Of course, both spatio-temporal and co-attention models can benefit more from 4ch adaptation; however, it is worth mentioning that they can still utilize spatial information provided by multi-channel inputs even if only 1ch recordings are used for adaptation. By freezing the parameters related to the calculation across channels during 1ch adaptation, the DERs of the co-attention model were reduced especially when four or more microphones were used, while those of the spatio-temporal model were not so improved.

Finally, we show the peak VRAM usage with $T = 500$ and batch size of 64 in Figure 4.2. VRAM usage of the co-attention model increased more slowly than the spatio-temporal model as the number of microphones increased because the multi-channel processing part is based on layers with a lower number of units. Thus, the co-attention model can be trained using a larger number of channels.

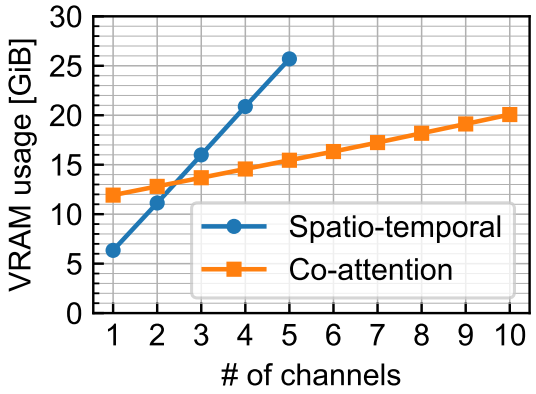


Figure 4.2: VRAM usage during training with $T = 500$ and batch size of 64.

4.3 Mutual Learning of Single-Channel and Multi-Channel End-to-End Speaker Diarization

4.3.1 Introduction

Speech processing under noisy and reverberant environments or the existence of multiple speakers expands the practicality of speech applications. While single-channel solutions for such conditions are widely studied, multi-channel approaches have shown promising performance in various speech applications such as speech recognition [122, 114], speech separation [123, 118], speaker recognition [124], and speaker diarization [68, 125, 126]. Especially, multi-channel processing based on distributed microphones rather than microphone-array devices is attracting much attention for its high versatility [126, 27, 117, 127].

Since multi-channel speech processing is powerful, its outputs are sometimes used as teacher labels when training a single-channel model, which is known as knowledge distillation or teacher-student learning [128, 129]. On the other hand, it has been reported that single-channel data is still useful in training multi-channel models, *e.g.*, single-channel pretraining [130, 131, 132] and simultaneous use of single- and multi-channel data [133, 126, 132]. This can be because the information captured by single- and multi-channel models are different. For example, when considering speech separation or speaker diarization, single-channel methods must rely on speaker characteristics, while multi-channel methods can use spatial information additionally (or even only). Another study demonstrated that incorporating spectral and spatial information boosts speech separation performance [134]. Let us consider a multi-channel model that can also handle single-channel inputs. Using single-channel data to train such a multi-channel model avoids falling into local minima that rely too heavily on spatial information and allow the model to

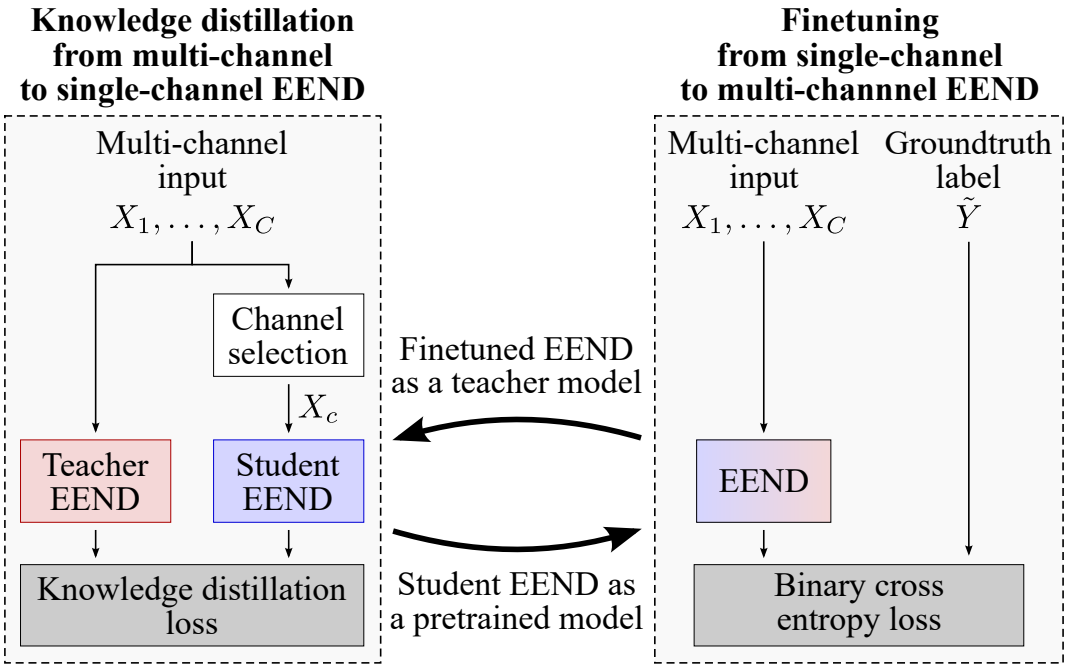


Figure 4.3: Mutual learning of single- and multi-channel EEND.

benefit more from speaker characteristics [126]. Here a research question arises—does iterative knowledge distillation from multi-channel to single-channel model and finetuning from single-channel to multi-channel model improve the performance of both single and multi-channel speech processing?

Given that question as motivation, this section proposes a mutual learning method of single- and multi-channel end-to-end neural diarization (EEND), illustrated in Figure 4.3. We focus specifically on speaker diarization here, but the method can be applied to other speech processing tasks such as speech recognition and separation. We first introduce a co-attention-based multi-channel EEND model invariant to the number and geometry of microphones. The multi-channel model is designed to be identical to the conventional Transformer-based single-channel EEND given single-channel inputs. We conduct the following processes iteratively: i) distilling the knowledge from multi-channel EEND to single-channel EEND (Figure 4.3 left) and ii) finetuning from the distilled single-channel EEND to multi-channel EEND (Figure 4.3 right)[1]. We demonstrate that the proposed method mutually improves both single- and multi-channel speaker diarization performance.

[1]The proposed method can also be applied to two multi-channel models, u-channel and v-channel models with $u < v$.

4.3.2 Related Work

Knowledge distillation or teacher-student learning is a scheme to train a student model to mimic a well-trained teacher model [128, 129]. It is widely used in speech applications such as speech recognition [135] and separation [136]. One typical use case is knowledge distillation between different network architectures: a large model to a small model [135, 137], a normal model to a binarized model [136], an ensemble of models to a single model [138], and a high-latency model to a streaming model [139].

The other type of knowledge distillation, which we focus on in this chapter, is based on different inputs, while the network architectures are not necessarily different. In some studies on unsupervised domain adaptation of speech recognition [140] and speaker verification [21], a far-field model is trained with knowledge distillation by using a close-talk model as a teacher, both of which take single-channel signals as inputs. Another series of studies leverage multi-channel signals; a student model is trained so that the output when the noisy features are input is close to the output of the teacher model when the enhanced features are input [141, 142]. Here, the enhanced features are calculated from a multi-channel signal using beamforming and the input to the model is still single-channel; thus, it is not applicable to speaker diarization where there is more than one speaker to be enhanced. In the context of continuous speech separation [143], a student VarArray model [127] is trained to produce similar outputs to a teacher model with a fewer number of channels [144]. This chapter, in contrast, tackles multi- to single-channel knowledge distillation with an end-to-end model rather than multi- to multi-channel knowledge distillation as in [144].

4.3.3 Proposed Method

This study aims to improve both single- and multi-channel diarization by alternating between knowledge distillation from a multi-channel model to a single-channel model and finetuning from a single-channel model to a multi-channel model (Figure 4.3). To achieve this, we first introduce the EEND model that can handle both multi-channel and single-channel inputs with exactly the same sets of parameters $\boldsymbol{\Theta}$, $\boldsymbol{\Phi}$, and $\boldsymbol{\Psi}$ in Section 4.3.3. Then, we describe a mutual learning method of single- and multi-channel models in Section 4.3.3.

Multi-Channel EEND with the Simplified Co-Attention Encoders

For multi-channel diarization, we used a co-attention-based extension of EEND, which was originally proposed in [126]. In this section, we used the simplified version of the encoder proposed in [126], which has the same number of parameters as the Transformer encoder introduced in Section 2.3.2.

Given frame-wise acoustic features for each of C channels $\boldsymbol{X} := [X_1, \ldots, X_C] \in \mathbb{R}^{F \times T \times C}$, each is first converted using (2.11) to obtain frame-wise embeddings for each channel independently $\left[E_1^{(0)}, \ldots, E_c^{(0)}\right] =: \boldsymbol{E}^{(0)}$. Then, the resulting tensor is processed using N-stacked co-attention encoder. The n-th encoder layer converts $\boldsymbol{E}^{(n-1)} \in \mathbb{R}^{D \times T \times C}$ into a tensor of the same shape $\boldsymbol{E}^{(n)} \in \mathbb{R}^{D \times T \times C}$ as follows:

$$\boldsymbol{E}^{(n)} = \mathsf{CoAttentionEncoder}\left(\boldsymbol{E}^{(n-1)}\right). \tag{4.19}$$

The output from the last encoder $\boldsymbol{E}^{(N)} := \left[E_1^{(N)} \ldots, E_C^{(N)}\right]$ is averaged across channels as

$$E^{(N)} = \frac{1}{C} \sum_{c=1}^{C} E_c^{(N)}, \tag{4.20}$$

and it is used for calculation of speech activities with (3.6).

Then, S speakers' speech activities Y are estimated based on inner products between the frame-wise embeddings from the last encoder $E^{(N)}$ and speaker-wise attractors B as

$$B = \mathsf{EDA}\left(E^{(N)}\right) \in (-1, 1)^{D \times S}, \tag{4.21}$$

$$Z = \left(B^{\mathsf{T}} E^{(N)}\right) \in \mathbb{R}^{S \times T}, \tag{4.22}$$

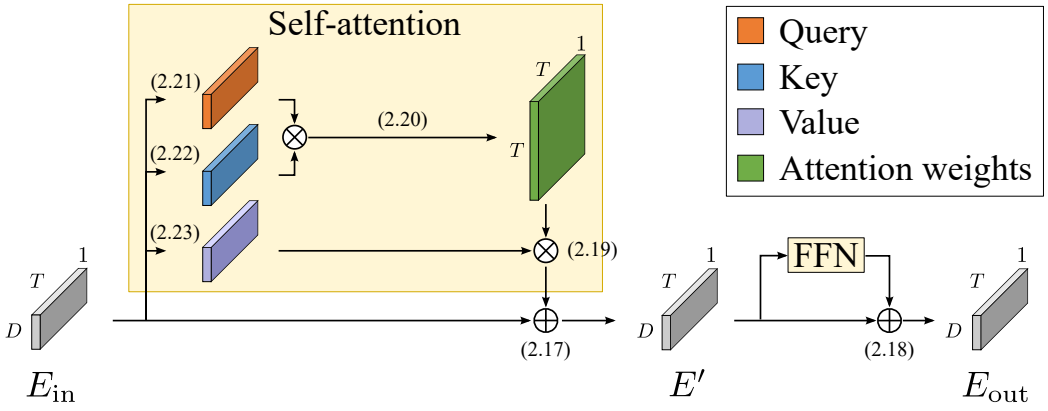
$$Y = \sigma(Z) \in (0, 1)^{S \times T} \tag{4.23}$$

where EDA is an encoder-decoder-based attractor calculation module, $(\cdot)^{\mathsf{T}}$ denotes matrix transpose, and $\sigma(\cdot)$ is an element-wise sigmoid operation. Note that the inner products Z between the embeddings and attractors become logits of the speech activities Y.

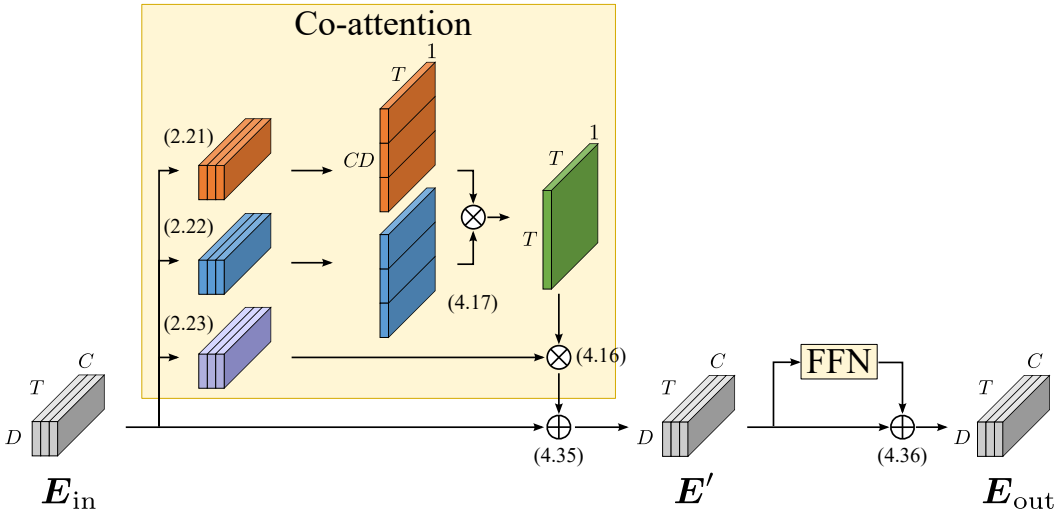
The speech activities are optimized to minimize the permutation-free loss, which is defined as

$$\mathcal{L}_{\text{BCE}}\left(\boldsymbol{\Theta} \mid X, \tilde{Y}\right) = \frac{1}{TS} \min_{P} \mathsf{BCE}\left(\tilde{Y}, PY\right) \tag{4.24}$$

where $P \in \{0, 1\}^{S \times S}$ denotes a $S \times S$ permutation matrix , $\tilde{Y} \in \{0, 1\}^{S \times T}$ is the groundtruth speech activities, and $\mathsf{BCE}(\cdot, \cdot)$ is the summation of the element-wise binary cross entropy. Ξ is a set of parameters of the network.



(a) Transformer encoder (Reprint of Figure 2.1)



(b) Simplified co-attention encoder

Figure 4.4: The architectures of encoders used in this chapter. D: the dimensionality of embeddings, T: sequence length, C: the number of channels.

The simplified co-attention encoder layer in (4.19) was illustrated in Figure 4.4(b). It has the same set of parameters to a Transformer encoder and converts input $\boldsymbol{E}_{\text{in}} = [E_{\text{in},c}]_{c=1}^{C} \in \mathbb{R}^{D\times T\times C}$ into $\boldsymbol{E}_{\text{out}} = [E_{\text{in},c}]_{c=1}^{C} \in \mathbb{R}^{D\times T\times C}$ as

$$E'_c = \mathsf{LN}\left(E_{\text{in},c} + \mathsf{MCA}\left(\boldsymbol{E}_{\text{in}}, \boldsymbol{E}_{\text{in}}, E_{\text{in},c}; \boldsymbol{\Theta}, \boldsymbol{\Phi}\right)\right), \tag{4.25}$$

$$E_{\text{out,c}} = \mathsf{LN}\left(E'_c + \mathsf{FFN}\left(E'_c; \boldsymbol{\Psi}\right)\right). \tag{4.26}$$

Here, MCA is the multi-head scaled dot-product co-attention defined in (4.16) and (4.17).

In multi-channel EEND, only the attention calculation in (4.17) is the inter-channel process, while the other processes are channel-independent. Also, there are no trainable channel-dependent parameters for the attention calculation in (4.17), and the parameters for the channel-independent processes are shared among channels. Therefore, multi-channel EEND is independent of the number and geometry of microphones. Note that when the number of channels is one, the multi-head scaled dot-product co-attention in (4.16) is identical to the multi-head scaled dot-product attention in (2.19), *i.e.*, the co-attention encoder is identical to Transformer encoder. Moreover, both encoders have the same set of parameters, $\boldsymbol{\Theta}$, $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$; thus, even if the model is trained as a single-channel model using Transformer encoders, it can handle multi-channel inputs by considering the attention mechanism as a co-attention in (4.16)–(4.17).

Mutual Learning of Single- and Multi-Channel EEND

In the proposed mutual learning method, given an initial multi-channel model parameterized by $\Xi^{(0)}_{\text{multi}}$, knowledge transfer from multi- to single-channel model and from single- to multi-channel model are iteratively conducted. More specifically, in the r-th round of mutual learning, the following two steps are carried out:

1. Train $\Xi^{(r)}_{\text{single}}$ from scratch on $\mathcal{L}_{\text{KD}}(\Xi \mid \boldsymbol{X}, \Xi^{(r-1)}_{\text{multi}})$,

2. Train $\Xi^{(r)}_{\text{multi}}$ initialized with $\Xi^{(r)}_{\text{single}}$ on $\mathcal{L}_{\text{BCE}}(\Xi \mid \boldsymbol{X}, \tilde{Y})$.

We detail two steps above in Section 4.3.3 and 4.3.3, respectively.

Knowledge transfer from multi- to single-channel EEND

To transfer the knowledge from multi-channel EEND to single-channel EEND, we use a knowledge distillation between network outputs (Figure 4.3 left). According

to the observation in [145], the mean squared error between logits is used as a loss to be minimized instead of Kullback-Leibler divergence. Given C-channel frame-wise acoustic features $\boldsymbol{X} = [X_c]_{c=1}^{C}$, we calculate the logits of frame- and speaker-wise speech activities $Z_{\text{multi}} \in \mathbb{R}^{S\times T}$ by using the teacher multi-channel model parameterized by $\Xi_{\text{multi}}^{(r-1)}$ with (2.11), (4.19), (4.20), (4.21), and (4.22). We also calculate single-channel results $Z_{\text{single}} \in \mathbb{R}^{S\times T}$ from randomly selected one of the C channels by using the student single-channel model with (2.11), (2.12), (4.21), and (4.22). To optimize the student model's parameters Ξ, here also as in (4.24), we introduce the following permutation-free knowledge distillation loss to be minimized:

$$\mathcal{L}_{\text{KD}}\left(\Xi \;\middle|\; \boldsymbol{X}, \Xi_{\text{multi}}^{(r-1)}\right) = \frac{1}{TS}\min_{P}\left\|Z_{\text{multi}} - PZ_{\text{single}}\right\|_F^2, \tag{4.27}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix and $P \in \{0,1\}^{S\times S}$ denotes a permutation matrix. We denote the obtained set of parameters of the student model by $\Xi_{\text{single}}^{(r)}$.

While some studies have investigated the weighted sum of hard-label-based loss $\mathcal{L}_{\text{BCE}}$ in (4.24) and knowledge-distillation-based loss $\mathcal{L}_{\text{KD}}$ in (4.27), we simply used $\mathcal{L}_{\text{KD}}$ instead of $\mathcal{L}_{\text{BCE}}$.

Knowledge transfer from single- to multi-channel EEND

As described in Section 4.3.3, single- and multi-channel models have the same network parameters. Therefore, even when a model is trained only on single-channel data, it can handle multi-channel inputs by using co-attention instead of self-attention. Thus, to transfer the knowledge from a single- to multi-channel model, we simply finetune the single-channel model to a multi-channel model. We initialize the network parameters with $\Xi_{\text{single}}^{(r)}$ obtained in Section 4.3.3 and finetune the model using the loss $\mathcal{L}_{\text{BCE}}$ in (4.24) with multi-channel data. The resulted set of parameters of the model is denoted as $\Xi_{\text{multi}}^{(r)}$.

4.3.4 Experimental Settings

Datasets

To prove the efficiency of the proposed method, we used the simulated two-speaker conversational datasets in Table A.4: Sim2spk-multi-train for training and Sim2spk-multi-eval for evaluation. While both datasets originally contain 10-channel recordings, we only used one- or four-channel subsets of Sim2spk-multi-eval for the evaluation in this section.

With domain adaptation on real datasets, we show that models trained on the simulated dataset using the proposed method are also good pretrained models. As the single-channel datasets, we used CALLHOME-2spk and CSJ [146], each of which is shown in Table A.2. CALLHOME-2spk Part 1 was used as the adaptation set and CALLHOME-2spk Part 2 and CSJ were used as the evaluation sets. We regard CALLHOME-2spk Part 2 as an in-domain dataset and CSJ as an out-of-domain dataset.

As the multi-channel datasets, we used CSJ-multi-train, CSJ-multi-eval, and CSJ-multi-dialog. CSJ-multi-train was used as the adaptation set and CSJ-multi-eval and CSJ-multi-dialog were used as the evaluation sets. Since CSJ-multi-train and CSJ-multi-eval are simulated conversations created using single-speaker recordings while CSJ-multi-dialog is actual conversations, we regard CSJ-multi-eval as an in-domain dataset and CSJ-multi-dialog as an out-of-domain dataset. Note that the original CSJ-multi-eval and CSJ-multi-dialog have nine channels, but we only used four-channel subsets for our experiments.

We conducted our experiments using two-speaker datasets because this chapter is aiming at proving the effectiveness of the proposed mutual learning method, but we note that EEND-EDA can handle the case where the number of speakers is unknown.

Implementation details

The inputs to single- and multi-channel models are 345-dimensional features extracted every 100 ms for each channel, which are prepared with the following procedure:

1. Extract 23-dimensional log-mel filterbanks for each 10 ms,
2. Apply frame splicing (± 7 frames) for them, resulting in 345 dimensions,
3. Subsample them by a factor of 10.

We used a four-layer encoder for each EEND model with the dimensionality of intermediate embeddings $D = 256$, and the number of heads in each encoder $h = 4$. The baseline single- and multi-channel EENDs were trained from scratch using Sim2spk-multi-train. During training, one or four of ten channels were randomly selected and fed to the models, respectively. Each model was optimized for 500 epochs with Adam [92] with the Noam scheduler [61] with warm-up steps of 100,000. For the training of the baseline multi-channel EEND, we used the channel dropout technique [126].

For the knowledge distillation from multi-channel to single-channel models, four channels of Sim2spk-multi-train out of ten were selected and fed to the multi-channel model to obtain Z_{multi} in (4.27), and one of the four channels was input to the single-channel model to obtain Z_{single} in (4.27). The same training strategy used to train the baseline models was used for knowledge distillation. Note that the single-channel model here was trained from scratch.

For the finetuning of the single-channel model using multi-channel data, four of ten channels of Sim2spk-multi-train were randomly selected at each iteration and fed to the model. The model was finetuned for another 100 epochs with Adam using the Noam scheduler with 20,000 warm-up steps.

For the adaptation on the real-recorded datasets, we used CALLHOME-2spk Part 1 for single-channel evaluation or CSJ-multi-train for multi-channel evaluation, respectively. Adam optimizer with a fixed learning rate of 1×10^{-5} was used for 100 epochs of adaptation.

We report diarization error rates (DERs) with 0.25 s collar tolerance. Note that speaker overlaps are included in the evaluation. While conventional studies generally apply a median filter as post-processing, in order to clarify the extent to which frame-level accuracy has been improved, this chapter discusses the results without median filtering. For reference, the results with 11-frame median filtering are also shown in parentheses in each table.

4.3.5 Results

Results on the simulated dataset

Table 4.3 shows the DER improvement using the proposed mutual learning on Sim2spk-multi-eval data. The DERs under mismatched conditions, *i.e.*, single-channel (four-channel) results using the model trained on the four-channel (single-channel) data, are written in gray. For clarification, we denote the DERs evaluated using single-channel Sim2spk-multi-eval as DER_{1ch} and those evaluated using four-channel data as DER_{4ch}.

The first and the second rows show the DERs of the baseline single-channel and four-channel models, respectively. It is clearly observed that the models can still decode data of mismatched conditions, but the resulting DERs are worse than those of matched conditions. Finetuning of the baseline single-channel model on four-channel data improved the DER_{4ch} from 4.11 % to 2.67 % as in the third row, but it did not reach the DER_{4ch} of the baseline four-channel model (2.32 %). These results indicate that single-channel pretraining does not always improve the performance

Table 4.3: DERs (%) improvement on Sim2spk-multi-eval with the proposed method. The values in gray indicate the mismatched condition in the number of channels. The values in the parentheses are with median filtering.

Method	DER_{1ch}	DER_{4ch}
① Baseline 4-ch model	5.79 (4.96)	2.32 (1.98)
② Baseline 1-ch model	4.11 (3.91)	4.90 (4.44)
③ Finetune ② using 4-ch data	12.76 (11.16)	2.67 (2.49)
④ Knowledge distillation from ①	3.34 (3.17)	4.04 (4.40)
⑤ Finetune ④ using 4-ch data	12.11 (9.83)	2.17 (2.02)
⑥ Knowledge distillation from ⑤	3.08 (2.94)	3.57 (3.25)
⑦ Finetune ⑥ using 4-ch data	10.83 (8.73)	2.08 (1.94)

Table 4.4: DERs (%) on single-channel real conversational datasets.

	Evaluation dataset	
Pretrained model	CALLHOME-2spk Part 2	CSJ
②	14.11 (12.56)	24.71 (24.15)
④	10.06 (8.95)	22.52 (22.09)
⑥	9.85 (8.45)	21.33 (20.66)

of the multi-channel model. We decided to use the baseline four-channel model ① as the initial model of mutual learning.

When we trained a single-channel model with the baseline four-channel model as a teacher using knowledge distillation, the DER_{1ch} was improved from 4.11 % to 3.34 % as in the fourth row. Finetuning the model ④ with four-channel data resulted in the DER_{4ch} of 2.17 %, outperforming the baseline four-channel model ①. This bi-directional improvement between single- and multi-channel models has proven the effectiveness of the proposed mutual learning.

We then ran another round of mutual learning starting from the model ⑤. Since the improvement of DER_{4ch} from ① to ⑤ was 0.15 ($= 2.32 - 2.17$) percentage points, the improvement of DER_{1ch} between their distilled models was larger: 0.26 ($= 3.34 - 3.08$). Finetuning the distilled model ⑥ with four-channel data led to a further improvement in DER_{4ch}, which was 2.08 % as in the seventh row.

Table 4.5: DERs (%) on multi-channel real conversational datasets.

	Evaluation dataset	
Pretrained model	CSJ-multi-eval	CSJ-multi-dialog
①	1.19 (0.66)	16.14 (15.78)
③	1.51 (0.93)	18.77 (18.07)
⑤	1.17 (0.67)	17.30 (16.60)
⑦	1.08 (0.66)	16.98 (16.42)

Results on the real datasets

We then evaluated each single-channel model, *i.e.*, ②, ④, and ⑥ in Table 4.3, with the adaptation on CALLHOME-2spk Part 1. The DERs on the CALLHOME-2spk Part 2 and CSJ datasets are shown in Table 4.4. It is clearly observed that pretraining using the proposed mutual learning method also improved the performance on the real datasets, regardless of in-domain or out-of-domain.

Each multi-channel model, *i.e.*, ①, ③, ⑤, and ⑦ in Table 4.3, was also adapted using CSJ-multi-train. The results on CSJ-multi-eval and CSJ-multi-dialog using those adapted models are shown in Table 4.5. For CSJ-multi-eval, the pretrained models trained using the proposed method, *i.e.*, ③, ⑤, and ⑦, helped to improve the DERs gradually with the best DER of 1.08 %, which also outperformed the model based on ①. In the case of CSJ-multi-dialog, the DERs were also reduced with respect to the pretrained model finetuned from the single-channel model, *i.e.*, ③, ⑤, and ⑦. However, the most accurate model was based on ① in terms of CSJ-multi-dialog. Since CSJ-multi-dialog is the out-of-domain dataset, it is more advantageous to rely on spatial information. That is why the model based on ① performed best on CSJ-multi-dialog because ① was trained using multi-channel data from the beginning and is considered to be highly dependent on spatial information.

4.4 Conclusion

This chapter investigated the use of multi-channel inputs for EEND.

First, we proposed a multi-channel end-to-end neural diarization method based on distributed microphones. We replaced Transformer encoders in the conventional EEND with two types of multi-channel encoders. Each showed better DERs with multi-channel inputs than the conventional EEND on both simulated and real-

recorded datasets. We also proposed a model adaptation method using only single-channel recordings and achieved comparable DERs as when using multi-channel recordings.

Second, we proposed a mutual learning method of single- and multi-channel models. With the model that can treat both single- and multi-channel inputs, we alternately execute 1) knowledge distillation from a multi-channel model to a single-channel model and 2) finetuning from the distilled single-channel model to a multi-channel model. Experimental results showed that the proposed method gradually improved DERs on the single- and multi-channel conditions. Future work will apply this method to other speech processing tasks such as speech separation and recognition.

Chapter 5

End-to-End Speaker Diarization as Post-Processing

5.1 Introduction

Cascaded approaches for speaker diarization ignore speaker overlaps, but they are still strong baselines over end-to-end methods on datasets of a large number of speakers, *e.g.*, DIHARD II dataset [95]. This is because they handle multiple speaker problems based on unsupervised clustering without using any speech mixtures as training data. Thus, the methods do not suffer from overtraining due to the lack of the overlap speech especially for a large number of speakers. On the other hand, while EEND-GLA in Section 3.3 can somewhat deal with a large number of speakers, its performance on that case is still poorer than the state-of-the-art cascaded approach. One reason is the training datasets. Mixtures of a large number of speakers are often rare in various datasets; thus, end-to-end models cannot produce diarization results for large number of speakers because they are overtrained on mixtures of a few number of speakers. Even if the issue on the number of mixtures is solved, the EEND depends on the permutation invariant training [66] so that it is still hard to train the model on a large number of mixtures in terms of the calculation cost. For these reasons, how to handle mixtures that contain overlapping speech of a large number of speakers is still an open problem for both cascaded and end-to-end diarization methods.

In this chapter, we propose to combine both cascaded and end-to-end methods effectively to deal with overlapping speech regardless of the number of speakers, namely, EEND as post-processing (EENDasP). We first obtain the initial diarization result using x-vector clustering, which does not produce overlapping results

in most cases. We then apply the following steps iteratively: i) frame selection to contain only two speakers and silence and ii) overlap estimation using a two-speaker EEND model. The frame selection is also used to adapt the EEND model to a dataset which contains mixtures of more than two speakers. We evaluate our method using various datasets including CALLHOME, AMI, and DIHARD II datasets.

5.2 EEND as Post-Processing

5.2.1 Overview

Given acoustic features $\{\boldsymbol{x}_t\}_{t=1}^{T}$, where $t \in \{1, \dots, T\} =: [T]$ denotes a frame index, diarization is a problem to predict a set of active frames $\mathcal{T}_k \subseteq [T]$ for each speaker $k \in \{1, \dots, K\} =: [K]$. K is the estimated number of speakers. For simplicity, we use $\mathcal{X}_{\mathcal{T}} := \{\boldsymbol{x}_t \mid t \in \mathcal{T}\}$ to denote the features of selected frames $\mathcal{T} \subseteq [T]$.

Cascaded methods assume that input recordings do not contain speaker overlap. It formulates diarization as a set partitioning problem, *i.e.*, $\mathcal{T}_k$ for $k \in [K]$ are predicted to be disjoint, *i.e.*, $\forall \{i, j\} \in \binom{K}{2}, \mathcal{T}_i \cap \mathcal{T}_j = \emptyset$. In EEND, on the other hand, diarization is formulated as a multi-label classification to handle overlapping speech; thus, they do not have to be disjoint. The formulation of EEND is appropriate for real conversations in which speakers sometimes utter simultaneously. However, it makes the problem too difficult to be solved; when K is large (*e.g.* 10), it rarely happens that K speakers speak together. Therefore, we assume that at most $K'(< K)$ speakers speak simultaneously, and refine the results of a cascaded method using an end-to-end model that is trained to process at most K' speakers. In this study, we set $K' = 2$. The detailed algorithm is explained in Section 5.2.2.

Table 5.1 summarizes the overlap ratio of various real multi-speaker datasets. If overlap regions are not taken into account at all, diarization error rates will exceed 10 % because they are bounded by the overlap ratio. On the other hand, Table 5.1 also suggests that overlap regions of more than two speakers are minor in natural conversations. From these observations, cascaded methods with two-speaker overlap detection could be a good choice for the diarization of natural conversations; thus, we set $K' = 2$.

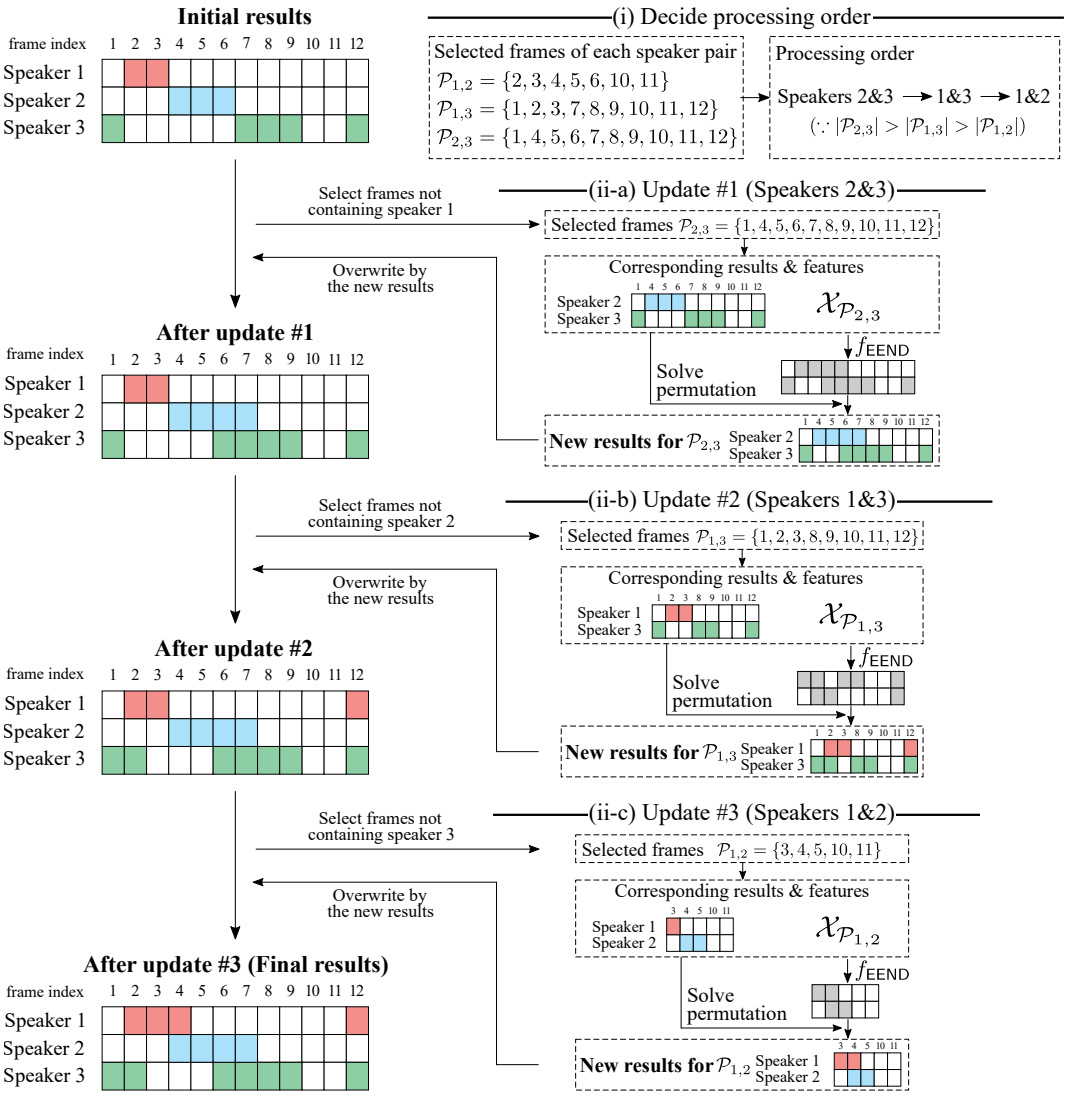


Figure 5.1: Diagram of EEND as post-processing when the number of speakers is three. It refines the diarization results of each pair of speakers iteratively. Given initial diarization results (top left), the proposed method (i) first determines the processing order on the basis of the number of frames and (ii) then refines the corresponding results of each pair using two-speaker EEND.

Table 5.1: Ratio (%) of region where at least n speakers are active over all speech region of various real multi-speaker datasets.

Dataset	# of speakers	$n = 2$	$n = 3$
CALLHOME [6]	2–7	16.9	0.2
AMI [7]	3–5	19.4	3.5
ICSI [8]	6 (ave.)	18.6	3.9
CHiME-6 [9]	4	33.9	8.6
Internal meeting [10]	N/A	14.7	N/A
Internal meeting [11]	4/6	16.3/16.0	N/A
Internal meeting [12]	5–8	13.2	1.4

5.2.2 Algorithm

Given initial diarization results $\{\mathcal{T}_k \mid \emptyset \neq \mathcal{T}_k \subseteq [T]\}_{k=1}^{K}$, we iteratively select two speakers among K and update the diarization results of the two speakers using an EEND model. The EEND model $f_{\mathsf{EEND}} : \mathbb{R}^{D\times L} \to (0,1)^{2\times L}$ was trained to estimate posteriors probabilities of two speakers from an L-length sequence of D-dimensional acoustic features. Figure 5.1 show the flow of EENDasP when $K = 3$.

Processing order determination

To apply the iterative refinement to each pair of speakers, the processing order influences the accuracy of final diarization results. This is because we cannot select frames to include only two speakers based on estimated diarization results because they include diarization errors. For example, if we select frames not containing Speaker 1 in Figure 5.1, the fourth frame contains Speaker 1 according to the final results. If the ratio of such impurities among the selected frames is high, the refinement using EEND may not perform well. We found that this problem is simply solved by processing the pairs of speakers in decreasing order of the number of selected frames (Figure 5.1(i)). For each speaker pair $\{i, j\} \in \binom{K}{2}$, we first select a set of frames $\mathcal{P}_{i,j}$ not containing speakers other than i and j as follows:

$$\mathcal{P}_{i,j} = [T] \setminus \underbrace{\bigcup_{k\in[K]\setminus\{i,j\}} \mathcal{T}_k}_{\text{frames in which speakers other than } i \text{ and } j \text{ are active}} . \tag{5.1}$$

We then apply the refinement described below for each speaker pair in descending order of $\left|\mathcal{P}_{i,j}\right|$ as in Figure 5.1 (ii-a)–(ii-c).

Iterative update of diarization results

To update the diarization results of speakers i and j, we first reselect a set of frames $\mathcal{P}_{i,j}$ using (5.1). This is because the diarization results $\{\mathcal{T}_k\}_{k=1}^{K}$ are updated at each refinement step so that we cannot reuse the one that is calculated to decide the processing order. Then the corresponding features $\mathcal{X}_{\mathcal{P}_{i,j}}$ are input to the EEND model to obtain posteriors of two speakers (a) and (b) by

$$\left(\left[q_t^{(a)}, q_t^{(b)} \right]^{\mathsf{T}} \;\middle|\; t \in \mathcal{P}_{i,j} \right) = f_{\mathsf{EEND}} \left(\mathcal{X}_{\mathcal{P}_{i,j}} \right) \in (0,1)^{2 \times \left| \mathcal{P}_{i,j} \right|}, \tag{5.2}$$

where $q_t^{(a)}$ and $q_t^{(b)}$ denote posteriors of the first and second speakers at frame index t, respectively, and $(\cdot)^{\mathsf{T}}$ denotes the matrix transpose. We simply apply the threshold value of 0.5 to obtain the indexes of active frames of the two speakers as

$$\mathcal{Q}^{(a)} = \left\{ t \in \mathcal{P}_{i,j} \;\middle|\; q_t^{(a)} > 0.5 \right\}, \tag{5.3}$$

$$\mathcal{Q}^{(b)} = \left\{ t \in \mathcal{P}_{i,j} \;\middle|\; q_t^{(b)} > 0.5 \right\}. \tag{5.4}$$

Note that we have speaker permutation ambiguity between (a)–(b) and i–j, and we solve permutation to find the optimal correspondence between $(\mathcal{T}_i, \mathcal{T}_j)$ and $(\mathcal{Q}^{(a)}, \mathcal{Q}^{(b)})$ as follows:

$$(\hat{\mathcal{T}}_i, \hat{\mathcal{T}}_j) = \underset{(u,v) \in \{(a,b),(b,a))\}}{\arg\max} \; s\left(\mathcal{Q}^{(u)}, \mathcal{T}_i \right) + s\left(\mathcal{Q}^{(v)}, \mathcal{T}_j \right), \tag{5.5}$$

where $s(\mathcal{U}, \mathcal{V})$ is a function to calculate similarity between speech and non-speech activities described by two sets $\mathcal{U}$ and $\mathcal{V}$ defined as

$$s(\mathcal{U}, \mathcal{V}) \coloneqq \underbrace{|\mathcal{U} \cap \mathcal{V}|}_{\text{speech similarity}} + \underbrace{|([T] \setminus \mathcal{U}) \cap ([T] \setminus \mathcal{V})|}_{\text{non-speech similarity}}. \tag{5.6}$$

Finally, we update the diarization results of speakers i and j. To confirm that the new results $\hat{\mathcal{T}}_i$ and $\hat{\mathcal{T}}_j$ are calculated for speaker i and j, we check whether they satisfy the following conditions:

$$\frac{\left| \hat{\mathcal{T}}_i \cap \left(\mathcal{T}_i \cap \mathcal{P}_{i,j} \right) \right|}{\left| \mathcal{T}_i \cap \mathcal{P}_{i,j} \right|} > \alpha, \tag{5.7}$$

$$\frac{\left| \hat{\mathcal{T}}_j \cap \left(\mathcal{T}_j \cap \mathcal{P}_{i,j} \right) \right|}{\left| \mathcal{T}_j \cap \mathcal{P}_{i,j} \right|} > \alpha, \tag{5.8}$$

where α is a lower limit of the ratio of the intersection between the new results $\hat{\mathcal{T}}_i$ (or $\hat{\mathcal{T}}_j$) and the previous results $\mathcal{T}_i \cap \mathcal{P}_{i,j}$ (or $\mathcal{T}_j \cap \mathcal{P}_{i,j}$). In this study, we set $\alpha = 0.5$. Only if the conditions in (5.7) and (5.8) are satisfied, we update the results of speakers i and j. When $K = 2$, we simply update the results with the new ones as

$$\mathcal{T}_i \leftarrow \hat{\mathcal{T}}_i \cup \left([T] \setminus \mathcal{P}_{i,j} \right), \tag{5.9}$$

$$\mathcal{T}_j \leftarrow \hat{\mathcal{T}}_j \cup \left([T] \setminus \mathcal{P}_{i,j} \right). \tag{5.10}$$

On the other hand, when $K \geq 3$, such fully-update strategy causes a performance drop due to impurities in the selected frames. Thus, we use the following instead of (5.9) and (5.10) to update only overlapped frames:

$$\mathcal{T}_i \leftarrow \mathcal{T}_i \cup \left(\hat{\mathcal{T}}_i \cap \hat{\mathcal{T}}_j\right), \tag{5.11}$$

$$\mathcal{T}_i \leftarrow \mathcal{T}_j \cup \left(\hat{\mathcal{T}}_j \cap \hat{\mathcal{T}}_j\right). \tag{5.12}$$

For the end-to-end model f_{EEND}, we use the self-attentive EEND model with an encoder-decoder attractor calculation module (EEND-EDA) [71]. It consists of a four-layer-stacked Transformer encoder to extract embeddings for each frame and the EDA module to calculate attractors from the extracted embeddings. The EDA includes long short-term memories but we shuffled the order of embeddings just before they are fed into the EDA, which improves the diarization performance. Thus, we can consider that all the components of f_{EEND} are independent of the order of embeddings and therefore the model can treat input features of selected frames even if they are not continuous in time.

5.2.3 Training Strategy of the EEND-EDA Model

In the original EEND and its derived methods [24, 71, 83] used *matched* dataset for model adaptation, *i.e.*, only two-speaker subset of the original dataset (*e.g.* CALLHOME [6]) was used to finetune the models in two-speaker evaluations. This strategy cannot be used to finetune two-speaker models when the dataset does not contain two-speaker mixtures (*e.g.* AMI [7]). Even if two-speaker mixtures are included in the dataset, it does not make full use of the datasets, which may cause performance degradation.

To cope with this situation, we adopt the frame-selection technique used in (5.1) for model adaptation. If the input chunk contains more than two speakers, we first choose two dominant speakers and then eliminate frames in which the other speakers are active as in (5.1). The model is trained only using the selected frames to output speech activities of the two speakers. This makes it possible to finetune two-speaker models from any kind of multi-speaker datasets without mixture-wise selection.

5.3 Experimental Settings

We use EEND-EDA trained on Sim2spk, which was evaluated in Section 3.2.5, as a pretrained model. The model was adapted on CALLHOME [6], AMI headset mix

Table 5.2: DERs (%) on CALLHOME-2spk. Collar tolerance of 0.25 s is allowed.

Model	Adaptation	DER
SA-EEND [147]	CALLHOME-2spk	9.54
SA-EEND	CALLHOME + frame selection	9.00
EEND-EDA (Section 3.2)	CALLHOME-2spk	8.07
EEND-EDA	CALLHOME + frame selection	**7.84**

(`pyannote.audio` split) [7] and DIHARD II [95] datasets for another 100 epochs, respectively. Refer to Section A.1.1 for the details of each dataset. Adam optimizer [92] was used in the adaptations step with the fixed learning rate of 1×10^{-5}.

We used diarization error rate (DER) and Jaccard error rate (JER) for evaluation. While some studies excluded overlapped regions from evaluation [36, 21], this study scored overlapped region. We also note that our evaluations are based on estimated speech activity detection (SAD), while some studies used oracle segments [40] or only reported confusion errors [21].

5.4 Results

5.4.1 Preliminary Evaluation of the Training Using Frame Selection

Before the evaluation of EENDasP, we first evaluated the training strategy explained in Section 5.2.3 using two transformer-based two-speaker EEND models: SA-EEND [24] and EEND-EDA [71]. They were trained on CALLHOME-2spk in the original papers, but we utilized mixtures that contain more than two speakers in the CALLHOME dataset. Table 5.2 shows DERs on the CALLHOME-2spk test set. Using the full CALLHOME improved DER of SA-EEND from 9.54 % to 9.00 % and that of EEND-EDA from 8.07 % to 7.84 %. According to these results, we show the effectiveness of our training strategy described in Section 5.2.3 based on the frame selection with (5.1).

Table 5.3: DERs (%) on CALLHOME. All the results include overlapped regions and are NOT based on oracle SAD. Collar tolerance of 0.25 s is allowed.

	#Speakers					
Method	2	3	4	5	6	All
EEND-EDA [71]	**8.50**	13.24	21.46	33.16	40.29	15.29
X-vector AHC	15.45	18.01	22.68	31.40	34.27	19.43
X-vector AHC + EENDasP	13.85	14.72	18.61	**28.63**	29.02	16.79
X-vector AHC + VB	12.62	16.82	21.27	31.14	31.80	17.61
X-vector AHC + VB + EENDasP	9.87	**13.11**	**16.52**	28.65	**27.83**	**14.06**

5.4.2 Evaluation of EENDasP

CALLHOME

We first evaluated EENDasP on CALLHOME dataset, which is composed of telephone conversations. As a cascaded baseline, x-vectors with AHC and PLDA[1] was used with TDNN-based speech activity detection[2]. We also prepared the results for which VB-HMM resegmentation [148] was applied. All the components were implemented in Kaldi recipe.

Table 5.3 shows the evaluation results. X-vector clustering without and with VB achieved 19.43 % and 17.61 % DERs, respectively, but they didn't outperform the 15.29 % DER scored by EEND-EDA trained to output diarization results on flexible number of speakers. However, we can also observe that the cascaded methods are better when the number of speakers is larger than four. Applying EENDasP for x-vector clustering baselines achieved 16.79 % and 14.06 % without and with VB, and the latter is 1.23 % better than the EEND-EDA model. In terms of the number of speakers, EENDasP performed well on both large and small number of speakers.

AMI

Second, we evaluated our method on AMI dataset, consisting of meeting recordings. We chose the system developed during JSALT 2019 [149] as a baseline. It is based on x-vector clustering followed by VB resegmentation and overlap detection and assignment for the second speaker candidate [5].

[1] https://github.com/kaldi-asr/kaldi/tree/master/egs/callhome_diarization/v2
[2] https://github.com/kaldi-asr/kaldi/tree/master/egs/aspire/s5

Table 5.4: DERs and JERs (%) on AMI eval. VB: Variational Bayes resegmentation, OVL: Overlap detection and speaker assignment [5]. All the results include overlapped regions and are NOT based on oracle SAD. No collar tolerance is allowed.

Method	DER	JER
X-vector AHC [149]	33.75	45.68
X-vector AHC + EENDasP	30.64	43.78
X-vector AHC + VB [149]	32.80	43.72
X-vector AHC + VB + EENDasP	29.66	42.63
X-vector AHC + VB + OVL [149]	28.15	41.00
X-vector AHC + VB + OVL + EENDasP	**27.97**	**40.57**

Table 5.4 shows DERs and JERs on AMI eval set. EENDasP reduced DERs of 3.07 %, 3.14 %, and 0.18 % of absolute improvement from the three baselines. Surprisingly, our method improved DER and JER of the results in which the overlap detection [5] was already applied.

DIHARD II

Finally, we evaluated EENDasP on DIHARD II dataset, which includes recordings from 10 different domains. We used the official baseline system [95] and the BUT system [49, 50], which is the winning system of the second DIHARD Challenge, to obtain initial diarization results. Both are based on the x-vector clustering, but the BUT system is more polished in that it extracts x-vectors in shorter intervals and uses VB resegmentation and overlap detection and assignment based on heuristics.

The results are shown in Table 5.5. EENDasP reduced DER and JER of the baseline system by 2.96 % and 2.91 %, respectively. Our method also improved DER and JER of 0.35 % and 0.66 % from the BUT system without overlap assignment and 0.38 % and 0.64 % from that with overlap assignment, respectively. These improvements are small, but it is significantly better than the heuristic-based overlap assignment in [49], which improved DER by 0.15 % ($= 27.26 - 27.11$) and JER by 0.08 % ($= 49.15 - 49.07$).

5.5 Conclusion

In this chapter, we proposed EEND as post-processing for cascaded diarization using an end-to-end diarization model. We iteratively selected two speakers, picked

Table 5.5: DERs and JERs (%) on DIHARD II eval. All the results include overlapped regions and are NOT based on oracle SAD. No collar torelance is allowed.

Method	DER	JER
DIHARD II baseline [38]	40.86	66.60
DIHARD II baseline + EENDasP	37.90	63.79
BUT system (w/o OVL) [49, 50]	27.26	49.15
BUT system (w/o OVL) + EENDasP	26.91	48.49
BUT system (w/ OVL) [49, 50]	27.11	49.07
BUT system (w/ OVL) + EENDasP	**26.88**	**48.43**

up frames that contain the two speakers, and process the frames by the end-to-end model to update diarization results. Evaluations on CALLHOME, AMI, and DIHARD II datasets showed that our proposed method improves the results of various types of cascaded methods.

5.6 Acknowledgment

We would like to thank Federico Landini for providing the results of the winning system [49, 50] of the second DIHARD Challenge.

Chapter 6

Meeting Transcription with Distributed Microphones

6.1 Introduction

Meeting transcription is one practical use case of multi-speaker ASR in which speaker diarization can play an essential role. To transcribe utterances from multiple speakers which may overlapped together, not only speaker diarization to detect speaker overlaps but also powerful speech separation to distinguish them is required. Therefore, most meeting transcription systems are based on a microphone array [150, 151, 152, 10], sometimes one with an omnidirectional camera [153, 154] for face tracking. This means that the system requires special equipment to be introduced. If the microphone arrays can be replaced by more general devices, such as participants' smartphones or tablets, ease of use, installation cost, and portability of the system will be greatly improved. The main challenge of the meeting transcription with such distributed devices is that they are asynchronous and thus general speech separation methods for synchronized signals cannot be simply applied.

Recently, some methods of meeting transcription using asynchronous distributed microphones have been proposed. One is the session-wise approach [27, 112], in which first synchronizes multi-channel observation by correcting sampling frequency mismatch, then performs session-wise speech separation using the minimum variance distortionless response (MVDR) beamformer, and finally feeds the enhanced signals into an ASR module to transcribe them. Here, the MVDR beamformer is applied for each frequency bin, so the well-known frequency-wise permutation problem has to be solved. The common approach for multi-speaker cases is to prepare initial spatial correlation matrices using training data with a

fixed number of speakers and their positions [155]. Therefore, it cannot deal with the case when the number of speakers during inference is different from, especially larger than, that in the training data. If we cannot obtain such spatial correlation matrices in advance, *e.g.*, we have no training data, we have to solve the permutation problem as a post-processing [156, 157], but there are few reports that they performed well on real noisy and reverberant data.

Another is the block-wise approach [28], in which first synchronizes input audio streams in a block-online manner, then applies block-wise speech enhancement followed by ASR and speaker diarization. The benefit of this approach is that the effect of sampling frequency mismatch can be ignored within each short-enough block; the scale of sampling frequency mismatch is about 100 ppm (parts per million) at most [158, 159]. However, the speech enhancement here does not consider speaker overlaps; each speech is separated only from background noise and thus cannot deal with multiple speakers speaking simultaneously.

This chapter investigates the utterance-wise approach, which is different from the session-wise or block-wise approaches described above. We first roughly synchronized audio signals recorded by distributed microphones and then applied speaker diarization. Speaker diarization is based on the clustering of features extracted from short segments, but we use features extracted from all the signals recorded by each microphone so that it can deal with overlapped speech. Then we applied guided source separation [16], which performed well for ASR in a dinner party scenario [73, 74]. This separation is conducted for each extracted utterance, which is short enough not to be suffered from sampling frequency mismatch between microphones. We applied ASR for each enhanced utterance, and finally, we conducted duplication reduction for ASR results to reduce the effect of errors on diarization or separation. Our approach can deal with speaker overlap without any methods to correct sampling frequency mismatch in the synchronization phase and solve the permutation problem in the speech separation phase. To evaluate our framework, we recorded eight sessions of real meetings using 11 distributed smartphones, each of which was equipped with a monaural microphone. The experimental results showed that our framework improved performance by using multiple microphones. We also showed that our framework could achieve performance comparable to that of headset microphone-based transcription if the oracle diarization results were known.

6.2 Method

We assume that a meeting is recorded using M asynchronous distributed microphones and the number of speakers K is known in advance. Figure 6.1 shows the

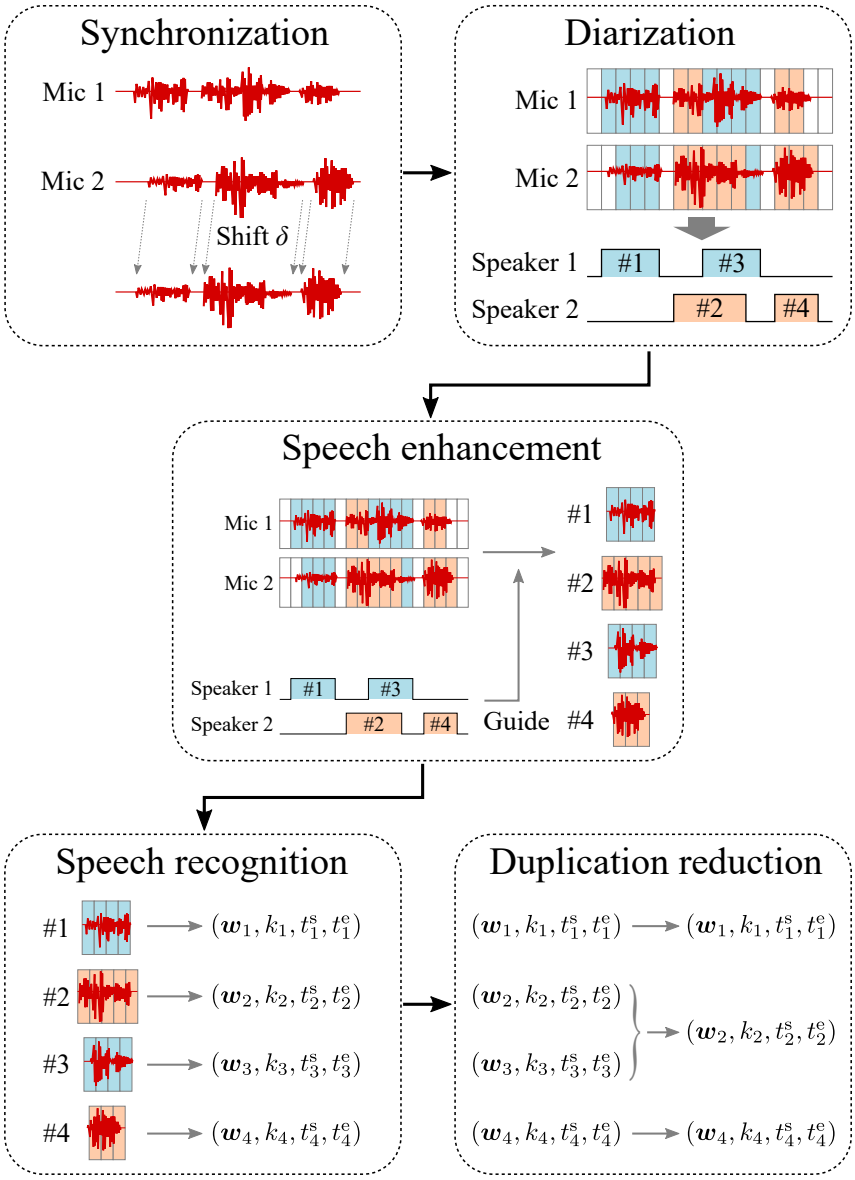


Figure 6.1: Overview of our meeting transcription system using asynchronous distributed microphones.

overview of our meeting transcription system. Given M-channel signals, we first synchronize them by maximizing their correlation coefficient. The correction of sampling frequency mismatch between signals is not conducted in the synchronization part. With the synchronized signals, clustering-based diarization is performed to obtain a set of utterances for each speaker. After that, we perform speech separation for each utterance by using the diarization results as guides to avoid the permutation problem [16]. The enhanced utterances are fed into the ASR module to obtain ASR results, followed by duplication reduction to reduce errors caused by diarization or separation. The following subsections detail the implementation of each module.

6.2.1 Blind Synchronization

In this part, we conduct a correlation-based synchronization to correct the start or end point differences of input signals. This rough synchronization can be operated under the existence of the sampling frequency mismatch. Assume that the time-domain observation of the m-th microphone ($m \in \{1, \ldots, M\}$) is defined as $\hat{\boldsymbol{x}}_m := [\hat{x}_{m,n}]_{n=1}^{N_m}$, where N_m is the number of samples in the m-th signal. We select an anchor m_a from the M microphones and calculate the shift δ_m between signals of the anchor m_a and each microphone $m \in \{1, \ldots, M\}$ as follows:

$$\delta_m = \begin{cases} \arg\max_{\delta \in \mathbb{Z}} \sum_\nu x_{m_a,\nu} x_{m,\nu+\delta} & (m \neq m_a) \\ 0 & (m = m_a) \end{cases}, \tag{6.1}$$

$$x_{m,\nu} = \begin{cases} \hat{x}_{m,\nu} & (\nu \in \{1, \ldots, N_m\}) \\ 0 & (\text{otherwise}) \end{cases}. \tag{6.2}$$

Synchronized signals $\boldsymbol{x}_m$ ($m \in \{1, \ldots, M\}$) are defined in the time interval recorded by all the microphones as follows:

$$\boldsymbol{x}_m = [\hat{x}_{m,n}]_{n=n_{\text{begin}}+\delta_m}^{n_{\text{end}}+\delta_m}, \tag{6.3}$$

$$n_{\text{begin}} = \max_{m' \in \{1,\ldots,M\}} (1 - \delta_{m'}), \tag{6.4}$$

$$n_{\text{end}} = \min_{m' \in \{1,\ldots,M\}} (N_{m'} - \delta_{m'}). \tag{6.5}$$

In this study we assume that all the utterances to be transcribed are within the time interval of $\boldsymbol{x}_m$.

6.2.2 Speaker Diarization

In this subsection, we conduct speaker diarization by clustering vectors. One drawback of the conventional clustering-based diarization using a monaural recording

is that it cannot deal with speaker overlap because each frame is assigned to one speaker. On the other hand, in our scenario, each meeting has been recorded by distributed microphones. Therefore, even when two speakers spoke simultaneously, it is expected that one microphone could have captured the one speaker's utterance at a sufficient signal-to-noise ratio (SNR) and another microphone could have captured the other speaker's utterance at a sufficient SNR. In this study, we extract features from all the signals from all the microphones and perform clustering for the extracted features altogether to deal with speaker overlap.

We first split each synchronized observation $\boldsymbol{x}_m$ into short overlapping segments $[\boldsymbol{x}_{m,1}, \ldots, \boldsymbol{x}_{m,T}]$ with 1.5 s of window size and 0.75 s of window shift. We apply power-based speech activity detection for each segment; as a result, each segment is classified as either speech or non-speech. From each speech segment, we extract features to be used for clustering. In this study, we concatenate two kinds of features: speaker-characteristics-based features and power-ratio-based features.

For features to represent speaker characteristics, we use x-vectors [39], which are commonly used in cascaded diarization systems [38, 40, 4]. We extract x-vectors from each microphone's audio; so we can obtain different speaker characteristics from one frame. Before we use the vectors for clustering, we subtract a mean vector within a session and normalized them to have a unit norm. As a result, we obtain microphone and frame-wise D-dimensional features $\boldsymbol{c}_{m,t} \in \mathbb{R}^D$ $(1 \leq t \leq T)$.

Although the x-vectors from distributed microphones are potentially beneficial to diarize overlapped speech, it becomes a problem that an utterance from the same speaker could be judged as one from multiple speakers because x-vectors suffer from speaker-microphone distance and noisy environments. Thus, we introduce power-based frame-wise features $\boldsymbol{p}_t := [p_{1,t}, \ldots, p_{M,t}]^\mathsf{T}$, where $p_{m,t}$ is the average power of $\boldsymbol{x}_{m,t}$. This speaker diarization part is a session-level one, so we avoid using phase-based features like GCC-PHAT [160] because they suffer from the sampling frequency mismatch.

Final $(D + M)$-dimensional features to be clustered are

$$\boldsymbol{v}_{m,t} = \begin{bmatrix} \boldsymbol{c}_{m,t} \\ \lambda \boldsymbol{p}_t / \left\| \boldsymbol{p}_t \right\| \end{bmatrix}, \tag{6.6}$$

where λ is the scaling factor to balance the effect of $\boldsymbol{c}_{m,t}$ and $\boldsymbol{p}_t$. We apply agglomerative hierarchical clustering for the features to divide the speech segments into K clusters. As a result, each feature from a speech segment belongs to one of the clusters $\mathcal{C}_2, \ldots, \mathcal{C}_{K+1}$, where $\mathcal{C}_{k+1}$ corresponds to the speech cluster of k-th speaker. We also define the additional noise cluster $\mathcal{C}_1 := \{\boldsymbol{v}_{m,t} \mid 1 \leq m, \leq M, 1 \leq t \leq T\}$. The diarization results including noise $D = \left\{ d_t^{(k)} \right\} \in \{0, 1\}^{(K+1) \times T}$ are calculated

as

$$d_t^{(k)} = \begin{cases} 1 & (\exists m \in \{1, \ldots, M\},\ \boldsymbol{v}_{m,t} \in \mathcal{C}_k) \\ 0 & (\text{otherwise})\,. \end{cases} \tag{6.7}$$

In the diarization results, utterances are sometimes divided into some short fragments due to the existence of backchannels, noises, *etc*. In this study, we treat silence of 1.5 s or less between speech fragments from the same speaker as a speech by applying two iterations of binary closing along the time axis.

Here each timeslot in the diarization results corresponds to 0.75 s, which is inconsistent with the signals used in speech separation in the next section. Thus, we upsample the diarization results so that each timeslot corresponds to 16 ms. Hereafter, $D = \left\{ d_t^{(k)} \right\}$ denotes the upsampled diarization results.

6.2.3 Speech Separation

In this study, we conducted speech separation for each utterance by using guided source separation (GSS) [16], which is explained in Section 2.4. While the original GSS utilized oracle speech activities, we instead use estimated diarization results described in the previous section.

We first apply weighted prediction error [76] to the input multichannel signals in a short-time Fourier transform (STFT) domain for dereverberation. The window length and the window shift for the STFT were set to 64 ms and 16 ms, respectively. With the dereverberated STFT signal and the diarization results D, we obtained utterance-wise enhanced signals using GSS. The length of pre- and post-context was set to 15 s, respectively. The number of iterations for the EM update was set to 10.

6.2.4 Speech Recognition

For each enhanced utterance, we apply ASR consisting of a CNN-TDNN-LSTM acoustic model (AM) [161] followed by 4-gram-based and recurrent neural network-based language models (LMs) [162]. The AM takes a 40-dimensional log-scaled Mel-filterbank and 40-dimensional Mel-frequency cepstral coefficients as input audio features. 100-dimensional i-vectors are also fed into the AM for online adaptation for speaker and environment [163]. It was trained by 1,700 hours of Japanese speech corpus using the lattice-free maximum mutual information criterion [164]. The LMs were trained by transcriptions of the corpus used for AM training and the Wikipedia corpus.

6.2.5 Duplication Reduction

The diarization and speech separation is not perfect, so the same transcription is sometimes included in multiple estimated utterances. Therefore, we apply duplication reduction for the ASR results. Widely used ensemble techniques such as ROVER [165] and confusion network combination [166] are for the different ASR results obtained from the same utterance; thus, they cannot be used in this situation where the utterances to be merged have different start and end points. To overcome this issue, we propose a combination technique for such utterances which have different time intervals. We first find which pairs of utterances should be merged. Given the set of U ASR results $\mathcal{W} = \{(\boldsymbol{w}_u, k_u, t_u^{\mathrm{s}}, t_u^{\mathrm{e}})\}_{u=1}^{U}$, where $\boldsymbol{w}_u$, k_u, t_u^{s}, and t_u^{e} denote the sequence of words, speaker, start time, and end time of u-th result, respectively, we calculate an adjacency matrix $A = \{a_{i,j}\}_{i,j} \in \{0,1\}^{U \times U}$ as follows:

$$a_{i,j} = \begin{cases} 1 & (\max(t_i^{\mathrm{s}}, t_j^{\mathrm{s}}) < \min(t_i^{\mathrm{e}}, t_j^{\mathrm{e}}) \ \wedge s(\boldsymbol{w}_i, \boldsymbol{w}_j) > \tau \ \wedge \ k_i \neq k_j) \\ 0 & (\text{otherwise}) \end{cases}, \tag{6.8}$$

where $\tau \in [0,1]$ is the threshold value. Here $s(\boldsymbol{w}_i, \boldsymbol{w}_j)$ is the similarity between $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ defined as follows:

$$s(\boldsymbol{w}_i, \boldsymbol{w}_j) := \frac{\max(|\boldsymbol{w}_i|, |\boldsymbol{w}_j|) - d(\boldsymbol{w}_i, \boldsymbol{w}_j)}{\min(|\boldsymbol{w}_i|, |\boldsymbol{w}_j|)}, \tag{6.9}$$

where $d(\boldsymbol{w}_i, \boldsymbol{w}_j)$ is the Levenshtein distance between $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$, and $|\boldsymbol{w}|$ denotes the number of words in $\boldsymbol{w}$. With this adjacency matrix, all the elements in $\mathcal{W}$ can be clustered into C clusters. We denote the clustering result as $\mathcal{C} = \{c_u\}_{u=1}^{U} \in \{1, \ldots, C\}^U$, which fulfill $c_i = c_j$ if a path between i-th and j-th elements exists in A and $c_i \neq c_j$ otherwise. Assuming that $\mathcal{W}_{k,c} \subseteq \mathcal{W}$ is the set of ASR results that belong to the cluster c and are uttered by speaker k, we obtain the representative speaker k^c of the cluster c by

$$k^c = \underset{k \in \{1,\ldots,K\}}{\arg\max} f(\mathcal{W}_{k,c}), \tag{6.10}$$

where $f(\cdot)$ is the selection function. In this study, we select the speaker with the longest utterance(s), *i.e.*, $f(\mathcal{W}_{k,c}) = \sum_{(\boldsymbol{w},k,t^{\mathrm{s}},t^{\mathrm{e}}) \in \mathcal{W}_{k,c}} |\boldsymbol{w}|$. The set of de-duplicated ASR results $\mathcal{W}'$ can be obtained as follows:

$$\mathcal{W}' = \bigcup_{c \in \{1,\ldots,C\}} \mathcal{W}_{k^c,c}. \tag{6.11}$$

6.3 Results

We investigated various combinations of asynchronous distributed microphones: 2 microphones (⑧ & ⑩ in Figure A.1), 3 microphones (⑦ & ⑨ & ⑪), 6 microphones

Table 6.1: CERs (%) obtained using various microphone combinations.

	Session								
#Mic	I	II	III	IV	V	VI	VII	VIII	All
1	31.2	30.1	37.1	37.6	28.2	48.4	50.4	52.5	38.2
2	22.9	25.3	30.5	37.0	21.8	41.7	36.8	45.7	31.4
3	26.8	24.4	35.9	37.2	23.2	43.1	41.9	46.6	33.7
6	22.3	22.2	36.0	32.1	21.0	38.1	35.1	44.3	30.2
11	21.2	21.1	32.5	30.9	19.6	37.6	34.0	41.0	28.7
11[†]	17.0	16.3	21.7	21.2	17.7	27.0	27.0	32.8	21.8
Headset	18.3	15.8	21.0	20.1	13.6	21.3	24.9	25.8	19.7

[†] The oracle diarization was used for speech separation.

(①–⑥), and 11 microphones (①–⑪). For comparison, we also evaluated the performance of one monaural microphone (⑨) and of headset microphones that the participants wore during each session.

The character error rates (CERs) obtained using various microphone combinations in each session are shown in Table 6.1. In these experiments, the weighting parameter λ in (6.6) was set to 1.0. By using multiple microphones, we could have reduced CERs, especially by using a large number of microphones. Note that in two-, three-, and six-microphone settings, using more microphones not always resulted in better CERs. This is because the sets of microphones in these settings are disjoint and the CERs highly depended on the positions of microphones and speakers. On the other hand, we observed the best CERs in almost every session by using all the 11 microphones. This result indicated that adding microphones has almost no negative effect on CERs. In Table 6.1, we also showed CERs with 11 microphones in the case when oracle diarization was used for GSS. It achieved the CER of 21.8 %, which is only 2.1 percentage points worse than the CER of 19.7 % obtained using headset microphones. It can be said that our method can potentially achieve nearly headset-level CERs when it is used with a more powerful diarization method [24, 167, 71].

In Table 6.2 we show the average CERs over sessions with various weighting parameters λ in (6.6). Combinations of speaker-characteristics-based features and power-ratio-based features improved transcription performance, especially when the number of microphones is smaller and the power ratio thus has less information about the directions of speakers.

Finally, we conducted ablation studies by removing binary closing in diarization, speech separation by using recordings of the reference microphone instead,

Table 6.2: CERs (%) obtained with various scaling factors λ.

	Scaling factor λ in (6.6)						
#Mic	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}
2	33.7	31.7	32.2	**31.4**	31.8	33.3	35.5
3	34.2	34.3	34.0	33.7	**33.0**	34.8	35.2
6	33.5	34.1	33.4	**30.2**	31.4	31.9	32.4
11	33.5	32.5	31.1	28.7	28.9	**28.4**	28.9

Table 6.3: Ablation study using 11 microphones.

Method	CER (%)
Baseline (11 mics)	28.7
w/o binary closing	30.6
w/o speech separation	37.8
w/o duplication reduction	31.9

and duplication reduction, respectively. Here we used 11 microphones with $\lambda = 1.0$. The results are shown in Table 6.3. We found 1.9, 9.1, and 3.2 percentage points degradation from the baseline by removing binary closing, speech separation, and duplication reduction, respectively. From these results, we concluded that these three components contributed to the improvement of the CER.

6.4 Conclusions

In this chapter, we proposed a meeting transcription system based on utterance-wise processing using asynchronous distributed microphones. It consists of the following modules: blind synchronization, speaker diarization, speech separation, speech recognition, and duplication reduction. Evaluation on the real meeting data showed the effectiveness of our framework and its components, and also showed that it could perform comparably to the headset microphone-based transcription if the oracle diarization was given.

Chapter 7

Block-Online Guided Source Separation

7.1 Introduction

Speech separation is essential to improve the performance of automatic speech recognition (ASR) under a noisy and speaker-overlapped condition. Although there have been recent successes in neural-network-based mask estimation [168, 169, 123, 10, 134] or end-to-end speech separation [19, 170, 171] for multi-channel signals, beamforming with unsupervised mask estimation is still a powerful speech separation method. Especially, guided source separation (GSS) [16], which involves constructing a beamformer by using diarization information, has performed well on the CHiME-5 corpus [73, 74] that consists of recordings at dinner parties. GSS was also adopted as a baseline method for the CHiME-6 Challenge [15] and is still a *de facto* standard for the CHiME-6 corpus [172, 167, 173]. The investigation in Chapter 6 has proven that diarization-first speech separation using GSS is also useful for an ASR system using asynchronous distributed monaural microphones.

There are mainly three advantages of GSS. The first is that the diarization information can give good initial parameters of a generative model of observations, which makes it possible to work well without pretrained parameters even when there are multiple speakers. The second advantage is that, although mask estimation is a frequency-wise algorithm, this initialization makes it free from the permutation problem of the frequency domain. The third advantage is that GSS calculates utterance-wise beamformers so that it works well when a session-level beamformer does not work well, *e.g.*, there is a sampling frequency mismatch between audio channels or speakers are moving around during a session.

The utterance-wise algorithm, however, incurs significant computational cost because several iterations of optimization are required for each utterance. It also produces latency according to the utterance length and its pre-context length. Thus, this algorithm limits the development of ASR systems based on GSS, especially those of real-time applications. If GSS is extended to an online algorithm, it is beneficial to implement a highly accurate speech recognition system for an overlapping and conversational speech by combining GSS with online diarization methods for an unrestricted number of speakers [43, 21, 174] and online ASR [175, 176, 177, 178].

This chapter proposes a block-online GSS algorithm to avoid the utterance-wise processing. A block-wise input is processed together with its pre-context to update time-frequency masks and estimate a minimum variance distortionless response (MVDR) beamformers. There are two benefits of using a pre-context. One is that the context is helpful to estimate the mask of the current block from only one expectation-maximization (EM) iteration because the context has been processed once in the previous step. The other is that the context is helpful to solve the frequency permutation problem, the same as with the conventional offline GSS algorithm [16]. To reduce the computational cost, the block-wise update only takes into account the parameters of active speakers during the block and its context. We evaluated the proposed algorithm in both synchronous and asynchronous settings using the CHiME-6 corpus and a meeting corpus recorded using distributed asynchronous microphones. The experimental results indicate that the proposed algorithm exhibits comparable performance to the conventional offline GSS algorithm with real-time processing.

7.2 Related Work

Conventional online mask-based beamforming methods are based on block-wise [155, 179, 180, 181] or frame-wise [182] estimation of time-frequency masks and updating of the beamformer. There are mainly two approaches for mask estimation: spatial-clustering- and neural-network-based estimation. Mask estimation based on spatial clustering empirically requires pretrained parameters for initialization [155, 179], especially when there are multiple speakers, to avoid iterative calculation and the frequency permutation problem. This is not suitable when the microphone and speaker arrangement is not known in advance. Mask estimation based on neural networks requires clean training data [180, 181, 182], which are inaccessible in real conversations. It is also a problem that such networks typically predetermine the number of input and output channels. Recently proposed methods accept variable channels of inputs [117, 118, 29], but the number of outputs still has to be known in advance. The original GSS does not have such limitations; an online ex-

tension of the GSS proposed in this chapter also does not have such restrictions on the requirement of clean data or the number of channels in input/output. Recently, Du *et al*. investigated an online update of the beamformer on the CHiME-6 corpus [172], but the preceding mask estimation based on GSS is an offline algorithm so it cannot work in an online manner.

7.3 Block-Online Guided Source Separation

7.3.1 Overview

In the CHiME-6 baseline system, speech separation based on GSS is applied for each utterance. It takes about 85.44 hours using a single CPU without utterance-wise parallel processing to enhance all the utterances in the development set, which includes about 4.46 hours of recordings. This processing speed is not sufficient for online processing or for offline ASR systems because it takes over 19x the recording duration for speech separation.

The reason the conventional offline GSS algorithm requires such a long calculation time is the redundancy of the utterance-wise operation. For example, if two speech signals are highly overlapped, as in Figure 7.1, the optimized cACGMMs should be almost the same. However, they are optimized independently in the conventional offline GSS algorithm. It should be also considered in online processing that this algorithm uses a few seconds of signals after each utterance as context. If we use such a post-context even in an online algorithm, it produces latency according to the length of the context. Moreover, we have to beware that the calculation cost is proportional to the number of speakers.

The proposed algorithm i) updates the parameters of a cACGMM in a block-online manner to avoid redundant calculation as in Figure 7.1, ii) only uses a pre-context of each block to reduce latency, and iii) only uses active sources to update parameters to reduce calculation. Note that the number of speakers in a session does not have to be known a priori because this algorithm determines this adaptively from block-wise input diarization information.

7.3.2 Proposed Algorithm

The proposed algorithm is shown in Algorithm 7.1. Let L be the length of a block along the time axis, C be the length of a pre-context of a block, N be the length of a sequence of blocks, and $K_n (K_1 \leq K_2 \leq \cdots \leq K_N)$ be the number of sources

Algorithm 7.1: Block-online guided source separation.

Input: $\left\{\mathbf{X}_n \in \mathbb{C}^{L\times F\times M}\right\}_{n=1}^{N}$ // STFT features

$\left\{\left(d_t^{(k)}\right)_{\substack{t\in\mathcal{T}_n\\ 1\le k\le K_n}}\right\}_{n=1}^{N}$ // Diarization

$C \in \mathbb{Z}_{\ge 0}$ // #Pre-context frames

$K_0 = 0$ // Initial #Sources

for $n = 1$ **to** N **do**

  $\mathbf{X}_n \leftarrow \mathsf{BlockOnlineWPE}\left(\mathbf{X}_n\right)$

  if $\sum_{t\in\mathcal{T}_n}\sum_{k=2}^{K_n} d_t^{(k)} = 0$ **then** // Silent block

    continue

  $\mathcal{K} \leftarrow \left\{k \;\middle|\; 1 \le k \le K_n, \sum_{t\in\mathcal{T}_n^+} d_t^{(k)} > 0\right\}$ // Set of active sources

  foreach $f \in \{1,\dots,F\}$ **do**

    for $k = K_{n-1}+1$ **to** K_n **do** // New sources

      $\gamma_{t,f}^{(k)} \leftarrow 0$ for $t \in \mathcal{T}_n^c$

      $\Gamma_f^{(k)} \leftarrow 0$

      $B_f^{(k)} \leftarrow O_M$

    Initialize $\gamma_{t,f}^{(k)}$ for $(t,k) \in \mathcal{T}_n \times \mathcal{K}$ by (2.46)

    Update $\alpha_f^{(k)}$ for $k \in \mathcal{K}$ using $\hat{\mathbf{X}}_n^+$ by (2.43)

    Calculate $B_{n,f}^{+(k)}$ for $k \in \mathcal{K}$ using $\hat{\mathbf{X}}_n^+$ by (7.1)

    Update $B_f^{(k)}$ for $k \in \mathcal{K}$ using $\hat{\mathbf{X}}_n^+$ by (7.2)–(7.3) or (7.4)

    Update $\gamma_{t,f}^{(k)}$ for $(t,k) \in \mathcal{T}_n^+ \times \mathcal{K}$ by (2.45)

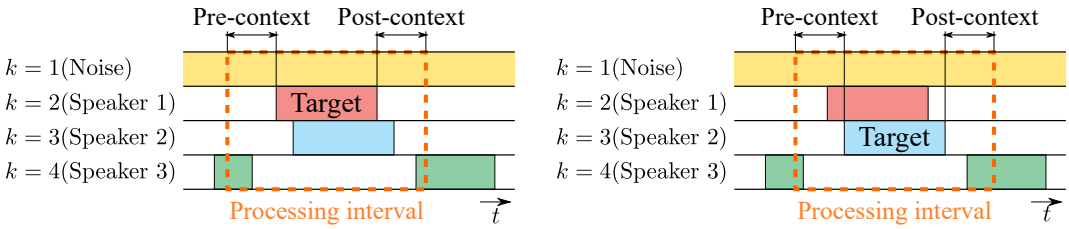
  foreach *utterance spoken during* $\mathcal{T}_n$ **do**

    /* assume that the utterance started at t_s and ended at t_e */

    Calculate R_f^{speech} and R_f^{noise} for $f \in \{1,\dots,F\}$ from $\mathbf{X}_n^+$ by (2.48)–(2.49)

    Calculate $\boldsymbol{w}_f$ for $f \in \{1,\dots,F\}$ by (2.50)

    Output an enhanced audio $\left(\boldsymbol{w}_f^{\mathsf{H}}\boldsymbol{x}_{t,f}\right)_{\substack{\max(t_s,(n-1)L+1)\le t\le \min(t_e,nL)\\ f\in\{1,\dots,F\}}}$



(a) Target: Speaker 1

(b) Target: Speaker 2

Figure 7.1: Case when almost same cACGMMs are obtained by conventional utterance-wise offline GSS algorithm.

appearing no later than the n-th block. For sake of simplicity, we define the set of time indexes in the n-th block as $\mathcal{T}_n \coloneqq \{(n-1)L+1, \ldots, nL\}$, that in the pre-context of the n-th block as $\mathcal{T}_n^c \coloneqq \{(n-1)L - C + 1, \ldots, (n-1)L\} \cap \mathbb{N}$, and the union of them as $\mathcal{T}_n^+ \coloneqq \mathcal{T}_n \cup \mathcal{T}_n^c$. To process the n-th block, we use samples in the previous blocks as the pre-context. Thus, we prepared a C-length queue and store the most recent C frames to use them in the block-online processing.

We assume that inputs are blocked STFT features $\{\mathbf{X}_n\}_{n=1}^N$, where $\mathbf{X}_n = \left(\boldsymbol{x}_{t,f}\right)_{\substack{t \in \mathcal{T}_n \\ f \in \{1,\ldots,F\}}} \in \mathbb{C}^{L \times F \times M}$, and their corresponding diarization results $\left\{\left(d_t^{(k)}\right)_{\substack{t \in \mathcal{T}_n \\ 1 \leq k \leq K_n}}\right\}_{n=1}^N$. Note that $k = 1$ corresponds to noise, whose activities $d_t^{(1)}$ are always one, and $k \geq 2$ corresponds to speakers. First, the initial number of sources K_0 is set to zero (Line 1 in Algorithm 7.1; L1). For each block (L2), the block-online WPE [183] is applied for dereverberation of the input features (L3). If there is no active speaker in the block, we finish processing for the input block (L4–5). If active speakers exist in the block, we extract the set of active sources $\mathcal{K}$ in the block and its context (L6).

The parameters of cACGMM are then updated for each frequency index f using the block and its pre-context (L7). To calculate the posteriors using (2.45), the mixture weight $\alpha_f^{(k)}$ and the matrix parameter $B_f^{(k)}$ for each active speaker are required. However, we do not have such $\alpha_f^{(k)}$ because the active source set $\mathcal{K}$ differs among blocks, and we also do not have reliable $B_f^{(k)}$ for new speakers. Therefore, in this online strategy, we first update $\alpha_f^{(k)}$ and $B_f^{(k)}$ using initial estimations of $\gamma_{t,f}^{(k)}$ calculated from the input diarization information, and then estimate $\gamma_{t,f}^{(k)}$ using the estimated $\alpha_f^{(k)}$ and $B_f^{(k)}$.

For each new speaker (L8), we set the posteriors $\gamma_{t,f}^{(k)}$ during the context by zero (L9) because the new speakers are not active during the context interval. The value

for the accumulation of posteriors $\Gamma_f^{(k)}$ is also initialized with zero (L10) and the matrix parameter $B_f^{(k)}$ is initialized with $M \times M$ zero matrix O_M (L11). We also initialize the posteriors of all the active speakers during the input block by (2.46) (L12). The mixture weight $\alpha_f^{(k)}$ for each source is then updated (L13) using (2.43). As described above, the sets of speakers differ from block to block; thus $\alpha_f^{(k)}$ is updated by (2.43) without any smoothing over blocks. On the other hand, in this study, we used two update strategies to update the matrix parameter $B_f^{(k)}$ (L14–15). One is the **accumulation** strategy, which updates $B_f^{(k)}$ to be closer to the offline estimation. We first calculate the matrix parameters using $\hat{\mathbf{X}}_n^+ := \left(\hat{\boldsymbol{x}}_{t,f}\right)_{\substack{t\in\mathcal{T}_n^+ \\ f\in\{1,\ldots,F\}}}$ by

$$B_{n,f}^{+(k)} = \begin{cases} M\dfrac{\sum_{t\in\mathcal{T}_n^+}\gamma_{t,f}^{(k)}\dfrac{\hat{\boldsymbol{x}}_{t,f}\hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}}}{\hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}}\left(B_f^{(k)}\right)^{-1}\hat{\boldsymbol{x}}_{t,f}}}{\sum_{t\in\mathcal{T}_n^+}\gamma_{t,f}^{(k)}} & (k \le K_{n-1}) \\ M\dfrac{\sum_{t\in\mathcal{T}_n^+}\gamma_{t,f}^{(k)}\hat{\boldsymbol{x}}_{t,f}\hat{\boldsymbol{x}}_{t,f}^{\mathsf{H}}}{\sum_{t\in\mathcal{T}_n^+}\gamma_{t,f}^{(k)}} & (k > K_{n-1}) \end{cases}. \tag{7.1}$$

By using this, $B_f^{(k)}$ is updated by

$$B_f^{(k)} \leftarrow \frac{\Gamma_f^{(k)}}{\Gamma_f^{(k)} + \sum_{t\in\mathcal{T}_n}\gamma_{t,f}^{(k)}} B_f^{(k)} + \frac{\sum_{t\in\mathcal{T}_n}\gamma_{t,f}^{(k)}}{\Gamma_f^{(k)} + \sum_{t\in\mathcal{T}_n}\gamma_{t,f}^{(k)}} B_{n,f}^{+(k)}, \tag{7.2}$$

where $\Gamma_f^{(k)}$ is an accumulation of the posteriors, which is updated after updating $B_f^{(k)}$ as follows:

$$\Gamma_f^{(k)} \leftarrow \Gamma_f^{(k)} + \sum_{t\in\mathcal{T}_n}\gamma_{t,f}^{(k)}. \tag{7.3}$$

The accumulation strategy is known to be effective when there is a beamformer that works well through a session [155, 180]. However, if there is a sampling frequency mismatch between audio channels or speakers are moving around during a session, such a session-wise beamformer is not sufficient and its block-level refinement is required for performance improvement [27, 112]. In such situations, the parameters of cACGMM should be updated to have temporal locality. Therefore, we also used the **decay** strategy, in which we update $B_f^{(k)}$ as follows:

$$B_f^{(k)} \leftarrow \eta B_f^{(k)} + B_{n,f}^{+(k)}, \tag{7.4}$$

where $\eta \in [0, 1)$ is a factor of decay. The posteriors $\gamma_{t,f}$ during $\mathcal{T}_n^+$ are then updated using (2.45) (L16).

The updates of $\alpha_f^{(k)}$, $B_f^{(k)}$, and $\gamma_{t,f}^{(k)}$ above are conducted using the samples of the block and its pre-context. By using the pre-context, the permutation problem can be

solved, as in the conventional offline GSS algorithm. Furthermore, the posteriors $\gamma_{t,f}^{(k)}$ for the context are computed once in the previous iteration; they are helpful for accurate estimation of the mixture weights $\alpha_f^{(k)}$ and matrix parameter $B_f^{(k)}$, and eventually calculation of posteriors for the current block only from one EM iteration. Note that the case of $\eta = 0$ corresponds to a block-wise calculation of the parameter $B_f^{(k)}$. In this study, we set $\eta = 0.9$.

After the update of the cACGMM, an MVDR beamformer is calculated using the optimized cACGMM. For each utterance spoken during the block, regardless of whether the utterance is finished, we calculate spatial covariance matrices from $\mathbf{X}_n^+$, calculate a beamformer, and output enhanced audio during the utterance (L17–20).

7.4 Experimental Settings

To evaluate the proposed algorithm, we used two corpora: the CHiME-6 dataset and the meeting dataset, listed in Table A.5. For speech recognition of the CHiME-6 dataset, we used an acoustic model based on a factorized time delay neural network and 3-gram language model with two-stage decoding as in the baseline system. For the meeting dataset, we evaluated the character error rates (CERs) using various combinations of microphones: 2 microphones (⑧&⑩), 3 microphones (⑦&⑨&⑪), 6 microphones (①-⑥), and 11 microphones (①–⑪). We used our meeting transcription system described in Chapter 6 for evaluation by replacing its speech enhancement module with the proposed online GSS.

A parameter set for the proposed algorithm is shown in Table 7.1. To enable real-time processing, the tap size of the WPE was set to two, which was set to 10 in the offline baselines. The block size L and the pre-context size C were varied among the experiments. From the view point of utterance, the number of pre-context frames for each utterance c_{pre} fulfills $C \leq c_{\text{pre}} \leq C + L - 1$ and that of post-context frames for each utterance c_{post} fulfills $0 \leq c_{\text{post}} \leq L - 1$. The offline baseline uses 10 s of pre- and post-contexts for the CHiME-6 corpus and 15 s of them for the meeting corpus, as in previous studies. For diarization information $d_t^{(k)}$, we used oracle speech segments. In the CHiME-6 evaluation, we also used estimated diarization results obtained by a single iteration of target-speaker voice activity detection (TS-VAD) [25][1].

Note that a block-wise input sometimes contains a new speaker k with a very limited number of active frames. In such a case, the estimated matrix parameter

[1] https://github.com/kaldi-asr/kaldi/tree/master/egs/chime6/s5b_track2

Table 7.1: Parameters used in online experiments.

Audio sampling rate	16 kHz
STFT window length	64 ms
STFT window shift	16 ms
STFT window function	Hanning
WPE taps	2 frames
WPE delay	2 frames
WPE decay factor	0.9

$B_f^{(k)}$ is not reliable. To avoid using such an unreliable parameter to process the next block, we treated the speaker k as a new speaker in the next block processing, *i.e.*, we conducted L9–11 in Algorithm 7.1 once again for the speaker k, if the duration of the active frames was less than 0.2 s.

7.5 Results

We first evaluated the performance of the proposed online GSS algorithm on the CHiME-6 development set using the oracle segments. The results are shown in Table 7.2(a). The decay strategy always performed better than the accumulation strategy when the same parameters (L, C) were used. This indicates that the decay strategy is suitable for home environments in which speakers are moving during a session. With the decay strategy, the WERs improved by using the pre-context from 57.3 % to 51.6 % when $L = 150$ (2.4 s) and from 56.0 % to 52.3 % when $L = 300$ (4.8 s). By comparing the results of $(L, C) = (300, 0)$ and $(150, 150)$, we can also observe that the pre-context improved WERs from 56.0 % to 51.6 % even if the lengths of the processing unit $L + C$ are the same. These results indicate that using pre-context for parameter update is important for accurate mask estimation only from one EM iteration.

We also evaluated the performance of the proposed online GSS algorithm using estimated diarization results. They were obtained by a single iteration of TS-VAD, which showed a diarization error rates of 46.5 % and 53.62 % on S02 and S09, respectively. The results shown in Table 7.2(b) indicate that the proposed method works as well as the offline GSS, even if it is based on the estimated diarization results.

Figure 7.2 shows WERs with various block sizes L and pre-context sizes C using the oracle segments. The decay strategy with $\eta = 0.9$ was used for the proposed online GSS algorithm. We found that the WERs slightly degraded as L became

Table 7.2: WERs (%) on CHiME-6 development set.

(a) With oracle segments.

Algorithm	Block L	Context C	Session S02	Session S09	All
Offline [16, 15]	—	—	52.2	51.1	51.8
Online (Accumulation)	150	0	59.4	64.7	61.5
Online (Accumulation)	300	0	59.1	64.3	61.1
Online (Accumulation)	150	150	62.9	63.6	63.1
Online (Accumulation)	300	300	62.8	63.8	63.2
Online (Decay, $\eta = 0.9$)	150	0	55.7	59.9	57.3
Online (Decay, $\eta = 0.9$)	300	0	54.1	59.0	56.0
Online (Decay, $\eta = 0.9$)	150	150	50.6	53.3	51.6
Online (Decay, $\eta = 0.9$)	300	300	51.4	53.7	52.3

(b) With estimated diarization results obtained by TS-VAD.

Algorithm	Block L	Context C	Session S02	Session S09	All
Offline [16, 15]	—	—	70.2	70.0	70.1
Online (Decay, $\eta = 0.9$)	150	150	70.1	71.3	70.6

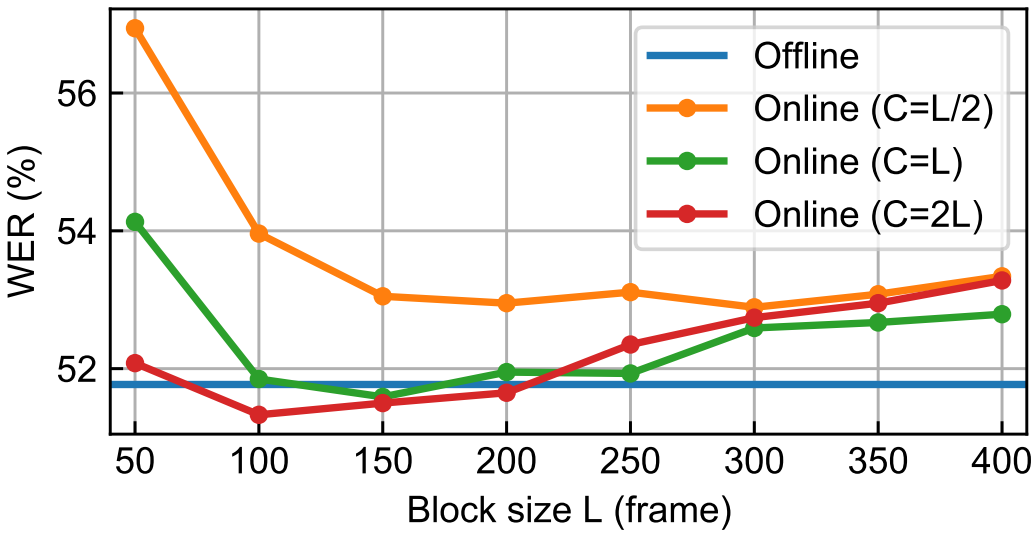


Figure 7.2: WERs (%) on CHiME-6 development set with various block size L and pre-context size C. Decay strategy with $\eta = 0.9$ was used for proposed algorithm.

Table 7.3: CERs (%) on the meeting corpus recorded using asynchronous distributed microphones. The block size L and context size C were set to 150 for proposed algorithm.

		Session								
#Mic	Method	I	II	III	IV	V	VI	VII	VIII	All
2	Offline	19.4	21.5	26.8	31.0	19.3	34.9	32.2	40.7	27.1
	Online (Accumulation)	19.9	25.7	27.9	32.1	20.0	36.7	32.2	40.2	28.2
	Online (Decay, $\eta = 0.9$)	20.3	20.8	27.3	31.5	19.7	36.2	32.3	39.9	27.4
3	Offline	21.7	22.0	31.4	29.3	20.0	38.2	36.0	41.0	28.9
	Online (Accumulation)	21.9	23.4	34.2	31.3	20.5	39.6	38.9	43.4	30.4
	Online (Decay, $\eta = 0.9$)	21.1	22.9	32.9	30.6	19.6	37.6	36.3	42.0	29.2
6	Offline	19.3	18.2	24.7	23.5	17.7	28.4	28.3	34.1	23.5
	Online (Accumulation)	18.6	20.1	24.1	24.2	17.7	31.8	31.0	37.0	24.6
	Online (Decay, $\eta = 0.9$)	18.7	17.9	25.6	23.8	17.9	32.2	31.9	37.2	24.6
11	Offline	17.0	16.3	21.7	21.2	17.7	27.0	27.0	32.8	21.8
	Online (Accumulation)	17.2	16.9	20.3	22.9	17.6	26.2	27.8	33.2	21.9
	Online (Decay, $\eta = 0.9$)	17.1	15.4	21.7	21.8	17.5	27.0	26.7	33.4	21.7

larger. This means that the MVDR beamformer should be calculated at short intervals under speaker-moving conditions, such as in CHiME-6. We also observed that the WERs are highly dependent on C when L is small, *i.e.*, $L = 50$. This indicates that the size of a unit to update parameters and calculate MVDR beamformers should be large to some extent. In this case, $L + C \geq 200$ was sufficient to avoid performance degradation due to the smallness of the unit.

We also evaluated the proposed algorithm on the meeting corpus using various combinations of asynchronous distributed microphones. In this experiment, the block size L and the pre-context size C were set to 150 frames and the oracle segments were used. The results are shown in Table 7.3. The proposed algorithm performed comparatively with the conventional offline GSS, *e.g.*, 21.8 % CER with the offline GSS and 21.7 % CER with the online GSS with the decay strategy, respectively, by using 11 microphones. In terms of the update strategy of the matrix parameter $B_f^{(k)}$, the decay strategy showed equivalent or better CERs than the accumulation strategy in overall performance, which is the same trend as the results from the CHiME-6 corpus.

Finally, we show the execution times on the CHiME-6 development set in Table 7.4. We showed the average and standard deviation of ten trials using Intel® Xeon® Gold 6132 CPU@2.60 GHz with a single thread. The proposed algorithm enhanced all utterances in the CHiME-6 development set within 2.65 hours, while

Table 7.4: Execution times on CHiME-6 development set. Mean and standard deviation among 10 trials are shown. The block size L and the pre-context size C were set to 150, which corresponds to 2.4 s.

		Duration (s)		Execution time (s)	
Session	#Mic	Total	Speech	Offline	Online
S02	12	8902	8492	183529 ± 9567	6135 ± 93
S09	10	7160	5552	124054 ± 7114	3418 ± 66

the conventional offline GSS algorithm required 85.44 hours by using the Kaldi CHiME-6 recipe. This means that the proposed algorithm is about 32x faster than the conventional offline GSS algorithm on the CHiME-6 corpus. In terms of real-time processing, the proposed algorithm skips most of the processing (L6–20 in Algorithm 7.1) if there is no speech activity in the input block; thus, the execution time should be compared with speech duration. As shown in Table 7.4, the execution time for each session is less than the speech duration; so we can fairly conclude that the proposed algorithm can be used in real-time applications.

7.6 Conclusion

In this chapter, we proposed a block-online algorithm of GSS. A block-wise input and its pre-context are used together to update the cACGMM parameters and posteriors of active speakers, which are then used to calculate MVDR beamformers. The proposed algorithm achieved almost the same performance as the conventional offline GSS algorithm on both the CHiME-6 and the meeting corpora, but with 32x faster calculation, which is sufficient for real-time processing.

Chapter 8

Conclusions

8.1 Contributions

This thesis has investigated ways of improving the practicality of speaker diarization from two aspects: various extensions for end-to-end speaker diarization and applications to utilize speaker diarization results. We summarize the contributions of each chapter below.

In Chapter 3, we proposed methods to handle flexible numbers of speakers with an end-to-end framework for speaker diarization. We first presented EEND-EDA, in which diarization results are estimated with flexible numbers of speaker-wise attractors calculated using the proposed EDA module. We further extended the method to EEND-GLA to deal with not only flexible but unlimited numbers of speakers by applying EDA-based diarization for each short-divided segment followed by unsupervised clustering. The experimental results showed that EEND-EDA and EEND-GLA successfully achieved overlap-aware speaker diarization with a single model even when the number of speakers is unknown. We also proposed BW-STB, which enabled block-online inference of EEND-GLA. It significantly outperformed the conventional cascaded and end-to-end approaches on a wide range of corpora.

In Chapter 4, we proposed methods to exploit spatial information from multi-channel inputs, especially from distributed microphones, in the end-to-end speaker diarization framework. First, we presented two types of multi-channel EEND based on spatio-temporal and co-attention encoders, respectively. The model based on co-attention encoders performed evenly to the conventional single-channel EEND given a single-channel input, and performed better as the number of input channels increased. It was also a remarkable result that the co-attention-based

EEND was adapted to a target domain only with single-channel data. Second, we presented a mutual learning method, in which knowledge distillation from multi-channel EEND to single-channel EEND and finetuning from single-channel EEND to multi-channel EEND were iteratively conducted. The experimental results on co-attention-based EEND showed mutual learning to improve the diarization performance of both single and multi-channel EEND.

In Chapter 5, we proposed a method to use a two-speaker EEND model as a post-processing step, *i.e.*, overlap detection and speaker assignment, of cascaded approaches for speaker diarization. For each speaker pair given initial diarization results, the frames that contain only the two speakers and silence are selected and processed with EEND to update the corresponding results. The proposed method successfully reduced DERs of various cascaded methods, even when another overlap handling method had already been applied.

In Chapter 6, we proposed a meeting transcription system based on asynchronous distributed microphones. In the system, the following modules were cascaded: correlation-coefficient-based blind synchronization, clustering-based speaker diarization, speech separation with GSS, ASR, and duplication reduction. It reduced CERs as the number of microphones increased, and also showed the CERs comparable to those of headset-based close-talk transcriptions if the oracle diarization was given. The results indicated the necessity of highly accurate speaker diarization.

In Chapter 7, we proposed a block-online algorithm of GSS. In the algorithm, each block-wise input is processed only with its pre-context to reduce the latency. The parameters of the generative model based on cACGMM are updated in an online manner with a decay factor, which enables the method to deal with moving speakers or sampling frequency mismatches between microphones. The transcriptions using the proposed method showed the almost same WERs / CERs as those using the conventional offline GSS even with sufficient processing speed for real-time purposes.

8.2 Future Directions

8.2.1 Simulated Datasets for the Training of Diarization Models

One drawback of end-to-end speaker diarization is that it relies on a large number of simulated mixtures; for example, the disk usage of the training set of the simulation data is shown below.

- Sim1spk: 115 GBi
- Sim2spk: 133 GBi
- Sim3spk: 226 GBi
- Sim4spk: 356 GBi
- Sim5spk: 492 GBi

On the other hand, speaker embedding extractors that are used in cascaded approaches only require one-speaker recordings in their training step, which is storage friendly. One possible solution is to create simulation data on the fly. However, some studies have revealed that the quality (or naturalness) of the simulated conversations has a large impact on the diarization performance on real datasets and thus proposed better simulation protocols based on autoregressive processes [184, 185]. Using these protocols for on-the-fly simulation will significantly increase the model training time. Methods that simultaneously achieve highly-accurate models, less training time, and better storage efficiency are important.

8.2.2 Speaker Diarization in Wilder Conditions

In this thesis, we have tested the effectiveness of the proposed methods on a variety of datasets, and it can be said that the methods work in a wide range of situations. However, there may be wilder situations, which have not yet been examined. For example, each recording in the DIHARD datasets has at most 10 speakers, but this number is by no means sufficient. For example, the recently released AVA-AVD data set [186] contains speech with more than 20 speakers, and even with EEND-EDA, the DER is reported to be about 50 % [187]. Whether EEND-GLA can work effectively in situations with a very large number of speakers remains to be explored. In addition, since the multi-channel diarization study was conducted using only two-speaker recordings in this thesis, there is room for further study on whether the combination with EEND-GLA can work in situations where the number of speakers is unknown. Another possible future work is to investigate whether the system can handle speaker movement. In the case where the speaker is moving around, how to generate large-scale simulation data and how to systematically collect a large amount of real data will be practical issues to be considered.

8.2.3 Tight Integration of Diarization and Subsequent Processes

Since speaker diarization is a preceding step of speech separation and ASR, only focusing on speaker diarization may not be a best Indeed, some studies have already

investigated a joint optimization of speaker diarization and speech separation [188] or speaker diarization and ASR [189]. There are also investigations on improving speaker diarization by using speech separation [190, 18, 191] and ASR [192, 53, 193]. However, these methods simply parallelized or cascaded multiple modules. Moreover, indeed speech separation and ASR can benefit speaker diarization as in the studies above, this thesis has revealed that the opposite is also true—speaker diarization can benefit speech separation and ASR. A promising future work will be a tight integration of speaker diarization and other speech processing, in which all the modules affect each other.

Bibliography

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

[2] Y. Xue, S. Horiguchi, Y. Fujita, Y. Takashima, S. Watanabe, P. Garcia, and K. Nagamatsu, "Online streaming end-to-end neural diarization handling overlapping speech and flexible numbers of speakers," in *Proc. INTERSPEECH*, 2021, pp. 3116–3120.

[3] M. Diez, L. Burget, F. Landini, and J. Černocký, "Analysis of speaker diarization based on bayesian HMM with eigenvoice priors," *IEEE/ACM TASLP*, vol. 28, pp. 355–368, 2020.

[4] F. Landini, J. Profant, M. Diez, and L. Burget, "Bayesian HMM clustering of x-vector sequences (VBx) in speaker diarization: Theory, implementation and analysis on standard tasks," *Computer Speech & Language*, vol. 71, p. 101254, 2022.

[5] L. Bullock, H. Bredin, and L. P. Garcia-Perera, "Overlap-aware diarization: Resegmentation using neural end-to-end overlapped speech detection," in *Proc. ICASSP*, 2020, pp. 7114–7118.

[6] "2000 NIST Speaker Recognition Evaluation," https://catalog.ldc.upenn.edu/LDC2001S97.

[7] J. Carletta, "Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus," *Language Resources and Evaluation*, vol. 41, no. 2, pp. 181–190, 2007.

[8] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke *et al.*, "The icsi meeting corpus," in *Proc. ICASSP*, vol. 1, 2003, pp. 364–367.

[9] S. Watanabe, M. Mandel, J. Barker, E. Vincent, A. Arora, X. Chang, S. Khudanpur, V. Manohar, D. Povey, D. Raj *et al.*, "CHiME-6 Challenge: Tackling multi-

speaker speech recognition for unsegmented recordings," in *Proc. CHiME-6*, 2020.

[10] T. Yoshioka, H. Erdogan, Z. Chen, X. Xiao, and F. Alleva, "Recognizing overlapped speech in meetings: A multichannel separation approach using neural networks," in *Proc. INTERSPEECH*, 2018, pp. 3038–3042.

[11] Z. Chen, X. Xiao, T. Yoshioka, H. Erdogan, J. Li, and Y. Gong, "Multi-channel overlapped speech recognition with location guided speech extraction network," in *Proc. SLT*, 2018, pp. 558–565.

[12] S. Horiguchi, Y. Fujita, and K. Nagamatsu, "Utterance-wise meeting transcription system using asynchronous distributed microphones," in *Proc. INTERSPEECH*, 2020, pp. 344–348.

[13] Ö. Çetin and E. Shriberg, "Analysis of overlaps in meetings by dialog factors, hot spots, speakers, and collection site: insights for automatic speech recognition," in *Proc. ICSLP*, 2006, pp. 293–296.

[14] J. Barker, S. Watanabe, E. Vincent, and J. Trmal, "The fifth 'CHiME' Speech Separation and Recognition Challenge: dataset, task and baselines," in *Proc. INTERSPEECH*, 2018, pp. 1561–1565.

[15] S. Watanabe, M. Mandel, J. Barker, E. Vincent, A. Arora, X. Chang, S. Khudanpur, V. Manohar, D. Povey, D. Raj, D. Snyder, A. S. Subramanian, J. Trmal, B. B. Yair, C. Boeddeker, Z. Ni, Y. Fujita, S. Horiguchi, N. Kanda, T. Yoshioka, and N. Ryant, "CHiME-6 Challenge: Tackling multispeaker speech recognition for unsegmented recordings," in *Proc. CHiME-6*, 2020.

[16] C. Boeddeker, J. Heitkaemper, J. Schmalenstoeer, L. Drude, J. Heymann, and R. Haeb-Umbach, "Front-end processing for the CHiME-5 dinner party scenario," in *Proc. CHiME-5*, 2018.

[17] M. Delcroix, K. Zmolikova, T. Ochiai, K. Kinoshita, and T. Nakatani, "Speaker activity driven neural speech extraction," in *Proc. ICASSP*, 2021, pp. 6099–6103.

[18] X. Fang, Z.-H. Ling, L. Sun, S.-T. Niu, J. Du, C. Liu, and Z.-C. Sheng, "A deep analysis of speech separation guided diarization under realistic conditions," in *Proc. APSIPA ASC*, 2021, pp. 667–671.

[19] T. von Neumann, K. Kinoshita, M. Delcroix, S. Araki, T. Nakatani, and R. Haeb-Umback, "All-neural online source separation, counting, and diarization for meeting analysis," in *Proc. ICASSP*, 2019, pp. 91–95.

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

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

[22] Q. Li, F. L. Kreyssig, C. Zhang, and P. C. Woodland, "Discriminative neural clustering for speaker diarisation," in *Proc. SLT*, 2021, pp. 574–581.

[23] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with permutation-free objectives," in *Proc. INTERSPEECH*, 2019, pp. 4300–4304.

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

[25] I. Medennikov, M. Korenevsky, T. Prisyach, Y. Khokhlov, M. Korenevskaya, I. Sorokin, T. Timofeeva, A. Mitrofanov, A. Andrusenko, I. Podluzhny, A. Laptev, and A. Romanenko, "Target-speaker voice activity detection: a novel approach for multi-speaker diarization in a dinner party scenario," in *Proc. INTERSPEECH*, 2020, pp. 274–278.

[26] F. Yu, S. Zhang, Y. Fu, L. Xie, S. Zheng, Z. Du, W. Huang, P. Guo, Z. Yan, B. Ma, X. Xu, and H. Bu, "M2MeT: The ICASSP 2022 multi-channel multi-party meeting transcription challenge," in *Proc. ICASSP*, 2022, pp. 6167–6171.

[27] S. Araki, N. Ono, K. Konoshita, and M. Delcroix, "Meeting recognition with asynchronous distributed microphone array," in *Proc. ASRU*, 2017, pp. 32–39.

[28] T. Yoshioka, D. Dimitriadis, A. Stolcke, W. Hinthorn, Z. Chen, M. Zeng, and X. Huang, "Meeting transcription using asynchronous distant microphones," in *Proc. INTERSPEECH*, 2019, pp. 2968–2972.

[29] D. Wang, Z. Chen, and T. Yoshioka, "Neural speech separation using spatially distributed microphones," in *Proc. INTERSPEECH*, 2020, pp. 339–343.

[30] T. J. Park, M. Kumar, and S. Narayanan, "Multi-scale speaker diarization with neural affinity score fusion," in *Proc. ICASSP*, 2021, pp. 7173–7177.

[31] T. J. Park, N. R. Koluguri, J. Balam, and B. Ginsburg, "Multi-scale speaker diarization with dynamic scale weighting," in *Proc. INTERSPEECH*, 2022, pp. 5080–5084.

[32] J.-W. Jung, H.-S. Heo, H.-J. Shim, and H.-J. Yu, "Short utterance compensation in speaker verification via cosine-based teacher-student learning of speaker embeddings," in *Proc. ASRU*, 2019, pp. 335–341.

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

[34] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE TASLP*, vol. 19, no. 4, pp. 788–798, 2010.

[35] S. Shum, N. Dehak, E. Chuangsuwanich, D. Reynolds, and J. Glass, "Exploiting intra-conversation variability for speaker diarization," in *Proc. INTERSPEECH*, 2011, pp. 945–948.

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

[37] M. Senoussaoui, P. Kenny, T. Stafylakis, and P. Dumouchel, "A study of the cosine distance-based mean shift for telephone speech diarization," *IEEE TASLP*, vol. 22, no. 1, pp. 217–227, 2014.

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

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

[40] M. Diez, L. Burget, S. Wang, J. Rohdin, and J. Černockỳ, "Bayesian HMM based x-vector clustering for speaker diarization," in *Proc. INTERSPEECH*, 2019, pp. 346–350.

[41] X. Xiao, N. Kanda, Z. Chen, T. Zhou, T. Yoshioka, S. Chen, Y. Zhao, G. Liu, Y. Wu, J. Wu, S. Liu, J. Li, and Y. Gong, "Microsoft speaker diarization system for the VoxCeleb speaker recognition challenge 2020," in *Proc. ICASSP*, 2021, pp. 5824–5828.

[42] S. H. Shum, N. Dehak, R. Dehak, and J. R. Glass, "Unsupervised methods for speaker diarization: An integrated and iterative approach," *IEEE TASLP*, vol. 21, no. 10, pp. 2015–2028, 2013.

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

[44] D. Garcia-Romero, D. Snyder, G. Sell, D. Povey, and A. McCree, "Speaker diarization using deep neural network embeddings," in *Proc. ICASSP*, 2017, pp. 4930–4934.

[45] M. Maciejewski, D. Snyder, V. Manohar, N. Dehak, and S. Khudanpur, "Characterizing performance of speaker diarization systems on far-field speech using standard methods," in *Proc. ICASSP*, 2018, pp. 5244–5248.

[46] D. Raj, Z. Huang, and S. Khudanpur, "Multi-class spectral clustering with overlaps for speaker diarization," in *Proc. SLT*, 2021, pp. 582–589.

[47] T. J. Park, K. J. Han, M. Kumar, and S. Narayanan, "Auto-tuning spectral clustering for speaker diarization using normalized maximum eigengap," *IEEE Signal Processing Letters*, vol. 27, pp. 381–385, 2020.

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

[49] F. Landini, S. Wang, M. Diez, L. Burget, P. Matějka, K. Žmolíková, L. Mošner, A. Silnova, O. Plchot, O. Novotnỳ, H. Zeinali, and J. Rohdin, "BUT system for the Second DIHARD Speech Diarization Challenge," in *Proc. ICASSP*, 2020, pp. 6529–6533.

[50] F. Landini, S. Wang, M. Diez, L. Burget, P. Matějka, K. Žmolíková, L. Mošner, O. Plchot, O. Novotnỳ, H. Zeinali, and J. Rohdin, "BUT system description for DIHARD Speech Diarization Challenge 2019," arXiv:1910.08847, 2019.

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

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

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

[54] X. Anguera, C. Wooters, and J. Hernando, "Acoustic beamforming for speaker diarization of meetings," *IEEE TASLP*, vol. 15, no. 7, pp. 2011–2022, 2007.

[55] K. Ishiguro, T. Yamada, S. Araki, T. Nakatani, and H. Sawada, "Probabilistic speaker diarization with bag-of-words representations of speaker angle information," *IEEE TASLP*, vol. 20, no. 2, pp. 447–460, 2011.

[56] S. Araki, M. Fujimoto, K. Ishizuka, H. Sawada, and S. Makino, "A DOA based speaker diarization system for real meetings," in *Proc. HSCMA*, 2008, pp. 29–32.

[57] S. Ding, Q. Wang, S.-y. Chang, L. Wan, and I. L. Moreno, "Personal VAD: Speaker-conditioned voice activity detection," in *Proc. Odyssey*, 2020, pp. 433–439.

[58] Q. Wang, I. L. Moreno, M. Saglam, K. Wilson, A. Chiao, R. Liu, Y. He, W. Li, J. Pelecanos, M. Nika, and A. Gruenstein, "VoiceFilter-Lite: Streaming targeted voice separation for on-device speech recognition," in *Proc. INTERSPEECH*, 2020, pp. 2677–2681.

[59] D. Wang, X. Xiao, N. Kanda, T. Yoshioka, and J. Wu, "Target speaker voice activity detection with transformers and its integration with end-to-end neural diarization," arXiv:2208.13085, 2022.

[60] K. Kinoshita, L. Drude, M. Delcroix, and T. Nakatani, "Listening to each speaker one by one with recurrent selective hearing networks," in *Proc. ICASSP*, 2018, pp. 5064–5068.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.

[62] Y. C. Liu, E. Han, C. Lee, and A. Stolcke, "End-to-end neural diarization: From transformer to conformer," in *Proc. INTERSPEECH*, 2021, pp. 3081–3085.

[63] S. Maiti, H. Erdogan, K. Wilson, S. Wisdom, S. Watanabe, and J. R. Hershey, "End-to-end diarization for variable number of speakers with local-global networks and discriminative speaker embeddings," in *Proc. ICASSP*, 2021, pp. 7183–7187.

[64] Y. Yu, D. Park, and H. Kook Kim, "Auxiliary loss of transformer with residual connection for end-to-end speaker diarization," in *Proc. ICASSP*, 2022, pp. 8377–8381.

[65] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. ICASSP*, 2016, pp. 31–35.

[66] D. Yu, M. Kolbæk, Z.-H. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *Proc. ICASSP*, 2017, pp. 241–245.

[67] S. Horiguchi, N. Yalta, P. Garcia, Y. Takashima, Y. Xue, D. Raj, Z. Huang, Y. Fujita, S. Watanabe, and S. Khudanpur, "The Hitachi-JHU DIHARD III system: Competitive end-to-end neural diarization and x-vector clustering systems combined by DOVER-Lap," in *Proc. DIHARD III*, 2021.

[68] K. Kinoshita, M. Delcroix, S. Araki, and T. Nakatani, "Tackling real noisy reverberant meetings with all-neural source separation, counting, and diarization system," in *Proc. ICASSP*, 2020, pp. 381–385.

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

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

[71] S. Horiguchi, Y. Fujita, S. Wananabe, Y. Xue, and K. Nagamatsu, "End-to-end speaker diarization for an unknown number of speakers with encoder-decoder based attractors," in *Proc. INTERSPEECH*, 2020, pp. 269–273.

[72] S. Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and P. García, "Encoder-decoder based attractors for end-to-end neural diarization," *IEEE/ACM TASLP*, vol. 30, pp. 1493–1507, 2022.

[73] N. Kanda, C. Boeddeker, J. Heitkaemper, Y. Fujita, S. Horiguchi, K. Nagamatsu, and R. Haeb-Umbach, "Guided source separation meets a strong ASR backend: Hitachi/Paderborn University joint investigation for dinner party scenario," in *Proc. INTERSPEECH*, 2019, pp. 1248–1252.

[74] C. Zorilă, C. Boeddeker, R. Doddipatla, and R. Haeb-Umbach, "An investigation into the effectiveness of enhancement in ASR training and test for CHiME-5 dinner party transcription," in *Proc. ASRU*, 2019, pp. 47–53.

[75] N. Ito, S. Araki, and T. Nakatani, "Complex angular central Gaussian mixture model for directional statistics in mask-based microphone array signal processing," in *Proc. EUSIPCO*, 2016, pp. 1153–1157.

[76] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B.-H. Juang, "Speech dereverberation based on variance-normalized delayed linear prediction," *IEEE TASLP*, vol. 18, no. 7, pp. 1717–1731, 2010.

[77] J. T. Kent, "Data analysis for shapes and images," *Journal of statistical planning and inference*, vol. 57, no. 2, pp. 181–193, 1997.

[78] E. Warsitz and R. Haeb-Umbach, "Blind acoustic beamforming based on generalized eigenvalue decomposition," *IEEE TASLP*, vol. 15, no. 5, pp. 1529–1539, 2007.

[79] Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, and N. Mesgarani, "Deep clustering and conventional networks for music separation: Stronger together," in *Proc. ICASSP*, 2017, pp. 61–65.

[80] Y. Luo and N. Mesgarani, "TasNet: time-domain audio separation network for real-time, single-channel speech separation," in *Proc. ICASSP*, 2018, pp. 696–700.

[81] ——, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM TASLP*, vol. 27, no. 8, pp. 1256–1266, 2019.

[82] Y. Luo, Z. Chen, and N. Mesgarani, "Speaker-independent speech separation with deep attractor network," *IEEE/ACM TASLP*, vol. 26, no. 4, pp. 787–796, 2018.

[83] Y. Fujita, S. Watanabe, S. Horiguchi, Y. Xue, J. Shi, and K. Nagamatsu, "Neural speaker diarization with speaker-wise chain rule," arXiv:2006.01796, 2020.

[84] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering," *IEEE/ACM TASLP*, vol. 29, pp. 2840–2849, 2021.

[85] N. Takahashi, S. Parthasaarathy, N. Goswami, and Y. Mitsufuji, "Recursive speech separation for unknown number of speakers," in *Proc. INTERSPEECH*, 2019, pp. 1348–1352.

[86] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *Proc. ICASSP*, 2017, pp. 246–250.

[87] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, "Set Transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. ICML*, 2019, pp. 3744–3753.

[88] B. B. Meier, I. Elezi, M. Amirian, O. Dürr, and T. Stadelmann, "Learning neural models for end-to-end clustering," in *Proc. ANNPR*, 2018, pp. 126–138.

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

[90] D. Raj, L. P. Garcia-Perera, Z. Huang, S. Watanabe, D. Povey, A. Stolcke, and S. Khudanpur, "DOVER-Lap: A method for combining overlap-aware diarization outputs," in *Proc. SLT*, 2021, pp. 881–888.

[91] A. Stolcke and T. Yoshioka, "DOVER: A method for combining diarization outputs," in *Proc. ASRU*, 2019, pp. 757–763.

[92] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[93] L. Drude, T. von Neumann, and R. Haeb-Umbach, "Deep attractor networks for speaker re-identification and blind source separation," in *Proc. ICASSP*, 2018, pp. 11–15.

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

[95] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "The Second DIHARD Diarization Challenge: Dataset, task, and baselines," in *Proc. INTERSPEECH*, 2019, pp. 978–982.

[96] N. Ryant, P. Singh, V. Krishnamohan, R. Varma, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "The third DIHARD diarization challenge," in *Proc. INTERSPEECH*, 2021, pp. 3570–3574.

[97] F. Landini, A. Lozano-Diez, L. Burget, M. Diez, A. Silnova, K. Žmolíková, O. Glembek, P. Matějka, T. Stafylakis, and N. Brümmer, "BUT system description for the Third DIHARD Speech Diarization Challenge," in *Proc. DIHARD III*, 2021.

[98] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE." *JMLR*, vol. 9, no. 11, pp. 2579–2605, 2008.

[99] K. Kinoshita, M. Delcroix, and N. Tawara, "Advances in integration of end-to-end neural and clustering-based diarization for real conversational speech," in *Proc. INTERSPEECH*, 2021, pp. 3565–3569.

[100] ——, "Integrating end-to-end neural and clustering-based diarization: Getting the best of both worlds," in *Proc. ICASSP*, 2021, pp. 7198–7202.

[101] Y. Takashima, Y. Fujita, S. Horiguchi, S. Watanabe, P. Garcia, and K. Nagamatsu, "Semi-supervised training with pseudo-labeling for end-to-end neural diarization," in *Proc. INTERSPEECH*, 2021, pp. 3096–3110.

[102] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy, "Semantic instance segmentation via deep metric learning," arXiv:1703.10277, 2017.

[103] S. Kong and C. C. Fowlkes, "Recurrent pixel embedding for instance grouping," in *Proc. CVPR*, 2018, pp. 9018–9028.

[104] K. Wagstaff, C. Cardie, S. Rogers, S. Schroedl *et al.*, "Constrained k-means clustering with background knowledge," in *Proc. ICML*, 2001, pp. 577–584.

[105] Y. Yang, T. Rutayisire, C. Lin, T. Li, and F. Teng, "An improved Cop-Kmeans clustering for solving constraint violation based on MapReduce framework," *Fundamenta Informaticae*, vol. 29126, no. 4, pp. 301–318, 2013.

[106] C.-Y. Cheng, H.-S. Lee, Y. Tsao, and H.-M. Wang, "Multi-target filter and detector for unknown-number speaker diarization," arXiv:2203.16007, 2022.

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

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

[109] Y. Yue, J. Du, M. He, Y. Yang, and R. Wang, "Online speaker diarization with core samples selection," in *Proc. INTERSPEECH*, 2022, pp. 1466–1470.

[110] Y. Zhang, Q. Lin, W. Wang, L. Yang, X. Wang, J. Wang, and M. Li, "Low-latency online speaker diarization with graph-based label generation," in *Proc. Odyssey*, 2022, pp. 162–169.

[111] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[112] S. Araki, N. Ono, K. Kinoshita, and M. Delcroix, "Meeting recognition with asynchronous distributed microphone array using block-wise refinement of mask-based MVDR beamformer," in *Proc. ICASSP*, 2018, pp. 5694–5698.

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

[114] T. Ochiai, S. Watanabe, T. Hori, and J. R. Hershey, "Multichannel end-to-end speech recognition," in *Proc. ICML*, 2017, pp. 2632–2641.

[115] X. Wang, R. Li, S. H. Mallidi, T. Hori, S. Watanabe, and H. Hermansky, "Stream attention-based multi-array end-to-end speech recognition," in *Proc. ICASSP*, 2019, pp. 7105–7109.

[116] F.-J. Chang, M. Radfar, A. Mouchtaris, and M. Omologo, "Multi-channel transformer transducer for speech recognition," in *Proc. INTERSPEECH*, 2021, pp. 296–300.

[117] Y. Luo, E. Ceolini, C. Han, S.-C. Liu, and N. Mesgarani, "FaSNet: Low-latency adaptive beamforming for multi-microphone audio processing," in *Proc. ASRU*, 2019, pp. 260–267.

[118] Y. Luo, Z. Chen, N. Mesgarani, and T. Yoshioka, "End-to-end microphone permutation and number invariant multi-channel speech separation," in *Proc. ICASSP*, 2020, pp. 6394–6398.

[119] N. Furnon, R. Serizel, I. Illina, and S. Essid, "Distributed speech separation in spatially unconstrained microphone arrays," in *Proc. ICASSP*, 2021, pp. 4490–4494.

[120] Z.-Q. Wang and D. Wang, "Multi-microphone complex spectral mapping for speech dereverberation," in *Proc. ICASSP*, 2020, pp. 486–490.

[121] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," in *Proc. NIPS 2016 Deep Learning Symposium*, 2016.

[122] T. N. Sainath, R. J. Weiss, K. W. Wilson, B. Li, A. Narayanan, E. Variani, M. Bacchiani *et al.*, "Multichannel signal processing with deep neural networks for automatic speech recognition," *IEEE/ACM TASLP*, vol. 25, no. 5, pp. 965–979, 2017.

[123] Z.-Q. Wang, J. Le Roux, and J. R. Hershey, "Multi-channel deep clustering: Discriminative spectral and spatial embeddings for speaker-independent speech separation," in *Proc. ICASSP*, 2018, pp. 1–5.

[124] H. Taherian, Z.-Q. Wang, and D. Wang, "Deep learning based multi-channel speaker recognition in noisy and reverberant environments," in *Proc. INTERSPEECH*, 2019, pp. 4070–4074.

[125] I. Medennikov, M. Korenevsky, T. Prisyach, Y. Khokhlov, M. Korenevskaya, I. Sorokin, T. Timofeeva *et al.*, "Target-speaker voice activity detection: a novel approach for multi-speaker diarization in a dinner party scenario," in *Proc. INTERSPEECH*, 2020, pp. 274–278.

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

[127] T. Yoshioka, X. Wang, D. Wang, M. Tang, Z. Zhu, Z. Chen, and N. Kanda, "VarArray: Array-geometry-agnostic continuous speech separation," in *Proc. ICASSP*, 2022, pp. 6027–6031.

[128] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. KDD*, 2006, pp. 535–541.

[129] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learning and Representation Learning Workshop*, 2014.

[130] J. Heymann, L. Drude, C. Boeddeker, P. Hanebrink, and R. Haeb-Umbach, "Beamnet: End-to-end training of a beamformer-supported multi-channel ASR system," in *Proc. ICASSP*, 2017, pp. 5325–5329.

[131] W. Zhang, A. S. Subramanian, X. Chang, S. Watanabe, and Y. Qian, "End-to-end far-field speech recognition with unified dereverberation and beamforming," in *Proc. INTERSPEECH*, 2020, pp. 324–328.

[132] K. An, J. Xiao, and Z. Ou, "Exploiting single-channel speech for multi-channel end-to-end speech recognition: A comparative study," in *Proc. ISCSLP*, 2022, pp. 180–184.

[133] X. Chang, W. Zhang, Y. Qian, J. L. Roux, and S. Watanabe, "MIMO-Speech: End-to-end multi-channel multi-speaker speech recognition," in *Proc. ASRU*, 2019, pp. 237–244.

[134] Z.-Q. Wang and D. Wang, "On spatial features for supervised speech separation and its application to beamforming and robust ASR," in *Proc. ICASSP*, 2018, pp. 5709–5713.

[135] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria," in *Proc. INTERSPEECH*, 2014, pp. 1910–1914.

[136] X. Chen, G. Liu, J. Shi, J. Xu, and B. Xu, "Distilled binary neural network for monaural speech separation," in *Proc. IJCNN*, 2018, pp. 1–8.

[137] T. Tan, Y. Qian, and D. Yu, "Knowledge transfer in permutation invariant training for single-channel multi-talker speech recognition," in *Proc. ICASSP*, 2018, pp. 5714–5718.

[138] T. Fukuda, M. Suzuki, G. Kurata, S. Thomas, J. Cui, and B. Ramabhadran, "Efficient knowledge distillation from an ensemble of teachers," in *Proc. INTERSPEECH*, 2017, pp. 3697–3701.

[139] S. Kim, M. Seltzer, J. Li, and R. Zhao, "Improved training for online end-to-end speech recognition systems," in *Proc. INTERSPEECH*, 2018, pp. 2913–2917.

[140] Z. Meng, J. Li, Y. Gaur, and Y. Gong, "Domain adaptation via teacher-student learning for end-to-end speech recognition," in *Proc. ICASSP*, 2019, pp. 268–275.

[141] S. Watanabe, T. Hori, J. Le Roux, and J. R. Hershey, "Student-teacher network learning with enhanced features," in *Proc. ICASSP*, 2017, pp. 5275–5279.

[142] A. S. Subramanian, S.-J. Chen, and S. Watanabe, "Student-teacher learning for BLSTM mask-based speech enhancement," in *Proc. INTERSPEECH*, 2018, pp. 3249–3253.

[143] Z. Chen, T. Yoshioka, L. Lu, T. Zhou, Z. Meng, Y. Luo, J. Wu, X. Xiao, and J. Li, "Continuous speech separation: Dataset and analysis," in *Proc. ICASSP*, 2020, pp. 7284–7288.

[144] X. Wang, D. Wang, N. Kanda, S. E. Eskimez, and T. Yoshioka, "Leveraging real conversational data for multi-channel continuous speech separation," in *Proc. INTERSPEECH*, 2022, pp. 3814–3818.

[145] T. Kim, J. Oh, N. Kim, S. Cho, and S.-Y. Yun, "Comparing Kullback-Leibler divergence and mean squared error loss in knowledge distillation," in *Proc. IJCAI*, 2021, pp. 2628–2635.

[146] K. Maekawa, "Corpus of spontaneous Japanese: Its design and evaluation," in *Proc. ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[147] Y. Fujita, S. Watanabe, S. Horiguchi, Y. Xue, and K. Nagamatsu, "End-to-end neural diarization: Reformulating speaker diarization as simple multi-label classification," arXiv:2003.02966, 2020.

[148] M. Diez, L. Burget, and P. Matějka, "Speaker diarization based on Bayesian HMM with eigenvoice priors," in *Proc. Odyssey*, 2018, pp. 102–109.

[149] P. Garcia, J. Villalba, H. Bredin, J. Du, D. Castan, , A. Cristia, L. Bullock, L. Guo, K. Okabe, P. S. Nidadavolu, S. Kataria, S. Chen, L. Galmant, M. Lavechin, L. Sun, M.-P. Gill, Ben-Yair, S. Abdoli, X. Wang, W. Bouaziz, H. Titeux, E. Dupoux, K. A. Lee, and N. Dehak, "Speaker detection in the wild: Lessons learned from JSALT 2019," in *Proc. Odyssey*, 2020, pp. 415–422.

[150] A. Stolcke, X. Anguera, K. Boakye, Ö. Çetin, A. Janin, M. Magimai-Doss, C. Wooters, and J. Zheng, "The SRI-ICSI spring 2007 meeting and lecture recognition system," in *Multimodal Technologies for Perception of Humans*. Springer, 2007, pp. 450–463.

[151] T. Hain, L. Burget, J. Dines, P. N. Garner, F. Grézl, A. El Hannani, M. Huijbregts, M. Karafiat, M. Lincoln, and V. Wan, "Transcribing meetings with the AMIDA systems," *IEEE TASLP*, vol. 20, no. 2, pp. 486–498, 2011.

[152] N. Ito, S. Araki, M. Delcroix, and T. Nakatani, "Probabilistic spatial dictionary based online adaptive beamforming for meeting recognition in noisy and reverberant environments," in *Proc. ICASSP*, 2017, pp. 681–685.

[153] T. Hori, S. Araki, T. Yoshioka, M. Fujimoto, S. Watanabe, T. Oba, A. Ogawa, K. Otsuka, D. Mikami, K. Kinoshita, T. Nakatani, A. Nakamura, and Y. Junji, "Low-latency real-time meeting recognition and understanding using distant microphones and omni-directional camera," *IEEE TASLP*, vol. 20, no. 2, pp. 499–513, 2011.

[154] T. Yoshioka, I. Abramovski, C. Aksoylar, Z. Chen, M. David, D. Dimitriadis, Y. Gong, I. Gurvich, X. Huang, Y. Huang, A. Hurvitz, L. Jiang, S. Koubi,

E. Krupka, I. Leichter, C. Liu, P. Parthasarathy, A. Vinnikov, L. Wu, X. Xiao, W. Xiong, H. Wang, Z. Wang, J. Zhang, Y. Zhao, and T. Zhou, "Advances in online audio-visual meeting transcription," in *Proc. ASRU*, 2019, pp. 276–283.

[155] T. Higuchi, N. Ito, S. Araki, T. Yoshioka, M. Delcroix, and T. Nakatani, "Online MVDR beamformer based on complex gaussian mixture model with spatial prior for noise robust ASR," *IEEE/ACM TASLP*, vol. 25, no. 4, pp. 780–793, 2017.

[156] H. Sawada, R. Mukai, S. Araki, and S. Makino, "A robust and precise method for solving the permutation problem of frequency-domain blind source separation," *IEEE TASLP*, vol. 12, no. 5, pp. 530–538, 2004.

[157] H. Sawada, S. Araki, and S. Makino, "Underdetermined convolutive blind source separation via frequency bin-wise clustering and permutation alignment," *IEEE TASLP*, vol. 19, no. 3, pp. 516–527, 2010.

[158] S. Miyabe, N. Ono, and S. Makino, "Blind compensation of inter-channel sampling frequency mismatch for ad-hoc microphone array based on maximum likelihood estimation," *Signal Processing*, vol. 107, pp. 185–196, 2015.

[159] S. Araki, N. Ono, K. Kinoshita, and M. Delcroix, "Estimation of sampling frequency mismatch between distributed asynchronous microphones under existence of source movements with stationary time periods detection," in *Proc. ICASSP*, 2019, pp. 785–789.

[160] C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," *IEEE TASSP*, vol. 24, no. 4, pp. 320–327, 1976.

[161] N. Kanda, Y. Fujita, and K. Nagamatsu, "Lattice-free state-level minimum Bayes risk training of acoustic models," in *Proc. INTERSPEECH*, 2018, pp. 2923–2927.

[162] ——, "Investigation of lattice-free maximum mutual information-based acoustic models with sequence-level Kullback-Leibler divergence," in *Proc. ASRU*, 2017, pp. 69–76.

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

[164] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. INTERSPEECH*, 2016, pp. 2751–2755.

[165] J. G. Fiscus, "A post-processing system to yield reduced work error rates: Recognizer output voting error reduction (ROVER)," in *Proc. ASRU*, 1997, pp. 347–352.

[166] G. Evermann and P. C. Woodland, "Posterior probability decoding, confidence estimation and system combination," in *Proc. NIST Speech Transcription Workshop*, vol. 27, 2000, pp. 78–81.

[167] I. Medennikov, M. Korenevsky, T. Prisyach, Y. Khokhlov, M. Korenevskaya, I. Sorokin, T. Timofeeva, A. Mitrofanov, A. Andrusenko, I. Podluzhny, A. Laptev, and A. Romanenko, "The STC system for the CHiME-6 Challenge," in *Proc. CHiME-6*, 2020.

[168] S. Araki, T. Hayashi, M. Delcroix, M. Fujimoto, K. Takeda, and T. Nakatani, "Exploring multi-channel features for denoising-autoencoder-based speech enhancement," in *Proc. ICASSP*, 2015, pp. 116–120.

[169] L. Drude and R. Haeb-Umbach, "Tight integration of spatial and spectral features for BSS with deep clustering embeddings," in *Proc. INTERSPEECH*, 2017, pp. 2650–2654.

[170] R. Gu, S.-X. Zhang, L. Chen, Y. Xu, M. Yu, D. Su, Y. Zou, and D. Yu, "Enhancing end-to-end multi-channel speech separation via spatial feature learning," in *Proc. ICASSP*, 2020, pp. 7319–7323.

[171] J. Zhang, C. Zorilă, R. Doddipatla, and J. Barker, "On end-to-end multi-channel time domain speech separation in reverberant environments," in *Proc. ICASSP*, 2020, pp. 6389–6393.

[172] J. Du, Y.-H. Tu, L. Sun, L. Chai, X. Tang, M.-K. He, F. Ma, J. Pan, J.-Q. Gao, D. Liu, C.-H. Lee, and J.-D. Chen, "The USTC-NELSLIP systems for CHiME-6 Challenge," in *Proc. CHiME-6*, 2020.

[173] H. Chen, P. Zhang, Q. Shi, and Z. Liu, "The IOA systems for CHiME-6 Challenge," in *Proc. CHiME-6*, 2020.

[174] B. Lin and X. Zhang, "Speaker diarization as a fully online bandit learning problem in MiniVox," in *Proc. ACML*, 2021, pp. 1660–1674.

[175] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," in *Proc. ICLR*, 2018.

[176] N. Arivazhagan, C. Cherry, W. Macherey, C.-C. Chiu, S. Yavuz, R. Pang, W. Li, and C. Raffel, "Monotonic infinite lookback attention for simultaneous machine translation," in *Proc. ACL*, 2019, pp. 1313–1323.

[177] R. Fan, P. Zhou, W. Chen, J. Jia, and G. Liu, "An online attention-based model for speech recognition," in *Proc. INTERSPEECH*, 2019, pp. 4390–4394.

[178] N. Moritz, T. Hori, and J. Le Roux, "Streaming automatic speech recognition with the transformer model," in *Proc. ICASSP*, 2020, pp. 6074–6078.

[179] S. Araki, N. Ito, M. Delcroix, A. Ogawa, K. Kinoshita, T. Higuchi, T. Yoshioka, D. Tran, S. Karita, and T. Nakatani, "Online meeting recognition in noisy environments with time-frequency mask based MVDR beamforming," in *Proc. HSCMA*, 2017, pp. 16–20.

[180] Y. Matsui, T. Nakatani, M. Delcroix, K. Kinoshita, N. Ito, S. Araki, and S. Makino, "Online integration of DNN-based and spatial clustering-based mask estimation for robust MVDR beamforming," in *Proc. IWAENC*, 2018, pp. 71–75.

[181] M. Togami, "Simultaneous optimization of forgetting factor and time-frequency mask for block online multi-channel speech enhancement," in *Proc. ICASSP*, 2019, pp. 2702–2706.

[182] T. Higuchi, K. Kinoshita, N. Ito, S. Karita, and T. Nakatani, "Frame-by-frame closed-form update for mask-based adaptive MVDR beamforming," in *Proc. ICASSP*, 2018, pp. 531–535.

[183] L. Drude, J. Heymann, C. Boeddeker, and R. Haeb-Umbach, "NARA-WPE: A python package for weighted prediction error dereverberation in Numpy and Tensorflow for online and offline processing," in *Proc. Speech Communication; 13th ITG-Symposium*, 2018, pp. 1–5.

[184] N. Yamashita, S. Horiguchi, and T. Homma, "Improving the naturalness of simulated conversations for end-to-end neural diarization," in *Odyssey*, 2022, pp. 133–140.

[185] F. Landini, A. Lozano-Diez, M. Diez, and L. Burget, "From simulated mixtures to simulated conversations as training data for end-to-end neural diarization," in *Proc. INTERSPEECH*, 2022, pp. 5095–5099.

[186] E. Z. Xu, Z. Song, S. Tsutsui, C. Feng, M. Ye, and M. Z. Shou, "AVA-AVD: Audio-visual speaker diarization in the wild," in *Proc. MM*, 2022, pp. 3838–3847.

[187] Z. Pan, G. Wichern, F. G. Germain, A. Subramanian, and J. L. Roux, "Towards end-to-end speaker diarization in the wild," arXiv:2211.01299, 2022.

[188] Y. Ueda, S. Maiti, S. Watanabe, C. Zhang, M. Yu, S.-X. Zhang, and Y. Xu, "EEND-SS: Joint end-to-end neural speaker diarization and speech separation for flexible number of speakers," in *Proc. SLT*, 2023.

[189] A. Khare, E. Han, Y. Yang, and A. Stolcke, "ASR-aware end-to-end neural diarization," in *Proc. ICASSP*, 2022, pp. 8092–8096.

[190] S.-T. Niu, J. Du, L. Sun, and C.-H. Lee, "Separation guided speaker diarization in realistic mismatched conditions," arXiv:2107.02357, 2021.

[191] G. Morrone, S. Cornell, D. Raj, L. Serafini, E. Zovato, A. Brutti, and S. Squartini, "Low-latency speech separation guided diarization for telephone conversations," 2023.

[192] T. J. Park, K. J. Han, J. Huang, X. He, P. Georgiou, and S. Narayanan, "Speaker diarization with lexical information," in *Proc. INTERSPEECH*, 2019, pp. 391–395.

[193] N. Kanda, X. Xiao, Y. Gaur, X. Wang, Z. Meng, Z. Chen, and T. Yoshioka, "Transcribe-to-diarize: Neural speaker diarization for unlimited number of speakers using end-to-end speaker-attributed asr," in *Proc. ICASSP*, 2022, pp. 8082–8086.

[194] D. Snyder, G. Chen, and D. Povey, "MUSAN: A music, speech, and noise corpus," arXiv:1510.08484, 2015.

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

[196] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, "`pyannote.audio`: neural building blocks for speaker diarization," in *Proc. ICASSP*, 2020, pp. 7124–7128.

[197] D. Diaz-Guerra, A. Miguel, and J. R. Beltran, "gpuRIR: A python library for room impulse response simulation with GPU acceleration," *Multimedia Tools and Applications*, vol. 80, no. 4, pp. 5653–5671, 2021.

[198] N. Kanda, S. Horiguchi, Y. Fujita, Y. Xue, K. Nagamatsu, and S. Watanabe, "Simultaneous speech recognition and speaker diarization for monaural dialogue recordings with target-speaker acoustic models," in *Proc. ASRU*, 2019, pp. 31–38.

Appendix A

Datasets

This chapter lists the datasets used for the experiments in this thesis.

A.1 Speaker Diarization Datasets

A.1.1 Single-Channel Datasets

Simulated Datasets

The simulated speech mixtures were created from single-speaker recordings of the following corpora.

- Switchboard-2 (Phase I & II & III)
- Switchboard Cellular (Part 1 & 2)
- NIST Speaker Recognition Evaluation (2004 & 2005 & 2006 & 2008)

We also used MUSAN [194] as a noise corpus and simulated room impulse responses [195] to emulate reverberated conditions. Note that these corpora are compatible with the Kaldi CALLHOME x-vector recipe[1].

We used the following simulation protocol to create multi-talker mixtures from single-speaker recordings:

1. Select N speakers,

[1] https://github.com/kaldi-asr/kaldi/tree/master/egs/callhome_diarization/v2

2. For each speaker, randomly sample speech segments and concatenate them with silences that are interlaid between speech segments,

3. For each of the N long recordings created, randomly select a room impulse response and convolve it with the recording,

4. Mix the N long recordings and a noise signal with a randomly determined signal-to-noise ratio.

The detailed algorithm for creating simulated mixtures can be found in [23], and the script for simulating mixtures is available online[2]. In the second process, we assume that the occurrence of an utterance is a Poisson process, so the duration of the silence between speech segments follows the exponential distribution $\frac{1}{\beta}\exp\left(-\frac{x}{\beta}\right)$, where β is the mean value. β can be used to control the overlap ratio of the mixtures. To obtain a similar overlap ratio among various numbers of speakers, we varied β according to the number of speakers as summarized in Table A.1. Note that the adaptation set of each simulated dataset was the subset of the corresponding training set.

Real Datasets

For real datasets, we employed five multi-talker datasets below.

- **CALLHOME** [6]: A dataset that consists of telephone conversations whose average duration is two minutes. We used the splits provided in the Kaldi x-vector recipe[1], which are denoted as Part 1 and Part 2, respectively. Two- and three-speaker subsets were used in the fixed-number-of-speakers evaluations, which are denoted as CALLHOME-2spk and CALLHOME-3spk.

- **CSJ** [146]: A dataset that consists of monologues and dialogues of Japanese speech. In this thesis, we used the dialogue part of the dataset. The average duration of the recordings is about 13 minutes. Following [73], we used 54 dialogue recordings out of 58.

- **AMI headset mix** [7]: A meeting dataset that consists of 100 hours of multimodal meeting recordings. Each meeting session is about 30 minutes. We used *headset mix* recordings, which were obtained by mixing the headset recordings of all the participants. We used the split and reference RTTMs provided in the VBx paper [4] in Section 3.2 and those provided in the `pyannote.audio` library [196] in Chapter 5.

[2] `https://github.com/hitachi-speech/EEND.git`

Table A.1: Statistics of single-channel simulated speaker diarization corpora.

Dataset	Split	#Spk	#Mixtures	β	Overlap ratio (%)
Sim1spk	Train	1	100,000	2	0.0
	Adaptation	1	1000	2	0.0
	Test	1	100,000	2	0.0
Sim2spk	Train	2	100,000	2	34.1
	Adaptation	2	1000	2	34.5
	Test	2	500	2	34.4
	Test	2	500	3	27.3
	Test	2	500	5	19.1
Sim3spk	Train	3	100,000	5	34.2
	Adaptation	3	1000	5	34.9
	Test	3	500	5	34.7
	Test	3	500	7	27.4
	Test	3	500	11	19.2
Sim4spk	Train	4	100,000	9	31.5
	Adaptation	4	1000	9	31.4
	Test	4	500	9	32.0
Sim5spk	Train	5	100,000	13	30.3
	Test	5	500	13	30.7
Sim6spk	Test	6	500	17	29.9

- **DIHARD II** [95]: A dataset used in the second DIHARD challenge. We used single-channel audio, which is used for tracks 1 and 2. The dataset consists of recordings from 11 domains (including telephone data) with an average duration of about 7 minutes.
- **DIHARD III** [96]: A dataset used in the third DIHARD challenge. It also consists of recordings from 11 domains (including telephone data) with an average duration of about 8 minutes. The test set has two evaluation conditions called *core* and *full*. The core set is a subset of the full set, in which the recordings are selected to balance the duration of each domain. In terms of the number of speakers, the full set contains more recordings of two speakers than the core set.

Their statistics are summarized in Table A.2. Note that the recordings in CSJ, AMI, DIHARD II, and DIHARD III were sampled at 16 kHz, so we downsampled them to 8 kHz to be aligned with those of the simulated datasets. We also note that the recordings of the CSJ corpus are in stereo, so we mixed them to create monaural

Table A.2: Statistics of single-channel real-recorded speaker diarization datasets.

Dataset	Split	#Spk	#Mixtures	Overlap ratio
CALLHOME-2spk [6]	Part 1	2	155	14.0 %
	Part 2	2	148	13.1 %
CSJ [146]	—	2	54	20.1 %
CALLHOME-3spk [6]	Part 1	3	61	19.6 %
	Part 2	3	74	17.0 %
CALLHOME [6]	Part 1	2–7	249	17.0 %
	Part 2	2–6	250	16.7 %
AMI headset mix [7]				
VBx split [4]	Train	3–5	136	13.4 %
	Dev	4	18	14.1 %
	Test	3–4	16	14.6 %
`pyannote.audio` split [196]	Train	3–5	118	19.4 %
	Test	3–4	24	18.6 %
DIHARD II [95]	Dev	1–10	192	9.8 %
	Test	1–9	194	8.9 %
DIHARD III [96]	Dev	1–10	254	10.7 %
	Test (Core)	1–9	184	8.8 %
	Test (Full)	1–9	259	9.2 %

recordings.

A.1.2 Multi-Channel Datasets

We created three multi-channel two-speaker conversational datasets based on the corpora that were used to create the single-channel simulated datasets listed in Section A.1.1. To emulate a reverberant environment, we generated room impulse responses (RIRs) using gpuRIR [197]. Following the procedure in [195], we sampled 200 rooms for each of the three room sizes: small, medium, and large. In each room, a table was randomly placed, 10 speakers were randomly placed around the table, and 10 microphones were randomly placed on the table. To create Sim2spk-multi-train and Sim2spk-multi-eval, two-speaker conversations were simulated following [23] then RIRs of the randomly selected room and two speaker positions were convolved to obtain a 10-channel mixture. MUSAN corpus [194] was also used to

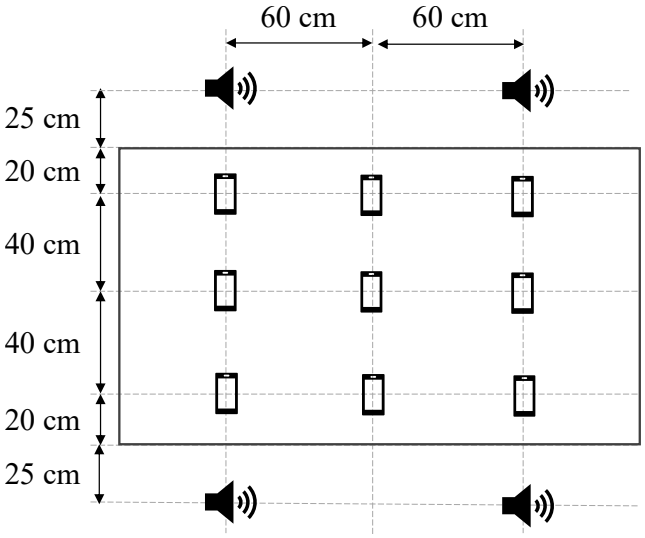


Table A.3: Recording environment of CSJ-multi-train, CSJ-multi-eval, and CSJ-multi-dialog.

add noise to each mixture. Sim2spk-multi-eval-hybrid was created using the same utterances in Sim2spk-multi-eval, but two speakers were placed at the same position. This dataset was designed to mimic the part of hybrid meetings, in which multiple speakers' utterances are played from a single loudspeaker.

We also prepared three real-recorded datasets on the basis of the corpus of spontaneous Japanese (CSJ) [146]: CSJ-multi-train, CSJ-multi-eval, and CSJ-multi-dialog[3], each of which is shown in Table A.4. For CSJ-multi-train and CSJ-multi-eval, 100 two-speaker conversations were simulated using single-speaker recordings in the CSJ training and evaluation sets, respectively. For CSJ-multi-dialog, we directly used the dialog portion of CSJ. To record each session, we distributed nine smartphone devices on a tabletop in a meeting room and four loudspeakers around the table, as shown in Table A.3. We played back two speakers' utterances from two of the four loudspeakers that were randomly selected and recorded them on the smartphone devices. Recorded signals were roughly synchronized to maximize the correlation coefficient and neither clock drift nor frame dropping was compensated.

Their statistics are listed in Table A.4.

[3]CSJ in Table A.2 and CSJ-multi-dialog in Table A.4 share the data source but the numbers of sessions are slightly different. This is because four of 58 sessions were eliminated in the single-channel CSJ following the conventional study [198].

Table A.4: Statistics of multi-channel speaker diarization corpora.

Dataset	Conversation	Record	#Mic	#Session	Average duration	Overlap ratio
Sim2spk-multi-train	Simulated	Simulated	10	20,000	88.7 s	34.1 %
Sim2spk-multi-eval	Simulated	Simulated	10	500	88.1 s	34.6 %
Sim2spk-multi-eval-hybrid	Simulated	Simulated	10	500	88.1 s	34.6 %
CSJ-multi-train	Simulated	Recorded	9	100	113.5 s	11.0 %
CSJ-multi-eval	Simulated	Recorded	9	100	102.2 s	9.6 %
CSJ-multi-dialog	Real	Recorded	9	58	755.2 s	17.3 %

Table A.5: Statistics of speech recognition datasets.

Dataset	Session	#Mic	Duration	#Spk	#Utt	Overlap
CHiME-6 dev [15]	S02	12	2:28:22	4	3,822	52.9 %
	S09	10	1:59:20	4	3,615	45.8 %
Meeting [12]	I	2/3/6/11	19:49	7	160	6.9 %
	II	2/3/6/11	14:27	8	150	14.0 %
	III	2/3/6/11	13:13	5	198	16.6 %
	IV	2/3/6/11	12:08	7	184	19.9 %
	V	2/3/6/11	12:37	6	80	5.5 %
	VI	2/3/6/11	16:50	6	256	14.7 %
	VII	2/3/6/11	16:25	7	223	11.3 %
	VIII	2/3/6/11	11:25	7	185	19.9 %

A.2 Speech Recognition Datasets

A.2.1 CHiME-6 Corpus

The CHiME-6 corpus [15] contains sessions each of which consists of a dinner party scenario with four participants. It was recorded using six distributed Kinect® v2 devices, each equipped with four microphones. The sampling frequency mismatch between devices was manually corrected. However, participants were moving around the kitchen, dining room, and living room; thus, a fixed beamformer for each participant does not exist. Following the CHiME-6 baseline system provided as a Kaldi recipe[4], we used 12-channel signals, *i.e.*, the outer two microphones of each device, for S02 and 10-channel signals for S09 because the recording of the fifth device was unavailable. The statistics are shown in Table A.5.

[4] `https://github.com/kaldi-asr/kaldi/tree/master/egs/chime6/s5_track1`

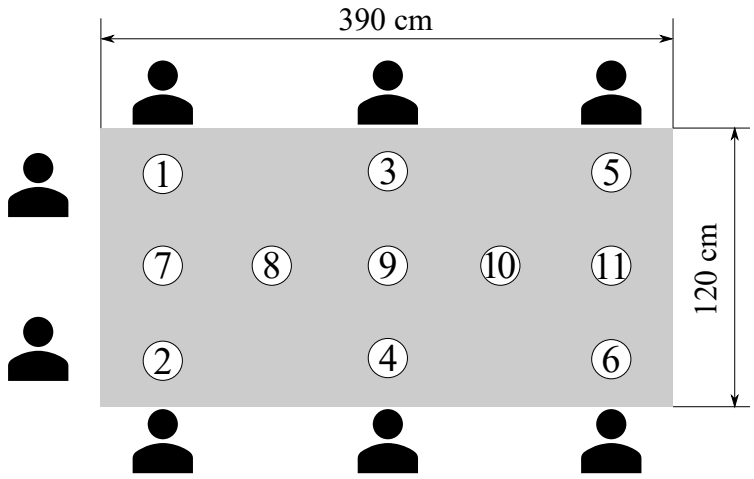


Figure A.1: Recording environment of the meeting corpus. 11 smartphones, each equipped with a monaural microphone, were distributed on the table.

A.2.2 Meeting Corpus

The meeting corpus is our internal dataset that consists of eight sessions of Japanese meeting data with 5–8 participants. The recording environment is shown in Figure A.1. Each session was recorded by 11 smartphones distributed on the table, each of which was equipped with a monaural microphone to record meetings at 16 kHz / 16 bit. Each participant wore a headset microphone and the groundtruth transcriptions were based on the headset recordings. Note that the sampling frequency mismatch between microphones is not corrected. The statistics of the dataset are also shown in Table A.5.

Appendix B

Publications

Publications Related to the Thesis

Journal Papers

[P1] Shota Horiguchi, S. Watanabe, P. Garcia, Y. Takashima, and Y. Kawaguchi, "Online neural diarization of unlimited numbers of speakers using global and local attractors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 704–720, Jan. 2023. → **Sections 3.3 & 3.4**

[P2] Shota Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and P. García, "Encoder-decoder based attractors for end-to-end neural diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1493–1507, Mar. 2022. → **Section 3.2**

International Conference Papers (Peer-Reviewed)

[P3] Shota Horiguchi, Y. Takashima, S. Watanabe, and P. García, "Mutual learning of single- and multi-channel end-to-end neural diarization," in *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, Jan. 2023, pp. 620–625. → **Section 4.3**

[P4] Shota Horiguchi, Y. Takashima, P. García, S. Watanabe, and Y. Kawaguchi, "Multi-channel end-to-end neural diarization with distributed microphones," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2022, pp. 7332–7336. → **Section 4.2**

[P5] Shota Horiguchi, P. García, S. Watanabe, Y. Xue, Y. Takashima, and

Y. Kawaguchi, "Towards neural diarization for unlimited numbers of speakers using global and local attractors," in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec. 2021, pp. 98–105. → **Section 3.3**

[P6] Shota Horiguchi, P. García, Y. Fujita, S. Watanabe, and K. Nagamatsu, "End-to-end speaker diarization as post-processing," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2021, pp. 7188–7192. → **Chapter 5**

[P7] Shota Horiguchi, Y. Fujita, and K. Nagamatsu, "Block-online guided source separation," in *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, Jan. 2021, pp. 236–242. → **Chapter 7**

[P8] Shota Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and K. Nagamatsu, "End-to-end speaker diarization for an unknown number of speakers with encoder-decoder based attractors," in *Proceedings of The Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Oct. 2020, pp. 269–273. → **Section 3.2**

[P9] Shota Horiguchi, Y. Fujita, and K. Nagamatsu, "Utterance-wise meeting transcription system using asynchronous distributed microphones," in *Proceedings of The Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Oct. 2020, pp. 344–348. → **Chapter 6**

Publications Non-Related to the Thesis

Journal Papers

[P12] Shota Horiguchi, D. Ikami, and K. Aizawa, "Significance of softmax-based features in comparison to distance metric learning-based features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 5, pp. 1279–1285, May 2020.

[P13] Shota Horiguchi, S. Amano, M. Ogawa, and K. Aizawa, "Personalized classifier for food image recognition," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2836–2848, Oct. 2018.

International Conference Papers (Peer-Reviewed, First Author)

[P14] Shota Horiguchi, N. Kanda, and K. Nagamatsu, "Multimodal response obligation detection with unsupervised online domain adaptation," in *Proceed-*

ings of The Annual Conference of the International Speech Communication Association (INTERSPEECH), Sep. 2019, pp. 4180–4184.

[P15] ——, "Face-voice matching using cross-modal embeddings," in *Proceedings of ACM International Conference on Multimedia (ACMMM)*, Oct. 2018, pp. 1011–1019.

[P16] Shota Horiguchi, K. Aizawa, and M. Ogawa, "The log-normal distribution of the size of objects in daily meal images and its application to the efficient reduction of object proposals," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 3668–3672.

International Conference Papers (Peer-Reviewed, Co-Author)

[P17] Y. Takashima, Shota Horiguchi, S. Watanabe, P. Garcia, and Y. Kawaguchi, "Updating only encoders prevents catastrophic forgetting of end-to-end ASR models," in *Proceedings of The Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2022, pp. 2218–2222.

[P18] T. Morishita, G. Morio, Shota Horiguchi, H. Ozaki, and N. Nukaga, "Rethinking Fano's inequality in ensemble learning," in *Proceedings of International Conference on Machine Learning (ICML)*, Jul. 2022, pp. 15 976–16 016.

[P19] N. Yamashita, Shota Horiguchi, and T. Homma, "Improving the naturalness of simulated conversations for end-to-end neural diarization," in *Proceedings of The Speaker and Language Recognition Workshop (Odyssey)*, Jun. 2022, pp. 133–140.

[P20] Y. Okamoto, Shota Horiguchi, M. Yamamoto, K. Imoto, and Y. Kawaguchi, "Environmental sound extraction using onomatopoeic words," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2022, pp. 221–225.

[P21] Y. Xue, Shota Horiguchi, Y. Fujita, Y. Takashima, S. Watanabe, P. Garcia, and K. Nagamatsu, "Online streaming end-to-end neural diarization handling overlapping speech and flexible numbers of speakers," in *Proceedings of The Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2021, pp. 3116–3120.

[P22] Y. Takashima, Y. Fujita, Shota Horiguchi, S. Watanabe, P. Garcia, and K. Nagamatsu, "Semi-supervised training with pseudo-labeling for end-to-end neural diarization," in *Proceedings of The Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2021, pp. 3096–3110.

[P23] Y. Xue, Shota Horiguchi, Y. Fujita, S. Watanabe, P. Garcia, and K. Nagamatsu, "Online end-to-end neural diarization with speaker-tracing buffer," in *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, Jan. 2021, pp. 841–848.

[P24] Y. Takashima, Y. Fujita, S. Watanabe, Shota Horiguchi, P. Garcia, and K. Nagamatsu, "End-to-end speaker diarization conditioned on speech activity and overlap detection," in *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, Jan. 2021, pp. 849–856.

[P25] K. Ito, Q. Kong, Shota Horiguchi, T. Sumiyoshi, and K. Nagamatsu, "Anticipating the start of user interaction for service robot in the wild," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Jun. 2020, pp. 9687–9693.

[P26] N. Kanda, Shota Horiguchi, Y. Fujita, Y. Xue, K. Nagamatsu, and S. Watanabe, "Simultaneous speech recognition and speaker diarization for monaural dialogue recordings with target-speaker acoustic models," in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec. 2019, pp. 31–38.

[P27] Y. Fujita, N. Kanda, Shota Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with self-attention," in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec. 2019, pp. 296–303.

[P28] N. Kanda, Shota Horiguchi, R. Takashima, Y. Fujita, K. Nagamatsu, and S. Watanabe, "Auxiliary interference speaker loss for target-speaker speech recognition," in *Proceedings of The Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2019, pp. 236–240.

[P29] N. Kanda, C. Boeddeker, J. Heitkaemper, Y. Fujita, Shota Horiguchi, K. Nagamatsu, and R. Haeb-Umbach, "Guided source separation meets a strong ASR backend: Hitachi/Paderborn University joint investigation for dinner party scenario," in *Proceedings of The Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2019, pp. 1248–1252.

[P30] Y. Fujita, N. Kanda, Shota Horiguchi, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with permutation-free objectives," in *Proceedings of The Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2019, pp. 4300–4304.

[P31] N. Kanda, Y. Fujita, Shota Horiguchi, R. Ikeshita, K. Nagamatsu, and S. Watanabe, "Acoustic modeling for distant multi-talker speech recognition with single- and multi-channel branches," in *Proceedings of IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2019, pp. 6630–6634.

[P32] M. Tamura, Shota Horiguchi, and T. Murakami, "Omnidirectional pedestrian detection by rotation invariant training," in *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2019, pp. 1989–1998.

[P33] S. Amano, Shota Horiguchi, K. Aizawa, K. Maeda, M. Kubota, and M. Ogawa, "Food search based on user feedback to assist image-based food recording systems," in *Proceedings of International Workshop On Multimedia Assisted Dietary Management (MADiMa)*, Oct. 2016, pp. 71–75.

Technical Descriptions

[P34] Shota Horiguchi, N. Yalta, P. Garcia, Y. Takashima, Y. Xue, D. Raj, Z. Huang, Y. Fujita, S. Watanabe, and S. Khudanpur, "The Hitachi-JHU DIHARD III system: Competitive end-to-end neural diarization and x-vector clustering systems combined by DOVER-Lap," in *Proceedings of The Third DIHARD Speech Diarization Challenge (DIHARD III)*, Jan. 2021.

[P35] S. Watanabe, M. Mandel, J. Barker, E. Vincent, A. Arora, X. Chang, S. Khudanpur, V. Manohar, D. Povey, D. Raj, D. Snyder, A. S. Subramanian, J. Trmal, B. B. Yair, C. Boeddeker, Z. Ni, Y. Fujita, Shota Horiguchi, N. Kanda, T. Yoshioka, and N. Ryant, "CHiME-6 Challenge: Tackling multispeaker speech recognition for unsegmented recordings," in *Proceedings of The 6th International Workshop on Speech Processing in Everyday Environments (CHiME-2020)*, May 2020, pp. 1–7.

[P36] T. Morishita, G. Morio, Shota Horiguchi, H. Ozaki, and T. Miyoshi, "Hitachi at SemEval-2020 task 8: Simple but effective modality ensemble for meme emotion recognition," in *Proceedings of The Forteenth Workshop on Semantic Evaluation (SemEval)*, Dec. 2020, pp. 1126—-1134.

[P37] N. Kanda, R. Ikeshita, Shota Horiguchi, Y. Fujita, K. Nagamatsu, X. Wang, V. Manohar, N. E. Yalta Soplin, M. Maciejewski, S.-J. Chen, A. S. Subramanian, R. Li, Z. Wang, J. Naradowsky, L. P. Garcia-Perera, and G. Sell, "The Hitachi/JHU CHiME-5 system: Advances in speech recognition for everyday home environments using multiple microphone arrays," in *Proceedings of The 5th International Workshop on Speech Processing in Everyday Environments (CHiME-2018)*, Sep. 2018, pp. 6–10.