Graph Clustering Approaches for Speaker Diarization of Conversational Speech

A THESIS

SUBMITTED FOR THE DEGREE OF

Doctor of Philosophy

IN THE

Faculty of Engineering

by

Prachi Singh

Department of Electrical Engineering

INDIAN INSTITUTE OF SCIENCE

Bangalore – 560 012

July, 2023

Dedicated to

My Mother & Father:

Shashi & Ram

Acknowledgements

I am profoundly grateful to my advisor, Dr. Sriram Ganapathy, for being a constant source of inspiration and guidance throughout my Ph.D. journey. Besides academic research, I have also learned invaluable life lessons from him, fostering my personal growth and well-being. His unwavering belief in me has enabled me to discover my true potential. He gave me enormous academic and non-academic freedom, allowing me also to explore extra-curricular activities in and outside the campus, which has helped me with my all-around development. His support and encouragement have played a pivotal role in transforming me into a better person.

This work would not have been possible without the help and inputs from my LEAP lab colleagues, Dr. Shreyas Ramoji and Dr. Neeraj Kumar Sharma. I have learned a lot from them regarding work ethics and their passion for research. I would like to extend my thanks to my all other colleagues and friends from the LEAP lab, Debarpan, Shikha, Chandrakant, Soumya, Akshara, and Anurenjan, for various technical and non-technical conversations and support. I would also like to thank my previous lab mates and collaborators, Shobhana, Harsha, Prashant, Venkat, Rajat, Amrit, and Srikanth.

I express my sincere gratitude to the esteemed faculties from my courses, Prof. P. S. Sastry, Prof. Prasanta Kumar Ghosh, Prof. K. V. S. Hari, Prof. Chiranjeeb Bhattacharya, Prof. Chandra Sekhar Seelamantula and all others. Their knowledge and insights motivated me to delve deeper into the subjects and helped shape my research. I also acknowledge the assistance provided by the Department of Electrical Engineering office. I am greatly indebted to Ms. Savitha from IISc Wellness Center for helping me to overcome my anxiety issues during the crucial phase of my Ph.D. I would like to thank the examiners for all the detailed and constructive comments that helped enhance the quality of the thesis.

I would like to thank the British Telecom research speech analytics team - Michael Free, Rohit Singh, and Shakti Srivastava for their valuable inputs in the work. I would also like to thank Neville Ryant, DIHARD challenge organizer, for giving me the opportunity to be a part of the DIHARD challenge baseline. I extend my appreciation to Observe.AI for the internship opportunity and to Dr. Jithendra Veppa and Srikanth Konjeti for mentoring during the internship. Additionally, I would like to thank Dr. Nandkishore Kambhatla, head of Adobe Research, India, and my mentors at Adobe Research, Srikrishna Karnam, and Sumit Shekhar, for the rewarding internship experience.

The journey would not have been the same without the support of my dearest friends. I want to thank my closest friends at IISc - Radhika, Subha, Sabnam, Shilpi, who have been constantly by my side when I needed them. My heartfelt appreciation goes out to my department friends - Supritam, Renuka, Siddhartha, and Abinay, for many discussions during coursework and thereafter. I am thankful for the innumerable memories we shared while at IISc.

I want to thank all my friends, especially Anjali and Vrushali, and my family. My deepest gratitude goes to my parents, Mr. Ram Singh and Mrs. Shashi Singh, for their constant and unconditional support, encouragement, and love. I am thankful to my uncle, Sanjeev Kumar, for motivating and guiding me to pursue a Ph.D. and secure admission at IISc, the best research institute in the country. I am also thankful to all my school and college teachers for their guidance and for making me capable of achieving this most memorable and significant milestone of my life.

Last but certainly not least, I extend my heartfelt gratitude to the Indian Institute of Science, Bangalore, for being a constant pillar of support, protection, and transformation. The institute has embraced me with everything I could have ever desired and dreamed of during college, including its captivatingly beautiful and tranquil campus. The long walks and cycling trails amidst lush greenery have always provided solace and serenity, bringing peace to my life. I shall forever cherish the precious memories of this remarkable place that has played a significant role in shaping me into a better version of myself.

Abstract

In this era of advanced machine intelligence, real-world speech applications need to be equipped to deal with conversations involving multiple speakers. An essential first step in speech information extraction from conversational speech is the task of finding "who spoke when", also referred to as speaker diarization. The focus of this doctoral thesis is to describe our efforts in investigating graph clustering techniques for this problem. While graph models have been used in several other domains, its application to temporal segmentation of speech is the first of its kind.

The thesis is divided into three main parts. In the first part of the thesis, we describe a novel proposal on self-supervised learning to perform joint representation learning and clustering, called self-supervised clustering (SSC) for diarization. On the learned representations, we explore path integral clustering (PIC), a graph-based clustering algorithm. The PIC is an agglomerative graph clustering method that performs clustering based on the edge connections of a node, called path integral. The proposed SSC with path integral clustering (SSC-PIC) is shown to achieve state-of-the-art performance for benchmark datasets.

The second part of the thesis is an extension of SSC-PIC to incorporate metric learning. We design a neural version of the probabilistic linear discriminant analysis (PLDA) approach with learnable parameters to compute a log-likelihood score between embeddings from two segments of the recording. We propose a joint self-supervised representation learning and metric learning approach called SelfSup-PLDA-PIC.

In the third part of the thesis, we introduce an end-to-end supervised graph clustering approach. We develop a supervised learning setup using labeled conversational data for training this model. In this setting, we propose a supervised clustering approach called

Supervised HierArchical gRaph Clustering (SHARC) for speaker diarization. This approach uses Graph Neural Networks (GNN) to capture the similarity between the speaker embeddings and performs hierarchical clustering. An extension of this work is the joint training of the speaker embedding extractor along with the GNN module, referred to as end-to-end SHARC (E-SHARC). To incorporate overlapped speech detection, the E-SHARC model is extended for diarization of overlapped speech recordings.

In summary, this thesis introduces innovative self-supervised and supervised methods that utilize hierarchical graph clustering to improve diarization performance. These approaches have demonstrated state-of-the-art results on different benchmark datasets. Despite their success, we also acknowledge the limitations of these methods, which open up opportunities for further advancements in diarization, especially in complex and challenging environments.

Publications

Peer-reviewed Journal Papers

1. **P. Singh** and S. Ganapathy, “Self-Supervised Representation Learning With Path Integral Clustering for Speaker Diarization", IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 1639-1649, 2021, doi: 10.1109/TASLP.2021.3075100.

2. **P. Singh** and S. Ganapathy, “Overlap-aware End-to-End Supervised Hierarchical Graph Clustering for Speaker Diarization", IEEE/ACM Transactions on Audio, Speech, and Language Processing (under review).

Peer-reviewed Conference Papers

1. **P. Singh**, A. Kaul and S. Ganapathy, “Supervised Hierarchical Clustering using Graph Neural Networks for Speaker Diarization", *Proc. ICASSP, 2023*, pp. 1-5, doi: 10.1109/ICASSP49357.2023.10095372.

2. **P. Singh** and S. Ganapathy, “Self-Supervised Metric Learning With Graph Clustering For Speaker Diarization", *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2021*, pp. 90-97, doi: 10.1109/ASRU51503.2021.9688271.

3. **P. Singh**, R. Varma, V. Krishnamohan, S. R. Chetupalli, and S. Ganapathy, “LEAP Submission for the Third DIHARD Diarization Challenge", *Proc. Interspeech, 2021*, pp. 3545-3549, doi: 10.21437/Interspeech.2021-728.

4. **P. Singh** and S. Ganapathy, “Deep Self-Supervised Hierarchical Clustering for Speaker Diarization", *Proc. Interspeech, 2020*, pp. 294-298, doi: 10.21437/Interspeech.2020-2297.

5. **P. Singh**, Harsha Vardhan MA, S. Ganapathy, A. Kanagasundaram, “LEAP Diarization System for the Second DIHARD Challenge", *Proc. Interspeech, 2019*, pp. 983-987, doi: 10.21437/Interspeech.2019-2716.

Contents

List of Figures

List of Tables

Notations

- $\mathbf{X}_r = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_{N_r}\}, \boldsymbol{x}_i \in \mathbb{R}^D \; \forall i \in \{1, ..., N_r\}$ denote the sequence of N_r segment embeddings for the recording r.
- $m \in \{1, 2, ..., M\}$ denote the level (iteration) of hierarchical clustering.
- $\mathcal{A}$ denote the affinity measure between two clusters.
- $\mathbf{S} \in \mathbb{R}^{N_r X N_r}$ denote pairwise similarity score matrix obtained using pair-wise similarity $s(i, j)$ between representations at time index i and j.
- $\mathbf{A} \in \mathcal{R}^{NXN}$ is the Adjacency matrix of graph $G = (V, E)$ where V and E are the set of vertices/nodes and edges, respectively. N is the number of nodes.
- $G_r^m = (V_r^m, E_r^m)$ is a graph of recording r at level m. $V_r^m = \{v_1, ..., v_{N_r^m}\}$ and $E_r^m = \{(v_1, v_i), ..., (v_{N_r^m}, v_j)\}$ are the set of nodes and set of edges of the graph, respectively.
- $\mathbf{z}^m = \{z_1^m, ..., z_{N_r}^m\}$ denote the cluster labels for recording r, where each element takes a discrete cluster index value.
- $\mathbb{C}_r^m = \{\mathbb{C}_1^m, ..., \mathbb{C}_{n_c}^m\}$ denote the set of clusters for level m.

Chapter 1

Introduction

Speech is a fundamental and vital means of communication, carrying information about content, speaker identity, emotions, and language. With the rapid advancement of machine intelligence, the development of robust speech technologies holds the potential to significantly enhance our lives, promoting better health, safety, and comfort. For instance, machines capable of transcribing and analyzing patient-doctor conversations in the healthcare sector can reduce the burden on medical staff by minimizing manual efforts. Moreover, speech is pivotal in human-computer interaction as a widely used input method to control devices.

However, numerous real-world speech applications involve multi-talker conversational scenarios, making it challenging for machines to understand and process such interactions. It is seamless for humans to understand the multi-speaker conversation, but replicating this fluency in machines is demanding. Consequently, addressing the obstacles presented by multi-talker conversational speech recordings and devising practical solutions becomes an imperative research area.

The conversational audio recordings entail speakers with varying intonations, swiftly switching between speakers (speaker turns), and frequent speech overlaps, posing difficulties for machine processing, transcription, and interpretation of the content. Overcoming these challenges through innovative research and technology advancements is crucial to unlocking the full potential of speech-based applications in diverse domains.

For applications such as rich speech transcription[1] of conversational audio and speaker detection in a multi-speaker recording [2], partitioning the input audio into segments based on speaker sources serves as an essential pre-processing step. This is referred to as speaker diarization. The focus of this thesis involves understanding some of the challenges faced by

current speaker diarization systems and proposing solutions in order to improve performance.

1.1 Motivation and Applications

In recent years, there has been a noteworthy increase in speaker diarization applications [3]. Some of the important applications are as follows:

- **Speech transcription and understanding:** One of the most sought-after application is meeting transcription, which requires automatically generating speaker-attributed transcripts during real-life meetings based on their audio recordings. Although this task was introduced by NIST in the RT evaluation series back in 2003 [4, 1, 5], the initial systems had very poor performance, and consequently, commercialization of the technology was not possible. However, recent advances in speech recognition [6, 7], far-field speech processing [8], have greatly improved the speaker-attributed transcription accuracy. Separating and identifying individual speakers in an audio recording[9, 10] helps produce more accurate and intelligible transcriptions.

- **Natural language processing:** Speaker diarization also provides a foundation for various natural language processing (NLP) tasks that require speaker-specific information. It enables the development of speaker-specific language, acoustic, and other speaker-dependent NLP models. Such models enhance the accuracy and effectiveness of speech recognition, sentiment analysis, speaker identification, and voice biometrics.

- **Information retrieval and indexing:** Speaker diarization facilitates efficient searching and indexing of audio content [11]. Identifying speakers and segmenting speech by speaker makes it easier to locate specific segments of interest within an audio recording. This is valuable in multimedia indexing, content-based retrieval, and information extraction from spoken documents.

- **Speaker-specific analysis:** Speaker diarization enables the analysis of individual speakers in various domains. Researchers can examine linguistic patterns, accent variations, speaker characteristics, and sociolinguistic aspects within a conversation

or audio dataset. This information can be leveraged for linguistic studies, sociological research, forensic investigations, and speaker-dependent analysis in fields such as psychology or voice pathology.

- **Behavioral Modeling:** Analyses of spoken conversational interactions relates to behavioral signal processing (BSP) [12, 13], which refers to the technology and algorithms for modeling and understanding human communicative, affective and social behaviors.
- **Improved user experience:** Speaker diarization contributes to enhancing user experiences in applications involving audio or speech interactions. It enables personalized responses from voice assistants, call center analytics, voice-controlled systems, and voice-enabled applications [14, 15]. Identifying speakers accurately can help personalize user experiences, adapt responses based on speaker preferences, and enable more natural and context-aware interactions.

Recent works in speaker diarization have focused on developing deep learning-based models that can effectively identify speaker-discriminative features in the audio. These models have shown promising results on a variety of datasets. But the performance of the speaker diarization system varies based on the application domain. Some of the challenges of speaker diarization have been highlighted in the recent DIHARD evaluations [16, 17], which evaluate diarization performance on various domains. These challenges include background noise, variable number of speakers, variable recording duration, fast transition between speakers, and overlapping speech. Therefore, there is still a need and scope for more research for building robust diarization systems.

1.2 Current State-of-the-art Diarization System

The early approaches to speaker diarization involved steps of speaker change detection and gender classification [18]. In recent years, speaker diarization involves multiple steps to automatically identify and group audio segments spoken by the same speaker in an audio recording. The main steps in a typical speaker diarization pipeline, shown in Figure 1.1, include the following:

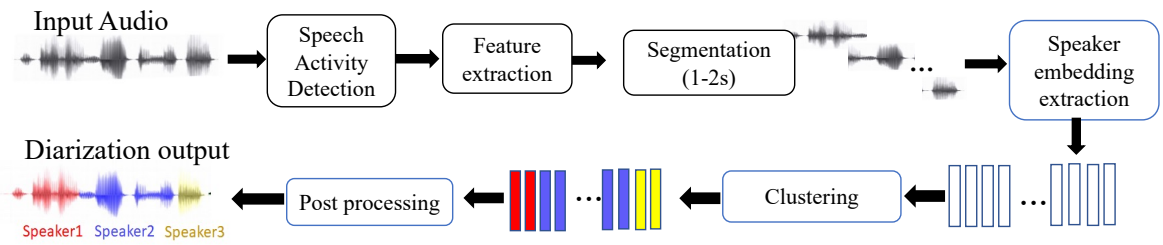

Figure 1.1: Block diagram of the multi-step speaker diarization approach.

1. **Speech activity detection (SAD):** SAD, also known as voice activity detection (VAD), is a pre-processing step that involves classifying audio regions into speech and non-speech, such as background noise. SAD plays an important role in extracting speech regions used for further processing in speaker diarization. There are two main stages in the SAD system. First is front-end acoustic feature extraction like zero crossing rate [19], pitch [20] or mel-frequency cepstral coefficients (MFCCs). Second, speech and non-speech classification using statistical [21, 22] or deep neural models [23, 24, 21]. SAD can largely affect the overall performance of the diarization system, adding false positive speech or miss speech segments [25]. A common practice in speaker diarization tasks is to report Diarization Error Rate (DER) with an "oracle SAD" setup, indicating that the system output uses SAD output identical to the ground truth.

2. **Feature extraction:** Speech is a quasi-stationary signal, i.e., it attains stationarity over a short period. To capture such transient information, speech is processed in one short-time window called frame after another. The window length (20-40 milliseconds) and shift (10-20 milliseconds) have fixed duration. A sequence of such frame-level feature vectors can be further modeled using various machine learning models or neural networks. Acoustic features, such as Mel frequency cepstral coefficients (MFCCs), linear predictive coding (LPC), spectrograms, and mel filterbanks, are extracted from each speech segment obtained from SAD. These features capture the spectral characteristics of the audio and serve as input for subsequent processing steps.

3. **Audio segmentation:** The audio recording is divided into smaller segments, often

called chunks, typically spanning a few seconds (1-2s). This segmentation step helps to extract units of audio for further processing.

4. **Speaker embedding extraction:** Speaker embeddings, which are high-dimensional representations, are derived from the corresponding short audio segments. For this extraction, deep neural networks like convolutional neural networks (CNNs) [26, 27] or recurrent neural networks (RNNs) [28] are commonly employed. The primary purpose of these speaker embeddings is to capture the distinguishing characteristics present in each speaker's voice.

5. **Clustering:** The speaker embeddings are then clustered into groups based on similarity. Clustering aims to group embeddings that belong to the same speaker while keeping embeddings from different speakers separate. The agglomerative hierarchical clustering (AHC) [29] has been the most promising approach for speaker diarization [30]. AHC aims to iteratively merge clusters until a stopping criterion is met to achieve a one-to-one correspondence between the clusters and speakers in the given recording [31].

6. **Post-processing:** Post-processing techniques are applied to refine the speaker diarization results. This may include techniques such as re-segmentation to refine the speaker change boundaries and overlap detection to identify overlapping speech regions for assigning multiple speakers.

1.3 Graphs and their Applications in Speech Segmentation

Graphs are powerful mathematical structures that represent complex relationships and interactions among entities. In a graph, entities are represented as nodes, and the relationships between entities are represented as edges connecting the nodes. Graph theory, the mathematical study of graphs, has found diverse applications across various fields, including computer science, social sciences, biology, transportation, and more [32, 33]. Graph algorithms are used to analyze the connectivity patterns, identify influential individuals, and detect communities within the network, e.g., social networks [34, 33]. Graphs represent complex biological interactions, such as protein-protein interaction networks or gene regulatory networks [35, 36].

Graph analysis aids in understanding biological processes and identifying potential drug targets. In Natural Language Processing, graphs are utilized to model syntactic and semantic relationships among words or sentences [37]. While graph models have been used in various domains, their applications explored in this thesis for temporal speech segmentation are the firsts of its kind. Graphs are an exciting paradigm for diarization problems due to several compelling reasons:

1. **Modeling Interactions:** Graphs allow us to naturally represent speaker interactions in a conversation. Nodes in the graph can represent individual speakers, and edges can capture turn-taking or overlapping speech between speakers during conversations. This representation enables a comprehensive view of the speaker relationships, aiding in identifying speaker clusters and their temporal dynamics.

2. **Flexibility in Representation:** Graph-based methods offer flexibility in incorporating various features and information. Speaker embeddings, acoustic features, and temporal cues can be efficiently integrated into the graph structure. This flexibility allows for a richer and more informative representation of speaker interactions, improving diarization performance.

3. **Utilizing Global and Local Information:** Graph helps leverage local and global information within the data. Local information involves the relationships between neighboring speech segments, while global information considers the entire conversation context. Graph-based diarization can make better-informed decisions by considering both scales, especially in scenarios with complex speaker interactions.

1.4 Outline of Contributions

Clustering is a crucial step in speaker diarization as it enables accurate speaker segmentation, turn-taking detection, speaker model creation, speaker adaptation, and evaluation. It contributes to the overall effectiveness and reliability of the diarization system, making it an important component in various applications involving speaker analysis. As shown in Figure

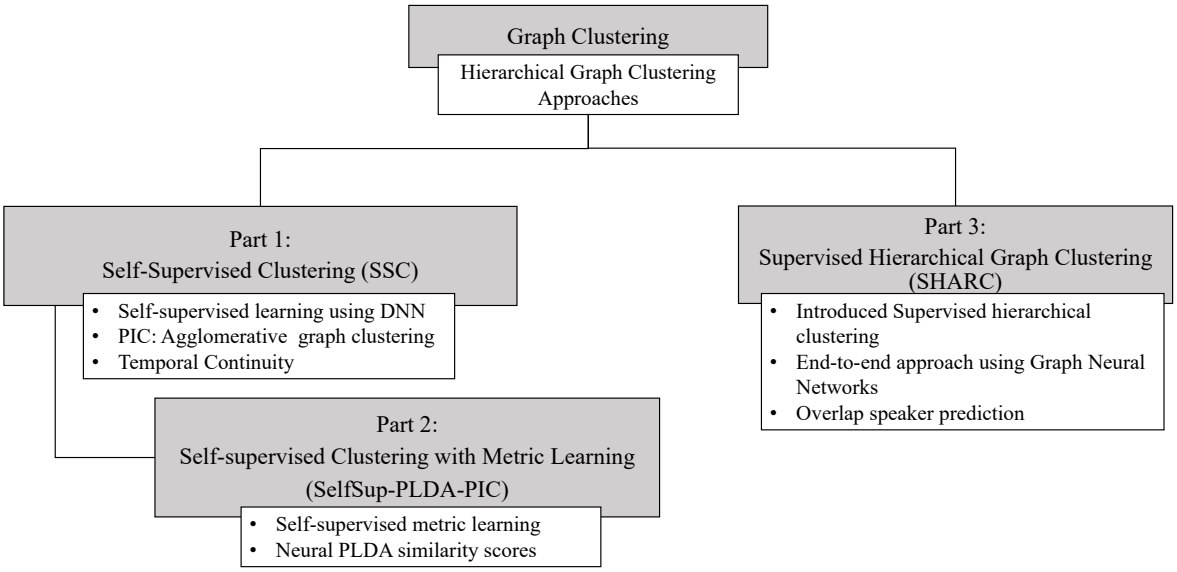

Figure 1.2: A summary of thesis contributions.

1.1, speaker embeddings are the input for the clustering stage. Therefore, better embeddings can lead to better clustering performance. Hence, our work is primarily concentrated on clustering and representation learning. The thesis centers around graph clustering, motivated by the manifold advantages of representing conversational speech as graphs (section 1.3). Consequently, our main objective is to enhance diarization performance through the introduction of robust self-supervised and supervised learning techniques tailored for graph clustering algorithms. In this thesis, we describe our efforts to investigate the shortcomings of the existing approaches and give a detailed account of our proposed self-supervised and supervised methodologies in developing robust speaker diarization systems.

The thesis is divided into three main parts, as shown in Figure 1.2. In the first part of this thesis research, we explore the state-of-the-art diarization model [10] and identify the key limitations. The conventional approach to speaker diarization is multi-step. The embedding extraction and unsupervised clustering are optimized separately. We provide a solution to the shortcoming by introducing a self-supervised learning approach to perform joint representation learning and clustering called self-supervised clustering (SSC). We iteratively perform representation learning using pseudo target labels generated from clustering and further cluster the learned representations. SSC helps to increase inter-cluster distance and

decrease intra-cluster distance, thus increasing speaker separability. In addition to the AHC, we explore other graph-based algorithms called path integral clustering (PIC). The PIC is an agglomerative graph clustering algorithm that first encodes the embeddings in a graph and then performs clustering based on the number of edge connections of a node called path integral. This algorithm is robust to noisy embeddings and generates reliable clustering output for SSC. Therefore, the proposed self-supervised clustering with path integral clustering (SSC-PIC) achieves state-of-the-art performance for benchmark datasets for speaker diarization.

The second part of the thesis is an extension of SSC-PIC to incorporate metric learning. The traditional approach to diarization uses a similarity matrix to perform clustering, which is obtained from Probabilistic Linear Discriminant Analysis (PLDA) model. It computes a similarity score or log-likelihood score between two embeddings of the recording, It yields better performance than cosine similarity with unsupervised clustering. The SSC-PIC uses cosine similarity, a non-parametric similarity measure, to perform clustering. This limits the learning capabilities of the model. Therefore, we propose a self-supervised representation learning and metric learning approach inspired by PLDA using PIC called Selfsup-PLDA-PIC. The model parameters are initialized with the PLDA parameters and then learned during the training. This increases the learning capability of the model, which further improves speaker separability and performance over the SSC-PIC approach.

In the third part of the thesis, we introduce an end-to-end supervised graph clustering approach. Although the self-supervised clustering performs well for conversations with up to seven speakers, it is found to underperform for recordings containing more than seven speakers. The initial clustering performance is found to be less reliable for such scenarios. Therefore, supervised learning using labeled conversational data will mitigate this issue. However, the existing end-to-end models are trained with permutation invariance loss in which the amount of computations increases with the higher number of speakers [38]. Therefore, to overcome the limitations of self-supervised clustering without requiring considerable computation, we propose a supervised clustering approach called Supervised HierArchical gRaph Clustering (SHARC) for speaker diarization. This approach uses Graph Neural Networks (GNN), which

jointly compute similarities between speaker embeddings and perform hierarchical clustering. The model is trained using input graphs created with speaker embeddings as the nodes for each recording with clustering loss. An extension of this work is to train the speaker embedding extractor along with the GNN module, referred to as End-to-End SHARC (E-SHARC). The E-SHARC model is further utilized to perform overlap speaker prediction.

1.5 Thesis Organization

In chapter 2, we discuss the related work in the literature, evaluation metrics for performance comparison and various datasets used in our experiments. In chapter 3, we discuss our efforts in developing a self-supervised clustering (SSC) method to perform speaker diarization. We discuss the experiments and results on two benchmark datasets. In chapter 4, we show the extension of the SSC to incorporate metric learning called SelfSup-PLDA-PIC. This improves the performance over the baseline methods. In chapter 5, we give a detailed account of our proposed supervised hierarchical clustering algorithm using graph neural networks. We show significant performance improvement across several test datasets. Finally, in chapter 6, we summarize the thesis and identify some critical research directions that can be explored in the future.

Chapter 2

Background Study

In this chapter, we will discuss some related work in the literature, the evaluation metrics, and the datasets used in the experiments. This chapter has been divided into three sections;

1. The first section describes the relevant related work, which includes clustering, metric learning, and end-to-end diarization approaches.

2. The second section describes the evaluation metrics used for comparing performance across different approaches.

3. The third and last section discusses the diarization datasets used in this thesis to train and evaluate the proposed methods.

2.1 Related Work

Early works on diarization were motivated to improve automatic speech recognition on air traffic control dialogues and broadcast news recordings by segmenting the audio based on homogeneous speaker segments or detecting changes in speakers, also known as speaker turns [39, 3]. Speaker change detection required comparing adjacent regions in the audio to identify the similarities. This led to the development of distance metrics like generalized likelihood ratio (GLR) [40], Bayesian information criterion (BIC) and ΔBIC metric [41], the Kullback–Leibler (KL) divergence [42] which find the distances between two adjacent segments to identify speaker changes. These same metrics were also utilized to perform clustering, identifying and grouping same-speaker segments that can be localized anywhere in the audio stream.

Gaussian mixture models (GMMs) [43] or hidden Markov models (HMMs) [44] were also introduced to perform unsupervised clustering using short-term spectral feature vectors. This section first describes various speaker embeddings and similarity measures used in the popular diarization systems and then discusses the approaches for clustering in speaker diarization.

2.1.1 Speaker embeddings

The speaker embeddings or representations are the feature vectors that are obtained from a model trained with speaker-labeled utterances. The model projects the variable-length utterances into fixed-length speaker-characterizing vectors, also called embeddings. These embeddings can then be utilized to perform clustering. This section discusses various models used for extracting speaker representations.

1. **i-vector:** In the last decade, the field of speaker diarization has focused on generating robust speaker embeddings from short segments (1-2s) from the recording and performing clustering. The embedding extraction has closely followed the developments in the speaker verification field. Joint factor analysis was initially proposed to separate channel and speaker information from the audio region [45]. This helped mitigate intersession variability when the same speaker is present in different recordings. First a Gaussian Mixture Model-based Universal Background Model (GMM-UBM), typically with 512 to 2048 mixtures, was trained on acoustic features (e.g., MFCCs) with mixture weights, mean, and covariance matrix parameters. Then, this GMM is adapted for speakers. The JFA factors the mean supervector M obtained from concatenating the mean of the mixtures of speaker-adapted GMM into UBM-GMM mean supervector, speaker information, channel information, and noise. Later, the speaker and channel factors are merged to form an i-vector to capture the shift in the mean from UBM-GMM to speaker-adapted GMM. The i-vector, referred to as the speaker identity vector, is modeled as follows:

$$M = m + Ty \tag{2.1}$$

where M is the speaker adapted GMM mean supervector, m is the UBM-GMM super-

vector , T is the total variability matrix. y is a latent variable with standard normal prior, weighing the column of T to capture the mean shift. The point estimate of the posterior probability of y given the acoustic features $X = \{x_1, ..., x_N\}$, $p(y|X)$, is called i-vector. The i-vector embedding extraction [46] using factor analysis showed promising improvements for speaker diarization [47].

2. **d-vector:** Initially introduced for text-dependent speaker verification, d-vectors [48] are the earliest neural network-based models trained to classify speakers at the frame level. The average of the speaker features from the last hidden layer of the DNN is called the d-vector. Later the model is adopted for text-independent speaker recognition and diarization [49].

3. **x-vector:** With the advancements in deep learning, the i-vector embeddings were successfully replaced with deep neural network (DNN) based embedding extractors. These embeddings referred to as x-vectors [26], are trained using time-delay neural networks (TDNNs) and improve the clustering performance [50]. A time delay neural network (TDNN) is specifically designed to process sequential or time-series data. It extends the traditional feed-forward neural network architecture to perform 1-D convolution over a sliding window using stride (shift) and dilation to capture the temporal context. Over the years, different variants of the x-vector model have been introduced. Some of them are discussed as follows:

   TDNN: The TDNN x-vector model introduced by Snyder et al. [26] consists of 5 layers of time-delay neural network and two layers of feed-forward architecture operating at frame-level followed by a pooling layer which generates mean and standard deviation statistics at the segment level. The segment-level statistics are passed through two feed-forward layers to the target layer. The model is trained for speaker classification using the asynchronous stochastic gradient descent (ASGD) algorithm. The first affine layer (before the non-linearity) after the pooling layer is used as the x-vector feature.

   ETDNN: To further improve the performance of x-vectors, a deeper version of TDNN

called extended time delay neural network (ETDNN) was introduced. The 13-layer ETDNN model follows the architecture described in [51, 52]. It has 4 TDNN layers which alternate with 4 fully connected layers of size 1024D. This is followed by 2 feed-forward layers containing {1024, 2000} units. The segment pooling layer is 4000D, containing mean and standard deviation computed at the segment level for the 2000D layer. From the segment-level features, the 512 dimensional output of the affine component from the 11^{th} layer is taken as the x-vector embedding.

FTDNN: The architecture of the FTDNN model is similar to that of ETDNN, with factorized TDNN layers [53] in place of the TDNN layers. The model has fewer parameters due to the factorization of each layer's weight matrix M as a product of two low-rank matrices. The model enforces the second factor in $M = AB$ to be semi-orthogonal ($MM^T = I$). The factorized TDNN layers are each size 1024 and perform a sequence of convolutions that maps the input to 256 dimensions and back to the hidden layer size 1024. The network has a longer temporal context of ±16 frames. 512D embeddings are extracted from the 12th affine layer of the model as the x-vector embedding.

ResNet: The Residual Network (ResNet) [54] is a type of deep convolutional neural network (CNN) architecture that introduced the concept of residual connections. It mitigated the vanishing gradient problem, where gradients become exponentially small during backpropagation by introducing skip or shortcut connections that bypass one or more layers. In this thesis, we have experimented with ResNet101 model, also used in [27]. The first layer is a 2D convolutional layer. This is followed by four residual blocks. Each residual block consists of 3 residual convolutions. The output of the residual blocks is fed to the statistics pooling layer (from each of the 8 heads). The dense network layer following the pooling layer extracts the ResNet x-vectors. The training data and the cost function used to train the ResNet model are similar to the ETDNN framework.

ECAPA-TDNN: The ECAPA-TDNN model comprises ResNet blocks followed by the squeeze and excitation (SE) block. Finally, it adds an attentive statistics pooling layer followed by a fully connected layer to obtain the x-vectors [55]. The SE block expands

the temporal context of the frame layer by rescaling the channels according to the global properties of the recording. It also performs aggregate and propagate features of different hierarchical levels. Dawalatabad et al. [56] illustrated the improved speaker diarization performance with ECAPA-TDNN x-vectors.

2.1.2 Similarity measure

Metric-based approaches were most commonly used for the similarity measurement between speech segments from the late 1990s to the early 2000s for speaker diarization systems. Nowadays, a metric function generates a similarity matrix containing pairwise similarity scores between speaker embeddings of the same recording for clustering. Initially, cosine similarity was used as a metric function [57]. The model-based similarity scoring using probabilistic linear discriminant analysis (PLDA) [58, 59] was successful in speaker verification [60]. Therefore, it has also been incorporated for speaker diarization. The details of the PLDA model formulation are given below.

- **PLDA model:** The PLDA is a generative model that factorizes the input into the speaker and channel factors. The simplified and widely used model is a generative linear-Gaussian model (GPLDA) [58], where x-vector $\boldsymbol{x} \in \mathbb{R}^D$ represents segment embedding. The vector $\boldsymbol{y} \in \mathbb{R}^D$ represents the speaker factor. The distribution of $\boldsymbol{x}$ given the speaker factor $\boldsymbol{y}$ is assumed to be a Gaussian distribution,

$$p(\boldsymbol{x}|\boldsymbol{y}) = N(\boldsymbol{x}; \boldsymbol{y}, \boldsymbol{\Phi_w}) \tag{2.2}$$

where $\boldsymbol{\Phi_w} \in \mathbb{R}^{DXD}$ is the within class covariance matrix. The latent vector $\boldsymbol{y}$ is assumed to be distributed according to the prior distribution:

$$p(\boldsymbol{y}) = N(\boldsymbol{y}; \boldsymbol{m}, \boldsymbol{\Phi_b}) \tag{2.3}$$

where $\boldsymbol{m} \in \mathbb{R}^D$ and $\boldsymbol{\Phi_b} \in \mathbb{R}^{DXD}$ are global mean and between-class covariance matrix, respectively. Further, $\boldsymbol{\Phi_w}$ and $\boldsymbol{\Phi_b}$ can be simultaneously diagonalized using diagonalizing

transform $\boldsymbol{V} \in \mathbb{R}^{DXD}$:

$$\boldsymbol{V}\boldsymbol{\Phi}_{\boldsymbol{w}}\boldsymbol{V}^T = \boldsymbol{I}, \quad \boldsymbol{V}\boldsymbol{\Phi}_{\boldsymbol{b}}\boldsymbol{V}^T = \boldsymbol{\Psi} \tag{2.4}$$

where $\boldsymbol{\Psi}$ is a diagonal covariance matrix. Given this model, the generative model can be expressed in terms of $\boldsymbol{u}, \boldsymbol{v}$, where $\boldsymbol{u}, \boldsymbol{v}$ denote the latent vectors representing speaker variable and the speaker embedding in the projected space, respectively. If $\boldsymbol{A} = \boldsymbol{V}^{-1}$, the generative model is expressed as:

$$\begin{aligned} p(\boldsymbol{v}) &= N(.|\boldsymbol{0}, \boldsymbol{\Psi}) \\ p(\boldsymbol{u}|\boldsymbol{v}) &= N(.|\boldsymbol{v}, \boldsymbol{I}), \\ \boldsymbol{x} &= \boldsymbol{m} + \boldsymbol{A}\boldsymbol{u} \end{aligned} \tag{2.5}$$

During the training, speaker embeddings and their corresponding speaker labels are used to learn parameters $\{\boldsymbol{m}, \boldsymbol{A}, \Psi\}$ using the expectation maximization (EM) algorithm.

- **PLDA similarity score:** A PLDA similarity score is computed between a pair of speaker embeddings $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ during inference. The PLDA score is the log-likelihood ratio $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$ computed using two hypotheses. Hypothesis $\mathcal{H}_s$ assumes that the two embeddings belong to the same speaker, and hypothesis $\mathcal{H}_d$ assumes they belong to different speakers. The PLDA similarity score or loglikelihood ratio $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is given by,

$$\begin{aligned} s(\boldsymbol{x}_i, \boldsymbol{x}_j) &= logp(\boldsymbol{x}_i, \boldsymbol{x}_j|\mathcal{H}_s) - logp(\boldsymbol{x}_i, \boldsymbol{x}_j|\mathcal{H}_d) \\ [\boldsymbol{S}]_{ij} &= s(\boldsymbol{x}_i, \boldsymbol{x}_j) \; \forall \; i, j \in \{1, .., N_r\} \end{aligned} \tag{2.6}$$

The higher value of s indicates that the two embeddings are from the same speaker. $\boldsymbol{S}$ is the similarity matrix and N_r is the number of segments in a recording.

2.1.3 Clustering approaches

Clustering is an essential step in the process of speaker diarization. By grouping similar acoustic features, clustering algorithms can determine boundaries between different speakers, allowing for the accurate extraction of individual speech segments. Clustering also enables the detection of speaker transitions and turn-taking points in a conversation. The audio is segmented into uniform short segments (1-2s). By clustering speech segments based on speakers, we can estimate speaker-specific models that can later be used for tasks like speaker identification or verification. The clustering algorithm, which does not use any training and finds the hidden patterns and insights from the given data, is called unsupervised clustering. Following are the details about various unsupervised clustering algorithms used in the literature for performing speaker diarization.

1. **The mean-shift algorithm :** The mean-shift algorithm [61, 62, 63] is an iterative non-parametric approach used to estimate the probability density function of a random variable (Fukunaga and Hostetler, 1975). It draws inspiration from the Parzen window method for non-parametric density estimation and offers several advantages: it doesn't require prior knowledge about the number of clusters. It doesn't make assumptions about the cluster shapes. In this algorithm, dense regions in the feature space correspond to local maxima or modes of the density function. The algorithm performs gradient ascent on the estimated local density for each data point until convergence is achieved. The stationary points obtained through this process represent the modes of the density function, and data points associated with the same fixed point are assigned to the same cluster.

2. **k-means clustering:** This method aims to partition N data points into k clusters [64]. First, it randomly selects "k" data points as the mean or centroid of "k" clusters. Then it iteratively assigns the data points to the nearest centroid (center) of a cluster and updates the centroids based on the associated data points. The process repeats till the centroids have stabilized or the defined number of iterations has been achieved.

3. **Hierarchical Clustering:** In this class of algorithm [65], the clusters are visually represented in a hierarchical tree called a dendrogram. There are two variations of hierarchical clustering - agglomerative and divisive. Agglomerative clustering is a bottom-up approach. It starts with the assumption that each data point or feature vector is a separate cluster and then merges them until one cluster remains or the required number of clusters is obtained. Divisive clustering is a top-down approach. It considers all data points belonging to one cluster, which are split repeatedly in a hierarchical manner till the required number of clusters is obtained. We will discuss the agglomerative approach in more detail.

   Agglomerative Hierarchical Clustering (AHC): AHC [66] is widely employed clustering method for speaker diariation task [47, 10]. AHC involves iteratively merging the existing clusters until the stopping criterion is achieved. Initially, each feature vector is considered as one cluster. The similarity score $s(i, j)$ between speaker embeddings x_i and x_j is computed using metric functions like cosine, PLDA [58, 67]. The similarity scores calculate affinity $\mathcal{A}$ between the clusters at each level of the hierarchy. In conventional AHC, the affinity $\mathcal{A}\ (\mathcal{C}_a, \mathcal{C}_b)$ between the two clusters $\mathcal{C}_a$ and $\mathcal{C}_b$ can be defined using different linkage choices [68] as follows:

   - Single-linkage (Nearest-neighbor): The highest similarity score between two elements (one element from each cluster). $max_{i,j}\{s(i, j) : \boldsymbol{x}_i \in \mathcal{C}_a, \boldsymbol{x}_j \in \mathcal{C}_b\}$.
   - Complete-linkage (Farthest-neighbor): The lowest similarity score between two elements (one element from each cluster). $min_{i,j}\{s(i, j) : \boldsymbol{x}_i \in \mathcal{C}_a, \boldsymbol{x}_j \in \mathcal{C}_b\}$.
   - Average-linkage: The average similarity score of all pairs of elements (one from each cluster). $mean_{i,j}\{s(i, j) : \boldsymbol{x}_i \in \mathcal{C}_a, \boldsymbol{x}_j \in \mathcal{C}_b\}$.

   The popular average linkage has been used in our baselines and proposed works. Based on the highest affinity between the clusters, a pair of clusters are merged at each time step given as: $\{\mathcal{C}_a, \mathcal{C}_b\} = \text{argmax}_{\mathcal{C}_i, \mathcal{C}_j} \mathcal{A}(\mathcal{C}_i, \mathcal{C}_j)$. The stopping criterion is based on the required number of clusters if known, or with the application of a threshold on

the similarity scores. This threshold is obtained from a development set for the best diarization performance.

4. **Graph Clustering:**

A graph G can be well described by the set of vertices V and edges E it contains, as $G = (V, E)$. The vertices are often called nodes. Adjacency matrix ($\mathbf{A} \in \mathcal{R}^{NXN}$, N - number of nodes) captures connections between nodes,

$\mathbf{A}_{ij} = 1$, if Node i is connected to j by an edge.

$\mathbf{A}_{ij} = 0$, if Node i and j are not connected.

$\mathbf{A}$ with real weights to the edges is called a weighted adjacency matrix. It can be obtained using similarity scores between data points.

The task of grouping the vertices of the graph into clusters in such a way that there should be many edges within each cluster and relatively few between the clusters is called *graph clustering*. We will discuss one popular graph clustering algorithm called spectral clustering.

Spectral Clustering: Spectral clustering [69] is based on the spectral graph theory, which allows it to capture complex relationships and structures in the data. Several advantages of spectral clustering make it popular for speaker diarization [28]. For example, it can handle non-linearly separable data by leveraging the spectral embedding of the data. It is not limited by assumptions of spherical or isotropic cluster shapes like k-means clustering, making it more flexible and suitable for datasets with irregular or overlapping clusters. Spectral clustering is less sensitive to noise and outliers compared to AHC. It considers the global structure of the data using the similarity matrix, which helps identify and separate noisy or outlier data points into their own clusters or treat them as background noise.

Spectral clustering involves identifying clusters of nodes in a network based on the edges that connect them using eigenvalues and eigenvectors of a modified similarity matrix. To perform spectral clustering, first, the normalized graph Laplacian matrix is

calculated using the adjacency matrix **A** as follows :

$$\boldsymbol{L} = \mathbb{I} - \boldsymbol{D}^{-1/2}\mathbf{A}\boldsymbol{D}^{-1/2} \tag{2.7}$$

where $\boldsymbol{D}$ is the diagonal matrix containing the sum of each row from the similarity matrix as the diagonal entry. The eigenvectors corresponding to the k-smallest eigenvalues of $\boldsymbol{L}$ are stacked to perform k-means clustering. The k value determines the number of clusters obtained using the maximum eigengap between two consecutive eigenvalues.

2.1.4 Metric learning approaches

A major research direction is focused on improving the similarity scores for speaker diarization and verification. Since out-of-set speakers are present in the test recordings, jointly learning optimal similarity metrics and robust representation became crucial to minimize within-speaker distance and maximize between-speakers distance. This is referred to as *metric learning*. It involves using neural networks and dedicated loss functions to *learn* a *custom* similarity/distance metric that can further improve diarization performance over standard PLDA/cosine over embeddings.

Lin et al. (2019) [70] used Bi-LSTM based networks to learn the pairwise scoring model with the metric learning objective. The Bi-LSTM model helps capture the sequential order of speech segments absent in PLDA scoring. It uses forward and backward segments to utilize the structural information and turn-taking behaviors of speakers in a conversation. Wang et al. (2020) [71] proposed Graph Neural Networks (GNN) for metric learning. The GNN takes a graph as the input where speaker embeddings are nodes, and the similarity scores are used to form edges by applying a threshold. The output of the model is a probability score of whether two nodes are connected or not. All these approaches perform spectral clustering on the learnt similarity scores matrix.

2.1.5 Post-processing re-segmentation approaches

To refine the boundaries of segmentation output in speaker diarization, a second re-segmentation step called variational-Bayes HMM (VB-HMM) involving frame-level (20-30ms) modeling

[72, 73] can be performed. This can be regarded as the joint optimization of segmentation and clustering. In this framework, the front-end MFCC features $\boldsymbol{X}$ are assumed to be generated from HMM, and each state of the HMM corresponds to one of the possible speakers. The state distribution is modeled using a hidden variable $\boldsymbol{Y}$. Suppose $\boldsymbol{Z} = \{z_1, ..., z_T\}$ indicates sequence of speakers present from $t = \{1, .., T\}$. The diarization problem can be expressed as an estimation of $\boldsymbol{Z}$ that maximizes the posterior distribution $P(\boldsymbol{Z}|\boldsymbol{X}) = \int P(\boldsymbol{Z}, \boldsymbol{Y}|\boldsymbol{X})d\boldsymbol{Y}$, which is intractable. Therefore, the VB inference estimates the model parameters that approximate $P(\boldsymbol{Z}, \boldsymbol{Y}|\boldsymbol{X})$. For the best performance, the posterior distribution is initialized using results from a clustering based model. The state distribution is initialized using Universal Background GMM (UBM-GMM) trained for i-vector modeling. The posterior distribution is updated using HMM forward-backward algorithm. VBx, a simplified version of VB-HMM, is also being adopted recently [27]. In VBx, the observations are the sequence of x-vectors, and the distribution $P(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{Y})$ is computed using the PLDA model.

2.1.6 Overlap detection approaches

Multiple speakers can often speak simultaneously in natural conversations, resulting in overlapped speech, e.g., news debates. Therefore, incorporating overlapping speech detection is important to improve diarization performance for real-life recordings. Medennikov et al. (2020) [74] proposed target-speaker voice activity detection (TS-VAD) to achieve accurate speaker diarization even under noisy conditions with speaker overlaps. TS-VAD uses the clustering output to estimate the i-vector for each speaker and then feeds it to the model along with the sequence of MFCC to estimate the speech activity of each of the speakers at each time frame. This allows us to predict multiple speakers at each time frame. The approach iteratively performs i-vectors estimation based on speaker activity and predicts the speech activity of each speaker. Bullock et al. (2020) [75] proposed a Long Short-Term Memory (LSTM) based architecture for overlap detection. The model called pyannote [76] is trained with hand-crafted MFCC features or trainable SincNet features. The model performs 2-class classification (overlapped versus clean speech) using binary cross entropy loss. The pyannote model is combined with VB-HMM to obtain the two most likely speakers in the predicted

overlapping frames.

2.1.7 End-to-end neural diarization approaches

In recent times, there has been a growing need for a more simplified and efficient approach to speaker diarization. The existing modular approach of segmentation and clustering can be time-consuming and challenging to optimize. In response to this demand, Zhang et al. (2019) [9] explored the concept of an end-to-end single neural model known as an unbounded interleaved recurrent neural network (UIS-RNN). This fully supervised speaker diarization system is trained on conversational data with speaker labels. It models each speaker by an RNN instance, and these instances share the same parameters. There is no limit on the number of RNN instances produced. The states of distinct RNN instances correspond to different speakers and are interleaved in the time domain. Given the input sequence of embeddings, UIS-RNN generates the diarization result as a sequence of speaker index for each time frame. This approach allows to predict speaker labels in an online fashion.

Separately, the application of end-to-end modeling for two-speaker conversational data has been explored in [77], which provides a complete neural solution to the problem of speaker diarization. In the end-to-end neural diarization (EEND) systems [78, 38], the input features are fed to a self-attention transformer model where the loss is permutation-invariant cross entropy (PIT) [79, 80]. For each speaker, it predicts the probability of the presence of the speaker at each time step. Permutation-invariance ensures that the model is indifferent to the true speaker identity and only learns the relative differences and similarities among the speakers. This also enables performance speech activity detection and predicts overlapping speech regions and corresponding speakers in those regions. The presence of an unknown number of speakers is handled using encoder-decoder based attractor [38] modeling. The attractor is an LSTM based model to count the number of speakers in the recording. However, training the EEND system with more speakers is challenging due to the number of permutations involved in the loss computation. To overcome this challenge, an unsupervised clustering process embedded in the attractor-based end-to-end diarization is introduced [81]. This allows to perform attractor-based diarization on short subsequences of a recording. Then inter-subsequence speaker

correspondence is obtained by unsupervised clustering of the vectors computed from the attractors from all the subsequences. This approach can handle more number of speakers and performs on par with the traditional clustering-based approaches at the cost of large computational time and data.

Bredin et al. (2020) [76] proposed an end-to-end speaker segmentation for overlap-aware resegmentation. This again involves performing speech activity detection, speaker change detection, and overlap-aware resegmentation. The model is based on BLSTM layers followed by fully connected layers to predict speaker presence probability, and it also used permutation-invariance while training. However, this approach deals with shorter chunks(5s) to avoid a large number of speakers and uses an external clustering output as initialization to perform overlap-aware resegmentation.

2.2 Speaker Diarization Evaluation Metrics

The Rich Transcription Time Marked (RTTM or rttm) [1] is a universally accepted format to store diarization output. It contains the recording id, start and duration of each speaker spoken continuously and other minor details in tab-separated format. First, an optimal mapping between reference speakers and predicted speakers is generated using the Hungarian algorithm [82] to evaluate the performance. The maximum intersection between regions of reference speakers (say A, B) with the corresponding speakers (say 1, 2) in the system output is computed to obtain an optimal mapping. This section describes the most widely used evaluation format and metrics for evaluating speaker diarization.

2.2.1 Diarization Error Rate (DER)

The performance of diarization systems is primarily measured using Diarization Error Rate (DER)[1][1, 17, 3]. DER is computed as follows:

$$DER = \frac{FA + Miss + Confusion}{Total} \tag{2.8}$$

[1] https://github.com/nryant/dscore

where *FA (false alarm)* is the duration of non-speech predicted as speech in the system generated rttm, *Miss or miss detection* is the duration of speech predicted as non-speech, and *Confusion(speaker confusion)* is the duration of speech of a reference speaker predicted as wrong speaker in the system rttm after optimal mapping. Total is the total duration of all the speakers present in the recording. FA captures false alarms in speech activity detection and overlapped speech detection. Miss captures miss detection error in the speech activity detection and overlapped speech detection. DER can exceed 100% if FA is higher than the total duration of the speakers.

2.2.2 Jaccard Error Rate (JER)

The Jaccard error rate (JER) was first introduced in the DIHARD II evaluation [17]. The goal of JER is to evaluate each speaker with equal weight. Unlike DER, which is estimated for the whole utterance, per-speaker error rates are computed and then averaged to compute JER [3]. Specifically, JER is computed as follows:

$$JER = \frac{1}{N} \sum_{i}^{N_{ref}} \frac{FA_i + Miss_i}{TOTAL_i} \tag{2.9}$$

In Equation 2.9, $Total_i$ is the union of the i-th speaker's speaking time in the reference transcript and the i-th speaker's speaking time in the hypotheses. N_{ref} is the number of speakers in the reference script. JER can range from 0-100%.

JER and DER are highly correlated. However, DER tends to give more weightage to the dominant speakers, whereas JER gives equal weightage to all the speakers. Therefore, JER is generally greater than DER, especially when the predicted rttm has missed the less dominant speaker.

2.3 Speaker Diarization Train Datasets

This section describes the speaker-labeled dataset used in this thesis for training the speaker embedding extraction models (e.g., x-vectors) and the PLDA model.

2.3.1 The Switchboard Cellular dataset

The Switchboard Cellular refers to a specific subset of the Switchboard corpus, which contains recordings of phone calls made on cellular networks, providing data to develop and evaluate speech recognition, speaker verification, language identification, and speech signal detection systems optimized for mobile phone scenarios. Switchboard Cellular Part 1 Audio [83] was developed by the Linguistic Data Consortium (LDC) and consists of approximately 109 hours of English telephone conversations collected by LDC between 1999-2000. During the study period, LDC collected 1,309 calls from 254 participants (129 male speakers, 125 female speakers) under varied environmental conditions. In 2000 LDC collected the Switchboard Cellular Part 2 Audio [84], which consists of approximately 200 hours of English telephone conversations.

2.3.2 The NIST Speaker Recognition Evaluation datasets

The National Institute of Standards and Technology (NIST) Speaker Recognition Evaluation (SRE) series aims to contribute to the direction of research efforts and the calibration of technical capabilities of text-independent speaker recognition. To this end, NIST has been coordinating Speaker Recognition Evaluations since 1996. The datasets comprise conversational telephone speech (CTS) data gathered from various handsets and mobile devices (PSTN and VoIP) and interview recordings from participants within and outside the United States of America. The audio recordings have a sampling rate 8kHz and are labeled with their corresponding speaker id to train the speaker classification networks, e.g., TDNN. They also encompass conversations in multiple languages, including English spoken by non-native speakers. These datasets are utilized to assess and compare speaker verification systems. In this thesis, the x-vector model was trained on SRE 2004-2008 datasets [85, 86, 87] for telephone test sets.

2.3.3 The Voxceleb 1 & 2 dataset

The VoxCeleb datasets are widely used for speaker verification and speaker recognition research. The VoxCeleb 1 [88] was introduced in 2017 by the Visual Geometry Group (VGG) from the University of Oxford. The dataset contains speech data from YouTube videos featuring

celebrities and public figures. The purpose of VoxCeleb 1 is to provide a large-scale and diverse dataset for training and evaluating speaker recognition models. It includes approximately 1,251 speakers, with over 200,000 utterances. Voxceleb 2 [89], an extension to Voxceleb 1, was released in 2018. VoxCeleb 2 provides a more extensive and diverse collection of speakers, making it suitable for training and evaluating state-of-the-art speaker verification systems. This dataset includes approximately 6,112 speakers with over one million utterances.

2.3.4 The LibriSpeech dataset

The LibriSpeech dataset [90] is a widely used open-source collection of English speech data primarily designed for training and evaluating automatic speech recognition (ASR) systems. It consists of a large amount of transcribed speech from various audiobooks. The complete dataset contains over 1,000 hours of 16kHz speech data prepared by Panayotov et al. It comprises a diverse set of speakers, providing variations in accent, pronunciation, and speaking style. It has been used in speaker diarization to simulate conversational speech for training the diarization systems.

2.3.5 The MUSAN dataset

MUSAN [91] is a corpus of music, speech, and noise recordings. The dataset consists of music from several genres, speech from twelve languages, and various technical and non-technical noises. The corpus comprises approximately 109 hours of WAV audio files sampled at 16kHz in the US Public Domain or under a Creative Commons license. It has been used extensively for data augmentation to train large and robust speaker classification models.

2.3.6 The Room Impulse Response (RIR) and noise dataset

It [92] includes the real RIRs and isotropic noises from the RWCP sound scene database, the 2014 REVERB challenge database, and the Aachen impulse response database (AIR); the simulated RIRs generated by ourselves, and also the point-source noises that extracted from the MUSAN corpus. It was released as part of data augmentation techniques for multi-condition training.

Table 2.1: Details of various speaker diarization test datasets used this thesis where SR - sampling rate, dur. - duration, rec. - recording, ovp. - overlap.

Test dataset	SR	Total dur. (hours)	Dur./rec. (mins)	#recs./split (dev, eval)	#spks/rec.	ovp./rec. (%)
CALLHOME (CH)	8kHz	~40	2-5	(250, 250)	2-7	0-15
AMI	16kHz	~25	20-60	(18, 16)	3-5	5-20
DIHARD III	16kHz	~40	0.5-10	(254, 259)	1-10	0-80
Voxconverse	16kHz	~64	0.5-20	(216, 232)	1-21	0-30
DISPLACE	16kHz	~32	20 -30	(27, 29)	3-5	5-20

2.4 Speaker Diarization Test Datasets

This section describes the conversational datasets used in this thesis for evaluating the performance of the proposed work reported in this thesis. Table 2.1 gives the statistics of the test datasets which are described below, such as the total duration of each dataset, range of duration per recording and number of speakers present in each recording and so on.

2.4.1 The CALLHOME dataset

The CALLHOME (CH) dataset [93] is a collection of multilingual telephone data sampled at 8kHz, containing 500 recordings, where the duration of each recording ranges from 2-5 mins. The number of speakers in each recording varies from 2 to 7, with most of the files having 2 speakers. The CH dataset is divided equally into 2 different sets, CH1 and CH2, with a similar distribution of the number of speakers.

2.4.2 The AMI dataset

The AMI dataset [5] contains meeting recordings from four different sites (Edinburgh, Idiap, TNO, Brno). The official speech recognition partition of the AMI dataset [5] comprises training, development (dev), and evaluation (eval) sets consisting of 136, 18, and 16 recordings sampled at 16kHz, respectively. The single-distant microphone (SDM) condition of the AMI dataset is used for experiments. The AMI train set contains 75 hrs of labeled speech. The number of speakers and the duration ranges of each recording from 3-5 and 20-60 mins,

respectively.

2.4.3 The Third DIHARD challenge dataset (DIHARD III)

The third DIHARD challenge dataset [94] was released as the third in a series of DIHARD speech diarization challenges. It consists of development and evaluation sets of recordings with a duration of 0.5-10 mins. These recordings are drawn from 11 domains, including audiobooks, telephone recordings, clinical interviews, restaurant conversations, web videos, etc. The number of speakers varies from 1-10 with diverse regions of overlapping speech and speaker turn behavior. There are 254 and 259 recordings in the development and evaluation sets, respectively.

2.4.4 The Voxconverse dataset

It is an audio-visual diarization dataset [95] consisting of multi-speaker human speech recordings extracted from YouTube videos. It is divided into a development (dev) set and an evaluation (eval) set of 216 and 232 recordings, respectively. The duration of a recording ranges from 22-1200s. The number of speakers per recording varies from 1-21.

2.4.5 The DISPLACE challenge dataset

The DISPLACE challenge 2023 [96] highlighted outstanding issues in speaker diarization (SD) in multilingual settings with code-mixing. The dataset released as part of the challenge comprises single-channel far-field natural multilingual, multi-speaker conversational speech recordings. The development (dev) and evaluation (eval) set contains 27 recordings (15.5 hours) and 29 recordings (16 hours), respectively. The speakers in the dev and eval sets are mutually exclusive. The number of speakers per recording varies from 3-5. The dataset contains conversations in Hindi, Kannada, Bengali, Malayalam, Telugu, Tamil, and Indian English. In each recording, the number of languages varies from 1-3.

2.5 Chapter Summary

This chapter discussed various unsupervised clustering approaches used in literature for speaker diarization, along with metric learning and end-to-end models. Then we briefly

explained different statistical and neural models for speaker representations/embeddings extraction, followed by similarity measures used to compute the similarity scores needed for clustering. Further, we described the evaluation metrics used for the performance evaluation of diarization systems. Finally, we discussed various train and test datasets used for training and evaluation of diarization model performance. In the following chapters, we describe our efforts in developing robust diarization system using novel graph clustering approaches which achieves the state-of-the-art performance on benchmark datasets.

Chapter 3

Self-Supervised Hierarchical Clustering

The state-of-the-art speaker diarization systems typically involve a two-stage processing approach where audio segments of fixed duration are converted to vector representations in the first stage. This is followed by an unsupervised clustering of the representations in the second stage. These two stages are performed in an isolated manner with independent optimization steps. In this chapter, we propose a novel solution to the task of speaker diarization termed Self-supervised Clustering (SSC) which combines speaker clustering and representation learning. The proposed approach is based on principles of self-supervised learning, where the self-supervision is derived from the clustering algorithm. The representation learning network is trained with a regularized triplet loss using the clustering solution at the current step. In contrast, the clustering algorithm uses deep embeddings from the representation learning step. In this work, we introduce two variations of SSC [1] *based on the choice of agglomerative clustering. The first approach, SSC-AHC, combines the self-supervision based representation learning with the Agglomerative Hierarchical Clustering (AHC) algorithm. We show that the proposed algorithm improves significantly (*29% *relative improvement) over the AHC algorithm with cosine similarity on the CALLHOME dataset. In addition, it also achieves* 10% *relative improvement in DER over the state-of-the-art system with PLDA affinity matrix. The second approach, SSC-PIC, involves combining representation learning with a graph-based clustering algorithm called path integral clustering (PIC). This is the first work to integrate PIC in speaker diarization, achieving huge gains. The representation learning step uses the cluster targets from PIC. We show that the SSC-PIC algorithm improves significantly over the baseline system (relative improvements of* 13% *and* 59% *on CALLHOME and AMI datasets, respectively.*

[1] This work has been published in Interspeech 2020 [97] and IEEE Transactions on Audio, Speech and Language Processing, 2021 [98].

3.1 Introduction

Self-supervised learning is a branch of unsupervised learning where the data provides supervision labels for model learning. Self-supervision can provide effective representations for downstream tasks without requiring the ground-truth labels [99]. Self-supervised learning has been explored in the clustering of images and text documents using losses such as k-means loss [100], spectral clustering loss [101] and agglomerative clustering loss [102] derived from the unlabelled data. In the speech processing domain, it has been attempted for phoneme and speech recognition tasks [103].

In this chapter, we propose an approach based on self-supervised representation learning for speaker diarization. The proposed approach involves iteratively updating embedding representations and cluster identities. This approach is inspired by Yang et al. [102], where joint representation learning using agglomerative clustering based loss was explored for image clustering applications. The approach alternates between merging the clusters for a fixed embedding representation and learning the representations using the given cluster labels. We show that the proposed approach can be applied to traditional AHC as well as to other graph-based clustering methods like path integral clustering (PIC) [104]. There are three main reasons to choose PIC as follows:

1. It measures the affinity of clusters based on the neighborhood graph instead of directly on pairwise distances/similarities and hence is more robust to noisy distances compared with linkage algorithms [68] commonly used in agglomerative clustering.

2. The graph structural merging strategy makes it more robust to noisy links than spectral clustering [105, 70], which utilizes eigen-decomposition.

3. It does not assume anything on the underlying data distributions and only needs the pairwise distances or similarities of samples. Therefore, it is flexible and better generalizable.

We refer to the proposed approach of joint representation learning and clustering as self-supervised clustering (SSC). The following are the key contributions from this work.

- A method to jointly learn deep representations and speaker clusters from an unlabeled recording.
- Introducing graph based PIC to perform robust clustering for updating representations.
- A stopping criterion based on eigenvalues of the affinity matrix.
- Incorporating a temporal continuity criterion in the SSC to improve the smoothness of the clustering decisions.

The representation learning framework in the proposed SSC approach uses a triplet similarity based learning. With the joint learning of the two processes in a single model, we aim to obtain good representations of the audio and precise speaker clusters. The diarization experiments are performed on the CALLHOME (CH) [93] and the Augmented Multi-party Interactions (AMI) [5] datasets. Using the proposed SSC approach with PIC affinity measure, we illustrate significant improvements over the x-vector AHC baseline system, discussed in sections 3.3.1 and 3.3.2. The proposed approach also shows advancements over other published results on these datasets. Furthermore, the proposed approach can be used as an initialization for frame-level refinement based on variational Bayes (VB) hidden Markov model (HMM) [72] (discussed in chapter 2, section 2.1.5).

3.2 Self-supervised Clustering

The block schematic of the self-supervised clustering (SSC) algorithm is given in Figure 3.1. The inputs to the model are x-vector embeddings extracted from fixed-length audio segments. The deep neural network (DNN) in Figure 3.1 is a two layer network. The output layer of the DNN generates representations used in the clustering.

The SSC implements iterative steps of clustering and representation learning for each recording separately. The clustering (forward operation) is performed using agglomerative clustering (described in Section 3.2.4). The representation learning (backward operation) is performed using the DNN training with modified triplet similarity (described in Section 3.2.3). The iterative process is repeated till the required number of clusters is reached, or the stopping

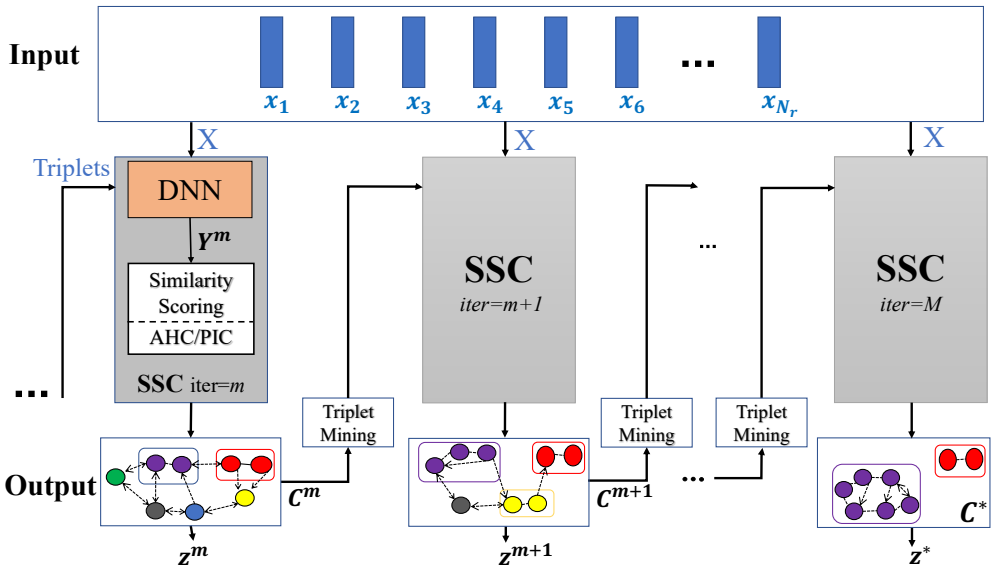

Figure 3.1: Block schematic of the self-supervised diarization (SSC). The DNN embeddings are used in the AHC/PIC algorithm. The clustering outputs create labels z^m used in sampling triplets for iteration $(m+1)$. The bottom row - circles represent embeddings, colors represent clusters, and the dashed lines correspond to connections based on scores.

criterion is met (discussed in Section 3.2.5).

3.2.1 Notations

The model parameters of the representation learning deep neural network (DNN) are denoted by $\boldsymbol{\theta}$. m denotes the iteration index. Further, let

- $\mathbf{Y}^m = \{\boldsymbol{y}_1^m, ..., \boldsymbol{y}_{N_r}^m\} \in \mathbb{R}^d$ denote the output representations from the DNN model.
- $\mathbb{C}^m = \{\mathbb{C}_1^m, ..., \mathbb{C}_n^m, ..., \mathbb{C}_{N^m}^m\}$, where $n-$th cluster, $\mathbb{C}_n^m = \{\boldsymbol{y}_i^m | z_i^m = n\}$.
- N^m denote the number of clusters at m-th iteration.
- N^* denotes the target number of clusters in the algorithm.
- Hyper-parameters:

  K - Number of nearest-neighbors used in PIC (Equation (3.3))

  σ - scaling factor in the computation of path integral in PIC (Equation (3.4))

  α - weighting factor in DNN learning (Equation (3.1))

  ϕ^m - threshold to estimate N^m.

  β, n_b - positive decaying factor and floor value respectively (Equation (3.8))

Algorithm 1: SSC algorithm for joint representation learning and clustering

Initialize:$(m = 0)$

$\boldsymbol{\theta}^0 \leftarrow$ (Whitening+PCA)

$Y^0 \leftarrow$ PCA outputs

If unknown $N^* \rightarrow N^* = 1$

while *continue* **do**

1. $m = m + 1$
2. Sample triplets based on $\boldsymbol{z}^{m-1}$
3. $\boldsymbol{\theta}^{m-1} \xrightarrow{\text{DNN with triplet training}} \boldsymbol{\theta}^m$
4. $\boldsymbol{X}_r \xrightarrow{\text{Forward Pass}(\boldsymbol{\theta}^m)} \boldsymbol{Y}^m$
5. $\tilde{N}^m$ =Estimate_$N^m(\boldsymbol{Y}^m, \phi^m, N^{m-1})$
6. $N^m = max(N^*, \tilde{N}^m)$
7. **if** $N^m == N^*$ *or* $m == M$ **then**

   $M = m$

   $N^* = N^m$

   break

   end
8. $\boldsymbol{Y}^m \xrightarrow{\text{AHC/PIC}(N^m \text{ clusters})} \boldsymbol{z}^m = \{z_1^m, ..., z_{N_r}^m\}$

end

Termination: $\boldsymbol{X}_r \xrightarrow{\text{DNN training + Forward Pass}(\boldsymbol{\theta}^*)} \boldsymbol{Y}^* = \{\boldsymbol{y}_1^*, ..., \boldsymbol{y}_{N_r}^*\}$

$\{\boldsymbol{y}_1^*, ..., \boldsymbol{y}_{N_r}^*\} \xrightarrow{\text{AHC/PIC}(N^* \text{ clusters})} \{z_1^*, ..., z_{N_r}^*\}$

3.2.2 SSC algorithm overview

The overview of the SSC[1] algorithm is given in Algorithm 1. The DNN model is trained using gradient ascent with triplet similarity. At the m-th iteration, positive and negative triplet pairs are formed using the cluster label indices $\mathbf{z}^{m-1}$ of the previous iteration. Using these triplets, we calculate the triplet similarity and learn the DNN model parameters $\boldsymbol{\theta}^m$ with the inputs as x-vector features $\boldsymbol{X}_r$.

[1]Our implementation of SSC-PIC is available at https://github.com/iiscleap/SSC.git

Once the DNN model is trained, we obtain embeddings Y^m from the last layer of DNN. We compute cosine similarity scores matrix S using the embeddings Y^m for the recording.

The agglomerative clustering is performed using the similarity measure such that $N^q \leq N^{m-1}$. We estimate N^m based on the stopping threshold ϕ^m discussed in the next section. We start with N_r clusters and perform multiple merges using the criterion defined in Equation (3.2), thereby reducing the total number of clusters in the m-th iteration. If $N^q = N^*$ or the stopping criterion is met, then the algorithm is stopped. We perform one more iteration of DNN training and clustering to generate the final diarization results. Else, the iteration index is updated, and the steps described above are repeated for the $(q+1)$-th iteration. If N^* is unknown, then the stopping criteria are based on a total number of iterations m or till we reach $N^m = 1$ (whichever is achieved first).

3.2.3 DNN training - dynamic triplet similarity

The clustering algorithm generates labels $\boldsymbol{z}^{m-1}$ where each z_i^{m-1} can take discrete values from $\{1, .., N^{m-1}\}$, with N^{m-1} being the number of clusters at the end of the iteration $(m-1)$. We use the dynamic triplet similarity to train the DNN model. The term dynamic is used to indicate the formation of new triplets after each iteration. We form a positive pair $\{\mathbf{y}_i^{m-1}, \mathbf{y}_j^{m-1}\}$ by selecting the anchor $\mathbf{y}_i^{m-1}$ and positive sample $\mathbf{y}_j^{m-1}$ from each cluster $\mathbb{C}_a^{m-1}$. Here, $\mathbf{y}^{m-1}$ denotes the representations formed at the DNN output using $\boldsymbol{\theta}^{m-1}$. The negative pair is formed by randomly sampling the embedding $\mathbf{y}_l^{m-1}$ from any other cluster $\mathbb{C}_b^{m-1}$ $(a \neq b)$. The DNN model parameters $\boldsymbol{\theta}$ are updated based on the following optimization criteria.

$$\boldsymbol{\theta}^m = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i,j,l} \left[s(i,j) - \alpha \left(s(i,l) + s(j,l) \right) \right] \tag{3.1}$$

where, $s(i,j)$ is pairwise similarity score between $\boldsymbol{y}_i^{m-1}$ and $\boldsymbol{y}_j^{m-1}$. Here, $0 < \alpha \leq 1$ is a weighting factor that controls the discriminability of the negative pairs. In the triplet formation, we also try to uniformly sample a similar number of anchors from all the clusters by suitably oversampling the negative pairs for the same positive pair.

3.2.4 Agglomerative clustering

The clustering algorithm forms the cluster labels for DNN training and plays an important role. The incorrect clustering results will make the diarization performance worse. In this work, we first explored the use of the conventional AHC algorithm. Then, a more robust graph-based agglomerative clustering called path integral clustering (PIC) was introduced to further boost the performance. The details are provided below:

3.2.4.1 AHC

The AHC (chapter 2, section 2.1.3) operates directly on short-segment embeddings (for the baseline system, these are the input x-vectors $\boldsymbol{X}_r$), and the algorithm merges two clusters at each step. The clusters to be merged at the m-th iteration are determined using,

$$\{\mathbb{C}_a, \mathbb{C}_b\} = \underset{\mathbb{C}_i^m, \mathbb{C}_j^m \in \mathbb{C}^m, i \neq j}{\operatorname{argmax}} \mathcal{A}\left(\mathbb{C}_i^m, \mathbb{C}_j^m\right) \tag{3.2}$$

The selected clusters $\{\mathbb{C}_a, \mathbb{C}_b\}$ are merged to form a new cluster $\{\mathbb{C}_a \cup \mathbb{C}_b\}^{m+1}$ for the next iteration. The merging process is stopped when the stopping criterion is met. The affinity $\mathcal{A}$ between clusters is computed using pairwise similarity between embeddings present in the similarity score matrix $\mathbf{S}$. The similarity score ($s(i, j)$) in state-of-the-art diarization systems uses the PLDA modeling [50], while in the proposed SSC algorithm, we use the cosine similarity measure.

3.2.4.2 Path Integral Clustering (PIC)

The use of pairwise similarities for affinity may fail to capture the global structure of data [106]. This can be addressed with the graph based agglomerative clustering algorithms. The PIC [104] is a graph-structural agglomerative clustering algorithm [106] where the graph encodes the structure of the embedding space. It exploits the connectivity of the directed graph to efficiently cluster the embeddings. The PIC has been attempted for image segmentation and document clustering [107].

The PIC uses path integral as a structural descriptor of clusters. The clustering process

Algorithm 2: Path Integral Clustering (Sec. 3.2.4.2)

Initialize: Construct a graph $G = (V, E)$ where, vertices V are the input data $\boldsymbol{X} = \{x_1, x_2, ..., x_{N_r}\}$. The weighted adjacency matrix $\boldsymbol{W}$ is computed and the transition probability matrix $\boldsymbol{P}$ is obtained by normalizing $\boldsymbol{W}$;

Form n_c initial clusters $\mathbb{C} = \{\mathbb{C}_1, ..., \mathbb{C}_{n_c}\}$ by assigning each sample x_i to a cluster, using nearest neighbor merging; N^* = required number of speakers

while $n_c > N^*$ **do**

1. Merge $\mathbb{C}_a$ and $\mathbb{C}_b$, if $\{\mathbb{C}_a, \mathbb{C}_b\} = \underset{\mathbb{C}_a, \mathbb{C}_b \in \mathbb{C}}{\operatorname{argmax}} \mathcal{A}(\mathbb{C}_a, \mathbb{C}_b)$ where, $\mathcal{A}(\mathbb{C}_a, \mathbb{C}_b)$ is given in Equation (4.4)
2. $\mathbb{C}_c \leftarrow \{\mathbb{C}_c \backslash \{\mathbb{C}_a, \mathbb{C}_b\} \cup \{\mathbb{C}_a \cup \mathbb{C}_b\}\}$ and $n_c = n_c - 1$
3. Recompute $\mathcal{A}$

end

Termination: $\mathbb{C}_c$

is treated as a dynamical system, with vertices as input features (states of the system) and edge weights as transition probabilities between states. The affinity between two clusters is defined as the incremental path integral when the two clusters are merged. This affinity is based on the assumption that if two clusters are closely connected, the stability of system can be enhanced by merging the two clusters.

The PIC algorithm described in (2) consists of the following steps,

- Defining the digraph: We build the directed graph based on similarity scores between embeddings. The digraph is denoted as $G = (V, E)$, where V is the set of vertices corresponding to the embeddings in $\boldsymbol{Y}$ and E is the set of edges connecting the vertices. The adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{N_r X N_r}$ is a sparse matrix defined as,

$$[W]_{ij} = \begin{cases} \frac{1}{1+exp(-s(i,j))}, & \text{if } \boldsymbol{y}_j \epsilon J_i^K. \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

  where, J_i^K is the set of K-nearest neighbors of $\boldsymbol{y}_i$. The transition probability matrix $\boldsymbol{P}$ is obtained from the adjacency matrix $\boldsymbol{W}$ by normalizing each row with its sum. The transition probability matrix $\boldsymbol{P}$ contains the edge weights of the graph. The directed

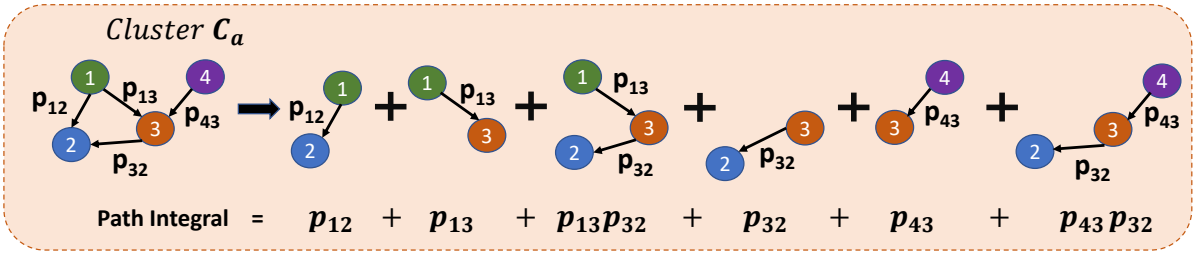

Figure 3.2: Computation of path integral of a graph. Cluster C_a is represented as a graph with edge weights as transition probabilities obtained using Equation (3.3). The path integral is the sum of probabilities of all possible paths in the graph.

edge connecting node i to node j exists only if the value of $[W]_{ij}$ is non-zero.

- Cluster initialization: We group the embeddings $\boldsymbol{Y}$ with their first nearest neighbour to form $N_r/2$ clusters. If the clusters have common elements, these are further merged.

- Defining path integral: We define path integral $\mathcal{S}_{\mathbb{C}_a}$ of a cluster $\mathbb{C}_a$ as the weighted sum of probabilities of all possible paths from any vertex i to any other vertex j, where i, j vertices belong to the cluster $\mathbb{C}_a$ and all the vertices along the path also belong to cluster $\mathbb{C}_a$. An illustration of path integral is shown in Figure 3.2. The unnormalized pairwise path integral over all paths (of lengths k ranging from 1 to ∞) from vertex i to j for $\mathcal{G}_{\mathbb{C}_a}$ is given as:

$$g_{ij} = \delta_{ij} + \sum_{k=1}^{\infty} \sigma^k \sum_{\gamma \in \Gamma_{ij}^{(k)}} \prod_{s=1}^{k} p_{u_{s-1},u_s} \tag{3.4}$$

where, $u_0 = i$, $u_k = j$, p_{u_{s-1},u_s} is transition probability from vertex u_{s-1} to u_s and δ_{ij} is Kronecker delta function defined as $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. $\Gamma_{ij}^{(k)}$ is the set of all possible paths from i to j of length k. The constant $0 < \sigma < 1$ gives more weight to shorter paths compared to longer paths. The path integral $\mathcal{S}_{\mathbb{C}_a}$ of the cluster $\mathbb{C}_a$ is then the summation of g_{ij} over all pairs of vertices $i, j \in \mathbb{C}_a$ normalized by $|\mathbb{C}_a|^2$.

We define the conditional path integral $\mathcal{S}_{\mathbb{C}_a|\mathbb{C}_a\cup\mathbb{C}_b}$ as the path integral of all paths in $\mathbb{C}_a \cup \mathbb{C}_b$ such that the paths start and end with vertices belonging to $\mathbb{C}_a$.

The unnormalized path in Equation 3.4 is:

$$\begin{aligned} g_{ij} &= \delta_{ij} + \sum_{k=1}^{\infty} \sigma^k \sum_{\gamma \in \Gamma_{ij}^{(k)}} \prod_{s=1}^{k} p_{u_{s-1},u_s} \\ &= \delta_{ij} + \sum_{k=1}^{\infty} \sigma^k [\boldsymbol{P}_{\mathbb{C}_a}^k]_{ij} \\ &= [\boldsymbol{I} + \sum_{k=1}^{\infty} \sigma^k \boldsymbol{P}_{\mathbb{C}_a}^k]_{ij} \\ &= [(\boldsymbol{I} - \sigma \boldsymbol{P}_{\mathbb{C}_a})^{-1}]_{ij} \end{aligned}$$

Therefore, the normalized path integral and the normalized conditional path integral converges to,

$$\mathcal{S}_{\mathbb{C}_a} = \frac{1}{|\mathbb{C}_a|^2} \mathbf{1}^T \left(\boldsymbol{I} - \sigma \boldsymbol{P}_{\mathbb{C}_a}\right)^{-1} \mathbf{1} \tag{3.5}$$

$$\mathcal{S}_{\mathbb{C}_a|\mathbb{C}_a \cup \mathbb{C}_b} = \frac{1}{|\mathbb{C}_a|^2} \mathbf{1}_{\mathbb{C}_a}^T \left(\boldsymbol{I} - \sigma \boldsymbol{P}_{\mathbb{C}_a \cup \mathbb{C}_b}\right)^{-1} \mathbf{1}_{\mathbb{C}_a} \tag{3.6}$$

where, $\boldsymbol{P}_{\mathbb{C}_a}$ and $\boldsymbol{P}_{\mathbb{C}_a \cup \mathbb{C}_b}$ are the sub-matrices of the transition probability matrix $\boldsymbol{P}$ obtained by selecting vertices that belong to $\mathbb{C}_a$ and $\mathbb{C}_a \cup \mathbb{C}_b$ respectively. Here, $|\mathbb{C}_a|$ denotes the cardinality (# of vertices) of cluster $\mathbb{C}_a$, $\mathbf{1}$ is a column vector of all ones of size $|\mathbb{C}_a|$ and $\mathbf{1}_{\mathbb{C}_a}$ is a binary column vector of size $|\mathbb{C}_a \cup \mathbb{C}_b|$ with ones at all locations corresponding to the vertices of $\mathbb{C}_a$ and zeros at all locations corresponding to the vertices of $\mathbb{C}_b$. Note that the identity matrix $\boldsymbol{I}$ used in Equation (3.5) and (3.6) is of dimensions $|\mathbb{C}_a|$ and $|\mathbb{C}_a \cup \mathbb{C}_b|$ respectively.

- Cluster merging: The cluster affinity measure for the PIC algorithm is computed as,

$$\mathcal{A}\left(\mathbb{C}_a, \mathbb{C}_b\right) = [\mathcal{S}_{\mathbb{C}_a|\mathbb{C}_a \cup \mathbb{C}_b} - \mathcal{S}_{\mathbb{C}_a}] + [\mathcal{S}_{\mathbb{C}_b|\mathbb{C}_a \cup \mathbb{C}_b} - \mathcal{S}_{\mathbb{C}_b}] \tag{3.7}$$

where the term inside each square bracket is called incremental path integral. The computation of the incremental path integrals is illustrated in Figure 3.3. A higher

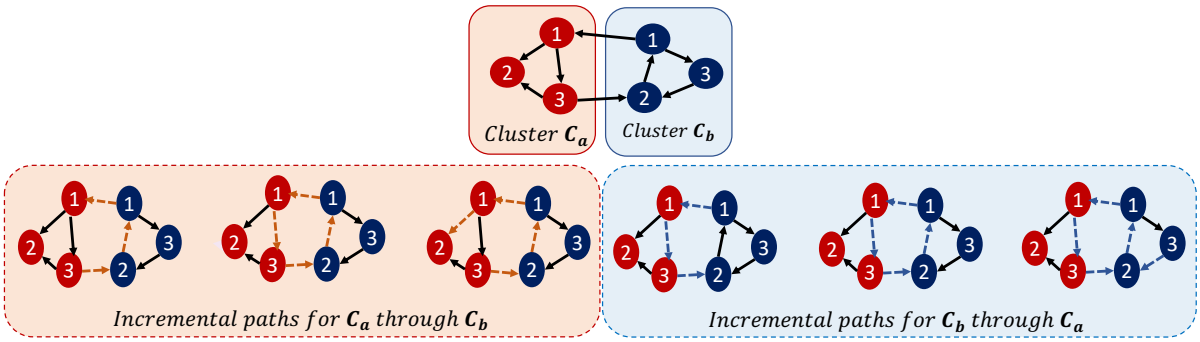

Figure 3.3: Illustration of incremental paths of clusters C_a and C_b for computing affinity using Equation (3.7). The second row shows paths for $C_a \cup C_b$. The second row, left side (orange block) highlights paths that start and end in cluster C_a with black dashed arrows. The second row, the right side (blue block), shows the paths which start and end in C_b with blue dashed arrows.

Algorithm 3: Estimate number of clusters N^m for PIC: Estimate_$N^m(Y^m, \phi^m, N^{m-1})$

1. $Y^m \xrightarrow{\text{PIC}(N^{m-1}\text{ clusters})} S^{m-1} \in \mathbb{R}^{N^{m-1} X N^{m-1}}$: Affinity matrix of N^{m-1} clusters such that $[S^{m-1}]_{ij} = \mathcal{A}(\mathbb{C}_i, \mathbb{C}_j)$
2. $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_{N^{m-1}}$: Eigen values of $\mathbf{A}$; $total\_energy = \sum_{i=1}^{N^{m-1}} \lambda_i$
3. $v \in \mathbb{R}^{N^{m-1}}$: cumulative explained variance ratio; $[v]_k = \frac{\sum_{i=1}^{k} \lambda_j}{total\_energy}$ where $k \in \{1, ..., N^{m-1}\}$
4. $N^m = \underset{k}{\text{argmax}}([v]_k \leq \phi^m)$
5. return N^m

value of affinity between two clusters indicates the presence of more dense connections between them. Thus, merging the clusters with maximum affinity will form a more coherent cluster. Using the definition of affinity (Equation (3.7)), the clusters with maximum affinity are merged at each iteration, as shown in Equation (3.2).

3.2.5 Estimation of number of clusters

The number of clusters N^m required for clustering at each iteration m is estimated using a threshold on the similarity scores in AHC and based on explained variance in PIC. In case of PIC, we compute affinity matrix S^{m-1} of N^{m-1} clusters using Equation (3.7). The

diagonal elements are set as the maximum value of non-diagonal elements of the matrix. Then, we compute the eigenvalues of the matrix and arrange them in descending order. We accumulate the eigenvalues until the cumulative explained variance ratio, the ratio of accumulated eigenvalues to the total sum of eigenvalues, reaches a stopping threshold ϕ^m for the m-th step. The number of accumulated eigenvalues at this step is denoted as N^m. The steps are mentioned in Algorithm 3.

3.2.6 Incorporating temporal continuity

In the given audio stream, the event that two neighboring x-vector embeddings come from the same speaker is more likely than the event that they belong to different speakers. This heuristic can be incorporated into the clustering algorithm. After DNN training, we obtain similarity score matrix $\boldsymbol{S}$ using the output embeddings and multiply the similarity score $s(i, j)$ with an exponentially decaying function of the temporal distance between i and j.

$$s'(i, j) = s(i, j)\beta^{min(n_b, |i-j|)} \tag{3.8}$$

where β is a positive decaying factor ($0 < \beta < 1$), $|i - j|$ is the absolute value of the time difference between segment i and j, n_b denotes a floor value which prevents the similarity score from going too low for segments that are farther in time. The above modification of similarity score encourages neighboring clusters to merge and have smooth transitions among clusters.

3.3 Experimental Setup

The CALLHOME (CH) dataset [93] and the AMI dataset [5] as described in chapter 2, section 2.4 are used for performance evaluation. The CH dataset containing 500 recordings is divided equally into 2 different sets, CH1 and CH2, with a similar distribution of speakers.

The x-vector extractor is a 7-layer TDNN model. The details of the model architecture are discussed in chapter 2, section 2.1.1. Table 3.1 shows the differences in x-vector training configuration for CH and AMI datasets.

Table 3.1: The x-vector training configurations for baseline of CH and AMI datasets.

Parameter	CH	AMI
Sampling Rate	8kHz	16kHz
Train set	SWBD, SRE 04-08	Voxceleb 1,2
# speakers	4,285	7,323
Input features	23D MFCCs	30D MFCCs
X-vector dimension	128	512

3.3.1 Experiments on CH dataset

Baseline

The baseline system used for the CH diarization task comes from the Kaldi recipe[1]. It involves the extraction of 23 dimensional mel-frequency cepstral coefficients (MFCCs). A sliding window mean normalization is applied over a 3s window after the feature extraction. For training the x-vector TDNN model and the PLDA model, we use the SRE 04-08 and Switchboard cellular datasets as given in the Kaldi recipe [108]. This training set had 4, 285 speakers. For the diarization task, speech segments (1.5s chunks with 0.75s overlap) are converted to 128 dimensional neural embeddings (x-vectors) [26]. The x-vectors are processed by applying the whitening transform obtained from the held-out set followed by length normalization [109]. An utterance level PCA is applied for dimensionality reduction [110] preserving 10% of total energy. These embeddings are used in the PLDA model to compute the similarity score matrix. The clustering is performed using the AHC algorithm discussed in section 3.2.4. The AHC stopping criterion is determined using held-out data (CH1 is used for diarization on CH2 data and vice-versa).

Implementation details

We use the same setup in the baseline system to extract the x-vector features. The number of x-vectors per recording (N_r) ranges from 50-700. The DNN model is a two-layer, fully connected DNN. The first layer has 128 input and hidden nodes. The second layer has 10 output nodes. The first layer of the DNN also realizes the unit length normalization as the

[1] https://kaldi-asr.org/models/m6

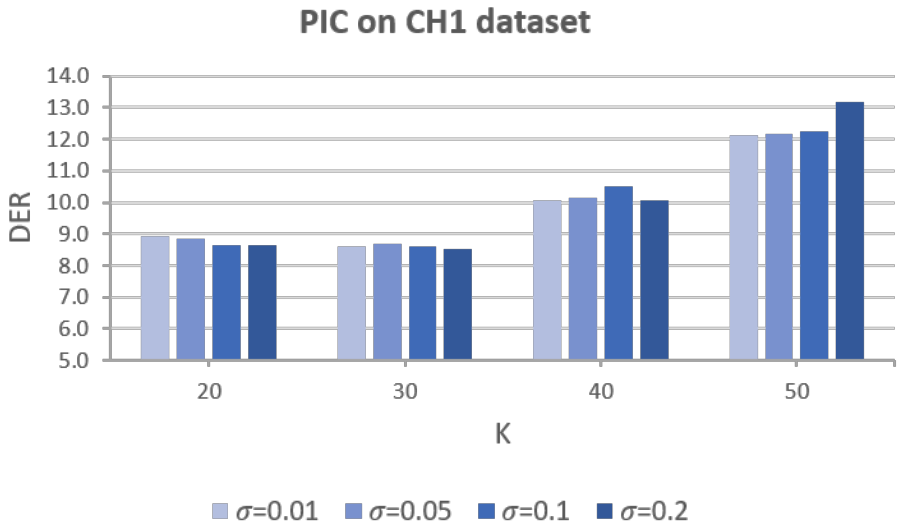

Figure 3.4: DER of CH1 dataset for different choices of number of nearest-neighbors K and scaling factor σ.

non-linearity. The learning rate is set at 0.001. The model is trained using the triplet similarity defined in Equation (3.1). We do not split the training triplet samples into mini-batches but rather perform a full batch training with Adam optimizer [111]. We use a parameter $\eta = 0.5$ (defined as the fraction of the training similarity measure at the zeroth iteration compared to the current iteration) for early-stopping of the DNN model training. Typically, this model training involves 5-10 epochs. The DNN outputs (10 dimensional embeddings) are used to compute the pairwise similarity score matrix using cosine similarity for the PIC.

3.3.2 Experiments on AMI dataset

Baseline

The baseline x-vector system uses the DIHARD Challenge recipe[1] for the AMI dataset. It involves the extraction of 30 dimensional MFCCs. The x-vectors are of dimension 512. For training the x-vector extractor and the PLDA model in AMI experiments, we trained the TDNN model using 16 kHz data from VoxCeleb-1 [88] and VoxCeleb-2 [112] datasets containing 7,323 speakers. In the post-processing of the x-vectors, we use the held-out dataset from the second DIHARD challenge [17] for computing the whitening transform. The rest of the steps in the PLDA-AHC baseline system follow the same processing pipeline used in the CH baseline system.

[1] https://github.com/iiscleap/DIHARD_2019_baseline_alltracks

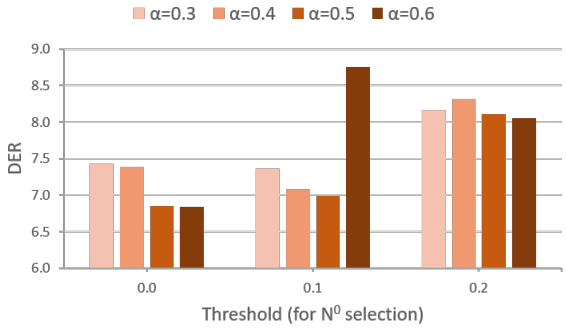

Figure 3.5: Bar plot of DER vs threshold for N^0 selection on CH1 subset of CALLHOME for different choices of α parameter.

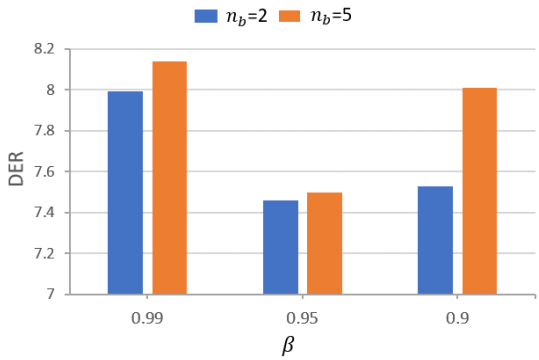

Figure 3.6: Effect of temporal weighting (β, n_b) on DER for CH1 dataset. The best DER is obtained for $n_b = 2$ with $\beta = 0.95$.

Implementation details

The DNN model is a two-layer feed-forward architecture (similar to the CH dataset). The first layer has 512 input and hidden nodes. The second layer has 30 output nodes (similar to the PCA dimension used in the baseline system). In the AMI dataset, the number of x-vectors per recording range from 1000-4000. Hence, a large number of triplets can be generated for each recording. We resort to the use of minibatches in DNN model training. A validation split is also formed in each recording. The loss on the validation set is used to decide the stopping point for training. We perform a learning rate annealing [70] in the DNN model training.

3.3.3 Choice of hyper-parameters

- **Number of nearest-neighbours K and scaling factor σ:** We choose K and σ values based on experiments on the CH1 subset of the CALLHOME dataset. Figure 3.4 shows the overall DER on CH1 for different values of K and σ. We observe that as we increase

K, the DER is increasing. The higher K value also increases computational complexity. Based on the results, the value of K is chosen as 30. For a fixed K, we vary σ from 0.01 to 0.2. The parameter σ has only a minor impact on the final DER performance. The value $\sigma = 0.1$ is chosen for all our experiments.

- **Initial number of clusters (N^0) and α:** The DNN training is performed only after the PIC/AHC clustering algorithm is performed for a few iterations to reduce the number of clusters from N_r to N^0. If the value of N^0 is too large, then the discriminative triplet similarity measure tends to increase the within-speaker diversity (controlled by factor α). However, if the value of N^0 is reduced significantly, then the clusters formed can be potentially impure in terms of the speaker identity of the elements in the cluster.

  For CH dataset, we initialize the model using AHC and vary the threshold applied on similarity scores to stop at N^0 number of clusters. In Figure 3.5, we observe that for lower threshold (smaller N^0), higher α (more discriminative) is better but as we increase threshold (larger N^0), lower α (less discriminative) is optimal. We obtain the best DER with threshold= 0.0 and $\alpha = 0.6$, which is used in CH experiments. In the AMI experiments, we choose N^0 as the smallest number obtained using the threshold ϕ for the explained variance on the affinity.

- **Temporal weighting factors β and n_b:** Figure 3.6 shows the effect of the hyper-parameters β and n_b used in incorporating temporal continuity (Equation 3.8). The best DER is obtained on the CH1 dataset using $n_b = 2$ and $\beta = 0.95$. The inclusion of temporal continuity discourages frequent speaker turns. Therefore, it is more useful for recordings that are longer in duration and for those with less frequent speaker turns. We found the best parameters on the CH1 subset and used it for both the CH and AMI experiments.

3.3.4 Triplet sampling strategy

We explore different sampling strategies:

1. **Hard Sampling**: We sample positive pairs from the same cluster and negative from the nearest neighbor of the anchor belonging to a different cluster.

2. **Random Sampling**: Here, we sample positive pairs from the same cluster and randomly sample negative pairs from any other cluster.

3. **Easy Sampling**: In this case, we sample positive pairs from the same cluster and the negative pair using the farthest data point from the anchor that belongs to a different cluster. As the clusters are unsupervised, easy sampling represents a safe approach for triplet mining.

The DER performance for each sampling strategy on subset CH1 is compared. Hard, random, and easy sampling techniques give DER of 9.4%, 6.8%, and 8.0%, respectively. Although hard sampling is typically preferred in triplet mining on supervised data, we find that random sampling works best in our case of unsupervised clusters. The hard sampling may potentially bias the model to disallow the merge of the same speaker clusters. However, in case of easy sampling, the SSC model learning is somewhat compromised as the negative samples do not provide a discriminative learning criterion.

3.3.5 SSC Initialization

The initialization of the DNN model uses the whitening transform and the recording level PCA from the baseline system for the first and second layers, respectively. The output of the model are the embeddings $\boldsymbol{Y} = \{\boldsymbol{y}_1, .., \boldsymbol{y}_{N_r}\}$ which are used to perform clustering (AHC/PIC) till N^0 number of clusters. We use AHC clustering with threshold= 0.0 for CH to generate N^0 cluster labels for each $\boldsymbol{y}_k$. For the AMI SSC-AHC training, we used PLDA scoring with learnable PLDA parameters. We use PLDA-AHC with threshold= 0.0 to generate initial speaker labels. For SSC-PIC, we perform PIC clustering using either $N^0 = N^*$ in the known N^* case or we generate N^0 using a stopping threshold $\phi = 0.7$ on the eigenvalue ratio (Algorithm 3) in the unknown N^* case.

Table 3.2: DER (%) for different systems when the number of speakers (N^*) is known and unknown for the full CH dataset.

System	Known N*	Unknown N*
x-vec. + cosine + AHC	8.9	10.0
x-vec. + cosine + Spec. Clus.	9.4	11.9
x-vec. + PLDA + AHC (Baseline)	7.0	8.0
x-vec. + cosine + PIC	7.7	9.3
Self-supervised AHC (SSC-AHC)	6.4	8.3
Self-supervised PIC (SSC-PIC)	6.4	7.5
+ Temporal continuity	**6.3**	**7.0**

3.3.6 Stopping criterion

In the SSC training, the stopping criterion for the DNN representation learning model is based on the triplet-loss. We use the stopping threshold of 0.5 times the triplet-loss at the initial epoch of the model training. The stopping criterion for the SSC merging process is done using the eigenvalue ratio in the affinity matrix (section 3.2.5). The exact threshold is selected based on the development data (similar to the AHC threshold estimation).

3.4 Results and Analysis

The performance metric in all the experiments used in this work is the diarization error rate (DER) computed with a 250 ms collar and ignores overlapping segments. For all our experiments, we use the oracle speech activity decisions.

3.4.1 CH Diarization results

The diarization results on the CH dataset are shown in Table 3.2. Here, the initial set of experiments uses x-vector embeddings (with post-processing) clustered with different algorithms (AHC/PIC/Spectral Clustering) and similarity measures (cosine/PLDA). Spectral clustering is a graph partitioning algorithm that uses eigen values and eigen vectors of the Laplacian matrix to perform clustering [113]. Further details are given in chapter 2.

The PLDA-AHC approach gives the best performance for x-vector embeddings and this

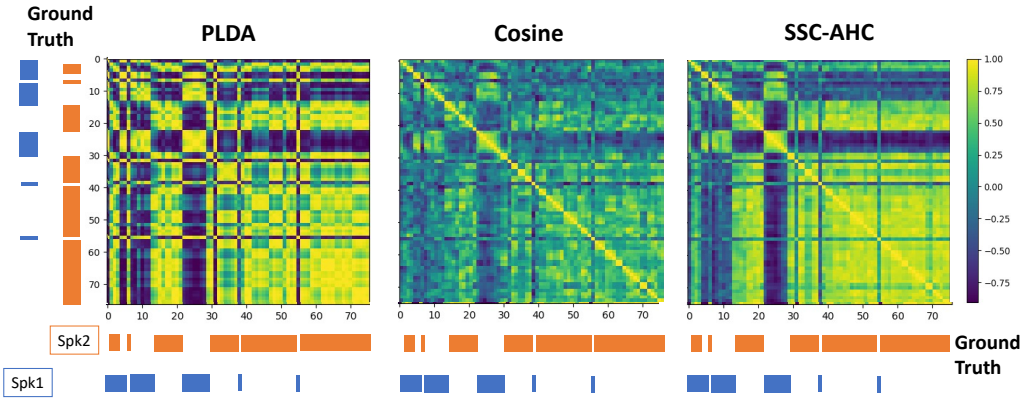

Figure 3.7: Affinity matrices using PLDA, cosine, and self-supervised AHC model with cosine for a two speaker recording from CH dataset. Ground truth labels are plotted across time on both sides of affinity matrices for comparison.

is used as the baseline system for comparison with the proposed SSC algorithm. For known N^*, Self-Supervised PIC (SSC-PIC) shows marginal improvement compared to SSC-AHC. However, for unknown N^*, we obtain considerable improvements for the SSC-PIC algorithm. The best results improve significantly over the baseline system for both the cases of known and unknown numbers of speakers. The relative improvements in known N^* and unknown N^* respectively for the SSC algorithm over the baseline system are 10% and 13%. We also performed paired student-t test to check the statistical significance of file-level DER improvements obtained for the SSC-PIC over the baseline system in Table 3.2. In this statistical t-test, comparing the baseline and SSC-PIC system, we find a t-statistic of $t = 2.2$ and p-value of $p = 0.03$. This indicates that the improvements seen in the proposed approach are statistically significant. In order to show the improvement in per-cluster variance, we computed the F-ratio based on the similarity scores. The F-ratio is a statistical measure that is used to compare the variances of two or more clusters or groups. It is a ratio of between-cluster variance to within-cluster variance. Hence it is always greater than or equal to 1. A larger F-ratio indicates that the variance between clusters is larger than the variance within clusters. This suggests that the clusters are different from each other in some way. We found the F-ratio (computed separately for each recording) to improve by 130% on an average on the CH1 dataset with the SSC training.

Table 3.3: DER (%) for different systems when number of speakers (N^*) is known and unknown for the AMI dataset.

System	Known N^*		Unknown N^*	
	Dev.	Eval.	Dev.	Eval.
x-vec + cosine + AHC	34.6	30.2	18.2	15.5
x-vec + cosine + Spec. Clus.	30.2	25.5	40.0	31.1
x-vec + PLDA + AHC (Baseline)	15.7	16.0	13.7	16.3
SSC-PLDA-AHC	9.4	11.1	10.7	11.6
x-vec + PLDA + PIC	9.4	9.3	9.8	10.4
x-vec + cosine + PIC	8.9	7.3	9.0	7.3
SSC-PIC	7.3	7.2	8.1	7.6
+ Temporal continuity	**6.2**	**6.4**	**6.4**	**6.7**

Figure 3.7 shows the PLDA and cosine affinity matrices for a 2-speaker recording from the CALLHOME dataset. Rightmost plot shows affinity matrix using SSC-AHC. The figure shows that representations have become more discriminative in the cosine space after self-supervised training. The affinity matrix is more smooth compared to PLDA affinity matrix.

3.4.2 AMI Diarization results

The diarization results on the AMI dataset are shown in Table 3.3. Here, the initial set of experiments reports using x-vector embeddings (with post-processing) clustered with different algorithms (AHC/PIC/Spec. Clus.) and similarity measures (cosine/PLDA). As the PLDA-AHC performance is significantly better than cosine-AHC, we performed SSC-AHC with PLDA scoring. Our SSC-PLDA-AHC system outperforms the baseline for all scenarios. However, the PIC algorithm achieves the best results compared to other clustering results on AMI. PIC algorithm can be used with PLDA and cosine scoring, but the performance is comparatively better with cosine because the cosine scores are bounded, which leads to uniformity while clustering. The cosine scoring with PIC gives 43% and 54% relative improvements for dev and eval, respectively for known N^* over the baseline system. The SSC algorithm performed with PIC provides additional gains. The incorporation of temporal continuity also improves

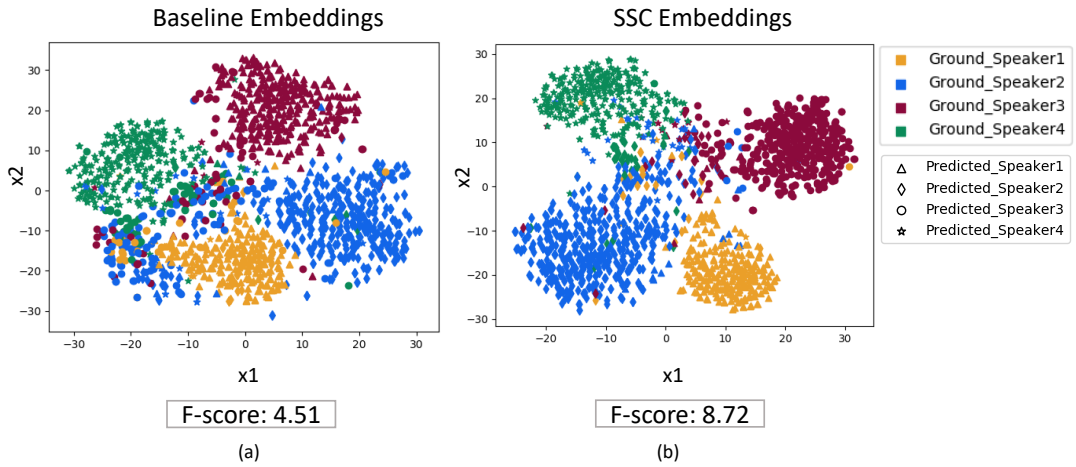

Figure 3.8: t-SNE based visualization of embeddings extracted on 1.5s audio segments from the recording AMI-ES2011c-SDM. (a) the baseline x-vectors post-processed with whitening transformation and PCA, (b) embeddings obtained after the SSC algorithm (DNN embeddings Y). The colors represent speakers in the ground truth, and the shapes represent predicted clusters.

results. We obtain significant relative improvements (60% for dev and eval) for known N^*. For unknown N^*, we get a relative improvement of 53% and 59% for dev and eval, respectively, over the baseline system.

Figure 3.8 shows the visualization of the embeddings from the baseline system (x-vec. + AHC) and the proposed SSC model for one recording from the AMI dataset. The visualization is performed using t-distributed stochastic neighborhood embedding (t-SNE), which performs an unsupervised dimensionality reduction of the embeddings [114]. The embeddings from the baseline system are shown in Figure 3.8(a), where the x-vectors are extracted from 1.5s segments of audio with half overlap followed by a whitening transform and PCA dimensionality reduction at the recording level. The embeddings from the SSC algorithm are shown in Figure 3.8(b) for the same recordings. The colors and shapes indicate the ground-truth and predicted speaker clusters, respectively. The ideal embeddings would have one-to-one mapping between shapes and colors. For understanding the importance of representations/embeddings in separating different clusters, the Fisher score or F-score is used here. It is a ratio of between-class variance and within-class variance computed using the embeddings. As seen in this Figure, the SSC algorithm provides a higher F-score (improved cluster separability) and improved association with the ground-truth speakers compared to the baseline.

Table 3.4: Comparison of the proposed SSC algorithm with other published works on the AMI dataset (after removing recordings from the TNO domain as reported in other works).

System	DER known N*		DER unknown N*	
	Dev.	Eval.	Dev.	Eval.
Semi-sup learning [115]	-	-	17.5	22.0
Incremental learning [116]	-	15.6	-	20.0
GAN clustering [117]	10.2	10.1	11.0	11.3
2-D self attention [118]	-	-	12.2	13.0
Baseline	14.4	16.5	12.9	13.6
SSC-PIC	**4.6**	**6.5**	**5.2**	**5.4**

Table 3.5: Comparison of the proposed SSC algorithm based results with other published works on the CH dataset. Here, VB refers to the use of VB-HMM based refinement.

System	DER Known N*	DER unknown N*
x-vec [50] (+VB)	-	12.8 (9.9)
VB modeling [72]	-	9.0
Bootstrap network [80]	6.2	8.6
LSTM scoring [70]	-	7.7
UIS-RNN [9]	-	7.6
Spec. Cluster [113]	-	7.3
Baseline (+VB)	7.0 (5.0)	8.0 (6.4)
SSC-PIC (+VB)	**6.3** (**4.8**)	**7.0** (**5.6**)

3.4.3 Comparison with other published works

Prior works on AMI report results on three domains (Edinburgh, Idiap, and Brno) out of the four domains. For comparison, we have also shown SSC-PIC results for these three domains in Table 3.4. As seen here, the proposed SSC algorithm significantly advances the state-of-the-art results for both the known/unknown N^* condition.

A similar comparison with other published results on the CH dataset is shown in Table 3.5. A much larger pool of published results exists on the CH dataset in recent years. Based on our survey of recent literature, the best reported results use the eigen-gap based spectral

clustering [113]. The SSC approach also improves over state-of-the-art in CH dataset results. However, improvements on the AMI dataset show more significant improvements compared to the CH dataset with PIC. One major reason is the difference in the duration of recordings between the datasets. In PIC, the graph is more stable when there are more nodes to build the edge connections. The duration of AMI recordings ranges from 20-60 mins which generates 1000-4000 embeddings. In the CH dataset, the recording duration ranges from 2 -5 mins generating 50-400 embeddings. Further, the SSC training also benefits from longer duration as we have more triplets ($> 200,000$) from AMI to train the network as opposed to CH dataset with $< 50,000$ triplets.

The SSC algorithm performance is also shown to be superior to the fully supervised diarization algorithms (for example, the RNN based algorithm [9]) which use a significantly large amount of speaker supervised conversational data. In contrast, the proposed SSC algorithm is purely unsupervised and developed without additional training data over the baseline system. Table 3.4 and 3.5 therefore highlight the advantages of self-supervised learning and graph structural clustering in the SSC algorithm. The segment-level SSC outputs can also be used as initialization for further refinement using frame-level VB-HMM modeling [72]. As seen in Table 3.5, this VB-HMM based refinement step further improves the diarization performance to achieve a DER of 4.8% for known N^* case and 5.6% for unknown N^* case on the CH dataset.

3.4.4 Computational complexity

The training of DNN and the PIC are two stages to be considered for time complexity at the recording level. Based on the triplet sampling strategy discussed in Section 3.2.3, the time complexity for training the DNN at each epoch is $\mathcal{O}(N_r^2)$. The time complexity for the clustering stage is based on the choice of algorithm. The baseline AHC has $\mathcal{O}(N_r^3)$ complexity. The proposed PIC algorithm is more efficient in computation as it calculates incremental path integral in linear time and does not require re-computation of similarity scores. It has a time complexity of $\mathcal{O}(N_r^2)$. The overall time complexity of SSC-PIC is $\mathcal{O}(RQN_r^2)$, where R is the number of DNN epochs and m is the number of SSC iterations performed. Comparing with

the baseline system complexity of $\mathcal{O}(N_r^3)$, the proposed SSC algorithm is more efficient when $N_r > RQ$. This condition is satisfied for recordings that are 30s or more in duration. We performed a compute-time measurement for the baseline (PLDA + AHC) and SSC-PIC system using x-vectors on an Intel CPU server (single core 64-bit). The compute time for the baseline and proposed SSC were 71.6s and 43.3s, respectively for a file of duration 26 minutes from the AMI dev set. This shows that the SSC algorithm achieves improved DER performance while reducing computational complexity.

3.5 Chapter Summary

In this chapter, we have proposed an algorithm for self-supervised embedding learning and clustering based on graph-structural path integral. The steps of embedding learning and agglomerative clustering (AHC/PIC) are performed iteratively for the given recording using x-vector features as inputs. The iterative steps validate the hypothesis that improved clustering can be achieved using discriminative representations while self-supervised representations can be learnt with well-separated clusters. The proposed approach, termed as self-supervised clustering (SSC), is applied for speaker diarization tasks in CH and AMI datasets. The diarization error rates achieved in these tasks are shown to improve the baseline system as well as several other recent state-of-the-art techniques proposed in the field. The computational complexity analysis and the visualization of the embeddings further illustrate the advantages of the SSC algorithm. To the best of our knowledge, the DER results reported in this work constitute the lowest error rates reported (till the date of publication) for diarization on CH and AMI datasets.

The proposed work involves use of a static cosine similarity metric to generate similarity scores for clustering. However, PLDA scoring performs better for diarization as shown in the literature. Therefore, we would like to incorporate PLDA scoring in the self-supervised model. In the next chapter, we perform metric learning along with representation learning using neural PLDA model to improve discriminability in similarity scores resulting in better diarization performance.

Chapter 4

Self-Supervised Metric learning

In this chapter, we propose a novel algorithm for speaker diarization using metric learning for graph-based clustering[1]*. The graph clustering algorithms use an adjacency matrix of similarity scores (discussed in chapter 3). These scores are computed between speaker embeddings extracted from pairs of audio segments within the given recording. We propose an approach that jointly learns the speaker embeddings and the similarity metric using principles of self-supervised learning. The metric learning network implements a neural model of the probabilistic linear discriminant analysis (PLDA). The self-supervision is derived from the pseudo labels obtained from a previous clustering iteration. The entire representation and metric learning model is trained with a binary cross entropy loss. By combining the self-supervision based metric learning along with the graph-based clustering algorithm, we achieve significant relative improvements of* 60% *and* 7% *over the x-vector PLDA agglomerative hierarchical clustering (AHC) approach on AMI and the DIHARD datasets respectively in terms of diarization error rates (DER).*

[1]This work is published in IEEE Automatic Speech Recognition and Understanding Workshop (ASRU '21) [119]

4.1 Introduction

The self-supervision can provide effective representations for downstream tasks without requiring the ground-truth labels [99] e.g., self-supervised clustering (SSC) for speaker diarization as discussed in section 3.2 of chapter 3. The SSC proposed representation learning and graph based clustering in an iterative self-supervised learning framework. The learned representations were used to derive a cosine similarity based adjacency matrix. This adjacency matrix, containing pairwise similarity scores, was used in path integral clustering for speaker diarization. But clustering based approaches that use PLDA similarity scores have been shown to outperform those that use cosine similarity [47] as described in chapter 2, section 2.1.2. However, PLDA is a generative model trained using statistical approaches like Expectation Maximization (EM). This means it is impossible to learn the parameters of a deep neural network using PLDA directly. A neural formulation of PLDA is required to address this issue. Ramoji et al. [120] proposed a neural PLDA or NPLDA, which poses the likelihood ratio score as a discriminative similarity function. The learnable parameters of the score function are then optimized using a verification cost.

This chapter extends our previous works on self-supervised learning and graph-based clustering (SSC-PIC). In this work, we explore a learnable metric based on neural PLDA in the self-supervised learning framework inspired by Ramoji et al. [120]. The PLDA parameters and the neural network parameters for the embedding extraction are learnable. In particular, the embeddings and the adjacency matrix for graph based clustering are jointly learned. The entire model is trained using binary cross entropy loss (BCE). The advantage of the proposed approach over the previous work is the direct optimization of the adjacency matrix used in the graph based clustering. Using this joint learning, we show significant performance improvements over baseline systems and previous models based on self-supervised graph clustering methods (chapter 3). We call the proposed approach as self-supervised PLDA based metric learning with path integral clustering (SelfSup-PLDA-PIC).

4.2 Background

This section describes the pre-processing steps, the probabilistic linear discriminant analysis (PLDA) model, and the path integral clustering algorithm used in our approach.

4.2.1 Pre-processing steps

The x-vectors of each segment in a given recording, extracted using the TDNN network [26], are mean normalized and whitened. The whitened x-vector features are then processed using length normalization [109]. Further, a principal component analysis (PCA) based dimensionality reduction at the recording level is also applied [110].

4.2.2 PLDA score computation

The PLDA is a generative model that factorizes the input into the speaker and channel factors (as discussed in section 2.1.2 of chapter 2). The embeddings from the PLDA model are used to obtain the pair-wise similarity score matrix $\boldsymbol{S}$, which captures the similarity between embeddings in the speaker space. The similarity score between a pair of embeddings $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, denoted as $s(i, j)$, is based on the log-likelihood ratio score between same-speaker hypothesis $\mathcal{H}_s$ and different-speaker hypothesis $\mathcal{H}_d$. This can be computed using the PLDA model. We project the embeddings $\boldsymbol{x}$ into latent space using Equation (2.5) (chapter 2),

$$\boldsymbol{u} = \boldsymbol{A}^{-1}(\boldsymbol{x} - \boldsymbol{m}) = \boldsymbol{V}(\boldsymbol{x} - \boldsymbol{m}) = \boldsymbol{V}\boldsymbol{x} - \boldsymbol{b} \tag{4.1}$$

The similarity score can be computed as [58]:

$$s(i, j) = log\{\tfrac{p(u_i,u_j|\mathcal{H}_s)}{p(u_i,u_j|\mathcal{H}_d)}\} \tag{4.2}$$

$$\begin{aligned} s(i, j) \quad &= -\tfrac{1}{2}\textstyle\sum_{k=1}^{d} log(\boldsymbol{\Psi}[k] + \tfrac{1}{2}) + 2log(\boldsymbol{\Psi}[k] + 1) \\ &+ \tfrac{(\bar{\boldsymbol{u}}[k])^2}{\boldsymbol{\Psi}[k]+\frac{1}{2}} + \textstyle\sum_{l\in\{i,j\}} \left((\boldsymbol{u}_l[k] - \bar{\boldsymbol{u}}[k])^2 + \tfrac{(\boldsymbol{u}_l[k])^2}{\boldsymbol{\Psi}[k]+1}\right) \end{aligned} \tag{4.3}$$

where $\boldsymbol{\Psi}[k]$ is the diagonal element of the k-th row of Ψ, $\boldsymbol{u}_i[k]$ is the k-th dimension of $\boldsymbol{u}_i$ and $\bar{\boldsymbol{u}}[k] = \frac{\boldsymbol{u}_i[k]+\boldsymbol{u}_j[k]}{2}$

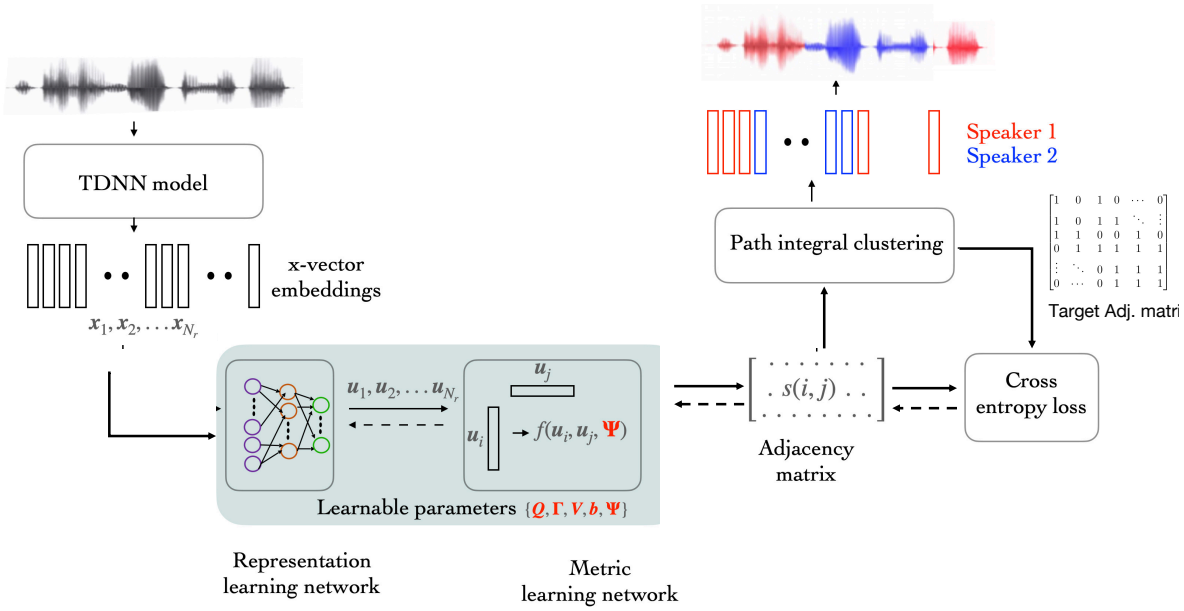

Figure 4.1: Block schematic of the proposed self-supervised metric learning approach to speaker diarization.

4.2.3 Path integral clustering

The path integral clustering (PIC) [104] is a graph-based agglomerative clustering algorithm, introduced in chapter 3, section 3.2.4 for speaker diarization. The PIC involves computation of path integral $\mathcal{S}_{\mathbb{C}_a}$ and conditional path integral $\mathcal{S}_{\mathbb{C}_a|\mathbb{C}_a\cup\mathbb{C}_b}$ for every cluster pair $\mathbb{C}_a$ and $\mathbb{C}_b$ at each step of merging as given in Equation (3.6) from chapter 3. The cluster affinity measure for the PIC algorithm is computed as,

$$\mathcal{A}\left(\mathbb{C}_a, \mathbb{C}_b\right) = \mathcal{S}_{\mathbb{C}_a|\mathbb{C}_a\cup\mathbb{C}_b} - \mathcal{S}_{\mathbb{C}_a} + \mathcal{S}_{\mathbb{C}_b|\mathbb{C}_a\cup\mathbb{C}_b} - \mathcal{S}_{\mathbb{C}_b} \tag{4.4}$$

where, $\mathcal{S}_{\mathbb{C}_a|\mathbb{C}_a\cup\mathbb{C}_b} - \mathcal{S}_{\mathbb{C}_a}$ is the incremental path integral of $\mathbb{C}_a$. It represents the sum of weighted paths between $\mathbb{C}_a$ and $\mathbb{C}_b$ such that the starting and ending vertices are in $\mathbb{C}_a$. Thus, higher affinity shows denser connections between clusters. The clusters with maximum affinity are merged at each time step. The pseudocode is given in Algorithm 2 (chapter 3).

4.3 Proposed Approach

The block schematic of the proposed self-supervised metric learning with graph based clustering algorithm (SelfSup-PLDA-PIC) is given in Figure 4.1. The model consists of a representation learning network and a metric learning network. The x-vectors extracted from short overlapping audio segments are used as inputs to the model. The model generates an adjacency matrix which is used in PIC.

The SelfSup-PLDA-PIC[1] jointly performs representation learning and metric learning using the initial clustering results. The output of clustering generates speaker labels. These labels form the same speaker and different speaker scores (1/0) for the target adjacency matrix. The model training uses the binary cross entropy (BCE) loss using the target adjacency matrix. Since the model updates the parameters based on the unsupervised clustering labels, it is known as self-supervised training. The previous chapter 3 discussed self-supervised representation learning using fixed cosine similarity scoring whereas, here, we have introduced metric learning block inspired by PLDA scoring (Sec. 4.2.2). The similarity score is computed using Equation (4.2), where, along with embeddings $\boldsymbol{u}$, the PLDA parameters $\boldsymbol{\Psi}$ are also learned. Therefore, it is also referred to as neural PLDA [120]. The following sub-section discusses the model architecture, joint representation learning, and metric learning approach.

4.3.1 Model architecture and training

The model is trained in two steps: forward pass (pseudo target generation) and backward pass (model update).

Forward Pass: First, a sequence of x-vectors of a recording r, $X_r \in \mathcal{R}^{D \times N_r}$, are fed to the representation learning network. The *representation learning network* is a three-layer DNN with $\{D, d, d\}$ units where D is the x-vector dimension. Let $\{\boldsymbol{Q}, \boldsymbol{\Gamma}, \boldsymbol{V}\}$ denote the learnable weights of each layer respectively, and let $\boldsymbol{b}$ denote the bias of the last layer. The representation

[1]github code link: https://github.com/iiscleap/SelfSup_PLDA.git

learning network performs the following operations:

$$\boldsymbol{H}^{(1)} = \sigma(\boldsymbol{Q}X_r) \tag{4.5}$$

$$\boldsymbol{H}^{(2)} = \boldsymbol{\Gamma}\boldsymbol{H}^{(1)} \tag{4.6}$$

$$\boldsymbol{U} = \boldsymbol{V}\boldsymbol{H}^{(2)} + \boldsymbol{b} \tag{4.7}$$

Where σ is the length normalization activation function, $\boldsymbol{H}^{(1)}$ and $\boldsymbol{H}^{(2)}$ are the output of the first and second layers, respectively. The final output, $\boldsymbol{U} = \{\boldsymbol{u}_1, ..., \boldsymbol{u}_{N_r}\} \in \mathcal{R}^{N_r \times d}$, is a matrix containing d-dimensional latent embeddings passed to the metric learning network.

The *metric learning network* is a function $f(u_i, u_j, \boldsymbol{\Psi})$ which takes pair of latent embeddings u_i and u_j which performs pairwise neural scoring using learnable parameter $\boldsymbol{\Psi} \in \mathbb{R}^{d \times d}$. This function is inspired by the log-likelihood ratio obtained using the PLDA model as discussed in Sec. 4.2.2.

$$\begin{aligned} f(u_i, u_j, \boldsymbol{\Psi}) &= -\frac{1}{2}\sum_{k=1}^{d} log(\boldsymbol{\Psi}[k,k] + \frac{1}{2}) + 2log(\boldsymbol{\Psi}[k,k] + 1) \\ &\quad -\frac{1}{2}\sum_{k=1}^{d} \frac{(\bar{\boldsymbol{u}}[k])^2}{\boldsymbol{\Psi}[k,k] + \frac{1}{2}} + \sum_{l \in \{i,j\}} \left((\boldsymbol{U}[l,k] - \bar{\boldsymbol{u}}[k])^2 + \frac{(\boldsymbol{U}[l,k])^2}{\boldsymbol{\Psi}[k,k] + 1} \right) \\ [\boldsymbol{W}]_{ij} &= sigmoid(f(u_i, u_j, \boldsymbol{\Psi})) \end{aligned} \tag{4.8}$$

where $sigmoid(.)$ is the sigmoid activation function, the adjacency matrix $\boldsymbol{W} \in \mathcal{R}^{N_r \times N_r}$ is the output of the metric learning network, which is used to perform graph based *path integral clustering* (section 4.2.3).

Backward Pass: In the backward pass, the target adjacency matrix is computed using the clustering solution from the PIC step. The target adjacency matrix $\boldsymbol{T} \in \mathcal{R}^{N_r \times N_r}$ is a binary matrix such that,

$[\boldsymbol{T}]_{ij} = 1$ if, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in same cluster,

$[\boldsymbol{T}]_{ij} = 0$ otherwise.

Using the target $\boldsymbol{T}$ and model output $\boldsymbol{W}$, a binary cross entropy (BCE) based loss function is

used to update the learnable parameters $\{\boldsymbol{Q}, \boldsymbol{\Gamma}, \boldsymbol{V}, \boldsymbol{b}, \boldsymbol{\Psi}\}$ as follows:

$$L_{ij} = [\boldsymbol{T}]_{ij} \log[\boldsymbol{W}]_{ij} + (1 - [\boldsymbol{T}]_{ij}) \log(1 - [\boldsymbol{W}]_{ij}) \tag{4.9}$$

$$L = \frac{1}{N_r^2} \sum_{i,j \in \{1,\ldots,N_r\}} L_{ij} \tag{4.10}$$

4.3.2 Model evaluation

After the model is trained and the model parameters $\{\boldsymbol{Q}, \boldsymbol{\Gamma}, \boldsymbol{V}, \boldsymbol{\Psi}\}$ are updated, forward pass is performed to generate the final clustering results.

4.4 Experiments

4.4.1 Evaluation data

The performance of the SelfSup-PLDA-PIC is evaluated on the AMI and the DIHARD III datasets. For experiments, we use the single distant microphone (SDM) condition of the AMI dataset containing 18 and 16 recordings in development (dev) and evaluation (eval) sets, respectively. We also compare beamformed multi-distant microphone (MDM) recordings results with other published results. For the DIHARD III dataset, we use the development and evaluation sets comprising 254 and 259 recordings, respectively. More details of the datasets are discussed in chapter 2, section 2.4.

For the AMI dataset experiments, we use the diarization error rate (DER) metric with a 250ms collar and by ignoring the overlap regions (as is the common practice on the AMI dataset). For the DIHARD dataset experiments, we use the DER metric with the overlaps and without providing a collar region.

4.4.2 Baseline model

Our baseline model is based on DIHARD III baseline recipe [94]. It involves feature extraction followed by x-vector embedding extraction. The x-vectors are extracted using the extended-TDNN (ETDNN) [51, 52] network. For training the ETDNN model, we use 40D mel-filterbank features using a 25ms window with 10ms shift.

The 13-layer ETDNN model follows the architecture described in chapter 2, section 2.1.1. The ETDNN model is trained on the VoxCeleb1 [88] and VoxCeleb2 [89] datasets for speaker identification task, to discriminate among the 7,146 speakers. The pre-processing steps mentioned in section 4.2.1 are applied to x-vectors, including whitening transform obtained from the DIHARD development set, length normalization, and recording level PCA. We preserve 30 dimensions for the AMI dataset. For the DIHARD dataset, we choose the number of dimensions which preserves 30% of total variance.

The PLDA model is trained using 3s x-vectors extracted from the subset of Voxceleb1 and 2. These x-vectors are whitened using a PCA transform learned from DIHARD development set. We use the same PLDA model for the DIHARD and AMI datasets. However, we extract the embeddings using a segment size of 1.5 s and a temporal shift of 0.75 s for the AMI dataset due to longer duration recordings while the segments of size 1.5 s are extracted with a shift of 0.25 s for the DIHARD dataset.

4.4.3 Model initialization

The initialization is a critical step for self-supervised training to generate reliable labels. We initialize our model parameters $\{\boldsymbol{Q}, \boldsymbol{\Gamma}\}$ using the whitening transform and the recording level PCA from the baseline system, respectively. The third layer's weight and bias $\{\boldsymbol{V}, \boldsymbol{b}\}$ of representation learning network are initialized with diagonalizing transform $\boldsymbol{V}$ and bias $\boldsymbol{b}$ from PLDA Equation (4.1), obtained after applying recording level PCA transform. Similarly, we initialize metric learning parameter $\boldsymbol{\Psi}$ using PLDA between-class covariance matrix defined in Equation (2.4) (chapter 3).

We perform initial clustering (AHC/PIC) till the initial N^0 number of clusters. The value of N^0 is based on a threshold applied to the similarity scores for the AHC system. With the PIC, we use the stopping threshold applied on eigenvalues of PIC affinity matrix (Equation (4.4)) to estimate N^0. We select a threshold higher than the optimal threshold to avoid over-clustering.

For the AMI dataset, the AHC threshold is set as $th = 0.0$ to obtain N^0 for SelfSup-PLDA-AHC training. For the SelfSup-PLDA-PIC system, the eigenvalues based threshold is set at $th = 0.7$ for initial clustering in the self-supervised training. For the DIHARD dataset, the

Table 4.1: DER (%) using ETDNN and ResNet x-vectors on the AMI dataset.

System	ETDNN				ResNet	
	Known N^*		**Unknown N^***		**Unknown N^***	
	Dev.	Eval.	Dev.	Eval.	Dev.	Eval.
x-vec + PLDA + AHC	15.9	12.2	13.1	12.3	-	-
x-vec + PLDA + PIC	5.1	10.2	5.8	11.4	6.0	6.2
SSC-Cosine-PIC	5.3	6.2	6.5	8.4	-	-
SelfSup-PLDA-AHC	7.9	7.3	7.7	9.4	-	-
SelfSup-PLDA-PIC	4.2	6.2	**4.4**	6.9	4.6	6.0
+ Temporal continuity	**4.2**	**4.2**	**4.4**	**4.9**	**4.4**	**4.3**
SelfSup-PLDA-PIC + VBx [121]	-	-	**2.9**	**4.2**	**3.4**	4.5

threshold for the SelfSup-PLDA-AHC system is set at $th = -0.7$. Since the DIHARD dataset has a huge variation in the number of speakers (1-10), the number of speakers estimated by AHC is used as N^0 in SelfSup-PLDA-PIC experiments.

4.4.4 Choice of hyper-parameters

The hyper-parameters in our approach are selected based on the performance of the development set of both datasets. The nearest neighbor K and scaling factor σ are the hyper-parameters for PIC. The values of $K = 30$ and $\sigma = 0.1$ are the best values for the AMI dataset. Similarly, for the DIHARD dataset, we found the best values of $K = 40$ and $\sigma = 0.5$. After model training, a temporal continuity of similarity scores can be incorporated (section 3.2.6) with an exponential decay given by,

$$s'(i,j) = s(i,j)\beta^{min(n_b,|i-j|)} \tag{4.11}$$

where β is a positive decay factor < 1, $|i-j|$ is the absolute segment index difference value of embeddings from the ith and jth segment, and n_b is a scalar. We use n_b and β as 2, and 0.95 respectively for both datasets.

4.5 Results and Analysis

4.5.1 AMI dataset

The results for various system configurations with the AMI development and evaluation datasets are reported in Table 4.1. These experiments use the ETDNN based x-vectors and residual network (ResNet101) based x-vectors [27]. For ETDNN x-vectors, we consider two cases in evaluation, with the known number of speakers N^* and with the unknown number of speakers. The baseline system is the x-vector with PLDA scoring and AHC. The use of graph based clustering with PIC improves the baseline system significantly. The self-supervised clustering (SSC) with cosine based affinity matrix proposed previously further improves over the PIC based system. The joint metric learning with the representation learning proposed in this chapter, denoted as SelfSup-PLDA, is shown to provide significant improvements over the previously proposed SSC-Cosine model. The relative DER improvement over the baseline system for the SelfSup-PLDA with temporal continuity is 66% and 60% for the AMI development and evaluation datasets for the condition with the unknown number of speakers. Further, the application of VBx re-segmentation [121] on the outputs from the SelfSup-PLDA-PIC system provides final DER values of 2.9% and 4.2% on the development and evaluation datasets. Our VBx setup is based on baseline ETDNN x-vectors with PLDA adapted from the DIHARD dev set. To the best of our knowledge, these results on the SDM data of the AMI corpus constitute the lowest DER reported in the literature.

The results using residual network (ResNet101) based x-vector embeddings [27] are also shown in Table 3.3 for the case with the unknown number of speakers. We use the ResNet 101 architecture explained in detail in chapter 2, section 2.1.1 for these experiments. The training data and the cost function used to train the ResNet model are similar to the ETDNN framework.

Comparing the system with PLDA scoring and PIC for the ETDNN x-vectors and the ResNet x-vectors, we find that the ResNet x-vectors improve significantly over the ETDNN based x-vectors. However, both models contain a similar number of parameters ($\sim 10^6$).

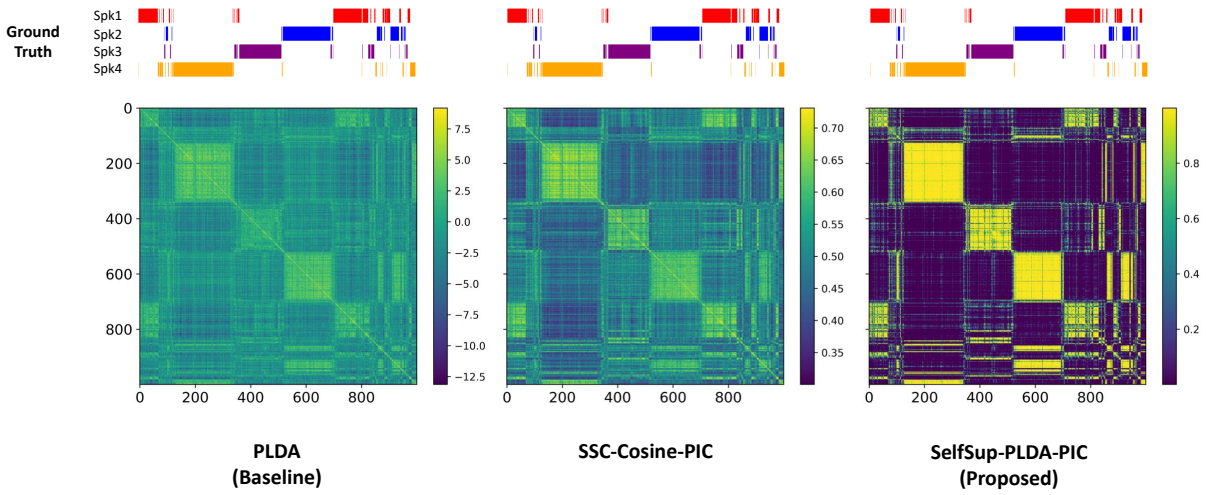

Figure 4.2: Similarity score matrices using PLDA, SSC-Cosine-PIC and SelfSup-PLDA-PIC (proposed) for a 4-speaker recording from AMI development set. The ground truth labels are plotted across time on top of affinity matrices for comparison.

The performance on the evaluation data for this system is 6.2% DER. Further, even with this improved baseline model, the proposed approach of SelfSup-PLDA with graph based clustering and the incorporation of the temporal continuity yields significant improvements. For the self-supervised metric learning, it is seen that the final results from either of the x-vector models achieve similar DER results on the AMI evaluation dataset.

Adjacency matrix analysis

The similarity score matrix (adjacency matrix used in graph clustering) for the baseline x-vector PLDA system (left), self-supervised representation learning with cosine scoring (SSC-PIC) (middle), and the proposed self-supervised metric learning (right) are shown in Figure 4.2. The adjacency score matrix used in the proposed approach is processed with sigmoid non-linearity for training with BCE loss. The ground truth speaker activity for the four speakers in this recording is also shown in this figure. The same speaker regions of the similarity matrix with the baseline x-vector PLDA system are not well pronounced. The self-supervised embedding learning with cosine scoring improves the contrast between the scores from the same speaker and across speaker segments. The proposed metric learning approach with self-supervised principles best contrasts the scores from the same speaker and across speaker regions of the

Table 4.2: DER (%) on the MDM recordings of AMI dataset (without TNO recordings).

System	Unknown N^*	
	Dev.	Eval.
x-vec(ResNet101)+AHC+VBx [27]	2.78	3.09
ECAPA-TDNN [56]	3.66	**3.01**
x-vec(ETDNN)+ SelfSup-PLDA-PIC	5.38	4.63
– + VBx [121]	**2.18**	3.27

given audio recording. This increase, in contrast, partly explains the improved DER results observed in Table 4.1 for the proposed SelfSup-PLDA+PIC model.

Comparison with prior literature

We attempt to compare the recent works proposed in Landini et al. [27] and Dawalatabad et al. [56] with the work proposed in this chapter. These previous works use beamformed audio from the AMI corpus (multi-distant microphone or MDM). In contrast, all the previous results reported in this work used the more challenging single-distant microphone (SDM) condition. We did not perform any adaptation or fine-tuning on the MDM data to make a direct comparison. Rather, the same model used for the SDM evaluations performs diarization on the MDM recordings. Further, keeping in line with the prior works, we have omitted the TNO recordings in the development and evaluation set in these results. The comparative analysis is shown in Table 4.2. Adding VBx-based re-segmentation to the proposed approach provides the best performance on the AMI development set compared to the prior works, with a final DER of 2.18%. Further, the result on the evaluation set (DER of 3.27%) is slightly inferior to the best-reported result of 3.01%. This analysis highlights that, even without fine-tuning or adapting the self-supervised model parameters to the MDM condition, the techniques reported in this chapter can match the best state-of-the-art results for the beamformed audio recordings.

Table 4.3: DER (%) when number of speakers (N^*) are unknown for the DIHARD III dataset.

System	Unknown N^*	
	Dev.	Eval.
x-vec + PLDA + AHC [122]	19.7	19.5
– + VBx [121]	17.0	16.6
x-vec + PLDA + PIC	19.7	18.9
– + VBx [121]	**16.8**	**16.3**
SSC-Cosine-PIC	23.9	21.1
SelfSup-PLDA-AHC	**18.9**	**18.2**
SelfSup-PLDA-PIC	19.2	**18.2**
– + VBx [121]	17.5	17.2

4.5.2 DIHARD dataset

The results on the DIHARD III dataset are reported in Table 4.3. The PIC approach improves slightly over the AHC approach in the baseline system [122]. The self-supervised learning approach with cosine similarity (SSC-PIC) degraded the performance over the baseline system. This was analyzed partly due to the reduced duration of the recordings, the large number of speakers within the given recording, and the lack of robustness in the simple cosine similarity scoring. The proposed approach of SelfSup-PLDA improves over both the AHC and PIC systems, respectively. Without the VBx re-segmentation, the best results for the SelfSup-PLDA-AHC model are achieved. However, the VBx re-segmentation did not improve over the re-segmentation applied to the baseline model.

As seen here, the improvements in the DIHARD dataset are less significant than those observed in the AMI dataset. The primary reason for this reduction in improvement is the shorter duration files (0.5-10min duration) in the DIHARD dataset compared to the 20-60min duration of the AMI recordings. The self-supervised metric learning approaches proposed in this work rely on recording level labels to improve the adjacency matrix used in the graph based clustering. With a reduced number of embeddings, the training of the SelfSup-PLDA model is compromised. Secondly, the DIHARD datasets have diverse domains, with some

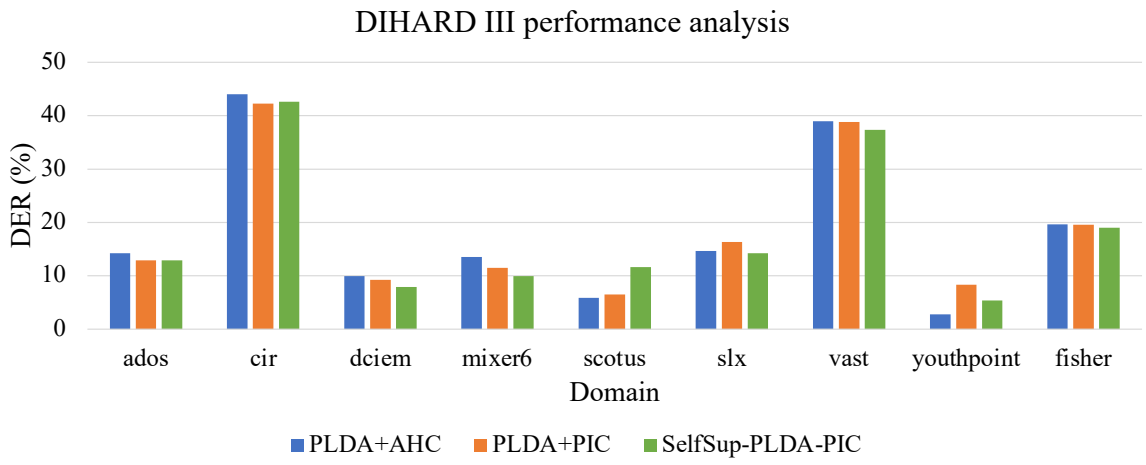

Figure 4.3: The plot shows the average DER (%) for different domains present in DIHARD III dataset.

Table 4.4: Average DER (%) on the DIHARD dataset for recordings with ≤ 7 speakers and > 7 speakers

System	≤ 7 **speakers**		> 7 **speakers**	
	Dev.	Eval.	Dev.	Eval.
x-vec + PLDA + AHC	18.0	19.3	36.6	27.1
x-vec + PLDA + PIC	17.7	17.8	**36.5**	**24.0**
SelfSup-PLDA-PIC	**17.0**	**17.2**	39.5	28.1

domains having a large number of speakers (more than 7 speakers) in the given recording.

Figure 4.3 shows the plot of performance comparison of PLDA-AHC, PLDA-PIC, and SelfSup-PLDA-PIC across nine domains of the DIHARD III dataset [122]. These include ados - clinical interviews, cir - restaurant conversations, dciem - map reading task, mixer6 - sociolinguistic interviews (lab), scotus - courtroom recordings, slx - sociolinguistic interviews (field), vast - youtube videos, youthpoint - broadcast interview, fisher - CTS (telephone recording). It can be observed that the performance of the proposed SelfSup-PLDA-PIC is better than the other two approaches except for the "scotus" domain. The reason is that this domain contains courtroom recordings that have more than seven speakers. The large number of speakers decreases the quality of the pseudo-cluster labels used in self-supervised learning. To analyze the impact of the increased number of speakers, we split the results reported in Table 4.3 into two conditions - recordings having less than or equal to 7 speakers

and those with more than 7 speakers. This analysis is reported in Table 4.4. As seen here, the model of self-supervised metric learning provides consistent performance improvements for the recordings having less than 8 speakers. On the other hand, for recordings with more than 7 speakers, the self-supervised metric learning results in performance degradation. As hypothesized earlier, the degradation is attributed to the errors in the pseudo-labels used in self-supervised learning.

4.6 Chapter Summary

We have proposed an approach to perform metric learning and clustering jointly for diarization. The metric learning is performed in a self-supervised manner by updating the neural PLDA model using cluster identities provided by graph based path integral clustering. Using an iterative metric learning and clustering procedure, we show that the proposed algorithm provides improved similarity scores and precise speaker clusters. With challenging diarization datasets, we have illustrated the performance improvements obtained using the proposed approach. In particular, the self-supervised metric learning algorithm provides the best results reported thus far for the AMI single-distant microphone conditions. With the more challenging DIHARD dataset evaluations, the proposed approach did not show improvements when the number of speakers in the given recording was greater than 7. However, for the recordings with less than 8 speakers, the model showed consistent performance improvements over the baseline systems.

The SelfSup-PLDA-PIC model fails to perform better for recordings with higher number of speakers because of erroneous initial clustering. To mitigate this issue, we would explore the supervised graph clustering approach using graph neural networks in the next chapter. This will remove the dependency of the model on the unsupervised clustering and also allow the model to learn from labeled conversational data using clustering-based loss.

Chapter 5

Supervised Hierarchical Graph Clustering

In this chapter, we propose a novel Supervised HierArchical gRaph Clustering algorithm (SHARC)[1] *for speaker diarization, where we introduce a hierarchical structure using Graph Neural Network (GNN) to perform supervised clustering. The supervision allows the model to update the representations and directly improve the clustering performance, thus enabling a single-step approach for diarization. In the proposed work, the input segment embeddings are treated as nodes of a graph with the edge weights corresponding to the similarity scores between the nodes. We also propose an approach to jointly update the embedding extractor and the GNN model to perform end-to-end speaker diarization (E-SHARC). It takes front-end time-frequency features and similarity matrix as input. It jointly learns the embedding extractor and GNN module, which performs representation learning, metric learning, and clustering using a single network. During inference, hierarchical clustering uses node densities and edge existence probabilities to merge the segments until convergence. The diarization experiments illustrate that the proposed E-SHARC approach achieves* 53% *and* 44% *relative improvements over the baseline systems on AMI and Voxconverse datasets, respectively. Later, this joint training network is further used to predict overlapping speakers for each segment based on the neighborhood speakers.*

[1]This work is published in IEEE ICASSP 2023 [123]. IEEE Transaction on Audio, Speech and Language Processing draft is under preparation.

5.1 Introduction

The metric learning approaches, including the one discussed in the previous chapter (chapter 4), help improve the discriminability between the embeddings of different speakers but eventually rely on unsupervised clustering algorithms to generate the desired output, resulting in sub-optimal performance. Another limitation of self-supervised approaches is that they show performance degradation as the number of speakers increases in a recording (>= 7) because the initial clustering algorithm fails to capture all the speakers.

On the other hand, end-to-end neural diarization (EEND) models that aim to perform the entire diarization process in a single neural network have been actively explored. However, such models require a large amount of labeled data and hundreds of hours of training. This can be difficult and time-consuming to obtain, especially for languages and accents that are underrepresented in existing speaker diarization datasets. Additionally, they can be computationally expensive and require powerful hardware to run in real-time, limiting their practicality in some applications.

We propose a simple approach to speaker diarization, which is not data intensive and can handle a large number of speakers (more than 7) during training and evaluation. The approach is called as Supervised HierArchical gRaph Clustering algorithm (SHARC). Our work is inspired by Xing et al. [124], where a supervised learning approach to image clustering was proposed. The self-supervised learning approaches required an external clustering algorithm to train and test the model. However, in this work, we perform supervised representation learning and clustering jointly without requiring an external clustering algorithm. SHARC trains a graph neural network using speaker embeddings as the nodes and similarity scores as the edges with a clustering based loss. The same network performs hierarchical clustering at the inference time and predicts the final speaker labels. This approach is further extended to incorporate the learning of embedding extractor and the graph neural network (GNN) [125] called E-SHARC, which achieved state-of-the-art performance on AMI and Voxconverse datasets. The framework allows any deep learning based embedding extraction of windowed audio segments of recording, like ETDNN [51], FTDNN [126], ResNet or ECAPA-TDNN [56]

models.

Multiple speakers can often speak simultaneously in natural conversations, resulting in overlapped speech, e.g., news debates. Therefore, incorporating overlapping speech detection is essential to improve diarization performance for real-life recordings. The proposed work further tackles the problem of overlapping speech. Our approach can assign multiple speakers to a region based on overlapping speech. The second speaker is assigned based on edge prediction probabilities for overlapping regions. The major contributions of our work are as follows:

- Introducing supervised hierarchical clustering using Graph Neural Networks (GNN) for diarization.
- Developing an end-to-end diarization system using supervised graph neural network based clustering (E-SHARC).
- Introducing an overlap detection approach called E-SHARC-Overlap to assign multiple speakers for the same audio region.
- Evaluating the performance on three benchmark datasets and showed improvement over baseline.

5.2 Background

Speaker diarization can be formulated as a link prediction problem between speaker embeddings of segments from the same recording. This motivated using graph neural networks (GNN) for speaker diarization. The GNNs are a class of deep learning methods designed to perform inference on data described by graphs. GNNs are neural networks that can be directly applied to graphs and provide an easy way to do node-level, edge-level, and graph-level prediction tasks. The details of two variants of GNN are discussed below.

Graph Convolution Network (GCN): The Graph Convolution Network (GCN) [127] is the most common variant of GNN. GCN behaves similar to convolution neural networks [128] by incorporating local connections, shared weights, and multiple layers. However, CNNs

can only operate on regular Euclidean data like images (2D grids) and texts (1D sequences), a subset of graphs. GCN generalizes the convolution operations for all types of structured and unstructured graphs. It learns the features by inspecting neighboring nodes where the number of nodes connections varies, and nodes are unordered. Each layer of the model takes a set of features $\boldsymbol{X}$ and corresponding adjacency matrix $\mathbf{A}$ (chapter 2, section 2.1.3) as inputs to generate hidden representations $\boldsymbol{H}^{l+1}$ as output. $\mathbf{A}$ is a sparse matrix containing weights of edges in the graph. The following equation describes the forward propagation.

$$\boldsymbol{H}^{(l+1)} = f(\boldsymbol{H}^{(l)}, \mathbf{A}) \tag{5.1}$$

$$= \sigma(\tilde{\boldsymbol{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\boldsymbol{D}}^{-1/2}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \tag{5.2}$$

where, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbb{I}_N$ is the adjacency matrix of the undirected graph G with added self-connections. $\mathbb{I}_N$ is the identity matrix, $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ and $\boldsymbol{W}^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(.)$ denotes an activation function, such as the $ReLU(.) = max(0, .)$. $\boldsymbol{H}^{(l)} \in \mathcal{R}^{N \times D}$ is the matrix of activations in the l^{th} layer; $\boldsymbol{H}^{(0)} = \boldsymbol{X}$.

GraphSAGE: The GCN is inherently transductive and does not generalize to unseen nodes. The GraphSAGE [129], another variant of GNN, is a representation learning technique suitable for dynamic graphs. It can predict the embedding of a new node without requiring a re-training procedure. GraphSAGE learns aggregator functions that can induce the embedding of a new node given its features and neighborhood. First, a graph is constructed using the embeddings as the nodes. The edges are connected using the similarity scores between the embeddings by using the k-nearest neighbor of each node. Instead of training individual embeddings for each node, a function is learned that generates embeddings by sampling and aggregating features from a node's local neighborhood. The aggregate function outputs a single neighborhood embedding by taking a weighted average of each neighbor's embedding. These neighborhood embeddings are further concatenated with the existing embedding of the

node and passed through a neural network as follows: The updated equations are as follows:

$$\begin{aligned} h^l_{N(v)} &= \text{Aggregate}(h^{l-1}_u, \forall u \in N(v)) \\ h^l_v &= \sigma(\boldsymbol{W}^l[h^{l-1}_v; h^l_{N(v)}]) \end{aligned} \tag{5.3}$$

where, $N(v)$ is the neighborhood of node v, $h^l_{N(v)}$ and h^l_v are the latent representations of $N(v)$ and node v respectively, at layer $l \in \{1, ..., L\}$. The σ is the activation function, and $\boldsymbol{W}^l$ is the weight matrix. h^l_v is the hidden representation of node v.

5.3 Proposed approach

The Supervised HierArchical gRaph Clustering algorithm (SHARC)[1] model involves two important steps: embedding extraction and GNN-based clustering. In this work, we have explored ETDNN model based embedding extractor. The ETDNN model is also fine-tuned for the diarization task and it is referred as end to end SHARC or E-SHARC.

5.3.1 Notations

- $\mathbb{H}^m_r = \{\boldsymbol{h}^{(m)}_1, \boldsymbol{h}^{(m)}_2, ..., \boldsymbol{h}^{(m)}_{N^m_r}\} \in \mathcal{R}^{D' \times N^m_r}$ denotes the node features. N^m_r is the number of nodes at level m for recording r and $N^0_r = N_r$.

- $\boldsymbol{h}^{(m)}_i = [\tilde{\boldsymbol{h}}^{(m)}_i; \bar{\bar{\boldsymbol{h}}}_i^{(m)}]$ where $\tilde{\boldsymbol{h}}^{(m)}_i$ is the identity feature of node i at level m. It is the identity feature of level $m-1$ that has the highest node density in the cluster i. $\bar{\bar{\boldsymbol{h}}}_i^{(m)}$ is the average feature of level m, which is the average of all the identity features from the previous level $m-1$ (Equation 5.6). $\boldsymbol{h}^{(0)}_i = [\boldsymbol{x}_i; \boldsymbol{x}_i]$.

- $\boldsymbol{S}^m_r \in \mathbb{R}^{N^m_r \times N^m_r}$ denotes pairwise similarity score matrix at level m for recording r such that $[\boldsymbol{S}^m_r]_{ij} = s(\tilde{\boldsymbol{h}}^{(m)}_i, \tilde{\boldsymbol{h}}^{(m)}_j)$, similarity score between identity features of node i and j.

- p^m_{ij} and $\hat{p}^m_{ij}$ denote the ground truth and predicted edge probabilities between node i and j, respectively. at level m. P^m and $\hat{P}^m$ are the groundtruth and predicted edge sets, respectively.

[1]The implementation code is available at https://github.com/prachiisc/SHARC.git

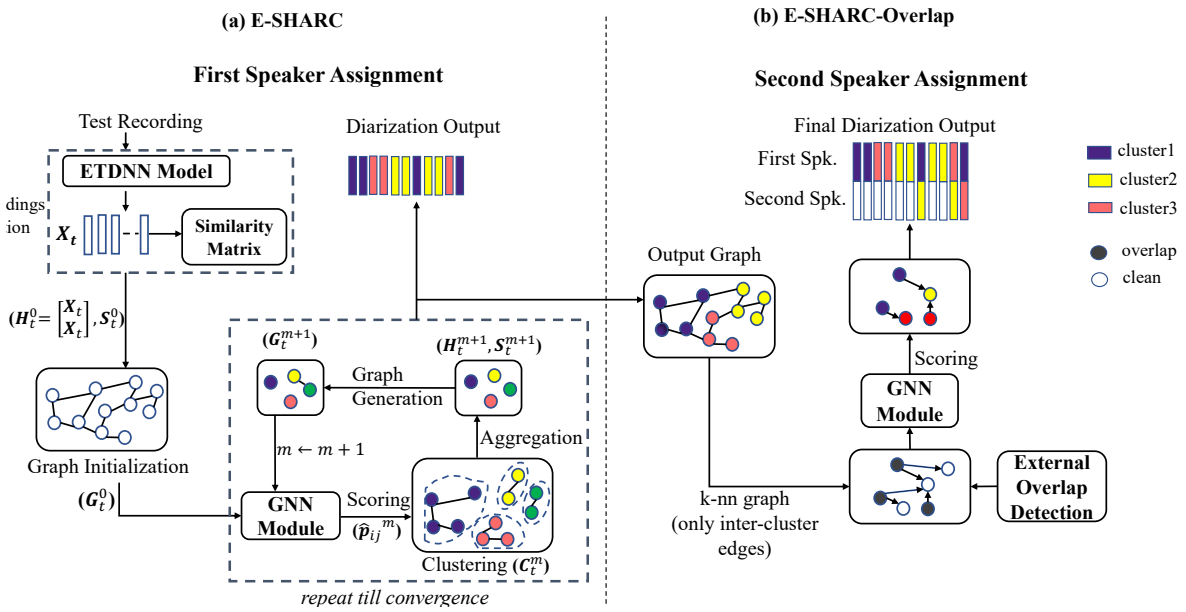

Figure 5.1: Block schematic of the E-SHARC and E-SHARC-overlap. (a) shows E-SHARC inference containing ETDNN and GNN module for the first speaker assignment. (b) shows E-SHARC-Overlap for the second speaker assignment approach using an external overlap detector and the GNN module.

- $\hat{e}_{ij}^m = 2\hat{p}_{ij}^m - 1 \in [-1, 1]$ denotes the edge coefficient or edge weight between node $i \in \{1, .., N_r^m\}$ and $j \in J_i^k$, where J_i^k represents set of k-nn (k-nearest neighbor) of i.

- Hyper-parameters:

  k - Number of nearest-neighbors

  τ - threshold on edge probabilities to stop merging.

5.3.2 Graph initialization

For both training and inference steps of the E-SHARC framework, the first step is the creation of a graph based on the input embeddings (x-vectors). The initial graph, termed as graph at level $m = 0$, $G_t^0 = (V_t^0, E_t^0)$, contains embeddings as the nodes in V_t^0 and k-nearest neighbor of each node based on S_t^0 similarity score matrix form the edges in E_t^0. The pre-trained PLDA model is used to generate similarity scores.

5.3.3 Forward pass - SHARC algorithm

Figure 5.1 (a) shows the block diagram of the inference step. For a test recording t, x-vectors $\boldsymbol{X}_t$ are extracted to perform hierarchical clustering to obtain speaker labels for each segment.

Algorithm 4: SHARC Inference

Initialize: $M \leftarrow$ maximum no. of levels; $m \leftarrow 0$;
$k \leftarrow$ no. of nearest neighbours; $t \leftarrow$ recording id;
$\mathbb{H}_t^0 \leftarrow [\boldsymbol{X}_t; \boldsymbol{X}_t]$
Graph Initialization: $G_t^0 \leftarrow graph(\mathbb{H}_t^0, \boldsymbol{S}_t^0, k)$
while $m \leq M$ **do**

1. GNN Scoring: $\hat{P}_t^m \leftarrow \Phi(G_t^m, \mathbb{H}_t^m)$
2. $\mathbb{C}_t^m \leftarrow Clustering(\hat{P}_t^m)$
3. Aggregation: $\mathbb{H}_t^{m+1} \leftarrow \Psi(\mathbb{H}_t^m, \mathbb{C}_t^m)$
4. Grah generation: $G_t^{m+1} \leftarrow graph(\mathbb{H}_t^{m+1}, \boldsymbol{S}_t^{m+1}, k)$
5. **If** $(G_t^{m+1} \leftarrow \{\phi\}) : M = m + 1; break$
6. $m \leftarrow m + 1$

end
Output: Predicted $\hat{\boldsymbol{Z}}_{\boldsymbol{t}} = \{\hat{z}_1, .., \hat{z}_{N_t}\}$ using $\mathbb{C}_t^{\{1:m-1\}}$

The SHARC is performed using a GNN module which takes graph as input and generates edge prediction probabilities. The SHARC algorithm during inference is summarized in Algorithm 4. The details are given below.

1. **GNN scoring:** The GNN scoring function Φ is a learnable GNN module designed for supervised clustering. The module jointly predicts node densities and edge probabilities using the input embeddings at each level. Each graph G_t^m, containing source and destination node pairs, is fed to the GNN scoring model. The output of the model is edge probability $\hat{p}_{ij}^m$ of the nodes v_i^m and v_j^m $\forall i = 1, .., N_t^m, j \in J_i^k$. The edge probabilities are also used to compute node density which measures how densely the node is connected with the cluster. A node with higher density is a better representative of the cluster than a node with lower density. A node density using predicted edge coefficients is called pseudo density $\hat{d}_i^m$ for node v_i^m and is defined as:

$$\hat{d}_i^m = \frac{1}{k} \sum_{j=1}^{k} \hat{e}_{ij}^m \boldsymbol{S}_t^m(i, j) \tag{5.4}$$

The ground truth density d_i^m is obtained using ground truth edge coefficient $e_{ij}^m =$

$2p_{ij}^m - 1 \in \{-1, 1\}$, where $p_{ij}^m = 1$ if nodes v_i^m and v_j^m belong to the same cluster, otherwise $p_{ij}^m = 0$.

2. **Clustering:** Clustering is the process of grouping the nodes based on presence of edge connections. After GNN scoring, clustering is performed hierarchically using the edge probabilities p_{ij}^m and the estimated node densities. At each level of hierarchy m, it creates a candidate edge set $\varepsilon(i)^m$, for the node v_i^m, with edge connection threshold τ, as given below.

$$\varepsilon(i)^m = \{j | (v_i^m, v_j^m) \in E_t^m, \quad \hat{d}_i^m \leq \hat{d}_j^m \quad \text{and} \quad \hat{p}_{ij}^m \geq \tau\} \tag{5.5}$$

For any i, if $\varepsilon(i)^m \neq \varnothing$, pick $j = argmax_{j \in \varepsilon(i)^m} \hat{e}_{ij}^m$ and connect v_i^m and v_j^m. After a full pass over every node, a set of clusters $\mathbf{C}_t^m$ is formed based on connected components. $\mathbb{C}_n^m \in \mathbf{C}_t^m$ represents set of all connected components/nodes in $n-$th cluster $\forall n \in \{1, ..., n_c\}$.

3. **Feature aggregation:** The cluster nodes at level m are agregated to form node features of next level. To obtain node representations for next level $\mathbb{H}_t^{m+1}$, the clusters $\mathbf{C}_t^m$ and the features $\mathbb{H}_t^m$ are used to compute and concatenate identity feature $\tilde{\boldsymbol{h}}_i^{(m+1)}$ and average feature $\bar{\bar{\boldsymbol{h}}}_i^{(m+1)}$ of each cluster i as,

$$\begin{aligned} \tilde{\boldsymbol{h}}_i^{(m+1)} &= \tilde{\boldsymbol{h}}_{z_i}^{(m)}; \qquad \bar{\bar{\boldsymbol{h}}}_i^{(m+1)} = \frac{1}{|\mathbb{C}_i{}^m|} \sum_{j \in \mathbb{C}_i{}^m} \tilde{\boldsymbol{h}}_j^{(m)} \\ \boldsymbol{h}_i{}^{(m+1)} &= [\tilde{\boldsymbol{h}}_i^{(m+1)}; \bar{\bar{\boldsymbol{h}}}_i^{(m+1)}] \end{aligned} \tag{5.6}$$

where $z_i = argmax_{j \in \mathbb{C}_i{}^m} \hat{d}_j^{(m)}$.

4. **Graph generation:** A new graph G_t^{m+1} is constructed for the next level using the node features $\mathbb{H}_t^{(m+1)}$. The edges are formed using $\boldsymbol{S}_t^{m+1}$ which is computed using the identity features $\mathbb{H}_t^{(m+1)}$.

The algorithm repeats until convergence when there are no connected components in the graph, as shown in Figure 5.1 (a).

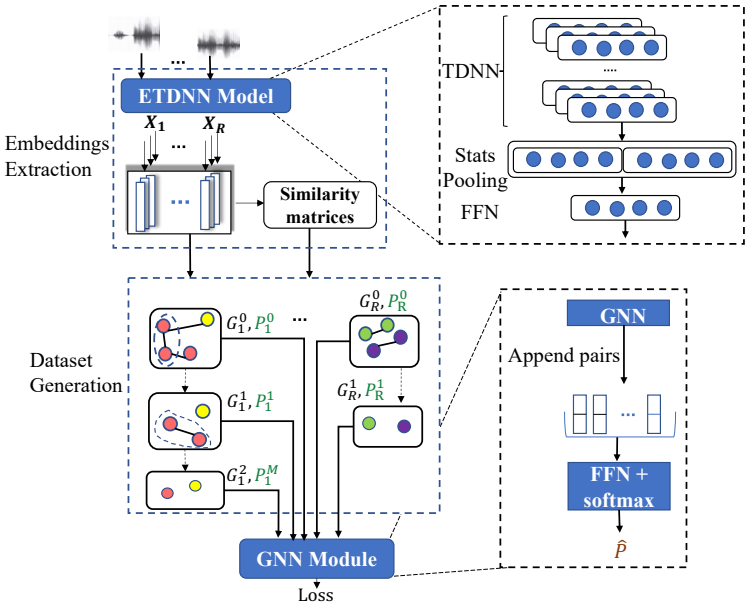

Figure 5.2: Block schematic of the E-SHARC training. The ETDNN and GNN Module contain learnable parameters. The GNN module generates edge prediction probabilities and edge weights which are used in loss computation. The blue colour represents learnable parameters.

5.3.4 Model training

E-SHARC training involves learning the embedding extractor and the GNN module (SHARC). During training, multiple graphs are constructed using input features and similarity matrix at different levels of clustering for each of the audio recordings in the training set. The loss is a combination of binary cross entropy loss and mean squared error loss. These losses across all graphs are then accumulated and backpropagated for each batch. The block diagram is shown in Figure 5.2. The steps involved in the E-SHARC training are as follows:

5.3.4.1 Dataset generation

For the training set, input graphs $G = \{G_1^0, G_2^0, .., G_1^1, .., G_R^{Mr}\}$ are constructed at different clustering levels for each recording r. M_r is the maximum number of levels created for r. $E = \{E_1^0, .., E_1^{M_1}, .., E_R^0, .., E_R^{M_R}\}$ and $V = \{V_1^0, .., V_1^{M_1}, .., V_R^0, .., V_R^{M_R}\}$ are the set of all possible edges and nodes, respectively. At level 0, all embeddings of a recording obtained from TDNN are considered as individual clusters. For each level, clustering is performed based on edge linkages, allowing only the same speaker embeddings to belong to a single cluster.

5.3.4.2 SHARC training

In this step, a pre-trained extended time delay neural network (ETDNN) [51] model for embedding extraction is used to extract x-vectors. These x-vectors are used to generate the training set graphs as described in the section 5.3.4.1. The weights of the ETDNN module are frozen, and only GNN module weights are learned.

- **GNN module architecture:** The model consists of one GNN layer with $D' = 2048$ units (neurons in a layer). It takes node representations $\mathbb{H}_r^m$ and their edge connections E_r^m as input and generate latent representations denoted as $\hat{\mathbb{H}}_r^{(m)} \in \mathcal{R}^{D' \times N_r^m}$. Each node v_i and it's groundtruth cluster (speaker) label z_i in the training set is used to learn the clustering criterion from the data. The pair of embeddings are concatenated $[\hat{\boldsymbol{h}}_i; \hat{\boldsymbol{h}}_j]$ and passed to a four-layer fully connected feed-forward network with a size of $\{2D', 1024, 1024, 2\}$ followed by softmax activation to generate edge probability p_{ij}. The architecture is selected based on validation experiments.

- **Loss function:** The GNN module is trained using the following loss function.

$$L = L_{conn} + L_{den} \tag{5.7}$$

where L_{conn} is the pairwise binary cross entropy loss based on edge probabilities across all the possible edges in E given as:

$$L_{conn} = -\frac{1}{|E|} \sum_{(v_i, v_j) \in E} l_{ij} \tag{5.8}$$

$$l_{ij} = \begin{cases} p_{ij} \log \hat{p}_{ij} + (1 - p_{ij}) \log(1 - \hat{p}_{ij}) & \text{if } d_i \leq d_j \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

Here, $|E|$ represents the total number of edges. L_{den} represents the neighborhood density average loss given by Equation 5.10. L_{den} represents mean squared error (MSE) loss

between ground truth node density d_i and predicted density $\hat{d}_i$,

$$L_{den} = \frac{1}{|V|} \sum_{i=1}^{|V|} ||d_i - \hat{d}_i||_2^2 \tag{5.10}$$

where $|V|$ is the cardinality of V.

5.3.4.3 Joint ETDNN and GNN training

In this step, the GNN module is first initialized using the SHARC model. The ETDNN model is also learned along with the GNN module.

ETDNN : The 13-layer ETDNN model [51, 52] follows the architecture described in section 2.1.1 of chapter 2 . The input to the model is 40-D mel-spectrogram features extracted from each segment (1-2s) of the training recording followed by cepstral mean normalization. The model is initialized with a pre-trained model for speaker classification. These x-vectors are used to generate the training set graphs as described in the section 5.3.4.1. The loss is back propagated till the TDNN layers.

5.4 Handling overlapped speech

We extend the E-SHARC model to also perform GNN-based overlap prediction called E-SHARC-Overlap. Our approach assumes that there are a maximum of two speakers present in an overlapping region. In order to accurately identify the speakers present in the overlapping region, we follow a two-pass approach. The first pass comprises E-SHARC modeling on the clean segments (single speaker segments). The first speaker, referred to as the parent cluster, is selected for each segment based on the first-pass E-SHARC algorithm. In the second pass, the proposed overlap model is trained using overlapping and clean segments. During training, the graph adjacency matrix is generated by connecting the nodes from one cluster to nodes from any other cluster except the parent cluster. This enables the overlap model to identify the second speaker present in the overlapping regions. A small percentage of intra-cluster connections (10%) are preserved randomly, enabling contrastive training. The k-nearest neighbors of each node are selected based on the final graph adjacency matrix. The training loss comprises of BCE and MSE losses, similar to the original SHARC model (Equation

5.7).

5.4.1 Initialization

The SHARC-Overlap model and the E-SHARC-Overlap model are initialized with the pre-trained SHARC and the E-SHARC models, respectively.

Initialization

The SHARC-Overlap model and the E-SHARC-Overlap model are initialized with the pre-trained SHARC and the E-SHARC models, respectively.

5.5 Inference

The inference steps comprises of first pass - clustering E-SHARC and second pass - overlap detection and second speaker assignment to generate the final diarization output for each recording as shown in figure 5.1.

5.5.1 First speaker assignment

First-pass clustering based on E-SHARC is performed to assign an initial parent cluster and to decide the number of speakers present in a recording as described in Section 2. The input x-vectors are the average of the pre-trained x-vectors and E-SHARC trained x-vectors.

5.5.2 Second/Overlap speaker assignment

The overlap speaker assignment or the second speaker assignment is done using the E-SHARC-Overlap model. The pyannote overlap detector [76, 75] is used to identify regions containing overlapped speech. Based on the E-SHARC cluster assignment, a new graph adjacency matrix is created after removing the within-cluster connections and outgoing connections of the single speaker nodes. Then k-nearest neighbors are selected from the remaining connections of each node. The E-SHARC-Overlap model computes the edge prediction probabilities. Then for each node, we select top $k'(k' \leq k)$ neighbors based on edge prediction probabilities. The dominant cluster identity of the neighbors is assigned as the second speaker of the corresponding node.

Table 5.1: Choice of hyper-parameters for train, dev, eval split of AMI and Voxconverse datasets. The parameters k^* and p_τ^* are used in E-SHARC training.

Parameters	AMI			Voxconverse		
	Train	Dev	Eval	Train	Dev	Eval
k	60	60	60	60	30	30
p_τ	-	0.0	0.0	-	0.5	0.8
k^*	30	50	50	60	30	30
p_τ^*	-	0.0	0.0	-	0.9	0.8

5.6 Experiments

5.6.1 Datasets

- **Testing**: The performance of SHARC, E-SHARC, and E-SHARC-Overlap is evaluated on the AMI [5] , the Voxconverse [95] and the DISPLACE [96, 130] datasets. The single distant microphone (SDM) condition of the AMI dataset comprising of development (dev) and evaluation (eval) sets is used for experiments. The Voxconverse dataset includes dev and eval sets extracted from YouTube videos. The DISPLACE dataset was part of DISPLACE challenge 2023 and comprises of code-mixed conversational recordings. The details of these datasets are provided in chapter 2, section 2.4.

- **Training**: The official training set of the AMI dataset, containing 75 hrs of labeled speech, is used for model evaluation on the AMI set. As Voxceonverse and DISPLACE do not have train sets, the dataset used for training the SHARC model for the these datasets is simulated using Voxceleb 1 and 2 [88, 89] and LibriSpeech [90] using the recipe from [78]. The recipe of mixture simulation is inspired by EEND speaker diarization [78]. It merges utterances from multiple speakers with variable silence intervals between utterances. We simulated 5000 mixtures containing 2-5 speakers with duration ranging from 150-440 s. This generates 1000 hrs of data with $6,023$ speakers. After simulating the recordings, using utterances from Voxceleb and LibriSpeech datasets, background noises and reverberations are added from MUSAN corpus [131] and Simulated Room

Impulse Response Database [92].

5.6.2 Baseline system

The baseline method is an x-vector-clustering approach followed in [132, 119]. The x-vectors are obtained from the ETDNN model to compute the PLDA similarity score matrix and perform clustering to generate speaker labels for each segment. The details of this approach are described in chapter 2, section 2.1.1. The ETDNN model and PLDA model for AMI dataset are trained using AMI train set whereas for Voxconverse dataset, Voxceleb 1,2 and LibriSpeech datasets are used. The PLDA model is trained using the x-vectors from 1.5s segment with 0.75s shift from the train sets. For comparison, we have used the two most popular clustering approaches - AHC [66] and spectral clustering (SC) [69]. To perform AHC, the PLDA similarity scores are used directly. For SC, we convert the PLDA scores s to $s' \in [0, 1]$ by applying sigmoid with temperature parameter $\tau = 0.1$ (best value obtained from experimentation) as: $s' = \frac{1}{1+exp(-s/\tau)}$.

5.6.3 Implementation details

Preprocessing steps: The graph adjacency matrix of each recording is obtained using a similarity matrix, allowing k-nearest neighbor (k-nn) for each row of the matrix. At each level of the hierarchy, the node identity features are passed to the PLDA model to generate the similarity scores matrix. The model first performs preprocessing on the features by applying mean subtraction, whitening transform obtained from held out set followed by length normalization. Then it applies recording-level PCA transform for dimensionality reduction to 30 and computes the log-likelihood score between pair of input speaker embeddings.

Model parameters: The SHARC model is trained with Stochastic Gradient Descent (SGD) optimizer with a learning rate lr=0.01 (for Voxconverse) and lr=0.001(for AMI) for 500 epochs. Similarly, the E-SHARC is also trained with an SGD optimizer. In this case, the learning rate is 1e-06 for the ETDNN model and 1e-03 for the GNN model, trained for 20 epochs. SHARC-Overlap is initialized with SHARC model weights and trained with lr=0.001 for 100

Table 5.2: DER (%) comparison on the AMI SDM and Voxconverse datasets with the baseline methods. SC: Spectral clustering, OVP: overlap, COL: collar.

AMI SDM System	**with OVP + no COL**		**w/out OVP + COL**	
	Dev.	Eval.	Dev.	Eval.
x-vec + PLDA + AHC [132]	24.50	29.51	7.61	14.59
x-vec + PLDA + SC	19.8	22.29	4.1	5.76
x-vec + PLDA + SHARC	**19.71**	21.44	**3.91**	4.88
E-SHARC	20.59	**19.83**	5.15	**2.89**
– + VBx [121]	**19.35**	**19.82**	**3.46**	**2.73**
Voxconverse System				
x-vec + PLDA + AHC [132]	12.68	13.41	7.82	9.28
x-vec + PLDA + SC	10.78	14.02	6.52	9.92
x-vec + PLDA + SHARC	10.25	13.29	6.06	9.40
E-SHARC	**9.90**	**11.68**	**5.68**	**7.65**
– + VBx [121]	**8.29**	**9.67**	**3.94**	**5.51**

Table 5.3: Performance comparison based on cluster purity and coverage for Voxconverse dataset. SC stands for Spectral Clustering.

Method	**Cluster Purity**	**Cluster Coverage**
Baseline with AHC	93.5	89.5
Baseline with SC	92.0	92.3
SHARC	93.0	92.4
E-SHARC	**93.0**	**92.9**

epochs. E-SHARC-Overlap is initialized with E-SHARC model weights.

Hyper-parameters: The hyper-parameters k, p_τ are selected based on the best performance on the dev set for the eval set and vice versa. The impact of different values for k, p_τ are discussed below. Table 5.1 shows the values of hyperparameters obtained for the AMI and Voxconverse datasets.

5.7 Results and Analysis

The proposed approaches are evaluated using the diarization error rate (DER) metric [132]. The DERs are computed for two cases. The first case considers overlaps and without collar

Table 5.4: DER (%) comparison on the AMI, Voxconverse, and DISPLACE datasets with the baseline methods considering overlaps and with no tolerance collar.

System	AMI Eval				Voxconverse Eval				DISPLACE Eval			
	FA	**Miss**	**Conf.**	**DER**	**FA**	**Miss**	**Conf.**	**DER**	**FA**	**Miss**	**Conf.**	**DER**
AHC	0.0	15.6	13.9	29.51	0.0	3.1	10.31	13.41	3.1	22.4	15.0	40.60
AHC+Overlap	0.7	11.6	14.37	26.67	0.8	1.6	9.65	12.05	3.8	21.0	15.67	40.47
SC	0.0	15.6	6.7	22.29	0.0	3.1	10.9	14.02	3.1	22.4	15.3	40.84
SC+Overlap	0.7	11.6	8.06	20.36	0.8	1.6	11.33	13.73	3.8	21.0	15.85	40.65
SHARC	0.0	15.6	5.7	21.27	0.0	3.1	10.2	13.29	3.1	22.4	7.47	33.07
SHARC-Overlap	0.7	11.6	7.2	19.50	0.8	1.6	10.16	12.56	3.8	21.0	7.9	32.73
E-SHARC	0.0	15.6	4.3	19.83	0.0	3.1	8.5	11.68	3.1	22.4	7.33	32.93
E-SHARC-Overlap	0.7	11.6	5.69	**17.99**	0.8	1.6	9.02	**11.42**	3.8	21.0	7.65	**32.45**

Table 5.5: DER (%) comparison on the AMI, Voxconverse, and DISPLACE datasets with the baseline methods after VBx resegmentation. considering overlaps and with no tolerance collar.

System	AMI Eval				Voxconverse Eval				DISPLACE Eval			
	FA	**Miss**	**Conf.**	**DER**	**FA**	**Miss**	**Conf.**	**DER**	**FA**	**Miss**	**Conf.**	**DER**
AHC+VBx	0.0	15.6	11.9	27.43	0.0	3.1	8.6	11.72	3.1	22.4	11.15	36.75
AHC+Overlap+VBx	0.7	11.6	11.63	23.93	0.8	1.6	8.45	10.85	3.8	21.0	12.26	37.06
SC+VBx	0.0	15.6	5.3	20.90	0.0	3.1	7.6	10.71	3.1	22.4	10.38	35.98
SC+Overlap+VBx	0.7	11.6	6.7	19.00	0.8	1.6	8.16	10.56	3.8	21.0	11.12	35.92
SHARC + VBx	0.0	15.6	4.8	20.34	0.0	3.1	7.2	10.30	3.1	22.4	6.29	31.89
SHARC-Overlap + VBx	0.7	11.6	6.26	18.56	0.8	1.6	7.8	10.19	3.8	21.0	6.77	31.57
E-SHARC+VBx	0.0	15.6	3.68	19.16	0.0	3.1	7.05	10.15	3.1	22.4	6.8	32.61
E-SHARC-Overlap + VBx	0.7	11.6	4.91	**17.21**	0.8	1.6	7.7	**10.11**	3.8	21.0	6.6	**31.40**

regions (DER), and the second case ignores overlaps and incorporates a tolerance collar of 0.25s (DER*).

In our work, we have used ground truth speech activity decisions for evaluating AMI and Voxconverse datasets. We also evaluate our trained model on the DISPLACE challenge dataset. As per challenge guidelines, we have used the DISPLACE challenge baseline SAD model. The model is based on TDNN architecture [96, 132] with speech and non-speech classification. The Voxconverse E-SHARC model is used for DISPLACE evaluations.

Table 5.6: DER (%, with overlap + without collar) and DER* (%, without overlap + with collar 0.25s) comparison with state-of-the-art on AMI SDM Eval, Voxconverse Eval, and DISPLACE Eval(phase 2) datasets.

AMI SDM System	DER	DER*
Pyannote [133]	29.1	-
x-vec+AHC+VBx [27]	27.4	12.6
SelfSup-PLDA-PIC +VBx [119]	23.8	5.5
Raj et al. [134]	23.7	-
Plaquet et al. [135]	22.9	-
GAE-based+ SC [136]	-	5.5
GADEC-based [136]	-	4.2
E-SHARC (proposed)	19.83	2.9
E-SHARC-Ovp +VBx (prop.)	**17.2**	2.6
Voxconverse System		
Pyannote [133]	11.9	-
Plaquet et al. [135]	10.4	-
GAE-based+ SC [136]	-	8.0
GADEC-based [136]	-	7.6
E-SHARC (prop.)	11.68	7.6
E-SHARC-Ovp +VBx (proposed)	**10.1**	6.3
DISPLACE System		
DISPLACE Baseline [96, 130]	32.2	14.6
E-SHARC-Ovp +VBx (prop.) + Baseline SAD	31.4	13.0
E-SHARC-Ovp +VBx (prop.) + pyannote SAD	29.9	12.0
Winning system [130]	**27.8**	7.3

5.7.1 Comparison with the baseline systems

Table 5.2 shows that the proposed SHARC model improves over the baseline systems, and the performance further improves with the E-SHARC model for both datasets. We also applied a re-segmentation approach using Variational Bayes inference (VBx) [121] (chapter 2, section 2.1.5) with the E-SHARC clustering labels as initialization, further boosting the performance. As shown in Table 5.2, for the AMI SDM dataset, we obtain 15.6% and 52.6% relative improvements for the dev and eval set, respectively over the PLDA-SC baseline (best). Similarly, we achieve 39.6% and 44.4% relative improvements over the Voxconverse baseline

(PLDA- SC) for the dev and eval set, respectively.

Table 5.3 discusses performance based on the cluster purity and coverage of Voxconverse dev set using pyannote-metric [137]. **Cluster purity** is defined as the percentage of segments from predicted speakers belonging to one speaker in the ground truth. **Cluster coverage** is defined as the percentage of segments from the ground truth speaker covered by the predicted speaker. From the table, it can be observed that the baseline with AHC has high purity but low coverage. However, the baseline with spectral clustering (SC) has lower purity but high coverage. But in our proposed approach, both the purity and the coverage are high, indicating that the supervised clustering can achieve the best of both worlds.

5.7.2 Evaluation of overlap detection system

Table 5.4 shows the False alarm (FA), Miss, and confusion error along with overall DER metric of baseline AHC and SC approaches and the proposed SHARC and E-SHARC models with and without overlap assignment for three different datasets: AMI Single Distant Microphone (SDM), Voxconverse and DISPLACE challenge datasets. From the table 5.4, it can be observed that SHARC and E-SHARC perform significantly better than AHC and SC baselines on all the datasets. E-SHARC achieves 11% and 13% relative improvements over best baseline on AMI and Voxconverse datasets, respectively.

To fairly compare the performance of SHARC-Overlap and E-SHARC-Overlap with the baseline clustering approaches, we have also integrated the overlap assignment for AHC and SC similar to the E-SHARC-Overlap approach. The details are described below.

Baseline overlap assignment (AHC/SC+Overlap): The first pass clustering (AHC/SC) is performed using the PLDA similarity scores matrix to generate the parent speaker labels for each x-vector. The pyannote overlap detector is used to select the x-vector segments containing overlaps. In the similarity matrix, the within-cluster similarity scores are set to the lowest value for these overlapping segments. Then the top k' similarity scores of each overlapping x-vector are selected for the second pass clustering. The parent speaker labels are

assigned for these top k' scores. The mode of the k' parent speaker labels is the second speaker assigned to the overlapping regions. The value of k' is selected as 30 based on validation experiments. The same value is selected for SHARC-Overlap and E-SHARC-Overlap.

From Table 5.4, it can be observed that adding overlap assignment to AHC and SC improves the DER performance. Further improvements are obtained with the proposed SHARC-Overlap and E-SHARC-Overlap approaches. We achieve 11.6%, 16.8%, and 25.3% relative improvements for AMI, Voxconverse, and DISPLACE, respectively, over the best baseline.

5.7.3 Comparison with the other published works

Table 5.6 shows the comparison of the proposed approach with state-of-the-art approaches for AMI, Voxconverse, and DISPLACE datasets. Pyannote [133] is the end-to-end pyannote model with SAD and overlap detection modules. Raj et. al. [134], and Plaquet et. al. [135] are the recent end-to-end models for diarization. The work reported in [136] proposed Graph Attention-Based Deep Embedded Clustering (GADEC), which performs graph attention-based clustering using multi-objective training. It also shows results of the Graph attentional encoder (GAE) based approach for metric learning followed by spectral clustering. Our E-SHARC-Ovp + VBx outperforms the state-of-the-art results for AMI SDM and Voxconverse systems. In the case of the DISPLACE System, we incorporated pyannote SAD to achieve 7.1% relative improvements over the challenge baseline. The winning system investigated different SAD and model combination strategies, while the proposed system was trained on out-of-set (Voxceleb mixtures) data.

5.8 Ablation Studies

5.8.1 Choice of hyper-parameters

Figure 5.3 shows the impact of different values of k and τ on DER on Voxconverse dev set. As k is increasing, the DER is also increasing because higher k generates less number of speakers. On the other hand, the DER is higher for a smaller value of $\tau < 0.2$. The best values of k and τ are 30 and 0.8.

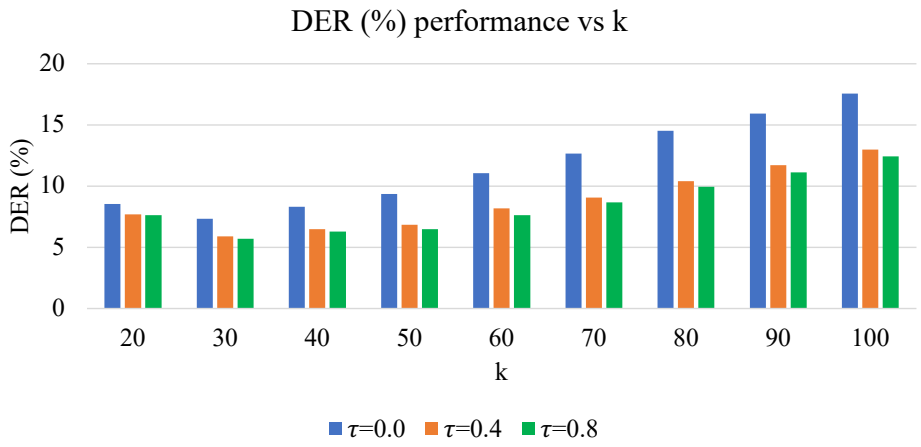

Figure 5.3: Plot comparing DER performance for k ranging from 20-100 and $\tau \in \{0.0, 0.4, 0.8\}$ for Voxconverse dev set.

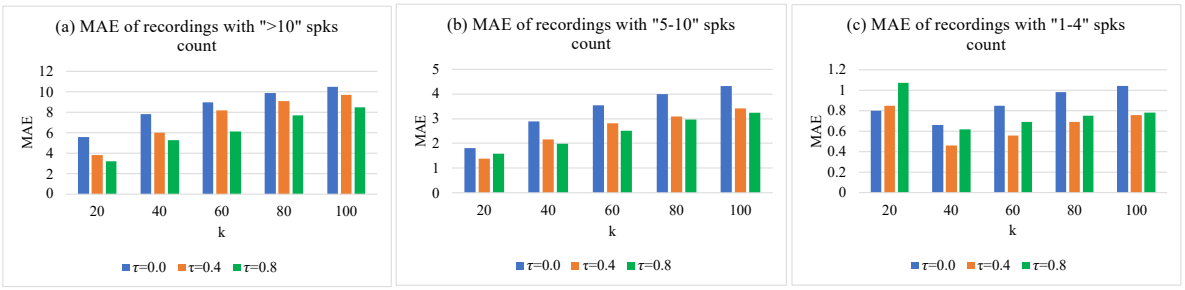

Figure 5.4: Plot comparing Mean Absolute Error (MAE) for the speaker counting task for k ranging from 20-100 and $\tau \in \{0.0, 0.4, 0.8\}$ for Voxconverse dev set.

5.8.2 Speaker counting task

We also performed the speaker counting task evaluation using the proposed model. Figure 5.4 shows performance of the speaker counting task for the Voxconverse dataset. It shows the impact of k and τ on the mean absolute error (MAE) between the ground truth number of speakers and the predicted number of speakers. The figure is divided into three categories of recordings based on the number of speakers present. The plot (a) shows the MAE for recordings with a higher number of speakers (> 10). As k increases, MAE increases and as τ increases, MAE reduces. The plot (b) shows the MAE for recordings containing a moderate number of speakers ($5 - 10$). Finally, the plot (c) shows the MAE for recordings containing a very small number of speakers ($1 - 4$). In this case, MAE is relatively low across different k and τ.

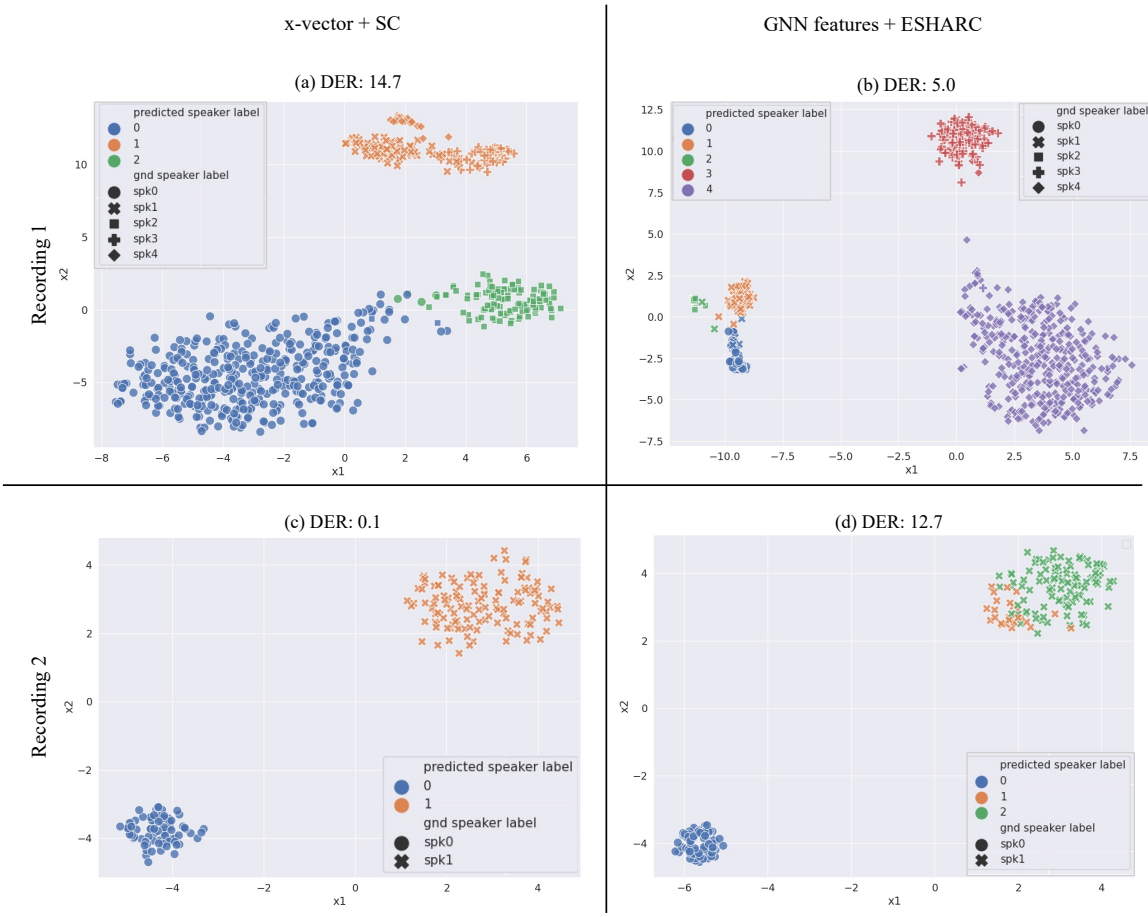

Figure 5.5: 2D t-SNE plot to compare x-vectors and GNN embeddings for two different recordings from Voxconverse dev set. The first column shows the x-vectors with SC labels in different colors, and the second column shows GNN features with E-SHARC labels in different colors. Ground truth labels are represented as different shapes. In both cases, the proposed E-SHARC yields representations with better separability. However, the DER detoriated due to early stoping in Recording 2.

5.8.3 Representation visualization

Figure 5.5 shows the t-SNE plots for two different recordings (two rows) from Voxconverse dev set for baseline and proposed ESHARC approach. The first column shows a t-SNE plot of x-vectors with SC, and the second column shows GNN features with ESHARC. Recording 1 contains 5 speakers represented by different shapes. The SC predicts only 3 speakers out of 5, as shown in different colors. However, E-SHARC is able to form 5 different clusters and provides a lower DER. It can also be observed that the within-cluster covariance is lower in GNN features compared to x-vectors. In the case of recording 2, SC is able to predict the correct speaker clusters. On the other hand, E-SHARC splits spk1 from ground truth into two

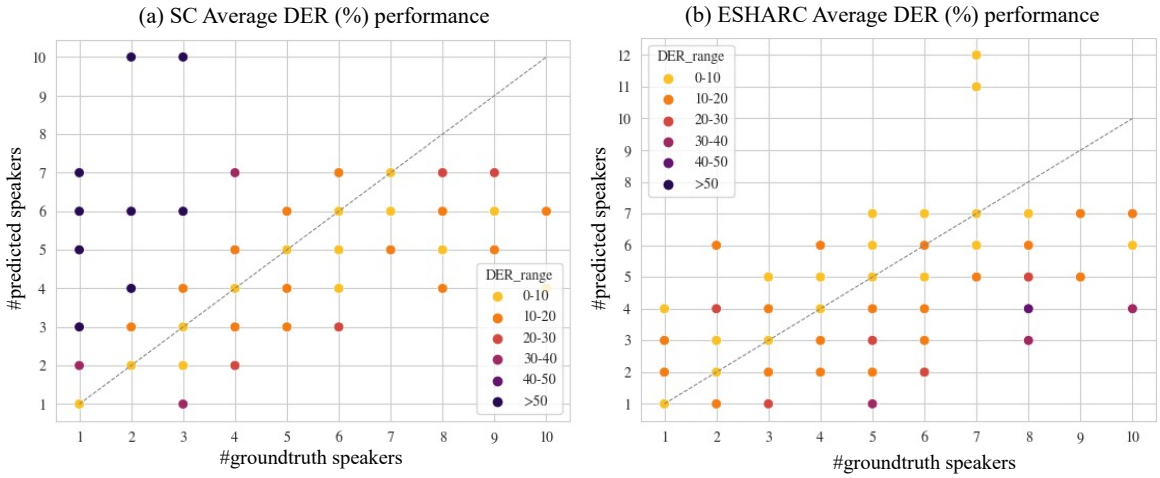

Figure 5.6: 2D scatter plot showing no. of ground truth speakers vs no. of predicted speakers using (a) SC and (b) ESHARC on Voxconverse dev set. The different colors represent different ranges of average DER (%) for a pair of (#ground speaker, #predicted speakers).

different speakers shown in different colors. Although the features are well separated in terms of speaker clusters, the stopping criterion has resulted in early stopping.

Figure 5.6 shows a 2D scatter plot comparing the performance of SC and ESHARC in terms of the predicted number of speakers on Voxconverse dev. The color represents different ranges of filewise DER for the particular pair of (#groundtruth speakers, #predicted speakers). It can be observed that for files with a lower number of speakers (< 4), the model predicts a very high number of speakers which leads to very high DERs (> 50) for SC (Figure 5.6 (a)). On the other side, Figure 5.6 (b) shows a scatter plot for E-SHARC model. The E-SHARC model is able to correctly predict speakers for files with a lower number of speakers in the ground truth, which results in lower DER (0-10). For other files, the E-SHARC predicts lesser number of speakers compared to the groundtruth but it does not affect the DER performance adversely.

5.9 Chapter Summary

We have proposed a supervised hierarchical clustering algorithm using graph neural networks for speaker diarization. The GNN module learns the edge linkages and node densities across all hierarchy levels. The proposed approach enables the learned GNN module to perform hierarchical clustering based on merging criteria which can handle many speakers. The method is further extended to perform end-to-end diarization by jointly learning the embedding extractor and the GNN module. With challenging diarization datasets, we have illustrated the performance improvements obtained using the proposed approach. Further, we also implement an overlapped speaker prediction using the same model with overlapped speech training. This enables prediction and assignment of multiple speakers in the overlapped speech regions which further improves the diarization performance.

Chapter 6

Conclusion And Future Directions

In this chapter, we summarize the key contributions of the thesis, discuss the limitations of our approaches (Table 6.1), and identify some critical unexplored directions that can be pursued to push the boundaries of the diarization research further.

6.1 Key Contributions of the Thesis

The central theme of the thesis revolves around hierarchical graph clustering. The clustering algorithms are key components in speaker diarization as they enable accurate speaker segmentation, speaker change detection, speaker model creation, speaker adaption, and evaluation. Therefore, improving clustering can indirectly enhance the diarization performance. In this doctoral thesis, we have proposed three approaches using graph clustering that significantly benefit the field of speaker diarization.

The first proposed approach is *SSC or self-supervised clustering* described in chapter 3 and published in [98, 97]. It involves alternately merging the clusters for fixed embeddings and then using the pseudo target labels from clustering to update representations till the required number of clusters/speakers are obtained. A graph agglomerative hierarchical clustering algorithm (PIC) was introduced to improve the performance further. The self-supervised representation learning with PIC is referred to as SSC-PIC. We also introduced temporal continuity for smoothening the output of the model. The results and analysis of the performance on two benchmark datasets is done in section 3.4. The relative improvements for the SSC algorithm over the baseline system are 13% and 59% for CH and AMI evaluation

Table 6.1: The table highlights the contributions and limitations of the proposed approaches based on hierarchical graph clustering.

Proposed Approach	Contributions	Limitations
SSC [98, 97] (chapter 3)	- Introduced self-supervised clustering using DNN - Introduced PIC: graph agglomerative clustering	- Similarity scoring is not learnable (cosine) - Performance depends on initial clustering
SelfSup-PLDA-PIC [119] (chapter 4)	- Introduced self-supervised metric learning	- Performance degrades with higher number of speakers
SHARC [123] (chapter 5)	- Introduced supervised hierarchical clustering using GNN	- Increased training time - Require domain specific training - Require external overlap detector - Not purely end-to-end

datasets, respectively. We also showed that the improvements seen in the proposed approach are statistically significant using measures like the student's t-test and Fisher score.

To improve the learning potential of SSC, an extension of SSC-PIC was introduced called *SelfSup-PLDA-PIC*. SelfSup-PLDA-PIC is described in detail in chapter 4 and in [119]. It is a self-supervised metric learning approach using graph clustering to perform joint representation learning and metric learning using the initial clustering results. A neural version of the PLDA model was introduced for metric learning. It is a discriminative model learned using the pairwise binary cross entropy loss that increases speaker separability. Section 4.5 highlights the performance of the *SelfSup-PLDA-PIC* on the AMI and the DIHARD dataset. The relative DER improvement over the baseline system with the temporal continuity is 60% for the AMI evaluation dataset for an unknown number of speakers. The final DER is 4.9%. The VBx re-segmentation [121] on the outputs improves the DER to 4.2%. The DIHARD dataset, a more challenging dataset with variability in the number of speakers and duration per recording, is also used for evaluation. Further analysis is done as the performance was relatively poor compared to the baseline.

The final proposed approach is called Supervised HierArchical gRaph Clustering (SHARC) [123]. As discussed in chapter 5, it is a supervised clustering algorithm introduced to remove the

dependence on the initial clustering results. This is the first attempt to perform supervised hierarchical clustering for diarization using Graph Neural Network (GNN). The supervision allows the model to update the representations and directly improve the clustering performance, thus enabling a single-step approach for diarization. We also propose an approach to jointly update the embedding extractor and the GNN model to perform end-to-end speaker diarization (E-SHARC). It inputs front-end time-frequency features and similarity matrix and jointly learns the embedding extractor and GNN module. It performs representation learning, metric learning, and clustering using a single network. It helps to increase the cluster purity and coverage compared to the baseline. The diarization experiments discussed in section 5.7 illustrate that the proposed E-SHARC approach achieves 53% and 44% relative improvements over the baseline systems on AMI and Voxconverse datasets. Later, this joint training network is further used to predict overlapping speakers for each segment based on the neighborhood speakers.

6.2 Limitations of the Thesis

While our proposed approaches have demonstrated superior performance compared to the baselines, they also come with certain limitations.

For the first approach, self-supervised clustering (SSC), the major limitation lies in using "cosine" similarity scoring, a non-parametric and non-learnable metric. This restriction limits the capabilities of the SSC algorithm and affects the quality of initial clustering, thereby impacting self-supervised learning.

In the second approach, SelfSup-PLDA-PIC, the performance still relies on the initial clustering output due to its self-supervised nature. However, it has been observed that the clustering tends to generate poor and unreliable results, mainly when dealing with a higher number of speakers (>7 speakers), leading to a degradation in the SelfSup-PLDA-PIC performance.

The third proposed approach, SHARC, which involves supervised training, has a few limitations to consider. The primary limitation is the increased training time and higher

computational resource requirements. Since the model is trained end-to-end, the training process takes approximately 24 hours, which is longer than self-supervised learning. Moreover, SHARC requires significantly higher GPU memory of 48 GB (10 times higher than the self-supervised approach), making it computationally demanding. Additionally, the process is not entirely end-to-end, as it relies on a PLDA similarity matrix to generate a graph. Lastly, SHARC necessitates an external overlap detector to predict overlapping speech, which opens up opportunities for further research directions.

Despite these limitations, the proposed approaches have demonstrated promising results, and addressing these shortcomings can lead to further advancements and improvements in diarization research.

6.3 Future Directions

6.3.1 Speaker and language diarization of multilingual conversations

In multilingual societies where people speak multiple languages and dialects, conversations often involve code-switching (switching between languages). Understanding, processing, and analyzing multilingual audio data is necessary to bring speech technologies to the masses. The first step in achieving this is to automatically segment and identify the spoken languages in an audio recording, referred to as language diarization. The applications of language diarization are similar to speaker diarization except that they are focused on language. However, multiple speakers speaking multiple languages make it a much more challenging task. Because different speakers speak the same language in different accents, which is difficult for the model to understand. The DISPLACE challenge 2023 [96] is a step towards understanding the complexity of the problem and proposing possible solutions. The graph clustering approaches discussed in this thesis can be extended to perform language diarization where the nodes represent language embeddings instead of speaker embeddings, as the edges represent the similarity in terms of languages. This can be further extended to perform speaker and language diarization together, which is the task of finding "who spoke when and in what language". One possible direction is generating embeddings with both speaker and language information and creating

a multi-edge graph. The multi-edge graph contains multiple edges between the nodes. In this scenario, it will represent speaker connections and language connections. Then it can be used to perform multi-task learning.

6.3.2 Target speaker detection in conversational speech

The task of automatic speaker verification involves detecting whether a target speaker from an enrollment recording is present in the test recording. Although earlier works assumed the presence of a single speaker in test utterance, in real-life applications, test recordings may contain multiple speakers, e.g., conversational speech [2]. Then it becomes more challenging as it involves two subproblems. The first subproblem is to perform speaker diarization of the test recording, which can help form speaker models of each speaker. The second subproblem is to verify if the target speaker is present in the test by comparing the target speaker model with the test speakers' model. The graph clustering approaches discussed in this thesis can solve these subproblems as one task by representing both the enrollment and test utterances as one graph and then performing clustering. However, the model should be trained to handle the channel mismatch between enrollment and test utterances.

6.3.3 Multi-speaker Automatic Speech Recognition

Transcription of conversational speech, e.g., meetings, uses speaker diarization to identify the regions of speakers and then transcribe regions of each speaker separately using an automatic speech recognition system (ASR). However, this results in errors propagating from two models. Another issue is that speaker diarization is trained to reduce the diarization error rate, which differs from the word error rate used in ASR. Therefore, the output of the diarization system does not consider word boundaries. This results in the requirement of developing a multi-speaker ASR system that can accurately separate and recognize the speech of individual speakers in a conversational speech.

Chetupalli et al. [138] proposed a speaker-conditioned acoustic model for multi-speaker conversational ASR. It involves a joint learning of diarization and the ASR acoustic model. The diarization model [77] predicts speaker activity of each speaker using which speaker

embeddings are extracted from a TDNN network. These two steps can be done jointly using the supervised hierarchical clustering called SHARC (chapter 5). The final part of this thesis was based on supervised hierarchical clustering called SHARC. SHARC directly generates one node embedding per cluster/speaker. Therefore, SHARC can be integrated directly with the ASR model.

Bibliography

[1] J. G. Fiscus, J. Ajot, M. Michel, and J. S. Garofolo, "The rich transcription 2006 spring meeting recognition evaluation," in *Machine Learning for Multimodal Interaction: Third International Workshop, 2006, Revised Selected Papers 3*. Springer, 2006, pp. 309–322.

[2] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (sitw) speaker recognition database." in *Interspeech*, 2016, pp. 818–822.

[3] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, "A review of speaker diarization: Recent advances with deep learning," *Computer Speech & Language*, vol. 72, p. 101317, 2022.

[4] 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 *IEEE ICASSP*, vol. 1, 2003, pp. I–I.

[5] I. McCowan, J. Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos *et al.*, "The AMI meeting corpus," in *Proceedings of Measuring Behavior 2005, 5th International Conference on Methods and Techniques in Behavioral Research*. Noldus Information Technology, 2005, pp. 137–140.

[6] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.

[7] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramab-

hadran, M. Picheny, L.-L. Lim *et al.*, "English conversational telephone speech recognition by humans and machines," *arXiv preprint arXiv:1703.02136*, 2017.

[8] B. Li, T. N. Sainath, A. Narayanan, J. Caroselli, M. Bacchiani, A. Misra, I. Shafran, H. Sak, G. Pundak, K. K. Chin *et al.*, "Acoustic modeling for google home." in *Interspeech*, 2017, pp. 399–403.

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

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

[11] A. Guo, A. Faria, and K. Riedhammer, "Remeeting-deep insights to conversations." in *Interspeech*, 2016, pp. 1964–1965.

[12] P. G. Georgiou, M. P. Black, and S. S. Narayanan, "Behavioral signal processing for understanding (distressed) dyadic interactions: some recent developments," in *Proceedings of the 2011 joint ACM workshop on Human gesture and behavior understanding*, 2011, pp. 7–12.

[13] S. Narayanan and P. G. Georgiou, "Behavioral signal processing: Deriving human behavioral informatics from speech and language," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1203–1233, 2013.

[14] Y. Li, "Speaker diarization system for call-center data," https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-286677, 2020.

[15] A. Addlesee, Y. Yu, and A. Eshghi, "A comprehensive evaluation of incremental speech recognition and diarization for conversational ai," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 3492–3503.

[16] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "First DIHARD challenge evaluation plan," *2018, tech. Rep.*, 2018.

[17] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, and S. Ganapathy, "The second DIHARD diarization challenge: Dataset, task, and baselines," in *Proc. of Interspeech*, 2019, pp. 978–982.

[18] S. E. Tranter and D. A. Reynolds, "An overview of automatic speaker diarization systems," *IEEE Transactions on audio, speech, and language processing*, vol. 14, no. 5, pp. 1557–1565, 2006.

[19] A. Itu, "silence compression scheme for g. 729 optimized for terminals conforming to recommendation v. 70," *ITU-T Recommendation G*, vol. 729, 1996.

[20] R. Chengalvarayan, "Robust energy normalization using speech/nonspeech discriminator for german connected digit recognition," in *Sixth European conference on speech communication and technology*, 1999.

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

[22] T. Ng, B. Zhang, L. Nguyen, S. Matsoukas, X. Zhou, N. Mesgarani, K. Veselỳ, and P. Matějka, "Developing a speech activity detection system for the darpa rats program," in *Thirteenth annual conference of the international speech communication association*, 2012.

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

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

[25] D. Haws, D. Dimitriadis, G. Saon, S. Thomas, and M. Picheny, "On the importance of event detection for asr," in *IEEE ICASSP*, 2016, pp. 5705–5709.

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

[27] F. Landini *et al.*, "Bayesian HMM clustering of x-vector sequences (VBx) in speaker diarization: theory, implementation and analysis on standard tasks," *arXiv preprint arXiv:2012.14952*, 2020.

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

[29] W. H. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of classification*, vol. 1, no. 1, pp. 7–24, 1984.

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

[31] X. Anguera, C. Wooters, and J. M. Pardo, "Robust speaker diarization for meetings: ICSI RT06s meetings evaluation system," in *International Workshop on Machine Learning for Multimodal Interaction*. Springer, 2006, pp. 346–358.

[32] A. Bondy and U. Murty, *Graph Theory with Applications*. Wiley, 1991. [Online]. Available: https://books.google.co.in/books?id=7EWKkgEACAAJ

[33] A. Majeed and I. Rauf, "Graph theory: A comprehensive survey about graph theory applications in computer science and social networks," *Inventions*, vol. 5, no. 1, p. 10, 2020.

[34] Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen, "Graph convolutional networks with markov random field reasoning for social spammer detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 1054–1061.

[35] G. A. Pavlopoulos, M. Secrier, C. N. Moschopoulos, T. G. Soldatos, S. Kossida, J. Aerts, R. Schneider, and P. G. Bagos, "Using graph theory to analyze biological networks," *BioData mining*, vol. 4, pp. 1–27, 2011.

[36] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," *Advances in neural information processing systems*, vol. 30, 2017.

[37] J. Seo and R. F. Simmons, "Syntactic graphs: A representation for the union of all ambiguous parse trees," *Computational Linguistics*, vol. 15, no. 1, pp. 19–32, 1989.

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

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

[40] H. Gish, M.-H. Siu, and J. R. Rohlicek, "Segregation of speakers for speech recognition and speaker identification." in *IEEE ICASSP*, vol. 91, 1991, pp. 873–876.

[41] S. Chen, P. Gopalakrishnan *et al.*, "Speaker, environment and channel change detection and clustering via the bayesian information criterion," in *Proc. DARPA broadcast news transcription and understanding workshop*, vol. 8. Citeseer, 1998, pp. 127–132.

[42] J. E. Rougui, M. Rziza, D. Aboutajdine, M. Gelgon, and J. Martinez, "Fast incremental clustering of gaussian mixture speaker models for scaling up retrieval in on-line broadcast," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5. IEEE, 2006, pp. V–V.

[43] D. A. Reynolds and P. Torres-Carrasquillo, "Approaches and applications of audio diarization," in *IEEE ICASSP*, 2005, pp. 953–956.

[44] C. Wooters and M. Huijbregts, "The icsi rt07s speaker diarization system," in *International Evaluation Workshop on Rich Transcription*. Springer, 2007, pp. 509–519.

[45] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," *CRIM, Montreal,(Report) CRIM-06/08-13*, vol. 14, no. 28-29, p. 2, 2005.

[46] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[47] G. Sell and D. Garcia-Romero, "Speaker diarization with plda i-vector scoring and unsupervised calibration," in *IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 413–417.

[48] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *IEEE ICASSP*, 2014, pp. 4052–4056.

[49] A. Torfi, J. Dawson, and N. M. Nasrabadi, "Text-independent speaker verification using 3d convolutional neural networks," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6.

[50] D. Garcia-Romero, D. Snyder, G. Sell, D. Povey, and A. McCree, "Speaker diarization using deep neural network embeddings," in *IEEE ICASSP*, 2017, pp. 4930–4934.

[51] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *IEEE ICASSP*, 2019, pp. 5796–5800.

[52] H. Zeinali, S. Wang, A. Silnova, P. Matějka, and O. Plchot, "But system description to voxceleb speaker recognition challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.

[53] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks." in *Interspeech*, 2018, pp. 3743–3747.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

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

[56] N. Dawalatabad *et al.*, "ECAPA-TDNN embeddings for speaker diarization," *arXiv preprint arXiv:2104.01466*, 2021.

[57] J. Silovsky and J. Prazak, "Speaker diarization of broadcast streams using two-stage clustering based on i-vectors and cosine distance scoring," in *IEEE ICASSP*, 2012, pp. 4193–4196.

[58] S. Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision*. Springer, 2006, pp. 531–542.

[59] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[60] Y. Lei, L. Burget, L. Ferrer, M. Graciarena, and N. Scheffer, "Towards noise-robust speaker recognition using probabilistic linear discriminant analysis," in *IEEE ICASSP*, 2012, pp. 4253–4256.

[61] T. Stafylakis, V. Katsouros, and G. Carayannis, "Speaker clustering via the mean shift algorithm," *Recall*, vol. 2, no. 7, 2010.

[62] I. Salmun, I. Shapiro, I. Opher, and I. Lapidot, "Plda-based mean shift speakers' short segments clustering," *Computer Speech & Language*, vol. 45, pp. 411–436, 2017.

[63] M. Senoussaoui, P. Kenny, T. Stafylakis, and P. Dumouchel, "A study of the cosine distance-based mean shift for telephone speech diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 217–227, 2014.

[64] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1. Oakland, CA, USA, 1967, pp. 281–297.

[65] B. Chander and K. Gopalakrishnan, "10 - data clustering using unsupervised machine learning," in *Statistical Modeling in Machine Learning*, T. Goswami and G. Sinha, Eds. Academic Press, 2023, pp. 179–204.

[66] W. H. E. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of Classification*, vol. 1, pp. 7–24, 1984.

[67] G. Sell and D. Garcia-Romero, "Speaker diarization with PLDA i-vector scoring and unsupervised calibration," in *IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 413–417.

[68] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning – Data Mining, Inference, and Prediction.* New York: Springer, 2009.

[69] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001.

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

[71] J. Wang, X. Xiao, J. Wu, R. Ramamurthy, F. Rudzicz, and M. Brudno, "Speaker diarization with session-level speaker embedding refinement using graph neural networks," in *IEEE ICASSP*, 2020, pp. 7109–7113.

[72] M. Diez, L. Burget, and P. Matejka, "Speaker diarization based on bayesian hmm with eigenvoice priors." in *Odyssey*, 2018, pp. 147–154.

[73] P. Singh, H. Vardhan, S. Ganapathy, and A. Kanagasundaram, "Leap diarization system for the second dihard challenge," in *Proc. of Interspeech): Crossroads of Speech and Language*. International Speech Communication Association, 2019, pp. 983–987.

[74] 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. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-1602

[75] L. Bullock, H. Bredin, and L. P. Garcia-Perera, "Overlap-aware diarization: Resegmentation using neural end-to-end overlapped speech detection," in *IEEE ICASSP*, 2020, pp. 7114–7118.

[76] 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 *ICASSP 2020, IEEE International Conference on Acoustics, Speech, and Signal Processing*, Barcelona, Spain, May 2020.

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

[78] Y. Fujita *et al.*, "End-to-End Neural Speaker Diarization with Self-attention," in *ASRU*, 2019, pp. 296–303.

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

[80] Q. Li, F. L. Kreyssig, C. Zhang, and P. C. Woodland, "Discriminative neural clustering for speaker diarisation," *arXiv preprint arXiv:1910.09703*, 2019.

[81] 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 *IEEE ASRU*, 2021, pp. 98–105.

[82] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[83] D. Graff, K. Walker, and D. Miller, "Switchboard cellular part 1 audio LDC2001S13," https://catalog.ldc.upenn.edu/LDC2001S13.

[84] ——, "Switchboard cellular part 2 audio LDC2004S07," https://catalog.ldc.upenn.edu/LDC2004S07.

[85] M. Przybocki and A. F. Martin, "Nist speaker recognition evaluation chronicles," in *ODYSSEY04-The Speaker and Language Recognition Workshop*, 2004.

[86] M. A. Przybocki, A. F. Martin, and A. N. Le, "Nist speaker recognition evaluations utilizing the mixer corpora—2004, 2005, 2006," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 1951–1959, 2007.

[87] A. F. Martin and C. S. Greenberg, "Nist 2008 speaker recognition evaluation: Performance across telephone and room microphone channels," in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

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

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

[90] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *IEEE ICASSP*, 2015, pp. 5206–5210.

[91] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[92] 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 *IEEE ICASSP*, 2017, pp. 5220–5224.

[93] M. Przybocki and A. Martin, "2000 NIST Speaker Recognition Evaluation," https://catalog.ldc.upenn.edu/LDC2001S97.

[94] N. Ryant, P. Singh, V. Krishnamohan, R. Varma, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "The Third DIHARD Diarization Challenge," *arXiv preprint arXiv:2012.01477*, 2020.

[95] J. S. Chung, J. Huh, A. Nagrani, T. Afouras, and A. Zisserman, "Spot the Conversation: Speaker Diarisation in the Wild," in *Proc. Interspeech*, 2020, pp. 299–303.

[96] S. Baghel, S. Ramoji, Sidharth, R. H, P. Singh, S. Jain, P. Roy Chowdhuri, K. Kulkarni, S. Padhi, D. Vijayasenan, and S. Ganapathy, "The DISPLACE Challenge 2023 - DIarization of SPeaker and LAnguage in Conversational Environments," in *Proc. Interspeech*, 2023, pp. 3562–3566.

[97] P. Singh and S. Ganapathy, "Deep Self-Supervised Hierarchical Clustering for Speaker Diarization," in *Proc. of Interspeech*, 2020, pp. 294–298.

[98] ——, "Self-supervised representation learning with path integral clustering for speaker diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, p. 1639–1649, 2021. [Online]. Available: http://dx.doi.org/10.1109/TASLP.2021.3075100

[99] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 663–15 674.

[100] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *international conference on machine learning*, 2017, pp. 3861–3870.

[101] U. Shaham, K. Stanton, H. Li, R. Basri, B. Nadler, and Y. Kluger, "Spectralnet: Spectral clustering using deep neural networks," in *International Conference on Learning Representations*, 2018.

[102] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5147–5156.

[103] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, and Y. Bengio, "Learning problem-agnostic speech representations from multiple self-supervised tasks," in *Proc. of Interspeech*, 2019, pp. 161–165.

[104] W. Zhang, D. Zhao, and X. Wang, "Agglomerative clustering via maximum incremental path integral," *Pattern Recognition*, vol. 46, no. 11, pp. 3056–3065, 2013.

[105] H. Ning, M. Liu, H. Tang, and T. S. Huang, "A spectral clustering approach to speaker diarization," in *Proc. of Interspeech and ICSLP*, 2006, pp. 2178–2181.

[106] M. Pavan and M. Pelillo, "Dominant sets and pairwise clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1, pp. 167–172, 2006.

[107] Y. Tang, X. Wu, and W. Bu, "Saliency detection based on graph-structural agglomerative clustering," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1083–1086.

[108] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *IEEE workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

[109] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Twelfth annual conference of the international speech communication association*, 2011.

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

[111] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[112] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, p. 101027, 2020.

[113] 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, 2019.

[114] L. V. D. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[115] M. Pal, M. Kumar, R. Peri, and S. Narayanan, "A study of semi-supervised speaker diarization system using gan mixture model," *arXiv preprint arXiv:1910.11416*, 2019.

[116] N. Dawalatabad, S. Madikeri, C. C. Sekhar, and H. A. Murthy, "Incremental transfer learning in two-pass information bottleneck based speaker diarization system for meetings," in *IEEE ICASSP*, 2019, pp. 6291–6295.

[117] M. Pal, M. Kumar, R. Peri, T. J. Park, S. H. Kim, C. Lord, S. Bishop, and S. Narayanan, "Speaker diarization using latent space clustering in generative adversarial network," in *IEEE ICASSP*, 2020, pp. 6504–6508.

[118] G. Sun, C. Zhang, and P. C. Woodland, "Speaker diarisation using 2D self-attentive combination of embeddings," in *IEEE ICASSP*, 2019, pp. 5801–5805.

[119] P. Singh and S. Ganapathy, "Self-supervised metric learning with graph clustering for speaker diarization," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 90–97.

[120] S. Ramoji, P. Krishnan, and S. Ganapathy, "NPLDA: A deep neural plda model for speaker verification," *arXiv preprint arXiv:2002.03562*, 2020.

[121] F. Landini *et al.*, "BUT system for the second DIHARD speech diarization challenge," in *IEEE ICASSP*, 2020, pp. 6529–6533.

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

[123] P. Singh, A. Kaul, and S. Ganapathy, "Supervised hierarchical clustering using graph neural networks for speaker diarization," in *IEEE ICASSP*, 2023, pp. 1–5.

[124] Y. Xing, T. He, T. Xiao, Y. Wang, Y. Xiong, W. Xia, D. Wipf, Z. Zhang, and S. Soatto, "Learning hierarchical graph neural networks for image clustering," in *Proc. IEEE ICCV*, 2021, pp. 3467–3477.

[125] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[126] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks," in *Proc. Interspeech*, 2018, pp. 3743–3747.

[127] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[128] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[129] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[130] S. Baghel, S. Ramoji, S. Jain, P. R. Chowdhuri, P. Singh, D. Vijayasenan, and S. Ganapathy, "Summary of the displace challenge 2023–diarization of speaker and language in conversational environments," *arXiv preprint arXiv:2311.12564*, 2023.

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

[132] 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.

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

[134] D. Raj, D. Povey, and S. Khudanpur, "Gpu-accelerated guided source separation for meeting transcription," *arXiv preprint arXiv:2212.05271*, 2022.

[135] A. Plaquet and H. Bredin, "Powerset multi-class cross entropy loss for neural speaker diarization," in *Proc. Interspeech*, 2023, pp. 3222–3226.

[136] Y. Wei, H. Guo, Z. Ge, and Z. Yang, "Graph attention-based deep embedded clustering for speaker diarization," *Speech Communication*, vol. 155, p. 102991, 2023.

[137] H. Bredin, "pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems," in *Interspeech*, Stockholm, Sweden, August 2017. [Online]. Available: http://pyannote.github.io/pyannote-metrics

[138] S. Raj Chetupalli and S. Ganapathy, "Speaker conditioned acoustic modeling for multi-speaker conversational ASR," in *Proc. Interspeech*, 2022, pp. 3834–3838.