Offline Speaker Diarization of Single-Channel Audio

Undergraduate Thesis

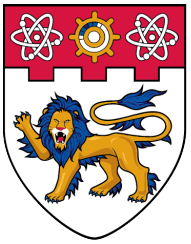
Submitted in partial fulfillment of the requirements of
BITS F422T Thesis

By

Madhur Panwar
ID No. 2016B4A70933P

Under the supervision of:

Dr. Eng Siong Chng
&
Dr. Poonam Goyal



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS
NANYANG TECHNOLOGICAL UNIVERSITY, SINGAPORE

May 2021

Declaration of Authorship

I, Madhur Panwar, declare that this Undergraduate Thesis titled, 'Offline Speaker Diarization of Single-Channel Audio' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: 18-May-2021

"If we knew what we were doing, it would not be called research, would it?"

Albert Einstein

Abstract

Offline Speaker Diarization of Single-Channel Audio

by Madhur Panwar

The Speaker Diarization task involves annotating a given audio file with the speaker labels. The requirement is to identify all the time segments in which each speaker is speaking. This is difficult because the audio file may contain arbitrary number of speakers and have variable length duration. The speakers may also have overlapping speech in the audio file. The traditional diarization systems were module based. Current literature tries to combine multiple modules into one for joint optimization. This thesis develops a fully pythonic pipeline for offline speaker diarization of single-channel audio which can be trained end-to-end. It can be used as a baseline over which several diarization systems can be built by changing the internal architecture of the pipeline.

To support this development, an overview of Speaker Diarization along with historical and current developments is presented. It is followed by the discussion of several neural based techniques which have emerged in recent years. We also discuss the two important baseline diarization models which yield state-of-the-art performance (7.06% and 7.12% DER) on 8 kHz CALLHOME speech dataset.

We employ a pythonic Time Delay Neural Network (TDNN) to extract the x-vector features from the audio. The extracted features are used by a Bi-LSTM to predict the affinity matrix of the speech segments. It is trained using the crossentropy loss between the predicted and the ground truth affinity matrix. Spectral clustering is used to cluster the rows of the affinity matrix and hence to detect the speakers in the various segments. We adapt the pythonic TDNN trained on 16 kHz speech to 8 kHz telephony data by fine-tuning it on speaker classification task. Fine-tuning the TDNN on a subset of the telephony data results in improvement of DER from 38.78% to 33.15% for the entire pipeline. Plots of x-vector embeddings also show the effectiveness of the resulting network.

Acknowledgements

I would like to express my gratitude to my supervisors, Associate Professor Eng Siong Chng from NTU, Singapore and Associate Professor Poonam Goyal from BITS Pilani, India, for passionately guiding me through each stage of my undergraduate thesis. I sincerely thank them for inculcating the scientific spirit within me and teaching me to conduct and communicate research effectively.

I would also like to thank Dr. Xionghu Zhong from Hunan University, China, for his patience and suggestions during our biweekly meetings.

I am highly indebted to my research mentor, Zhi Hao Lim, whose expertise was invaluable in formulating the research questions and methodology. Our discussions have immensely helped me develop a research mindset. I hope to preserve and hone it further in life.

I would like to thank my family members, whose love and support are with me in everything I pursue.

Contents

List of Figures

List of Tables

Abbreviations

ASR	**A**utomatic **S**peech **R**ecognition
ANN	**A**rtifical **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
VAD	**V**oice **A**ctivity **D**etection
WebRTC	**W**eb **R**eal-**T**ime **C**ommunications
GMM	**G**aussian **M**ixture **M**odels
UBM	**U**niversal **B**ackground **M**odel
EM	**E**xpectation **M**aximization
HMM	**H**idden **M**arkov **M**odels
DER	**D**iarization **E**rror **R**ate
JER	**J**accard **E**rror **R**ate
JFA	**J**oint **F**actor **A**nalysis
MFCC	**M**el-**F**requency **C**epstral **C**oefficients
LSTM	**L**ong **S**hort-**T**erm **M**emory
STFT	**S**hort-**T**ime **F**ourier **T**ransform
NMS	**N**on **M**aximum **S**uppresion
RCNN	**R**egion **b**ased **C**onvolutional **N**eural **N**etworks
RPN	**R**egion **P**roposal **N**etwork
UIS-RNN	**U**nbounded **I**nterleaved-**S**tate **R**ecurrent **N**eural **N**etwork
PyTDNN	**Py**thonic **T**ime **D**elay **N**eural **N**etwork
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
SRE	**S**peaker **R**ecognition **E**valuation
PCA	**P**rincipal **C**omponent **A**nalysis
t-SNE	**t**-distributed **S**tochastic **N**eighbor **E**mbedding

To my dear family

Chapter 1

Introduction

Diarize means to make a note or to list an event in some diary. Speaker Diarization systems, much like record keeping, address the problem of "who spoke when" in an audio recording. They are important because they improve the Automatic Speech Recognition (ASR) System. They also find applications in speaker indexing, audio information retrieval, speaker verificaton and speech-to-text transciption.

Traditionally, diarization systems consist of various components which are independently optimized. This is tedious because it requires us to maintain and handle various models, serving as different stages of the pipeline. Recently, end-to-end diarization systems have emerged and due to the advent of deep learning, multiple modules of the pipeline are being collapsed into one. This thesis will focus on the development of a fully pythonic pipeline where the feature extraction and similarity scoring modules can be optimized together.

1.1 Motivation

Speech is the most used way by which humans share ideas. It is not exaggerating to say that our development, as a species, is dependent on the sharing of ideas, which is mostly done via medium of speech. With the evolution in technology, we are now in an age of information explosion where data can be stored cheaply. However, this collection of data poses a new challenge on us i.e. to make sense out of it. We have a lot of speech data and if we can understand it better, that can lead to faster scientific progress.

Speaker Diarization is an important task in that regard since it tells us the events that took place in an audio file. Even though speaker diarization has been researched on for long time and several advancements have been made, a general diarization system does not exist. Diarization is largely

a domain specific task and treated like so. A system trained on meeting room conversations cannot be used to diarize the audio from a noisy environment.

Not only is the task of diarization solved for a particular environment, the many components in the diarization pipeline are independently optimized. The research is largely being carried out to employ the power of neural networks and solve this task end-to-end. This is hard due to the permutation invariant nature of the diarization output. The current approaches solve this issue by considering all possible permutations [23], however they do not scale for large number of speakers and cannot be put to use in a practical environment.

The motivation of this thesis is to explore the field of diarization, evaluate few state of the art diarization models and to develop a fully pythonic pipeline for diarization. It is hoped that the developed pipeline will ease the formation of diarization systems and aid to speed up the research in this field.

1.2 Contribution

In this thesis, we develop a fully pythonic pipeline for offline speaker diarization of single-channel audio, and evaluate it empirically over CALLHOME dataset and synthetic datasets prepared from NIST SRE10 and VoxCeleb1 [63]. We also present the qualitative results in terms of the x-vector visualizations.

Our developed pipeline uses the x-vector embeddings from a Pythonic TDNN (PyTDNN), which achieves state-of-the-art on Speaker Verification when evaluated over 16 kHz speech. We observed that using the PyTDNN over 8 kHz CALLHOME speech as-is results in high DER. We hypothesize that this problem is due the mismatch in the x-vector embedding due to difference in sampling rates. To that end, we fine-tune the PyTDNN over 8 kHz telephony data subset from Switchboard (SWBD) and SRE-databases (SRE 2004, 2005, 2006, 2008). This results in decreasing of the DER and thereby verifying our hypothesis.

We also set up and evaluate two important diarization baselines in the process: kaldi [70] diarization pipeline, and the LSTM based Similarity Measurement with Spectral Clustering (LSTM-SC) [52] pipeline. These pipelines achieve the state-of-the-art DER of 7.06% and 7.12% respectively on the CALLHOME dataset. Our fine-tuning procedure improves the DER of PyTDNN diarization pipeline from 38.78% to 33.15% on CALLHOME and from 35.55% to 32.97% on synthetic dataset prepared from SRE10.

1.3 Organisation

This thesis is organised as follows:

Chapter 2 presents an overview of the speaker diarization field starting from the traditional approaches up to the current neural advancements. Particularly, we shall see the components of diarization pipeline in detail and discuss various recent state-of-the-art approaches. We will also look at the datasets and evaluation metrics used in speaker diarization.

In Chapter 3, we shall discuss in detail the two diarization pipelines: kaldi [70] diarization pipeline, and the LSTM based Similarity Measurement with Spectral Clustering (LSTM-SC) [52] pipeline. These are the baselines over which our pythonic pipeline is built.

Chapter 4 introduces the fully pythonic pipeline developed by employing a Pythonic TDNN. We discuss the fine-tuning procedure and preparation of the synthetic dataset over which the pipeline was evaluated. We also witness qualitative visualizations of the x-vector embeddings which depict the effectiveness of the fine-tuning procedure.

Finally, Chapter 5 concludes with the summary of our developments and lists several possible future directions in which further research can be carried out.

Chapter 2

Literature Review of Speaker Diarization

Speaker Diarization is the task of annotating segments of an audio with the speaker identities. It is identical to keeping records of the events happening in the audio, hence the term "diarize" is used. Automatic Speech Recognition (ASR) systems have a module for diarization, thus improving on this task has widespread applications. Diarization is therefore a very popular problem in ASR and annual contests are held in its name [79] [49] [48].

Traditional diarization systems were module based and treated the problem as a sequence of multiple problems all of which were independently solved. The current research of diarization is headed towards the development of an efficient end-to-end pipeline which solves the entire task at once [23]. In this chapter we shall see the development of diarization research over the years and gain background knowledge specific to this task.

The rest of this chapter is organised as follows: In section 2.1 we see the history and developments of the speaker diarization, as well as discuss in detail each component of the pipeline.

In section 2.2, we discuss the recent advances made in speaker diarization by the advent of deep learning.

In section 2.3, we note the diarization datasets and mechanisms used to evaluate speaker diarization systems.

Finally, in section 2.4, we look at the various open source toolkits which can be used to develop speaker diarization systems.

2.1 Overview of Speaker Diarization

Traditional speaker diarization systems contain independent modules as shown in Figure 2.2. To alleviate from acoustic environmental artifacts, a lot of processing techniques are utilised. Speech Activity Detection (SAD) is applied to separate speech and non-speech segments. Signals for the segments classified as speech are transformed to embeddings. Clustering stage groups the speech portion representing the embedding vectors, and Post Processing refines the results of clustering stage. This section discusses the history and development of these components of diarization pipeline.

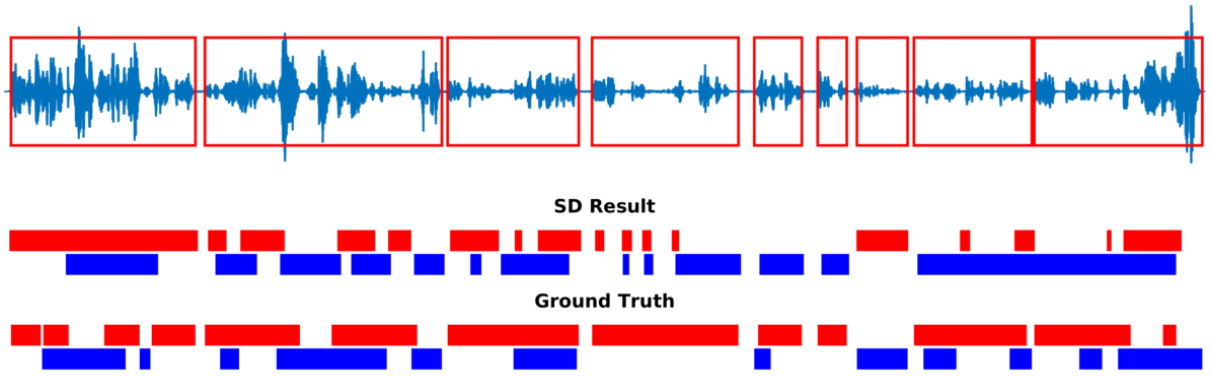


FIGURE 2.1: A visualization of speaker diarization.

2.1.1 History and Development of Speaker Diarization

In the 1990s, the diarization research focussed on clustering of sound events and on unsupervised speech segmentation [27, 76, 39, 92]. Fundamental approaches like Bayesian Information Criterion (BIC) and Generalized Likelihood Ratio (GLR) were developed in this period. Automatic Speech recognition (ASR) on broadcast news recordings benefited from these advancements by training the models in speaker adaptive fashion [26, 25, 53]. Research groups and committees started forming around the world and diarization challenges started being organised in the early 2000s. Among these groups were the Augmented Multi-party Interaction (AMI) Consortium [1] and Rich Transcription Evaluation[2] by the National Institute of Standards and Technology (NIST). Formation of these organizations brought in various revolutions in the speaker diarization field like Information Bottleneck Clustering (IBC) [9] and Joint Factor Analysis (JFA) [45, 43]. They positively impacted different data domains like meeting room conversations [40, 38, 102], Conversational Telephone Speech (CTS) [75, 100, 73], and broadcast news [2, 62, 61].

Since 2010s, a lot of research has happened to model the diarization using neural networks. One area which benefitted the most is the speech embedding extractors i.e. d-vector [31, 106] and

[1] AMI Consortium.
[2] NIST, Rich Transcription Evaluation.

x-vector networks [94]. Speaker representations are embeddings from some latent space modeled by a bottleneck layer in the speaker recognition neural networks. The transition from i-vectors [90, 20, 91] to neural embeddings led to the development of more robust speaker diarization systems.

Recent research in diarization is towards combining multiple components of the pipeline and optimizing them jointly. An example of such effort is the End-to-End Neural Diarization (EEND) [23, 24] model which gives promising results. A wave of such jointly optimized models has struck speaker diarization and a lot of interesting development awaits the field. This would open up unparalleled opportunities of tackling challenges like jointly optimizing diarization with other speech systems, handling of overlapping speech, etc.

2.1.2 Components of Speaker Diarization Systems

In this section, we shall learn about the different components of speaker diarization systems in detail.

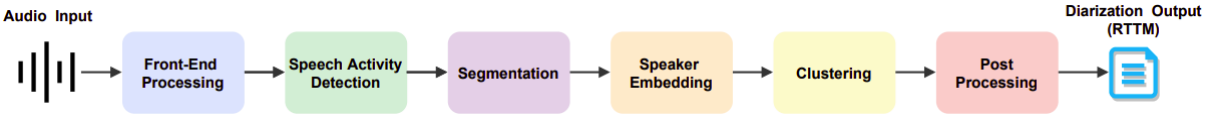


FIGURE 2.2: Pipeline of speaker diarization, consisting of various components.

2.1.2.1 Front-end Processing

To refine the input audio, the first step in the diarization pipeline is to apply the various front-end processing techniques. They deal with enhancement of speech, speech separation, dereverberation and speech extraction. The idea is to estimate the original source signal given the observed (possibly noisy) signal, where, this estimation takes place considering the Short-time Fourier transform (STFT) representation of the input signal at frame level defined as follows. Let $s_{i,f,t} \in \mathbb{C}$ be the STFT of source speaker i at frame t on frequency bin f. Observed noisy signal $x_{t,f}$ is then given by a mixture of the source signals, an impulse response from the room $h_{i,f,t} \in \mathbb{C}$, and an additive noise $n_{t,f} \in \mathbb{C}$,

$$x_{t,f} = \sum_{i=1}^{K} \sum_{\tau} h_{i,f,\tau} S_{i,f,t-\tau} + n_{t,f}, \tag{2.1}$$

where K is the total number of speakers in the audio signal [67].

Denoising (or speech enhancement) focuses on suppressing the noisy parts from the speech. Deep learning has shown significant improvement in single-channel denoising [54, 108, 21].

In contrast to the other front-end processing techniques, **dereverberation** uses statistical signal processing techniques. Weighted Prediction Error (WPE) [64, 109, 18] is the most prevalent among them. The fundamental notion of WPE (for single source case i.e. $K = 1$) is to break down the original signal model in Eq. 2.1 into two components: the early reflection $x_{t,f}^{early}$ and the late reverberation $x_{t,f}^{late}$ as follows:

$$x_{t,f} = \sum_{\tau} h_{f,\tau} S_{f,t-\tau} = x_{t,f}^{early} + x_{t,f}^{late}$$

WPE estimates the filter coefficients $\hat{h}_{f,t}^{wpe} \in \mathbb{C}$, maintaining the early reflection and suppressing the late reverberation based on maximum likelihood estimation (MLE).

Speech separation is popular when the overlapping speech regions are present in significant amounts. Deep learning based speech separation techniques have gained popularity, e.g. Permutation Invariant Training (PIT) [47] and Deep Clustering [32]. Multi-channel speech separation based on beamforming has also proven to be effective [110, 6].

2.1.2.2 Speech Activity Detection (SAD)

The task of SAD is to distinguish between speech and non-speech segments. SAD consists of two parts: first is a feature extraction front-end where features like MFCCs are extracted, and the other part is a classifier model which predicts the fate of the input frame being speech or not. These models involve either Hidden Markov Models (HMMs) [83], Gaussian Mixture Models (GMMs) [65],or Deep Neural Networks (DNNs) [19].

It is important to note that the performance of of SAD directly determines the overall performance of the diarization system. This is due to the possibilities of large false positives being created by the SAD system [30]. It is also a common practice in diarization to use the ground truth SAD (called “oracle SAD”). When the system employs its own speech activity detector, it is referred to as ”system SAD” setup.

2.1.2.3 Segmentation

Breaking the input audio into multiple segments is called segmentation. It is usually performed in two ways: either segmenting the audio at the speaker change points, or using uniform segmentation. The first approach was used traditionally, however, second approach is prevalent nowadays. Uniform segmentation, however, poses certain problems: the segments need to be short enough to ensure them being speaker homogeneous, but simultaneously we need to have sufficient audio to extract meaningful representations as well. There is usually a trade-off between

the two. Traditional diarization systems detected speaker change points by comparing two contrasting hypotheses:

- H_0: Both samples are from the same speaker
- H_1: Both the samples are from the different speakers.

Although many methods for hypothesis testing were proposed viz. Generalized Likelihood Ratio (GLR) [61], Kulback Leibler 2 (KL2) [92] and Bayesian Information Criterion (BIC) [10, 13]. BIC was the most used approach. It is applied to segmentation in the following manner: Assume that the audio stream's features are represented by the sequence $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..\mathbf{x}_N]$, with each $\mathbf{x}_i$ being drawn from an independent multivariate Gaussian distribution:

$$\mathbf{x}_i \sim N(\mu_i, \Sigma_i),$$

where Σ_i and μ_i are the covariance matrix and mean of i-th feature window respectively. Now, the two hypotheses H_0 and H_1 are denoted as follows:

$$H_0 : \mathbf{x}_1 ... \mathbf{x}_N \sim N(\mu, \Sigma)$$

$$H_1 : \mathbf{x}_1 ... \mathbf{x}_i \sim N(\mu_1, \Sigma_1)$$

$$\mathbf{x}_{i+1} ... \mathbf{x}_N \sim N(\mu_2, \Sigma_2)$$

The maximum likelihood estimator is represented as:

$$R(i) = N \; log \; |\Sigma| - N_1 \; log \; |\Sigma_1| - N_2 \; log \; |\Sigma_2|$$

where covariances Σ, Σ_1, and Σ_2 are respectively from $\{\mathbf{x}_1, ..\mathbf{x}_N\}$, $\{\mathbf{x}_1, ..\mathbf{x}_i\}$, and $\{\mathbf{x}_{i+1}, ..\mathbf{x}_N\}$ Now, the BIC value between the two models is given by:

$$BIC(i) = R(i) - \lambda P \tag{2.2}$$

where P denotes the penalty term [10]:

$$P = \frac{1}{2}\Big(d + \frac{1}{2}d(d+1)\Big) \; log \; N,$$

d being the feature dimensionality. λ is usually chosen as 1. The speaker change point is then said to have occured if:

$$\left\{ \max_i \; BIC(i) \right\} \; > \; 0.$$

When using the speaker change point detection for segmentation, the segments are of varied lengths. Therefore, the arrival of *i*-vectors [12] and the neural network based embeddings [103] replaced it by uniform segmentation [87, 86], the reason being that variable lengths of segments created additional variability for the model to deal with, which decreased the efficacy of speaker representations.

2.1.2.4 Speaker Representations and Speaker Embeddings

This section explains popular methods for measuring speech segments' similarity. These methods are often paired with clustering algorithms which will be the topic of section 2.1.2.5. First, we shall see GMM-based hypothesis testing methods which are used with speaker point change segmentation, then, we see the widely used uniform segmentation based speaker representations i.e. *i*-vector and *d*-vector.

GMM Speaker Model for Similarity Measure

Starting models of diarization were based on Gaussian mixture models (GMM) built on features like MFCCs. Agglomerative Hierarchical Clustering (AHC) was also put to use alongside and resulted in clusters which were speaker homogeneous. There are several hypothesis testing methods for process of speech segment clustering like GLR [101], KL [77] and greedy BIC [10] (most popular). Greedy BIC method uses the BIC value during the AHC process to measure similarity between two nodes. Assume a set of nodes $S = s_1, s_2, ...s_k$. Greegy BIC models each s_i as a multivariate Gaussian distribution (denoted by $N(\mu_i, \Sigma_i)$ where μ_i & Σ_i respectively represent the mean and covariance of samples merged in node s_i). The BIC value for node merging of s_1 and s_2 is computed by:

$$BIC = n\ log\ |\Sigma| - n_1\ log\ |\Sigma_1| - n_2\ log\ |\Sigma_2| - \lambda P \tag{2.3}$$

where P and λ are same as in Eq. 2.2 in section 2.1.2.3, and n is the merged node size $n = (n_1 + n_2)$. We merge these nodes during the clustering procedure if Eq. 2.3 < 0. GMM based clustering was prevalent until i-vector and later d-vector speaker representations came into picture.

Joint Factor Analysis and i-vector

Before the arrival of representations such as i-vector [12] or x-vector [94], the Universal Background Model (UBM) [74] showed persistent success in tasks of speaker recognition by using large mixture of Gaussians which covered quite a large amount of speech data. The arrival of Joint Factor Analysis (JFA) [46, 44] drastically improved the testing and modeling of similarity of vocal characteristics with GMM-UBM [74].

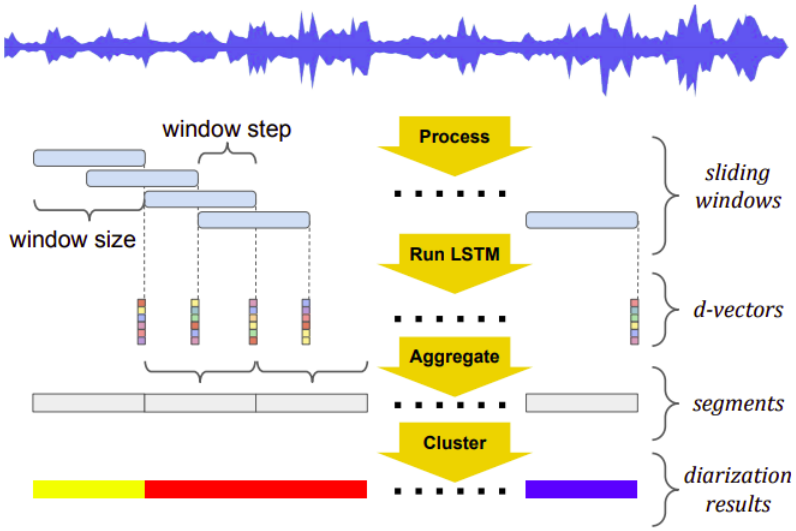


FIGURE 2.3: Flowchart of Google Speech's d-vector diarization pipeline.

Hypothesis testing based on GMM-UBM suffered from the drawback of 'Maximum A Posteriori' (MAP) which is affected not only by specific characteristics but by various inconvenient factors e.g. background and channel noise, etc. as well. Hence, the supervector concept developed by GMM-UBM was unideal.

JFA alleviates this problem by breaking down the supervector into several component matrices viz. speaker dependent ($\mathbf{D}$) & independent ($\mathbf{m}$) components, channel dependent ($\mathbf{U}$) components and residual components. Hence, the supervector s can be decomposed as shown in Eq. 2.4, where, alongside the component matrices, vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are for channel factors, speaker factors and speaker-specific residual factors. These vectors are assumed to have a standard normal distribution.

$$\mathbf{M(s)} = \mathbf{m} + \mathbf{Vy} + \mathbf{Ux} + \mathbf{Dz} \tag{2.4}$$

The 'Total Variability' matrix $\mathbf{T}$, modeling channel and speaker variabilities together, further simplifies the idea of JFA. The supervector $\mathbf{M}$ is then given as:

$$\mathbf{M} = \mathbf{m} + \mathbf{Tw}, \tag{2.5}$$

where $\mathbf{w}$ is referred to as the i-vector [12] and assumed to have a standard normal distribution. It is computed by the MAP estimation [42]. $\mathbf{m}$ is the channel and sesion-independent component of mean supervector. i-vector representation popularized the idea of speaker representations and is said to characterize the vocal tract of the speaker. They have been used in speaker recognition as well as speaker diarization studies [87, 85, 112] and have outperformed the hypothesis testing approaches based on GMM.

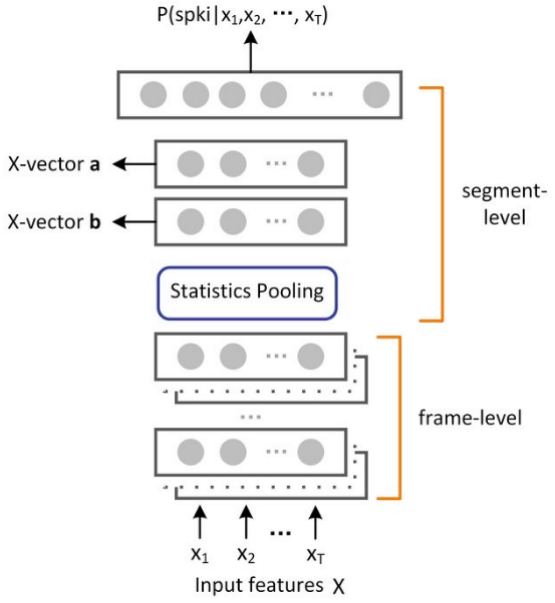


FIGURE 2.4: Overview of the x-vector embedding network.

Neural Network Based Speaker Representations

Speaker representation systems in diarization have largely been impacted by the rise of neural networks and deep learning. Representation learning was first introduced for tasks in face recognition domain [98, 99]. It involves using the neural network model to map the input (image, audio, or text) to a high dimensional vector by sampling the activations from an intermediate layer. This alleviates the problem of creating manual and handcrafted features which involve eigenvalue decomposition and factor analysis. The input data also need not satisfy the normality criteria. Therefore, representation learning for audio has become simpler and inference speeds have also increased drastically.

Audio representations obtained by using deep neural networks are called ***d*-vectors** [103]. One popular *d*-vector utilizes stacked filterbank features including the context frames as an input and trains a sequence of multiple dense layers with the binary cross entropy (BCE) loss function. The penultimate layer gives the *d*-vector representations of the input. *d*-vectors have been used in many recent diarization systems [106, 111]. Figure 2.3 represents the diarization system [106] developed by Google Speech team which used *d*-vector representations.

The arrival of **x-vectors** [93, 94] further improved the DNN-based speaker representations. They immediately captured the scene by winning the first DIHARD challenge [86] and the NIST Speaker Recognition Challenge [104]. Figure 2.4 (adapted from [89]) shows the architecture of x-vector framework. x-vector differs from d-vector in terms of statistical pooling and the time-delay architecture which reduces the effect of input length. This is useful in diarization where segments shorter than normal window length need to be processed.

2.1.2.5 Clustering

The next step after extraction of speaker representations is to cluster the audio segments. In this section, we shall see the most prevalant clustering schemes used in speaker diarization.

Mean shift is a technique to locate the maxima (the modes) of a (non-parametric) density function by considering discrete samples from it. Mean shift algorithm uses circular sliding window whose radius is user-determined. It works as follows:

- Assign a unique cluster to each of the data points.
- Place the circular window at a random point and calculate the mean of all the points lying within the window.
- Shift the window to the location of the computed mean.
- Repeat until the mean converges.

This algorithm has been applied to diarization in different settings: i-vector and PLDA [82], i-vector and cosine distance [87, 88] and KL distance [95]. It is similar to k-means in terms of having an update step but does not require the number of clusters beforehand. This aspect especially makes it useful for diarization where an arbitrary number of speakers might be present.

Agglomerative Hierarchical Clustering (AHC) approach has been constantly utilized in many speaker diarization systems besides multiple different distance metrics e.g. KL [77], BIC [10, 29], and PLDA [86, 4, 66]. It is an iterative algorithm which works as follows:

- Assign all data points to individual clusters.
- Compute the similarity between the different clusters using the distance metric.
- Merge the clusters with the highest similarity.
- Repeat until the provided similarity threshold or number of clusters criteria is met.

For diarization, the stopping criteria is usually a target number of speakers. In practice, the clustering algorithm calculates the optimum number of speakers by supervised calibration over a development set.

Spectral Clustering is another type of clustering approach. It is a graph based method and has been widely used in speaker diarization. There are many variants of spectral clustering, but they all share the following steps:

- **Computation of Affinity Matrix**: Affinity matrix $\mathbf{A}$ is generated by processing of the affinity value d by a kernel. Zeroing the affinity values below a certain threshold can also be carried out before computing $\mathbf{A}$.
- **Computation of Laplacian Matrix** [55]: It is computed in two ways: normalized and unnormalized. $\mathbf{D}$ (the degree matrix) has diagonal entries $d_i = \sum_{j=1}^{n} a_{ij}$ where a_{ij} denotes the affinity matrix's element at row i and column j. Then, $\mathbf{L} = \mathbf{D^{-1/2}AD^{-1/2}}$ (normalized), or $\mathbf{L} = \mathbf{D} - \mathbf{A}$ (unnormalized).
- **Eigen Decomposition**: Laplacian matrix $\mathbf{L}$ is decomposed into a matrix of eigenvectors $\mathbf{X}$ and a diagonal matrix $\mathbf{E}$ of eigenvalues, i.e. $\mathbf{L} = \mathbf{XEX}^\top$.
- **Speaker Counting**: Maximum eigengap is found and the speaker number is estimated.
- **k-means Clustering**: $\mathbf{U} \in \mathbb{R}^{m \times n}$ is formed by using the k-smallest eigenvalues and their corresponding eigenvectors. The row vectors in $\mathbf{U}$ are clustered by k-means.

Some of the diarization models that employ spectral clustering are [106, 55, 52, 68].

2.1.2.6 Post Processing

Once clustering has been carried out, we get the diarization results. An additional step of post-processing can be leveraged to further improve these results. It is of two types: *Resegmentation*, where we refine the results obtained by a diarization system, and *System Fusion*, which combines the results from multiple diarization systems. We shall now look at these two approaches.

Resegmentation refines the approximate speaker boundary calculated by the clustering method.Viterbi resegmentation had been used in the diarization systems [43]. It is based on the Baum-Welch algorithm, where we estimate the GMM for each speaker and apply the Viterbi technique alternately.

A resegmentation technique based on Variational Bayesian Hidden Markov Model (VB-HMM) later emerged and proved to be better than the Viterbi resegmentation [14, 15, 16]. This technique has since then been widely applied as the final step in a diarization pipeline [84, 86].

System Fusion deals with the combination of diarization results from the multiple systems which increases accuracy. However, unlike other speech tasks, system fusion in diarization suffers from various issues due to the non-standard speaker labels in each system. The estimated number of speakers and the segment time boundaies may also differ among different diarization models. Different fusion techniques tackle these problems differently. [36] proposed an AHC-based fusion approach using the symmetric DER as the distance function. [7] combines two diarization results by finding the matching between them and then performing resegmentation on the matching

result. A very recent diarization fusion scheme is DOVER (diarization output voting error reduction) [96] which aggregates the different results based on a voting scheme.

2.2 Recent Advances in Speaker Diarization using Deep Learning

In this section we shall discuss various Neural Network based diarization methods which have been recently proposed. We will organise our discussion by first outlining the models which utilize neural networks in one component of the diarization pipeline in section 2.2.1. Then, in section 2.2.2, we will see the models which combine multiple parts of diarization pipeline and optimize them jointly.

2.2.1 Single-Module Optimization

The models which involve neural networks in one module of the pipeline can be categorized based on where the neural approach was used. We shall see two stages where DNNs have been used: DNN-based clustering and DNN-based post processing. We shall cover each of them one by one.

DNN-based Clustering: Improved Deep Embedded Clustering (IDEC) was introduced in [17]. The goal of this algorithm is to modify the input features, i.e. speaker embeddings, to make them more separable. The key notion is that each speaker embedding has some probability of being in each of the available clusters [107, 28]. The clusters are refined in iteration based on a target distribution which relies on critical features estimated by an autoencoder. [105] proposed an approach to purify the affinity matrix in spectral clustering based on Graph Neural Networks (GNNs). [52] computes the affinity matrix by learning the distance function through a bidirectional LSTM network. We shall see the details of its working in next chapter. [51] is an improvement over the same model which incorporates self attention into the network.

DNN-based post processing: It has recently been studied to train a neural network on top of the clustering-based diarization. These approaches are an extension of post processing. Target-Speaker Voice Activity Detection (TS-VAD) has been proposed to diarize the audio in overlapping and noisy conditions [58, 59]. It inputs the MFCCs and i-vectors of all the possible speakers. The output layer outputs a vector whose i-th element is 1 if the i-th speaker is present in the segment, else it is 0. The initialization of i-vectors takes place based on the usual clustering based diarization result. The inference by TS-VAD and refinement of the i-vector by the result can be repeated till convergence. TS-VAD significantly improved the DER over clustering based approaches [58, 71]. As a separate effort, EEND model (discussed in 2.2.2.3) was introduced

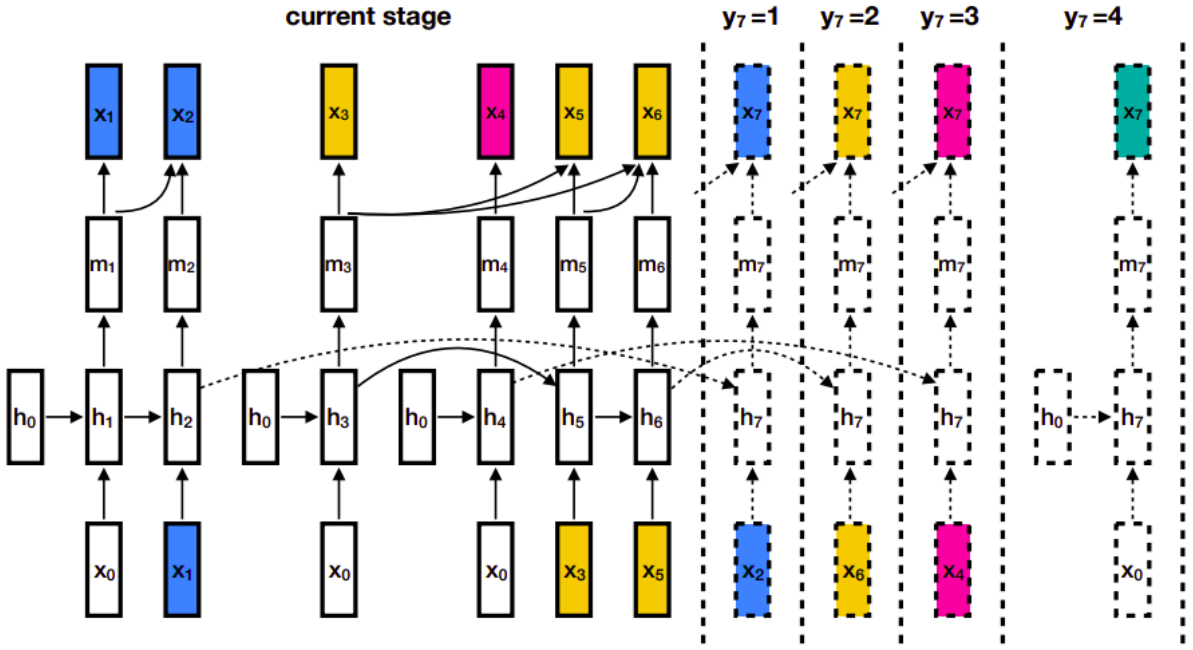


FIGURE 2.5: UIS RNN's generative process. Colors indicate labels for speaker segments. Given $x_{[6]}$ & $y_{[6]}$, there are four options for y_7.

which refined the result of a clustering based diarization [33]. The characteristic of a clustering based approach is that it can handle large number of speakers while it cannot efficiently manage overlapped speech. EEND has exactly opposite characteristics. In an iterative fashion, authors first apply a clustering approach and then utilize a two-speaker EEND per detected speaker-pair for refining the time boundaries of overlapped regions.

2.2.2 Joint Optimization for Speaker Diarization

In this section, we shall witness the diarization techniques which combine multiple parts of the pipeline and optimize them jointly as a single network.

2.2.2.1 Joint Segmentation and Clustering

Unbounded Interleaved-State Recurrent Neural Network (UIS-RNN) has been proposed by the Google Speech team [111]. It eliminates the segmentation and clustering steps and does them together by the means of an RNN. The motivation is that clustering algorithms cannot learn from data and neglect the temporal order of segments. Hence, it makes sense to take the temporal order of segments into account while predicting speaker labels, which is precisely what UIS-RNN does.

Denote the speaker embeddings by $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^d \mid t = 1, ...T)$, the speaker labels by $\mathbf{Y} = (y_t \in \mathbb{N} \mid t =, ...T)$, and speaker change indicators by $\mathbf{Z} = (z_t \in \{0, 1\} \mid t = 2, ...T)$. The idea is to model a joint probability distribution $P(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ as follows:

$$P(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = P(\mathbf{x}_1, y_1).\prod_{t=2}^{T} P(\mathbf{x}_t, y_t, z_t \mid \mathbf{x}_{[t-1]}, y_{[t-1]}, z_{[t-1]}),$$

where $[n]$ denotes the ordered set $(1, 2, ...n)$.

Each product term here can be factorized into three meaningful terms:

$$\begin{aligned} &P(\mathbf{x}_t, y_t, z_t \mid \mathbf{x}_{[t-1]}, y_{[t-1]}, z_{[t-1]}) \\ &= P(\mathbf{x}_t \mid \mathbf{x}_{[t-1]}, y_{[t]})).P(y_t \mid z_{[t]}, y_{[t-1]}).P(z_t \mid z_{[t-1]}), \end{aligned}$$

where the three resulting product terms respectively mean the following.

Speaker generation: It is assumed that each speaker's embedding is generated by a parameter sharing RNN (a gated-recurrent unit (GRU)).
Speaker assignment: It is modeled as distance-dependent Chinese Restaurant Process [5].
Speaker change: z_t is a binary random variable modeled as a coin-flipping process.

At any time t, having seen k speakers, if the next speaker is new, then we instantiate the RNN. If the speaker has already been seen before, then we update the existing RNN instance of that speaker. The model is trained by maximizing the log likelihood of the distribution $P(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ conditioned on all the parameters.

Inference is conducted by finding $\mathbf{Y}$ that maximizes $log\ P(\mathbf{X}, \mathbf{Y})$ provided $\mathbf{X}$ depending on beam search carried out in online fashion. Even though it works online, UIS-RNN showed better DER than the offline counterparts which use spectral clustering.

UIS-RNN in briefly summarised in Figure 2.5. At the current stage (shown by solid lines) $y_{[6]} =$ (1, 1, 2, 3, 2, 2). There are four options for y_7: 1, 2, 3 (existing speakers), and 4 (a new speaker). The probability for generating a new observation x_7 (shown by dashed lines) is dependent both on previous label assignment sequence $y_{[6]}$, and previous observation sequence $x_{[6]}$.

An improvement over UIS-RNN was introduced in paper [22]. It was called UIS-RNN-SML and it put forth a novel loss function, called the Sample Mean Loss (SML), and presented a better modeling of the speaker turn behaviour.

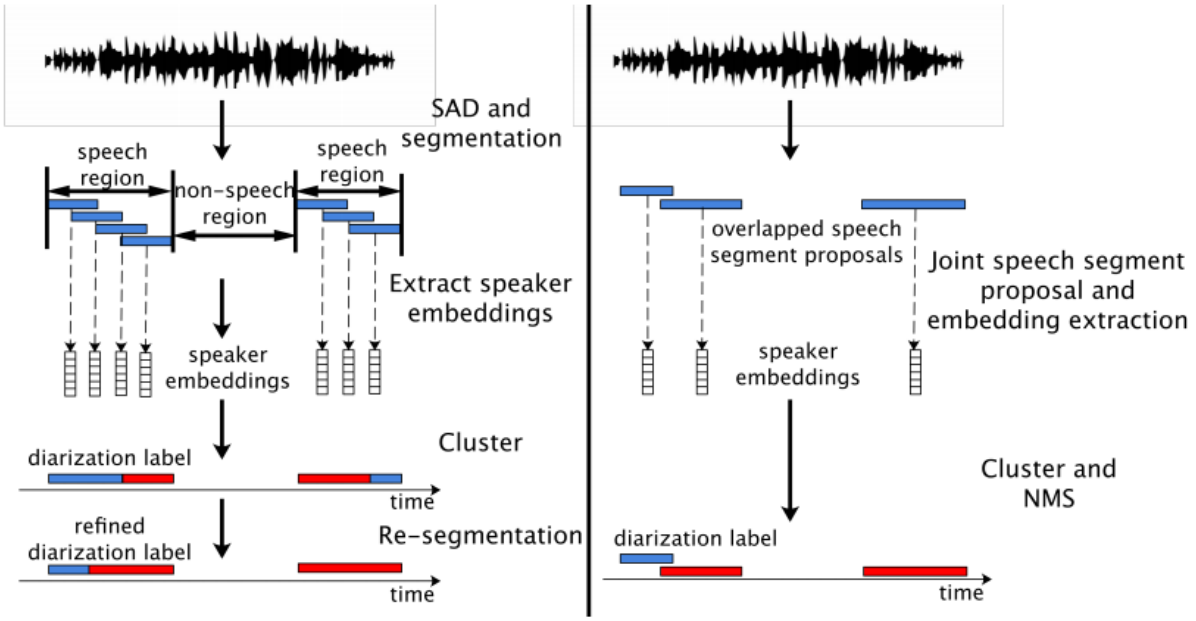


FIGURE 2.6: Difference between the standard diarization system pipeline (left) and the RPNSD system (right).

2.2.2.2 Joint Segmentation, Embedding extraction, and Resegmentation

Region Proposal Network based Speaker Diarization (RPNSD) was proposed based on Region Proposal Networks (RPN) [35]. It jointly optimizes the segmentation, embedding extraction and resegmentation stages of the pipeline via a single network. Region Proposal Networks were originally introduced in the object detection domain [72]. 1-d variant of RPN is used in diarization to give time segment proposals. RPNSD inputs STFT features and converts them into a feature map. For each time segment proposal with speech activity (referred to as 'anchor'), the network jointly performs tasks for (a) estimating whether anchor has speech or not, (b) extracting the speaker embedding for the anchor, and (c) approximating the difference between the center of reference speech and anchor. These three tasks respectively correspond to the segmentation, extraction of embedding and resegmentation. Difference between the RPNSD and other diarization systems is shown in Figure 2.6.

The inference procedure involves the RPN being applied to each of the anchors on test audio data, and listing the regions of higher speech probability as candidate regions. These regions are clustered using k-means based on their speaker embeddings. Finally, non-maximum suppression (NMS) is applied to remove segments with very high overlap. The advantage of this approach is that it can handle overlapped speech with arbitrary number of speakers. The authors empirically show in the paper that RPNSD outperformed clustering-based systems on multiple datasets [35, 71]. We set up RPNSD ourselves and evaluated it on CALLHOME dataset using 5-fold

validation. We got DERs of 12.11% & 16.75% respectively for without and with overlapped speech consideration. The set up can be found in the Google Colab Notebook[3].

2.2.2.3 Fully end-to-end Neural Diarization

End-to-End Neural Diarization (EEND) has recently been proposed and it aims to perform all the tasks of a diarization pipeline by a single network which can be trained end-to-end [23, 24]. The network of EEND is shown on the left of Figure 2.7. EEND takes as input the sequence of sound features $\mathbf{X} = (\mathbf{x}_t \in \mathbb{R}^F \mid t = 1, ...T)$. The system then outputs the speaker label sequence $\mathbf{Y} = (\mathbf{y}_t \mid t = 1, ...T)$ where $\mathbf{y}_t = [y_{t,k} \in 0, 1 \mid k = 1, ...K]$. $y_{t,k} = 1$ denotes the speech activity of speaker k at time t. K denotes the maximum number of allowed speakers. Overlapping speech is naturally accommodated in the above description since y_{t,k_1} and y_{t,k_2} $(k_1 \neq k_2)$ can both be 1 simultaneously at time frame t.

The network is trained by maximizing the log likelihood of $P(\mathbf{Y} \mid \mathbf{X})$ over the training data under the approximation $log\ P(\mathbf{Y} \mid \mathbf{X}) \sim \sum_t \sum_k\ log\ P(y_{t,k} \mid \mathbf{X})$. Under this model, there can be multiple sequences for $\mathbf{Y}$, all differing in swaps of speaker indices k from each other. The loss function is calculated for all the possible reference labels and the one which corresponds to minimum loss is used to update the network's weights. This is referred to as the Permutation Invariant Training (PIT) loss, and is inspired by the permutation free objective in speech separation [47]. The initial proposal of EEND was with a Bi-LSTM network [23], which was later extended to incorporate self attention based Transformer architecture [24]. The differences between the two architectures are shown in Figure 2.7.

EEND has advantages of dealing with overlapped speech and of optimizing the DER metric directly. EEND outperforms in the settings where maximum number of speaker is bounded by a small value. Note that K, the maximum number of speakers, must be known to the model in advance and the loss computation over all the permutations in PIT is of exponential complexity. Therefore, EEND cannot be applied to real world scenario where inference time is crucial. To deal with arbitrary number of speakers, an extension of EEND was proposed with encoder-decoder based attractors (EDA) [34]. An LSTM-based encoder-decoder is applied on the EEND output for generating multiple attractors. Attractors are generated until the probability of existing attractors goes below a threshold. Each attractor is then multiplied with the embeddings generated from EEND for computing speech activity of each speaker.

According to the paper [24], EEND with self attention attains a DER of 10.99% on CALLHOME. This is worse than 7.06% and 7.12% achieved by kaldi's [70] x-vector diarization system and LSTM-SC [52] model respectively (refer Tables 3.1 & 3.2). Also, LSTM-SC is simpler than

[3]Google Colab Notebook for RPNSD.

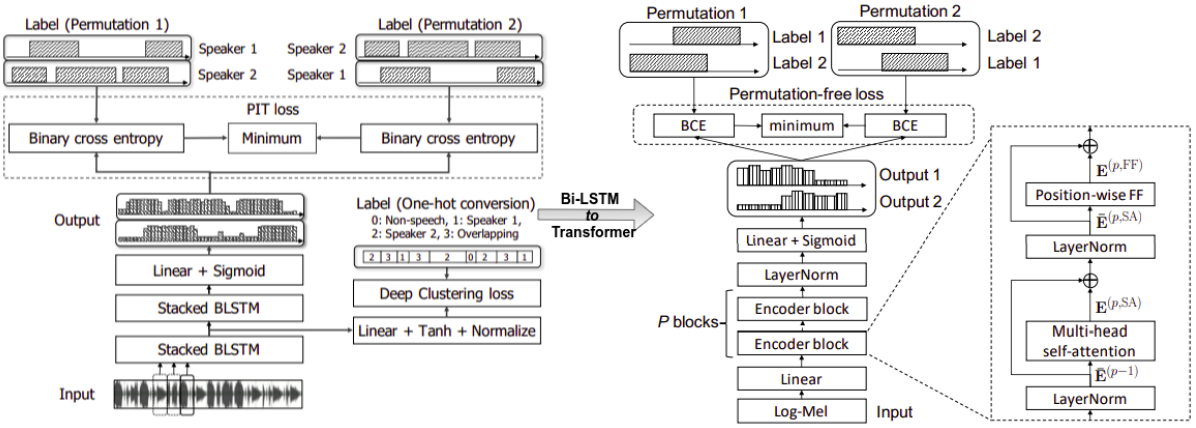


FIGURE 2.7: Difference between the EEND (left) and EEND-SA (right) architectures. EEND-SA results by replacing the Bi-LSTM block by a transformer block in EEND.

EEND in its base architecture. Therefore, we chose the same as our baseline and built the fully pythonic pipeline using it.

Starting from the next chapter, we shall be focussing on the LSTM based Similarity Measurement [52] diarization approach. Chapter 3 will discuss its theoretical background and results, while in Chapter 4, we shall use it to build our fully pythonic diarization pipeline.

2.3 Evaluation

Having seen the history and development of the Speaker Diarization, we now divert our attention to the evaluation of these systems. The output of any diarization system is the Rich Transcription Time Marked (RTTM) file. It is also called the hypothesis, since this is what our model hypothesizes. It maps the time segments with speaker labels. A ground truth RTTM file (also called the reference RTTM), which is provided along with the dataset, is then used to match with the hypothesis and yield a numerical value quantifying the performance of the model.

First, we shall see the details of the RTTM file. Next, we shall look into the various datasets available for diarization. Then, we see two metrics viz. DER and JER, used to score the hypothesis RTTM using the reference RTTM.

Following are the few lines from a RTTM file.

```
SPEAKER CMU_20020319-1400_d01_NONE 1 130.430000 2.350 <NA> <NA> juliet <NA> <NA>
SPEAKER CMU_20020319-1400_d01_NONE 1 157.610000 3.060 <NA> <NA> tbc <NA> <NA>
SPEAKER CMU_20020319-1400_d01_NONE 1 130.490000 0.450 <NA> <NA> chek <NA> <NA>
```

Let us see what each of the columns actually mean (*The following description is adapted from [78]*).

- **Type**: segment type; should always by **SPEAKER**.
- **File ID**: file name; basename of the recording minus extension (e.g.,**rec1_a**).
- **Channel ID**: channel that turn is on; should match between the hypothesis and reference.
- **Turn Onset**: onset of turn in seconds from beginning of recording.
- **Turn Duration**: duration of turn in seconds.
- **Orthography Field**: should always by **<NA>**.
- **Speaker Type**: should always be **<NA>**.
- **Speaker Name**: name of speaker of turn; should be unique within scope of each file.
- **Confidence Score**: system confidence (probability) that information is correct; should always be **<NA>**.
- **Signal Lookahead Time**: should always be **<NA>**.

2.3.1 Datasets

The task of diarization is non-trivial. Even after the presence of a widespread number of approaches, none of them performs best in all voice environments. This is why we have different types of datasets based on the environments they were recorded in. Another aspect in which the diarization datasets differ is the sampling rate of the audio. Traditional datasets primarily used 8 kHz sampling frequency since storage was a constraint. This is evident by the fact that NIST's datasets prior to 2010 were in 8 kHz. Nowadays 16 kHz is the normal sampling rate used.

It must also be noted that due to the evidently hard nature of the diarization task, the datasets cannot be automatically annotated. This means all of the datasets must be annotated manually by humans. This is reason that only a few large scale diarization datasets are available.

It is also common practice to synthetically generate diarization datasets and evaluate the model using them. To that end, NIST's SRE-datasets are used. We shall see an algorithm to generate the synthetic dataset in section 4.2.1.2.

Below are mentioned a few of the prominent datasets used to train diarization models:

- 2000 NIST Speaker Recognition Evaluation[4] (paid).
  - Disk-6 (Switchboard).

[4] https://catalog.ldc.upenn.edu/LDC2001S97

  - Disk-8 (CALLHOME) - multilingual.

- CALLHOME American English[5] -English (paid).
- The ICSI Meeting Corpus[6] (freely available).
- The AMI Meeting Corpus[7] - English (freely available).
- VoxConverse [11] - English (freely available).
- DIHARD Datsets[8].

Since we shall be using Disk 8 - SRE 2000 CALLHOME (LDC2001S97) extensively, we must note that this dataset contains multiple languages including English and Spanish. The number of speakers ranges from 2-7. The number of audio files with 2, 3, 4, 5, 6 & 7 speakers are 303, 136, 43, 10, 6 & 2 respectively. The dataset contains a total of 500 utterances which add up to a total play time of about 18 hours.

2.3.2 Metrics

To evaluate the performance of diarization systems, we use the Diarization Error Rate (DER). A less frequently used metric, the Jaccard Error Rate, which is based on the Jaccard index[9], is also used. It was first used in DIHARD II [80, 81] challenge alongside DER for a holistic evaluation.

The most commonly used script to compute the DER is the NIST's **md-eval.pl**[10]. It takes as input the collar value (i.e. the error while matching time segments we are willing to tolerate; usually 0.25s; '-c' parameter), whether we want to consider the overlapped speech segments or not (parameter '-1' and '-o' respectively) and the path to the reference and the hypothesis RTTMs. A sample call to the script is as follows:

```
$ md-eval.pl -o -c 0.25 -r reference.rttm -s hypothesis.rttm 2> DER.log > DER.txt
```

Below we see the evaluation criteria of DER and JER in detail.

[5] https://catalog.ldc.upenn.edu/LDC97S42
[6] https://groups.inf.ed.ac.uk/ami/icsi/
[7] https://groups.inf.ed.ac.uk/ami/corpus/
[8] https://dihardchallenge.github.io/dihard3/
[9] Jaccard Index wikipedia link.
[10] md-eval.pl script link.

2.3.2.1 DER

Diarization Error Rate (DER) is the main metric used to evaluate Speaker Diarization systems. It was described and used by NIST in their 2006 RT (Rich Transcriptions) evaluations. Put simply, DER measures the time wrongly assigned as non-speech or assigned to an incorrect speaker. The measurement is done by md-eval.pl script provided by NIST [3]. Other open source alternatives are mentioned in section 2.4 [78].

Since the problem of diarization is not to identify the speaker but rather to just discriminate different speakers, the ID labels assigned to speakers in reference and hypothesis RTTM need not be same. However, the non-speech tags must identically match between the reference and hypothesis RTTM.

md-eval.pl does the optimum one-to-one mapping of speaker labels between hypothesis and reference RTTMs. Based on this mapping the scoring takes place. The Diarization Error Rate is computed as:

$$DER = \frac{\sum_{s=1}^{S} dur(s).(max(N_{ref}(s), N_{hyp}(s)) - N_{correct}(s))}{\sum_{s=1}^{S} dur(s).N_{ref}}$$

where, S is the number of speech segments in which both the reference and hypothesis have the identical speakers. $N_{ref}(s)$ and $N_{hyp}(s)$ denote the number of speakers in a segment s. $N_{correct}(s)$ is the number of speakers which are in a segment s and have been matched correctly between the reference and hypothesis.

It is easier to see the DER by decomposing it into the different sources that these errors come from:

- **Speaker Error**: % of time that a speaker is misidentified. This error only accounts for the segments which were detected as speech and for the segments where overlap was detected succesfully. We denote it by E_{Spkr}.
- **False alarm speech**: % of time that a segment is wrongly labelled as speech in the hypothesis. It is denoted by E_{FA}.
- **Missed speech**: % of time that a segment is wrongly labelled as non-speech in the hypothesis. It is denoted by E_{MISS}.
- **Overlap speaker**: % of time that some of the many speakers in a segment are not assigned to any speaker. It is denoted by E_{ovl}.

Given these types of errors, we can redefine DER by the following equation:

$$DER = E_{spkr} + E_{MISS} + E_{FA} + E_{ovl}$$

2.3.2.2 JER

Jaccard Error Rate (JER) is based on the Jaccard Similarity Index, which is a metric used commonly for evaluating the output of image segmentation systems. It is defined as the ratio of the sizes of intersections and unions of two sets of segments. Similar to the case with DER computation, an optimal mapping between speakers in the reference and hypothesis is determined. Then, for each pair the Jaccard index of the segmentations is calculated. JER is expressed as a percentage and given as 1 minus the mean of these scores over all speaker pairs [79].

$$JER_{ref} = \frac{FA + MISS}{TOTAL}$$

Hence, JER is the average of the above equation over all the reference speakers ref. $TOTAL$ is the duration of union of hypothesis and reference speaker segments. FA and MISS are False Alarm and Missed Speech as defined above in section 2.3.2.1.

2.4 Open Source Toolkits for Speaker Diarization

The various modules of the diarization pieline and the evaluation metrics are implemented in various open source toolkits. This eases our task of not having to implement everything ourselves. The most popular open source toolkits prevalent in the community are mentioned below along with the specialities of each toolkit.

The task of speaker diarization, as we saw starts with **Voice Activity Detection (VAD)**. The most popular voice activity detectors are the **Pythonic WebRTC VAD**[11] by Google and the kaldi's energy based VAD which is applied as a part of kaldi's recipe[12].

Audio feature extraction and Augmentation can be achieved by using the popular audio processing library **librosa** [57] or PyTorch's [69] own audio processing framework viz. **torchaudio**. Figure 2.8 shows the MFCCs of a CALLHOME recording extracted using librosa.

The **clustering** module of Speaker Diarization pipeline has been widely studied. A few of the GitHub implementations of the clustering algorithms from the approaches seen in section 2.1 are given below:

[11] Google WebRTC VAD GitHub link.
[12] kaldi diarization recipe GitHub link.

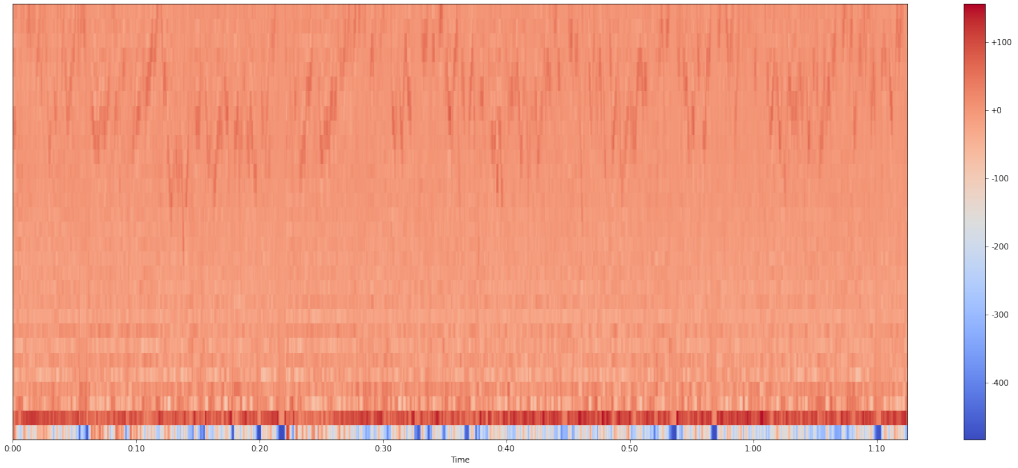


FIGURE 2.8: Visualization of MFCCs (30 mel bands) for CALLHOME audio recording iaaa using librosa.

- Probabilistic Linear Discriminant Analysis implementation in Python[13], implementation of the paper [37].
- Supervised Online Clustering[14], an implementation of the UIS-RNN [111].
- Supervised Online Clustering[15], an implementation of the UIS-RNN-SML [22].
- Discriminative Neural Clustering for Speaker Diarisation [50] implementation[16].
- Spectral Clustering[17], as implemented in the paper [106].

For **evaluating** the output RTTM, NIST's **md-eval.pl** is the most widely used script as mentioned in the section 2.3 above. However, for Python based evaluation **pyannote-metrics**[18] and **nyrant-descore**[19] are increasingly gaining popularity.

There are several software which implement all parts of the pipeline and the three most used ones are **kaldi-asr** [70], **pyannote-audio** [8], and **LIUM SpkDiarization** [60]. **kaldi-asr** is the most eature rich speech processing library. It is written in C++, bash, Python and Perl. For diarization, it provides two recipes, one using i-vectors and the other using the x-vectors. **pyyannote-audio** is a great toolkit which is still in development. It is not very flexible and can only be used as-is like an executable using its Command Line Interface (CLI). **LIUM SpkDiarization** dates back to 2010 and was one of the first toolkits for diarization. It is however written in Java and therefore relatively less used nowadays.

[13] Python PLDA link.
[14] uis-rnn link.
[15] uis-rnn-sml link.
[16] DNC link.
[17] google spectral clustering link.
[18] Pyannote-metrics GitHub link.
[19] nyrant dscore GitHub link.

Chapter 3

Baselines: Speaker Diarization System

3.1 Setting up Baselines

Setting up the baselines is an important step in any machine learning research. It is where we start our improvements from. In this chapter we shall see the establishment of two baselines which give state-of-the-art results on CALLHOME. The two baselines are kaldi's [70] x-vector baseline and the LSTM based Similarity Measurement (LSTM-SC) [52] baseline. To verify these baselines, we evaluate them on CALLHOME and report the DER. kaldi is evaluated on 2-fold CALLHOME, while LSTM-SC is evaluated on 5-fold CALLHOME.

The rest of this chapter is organised as follows:
Sections 3.1.1 and 3.1.2 describe the theory behind the two baseline systems.

Section 3.2 describes the details of the setup used to perform the experiments and section 3.3 mentions the results of these experiments.

Lastly, we summarise our observations and findings in the section 3.4.

3.1.1 kaldi x-vector system

Figure 3.1 represents an overview of the kaldi [70] diarization pipeline. We shall see each part of the pipeline in a little more detail:

- **Feature Extraction**: The features of choice used by kaldi are the Mel-Frequency Cepstral Coefficients (MFCCs). The Cepstral Mean and Variance Normalization (CMVN) is also

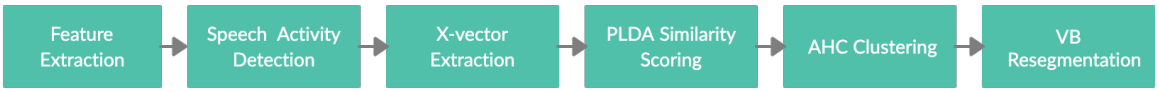


FIGURE 3.1: Overview of kaldi's diarization pipeline.

applied to normalize the MFCCs. MFCCs favour the sound that are best heard by huamn ears. Therefore, their use is prevalent to produce state-of-the-art systems.

- **Speech Activity Detection**: Kaldi uses an energy based SAD. The signal energy is used to classify a segment as speech or non-speech.
- **X-vector extraction**: The x-vector extractor used by kaldi is same as in Figure 2.4. It is a Time Delay Neural Network (TDNN) with frame-level layers, statistical pooling layers, fully connected (FC) layers and the softmax layers. The frame level layers are essentially FC layers taking the input from the sliding window across the audio sample. Since each frame produces a different vector, we concatenate their means and standard deviations to get one single vector which characterizes the entire segment. The vector is fed through FC layers and finally a softmax callifier is applied after ReLU. kaldi produces 128-dimensional x-vectors.
- **PLDA Similarity Scoring**: Once we have the x-vector representations of our windowed segments, we can use a similarity metric (PLDA in this case) on them to compute affinity between different segments.
- **AHC Clustering**: These segment affinities serve as the input to the AHC clustering module which groups segments of same speakers together.
- **VB Resegmentation**: Resegmentation is a refinement step on top of the clustering output. VB resegmentation treats the speech features to be coming from a Hidden Markov Model. This models the diarization problem as an inference problem, maximizing the posterior distribution.

3.1.2 LSTM based Similarity Measurement

LSTM-SC pipeline is special in that it is simple to understand and easy to experiment with. At the heart of it is the Bi-LSTM sequence model used to learn the similarity metric. It also produces state-of-tha-art results on CALLHOME dataset. Therefore, we chose this as one of our baselines. It provides a firm ground over which efficient diarization systems can be built.

We shall first see an overview of the pipeline as in Figure 3.2 and then dive deep into each of the aspects.

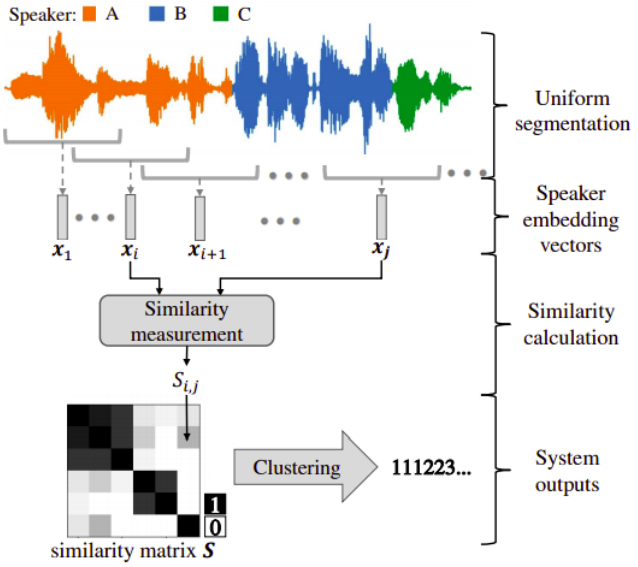


FIGURE 3.2: LSTM-SC framework for speaker diarization.

Oracle VAD is used to remove non-speech regions. **Sliding window with overlap** is used to uniformly segment the audio signal. From each of the segments, **MFCCs** are extracted and **CMVN** normalized. The MFCC features are used as an input to the **x-vector extractor**. The **x-vectors for one recording are then tiled** to create one training batch. This batch is passed to the **Bi-LSTM** which yields an **affinity matrix**. It is used for evaluating **crossentropy loss** with the reference matrix generated from the reference RTTM file. During testing, we cluster the output affinity matrix using **Spectral Clustering**. Finally, the **output RTTM** is generated based on clustering of the rows of the affinity matrix. [52]

The number of mel bands used for MFCC extraction is 23. Augmentations in the form of babble, noise, reverberation and music are added to the audio data to ensure robustness. Let us denote the segment level x-vector embeddings as x_1, x_2, ... x_n, where each x_i is a d-dimensional vector, d being 128 for kaldi's x-vector. We assume that each of these segments correspond to single speaker. This assumption is good enough as long as our segmentation is fine enough (i.e. we have smaller window length). Further details about the approach are mentioned below.

- **Construction of reference similarity matrix**: The reference similarity matrix S, for a recording R, is of shape $n \times n$. n is the number of segments extracted by windowing over the recording R. It can be determined using the reference RTTM whether segment i and segment j $(i, j \in 1, 2, .., n)$ have same speaker or different speakers. $S[i, j] = 1$ if segment i and segment j have same speaker, otherwise $S[i, j] = 0$. Note that by definition of S, it is robust against flipping or changing the index of the speaker. Therefore, S serves as a good label for the entire speaker embedding sequence x to train the Bi-LSTM model.

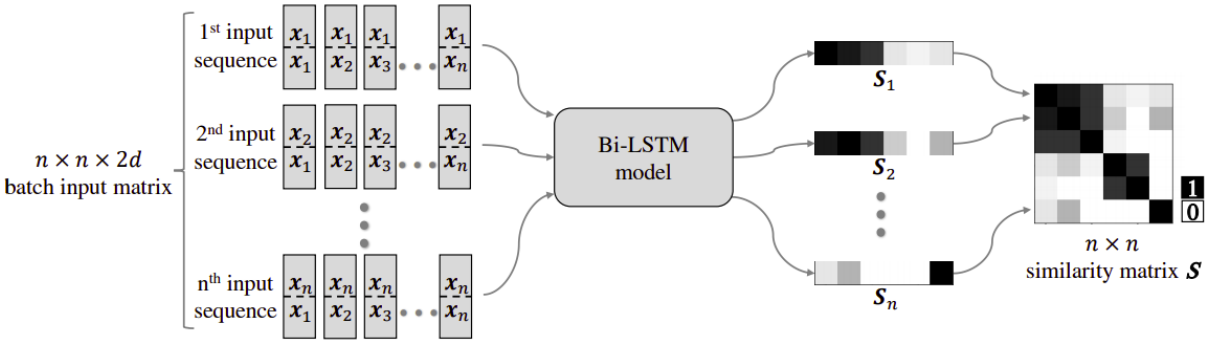


FIGURE 3.3: Bi-LSTM batch workflow.

- **Stacking and Tiling**: To predict an entry $S[i, j]$ in the similarity matrix, we need the x-vectors for segment i and segment j. Following similar chain of reasoning, to predict the row i of the similarity matrix, we need x_i paired with all the segments from x_1 to x_n.

$$S_i = [S_{i1}, S_{i2}, ...S_{in}] = f_{LSTM}\left(\begin{bmatrix} x_i \\ x_1 \end{bmatrix}, \begin{bmatrix} x_i \\ x_2 \end{bmatrix}, ... \begin{bmatrix} x_i \\ x_n \end{bmatrix} \right).$$

  Therefore, we duplicate each of the n x-vectors n times and tile them with x_1, x_2, ... x_n. This results in one batch of Bi-LSTM training as shown in Figure 3.3. Input sequence k is responsible to predict the k^{th} row of the similarity matrix S.

- **Mini-batching for larger matrices**: Note that the length of the audio is arbitrary and this can result in sequence x to be very large due to uniform segmentation. This ends up requiring huge GPU memory and also posing a computational challenge on the LSTM to process arbitrarily long sequences. To alleviate this, the authors propose to split the $n \times n \times 2d$ batch input matrix as well as the $n \times n$ similarity matrix S into 4 parts as shown in Figure 3.4. This does increase the number of batches that the model must train on but it ensures that the computation of the predictions is tractable and LSTM can sufficiently generalize. The practical limit set on n for such a split is 400. We split whenever n exceeds 400.

- **Spectral Clustering**: This step is identical to the Spectral Clustering procedure seen in section 2.1.2.5.

- **Network Architecture**: The specifics of the network are as follows. Two Bi-LSTM layers are followed by two fully connected layers. Since $d = 128$ for kaldi's x-vector, the input to the Bi-LSTM is $2d = 256$ dimensional. The hidden size is also kept identical to the input size and equals 256. Hence, Bi-LSTM produces 512 outputs (256 forward and backward hidden states). The fully connected layers gradually reduce the output dimension from $512 \rightarrow 64 \rightarrow 1$. The final layer's output is passed through a sigmoid activation function to yield a similarity score in (0,1).

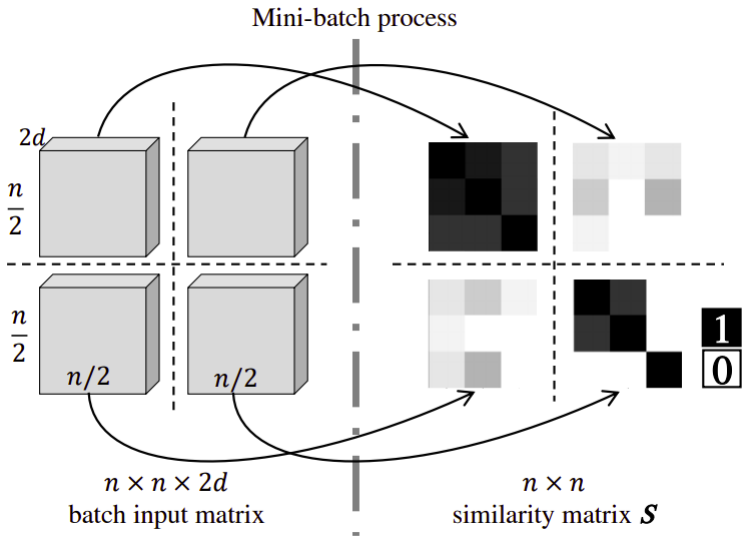


FIGURE 3.4: Partitioning of large matrix into multiple submatrices, which are processed as mini-batches.

- **Enhancement of Similarity Matrix**: Following [106], authors apply the matrix refinement operations to denoise and smoothen the similarity matrix. They, however, remove the Gaussian Blur step. Their procedure is as follows:
  - Symmetrization: $A_{i,j} = max(S_{ij}, S_{ji})$.
  - Diffusion: Assign AA^T to A.
  - Max-normalization (row-wise): $S_{ij} = A_{ij}/max_k Y_{ik}$
- **Data for training the x-vector system**: Authors train the x-vector models on a combinaton of Switchboard (SWBD) and SRE-databases. In particular, SRE 2004, 2005, 2006 & 2008 are used.

3.2 Experimental Setup

The experimental setup of kaldi is straigtforward for CALLHOME and involves following their diarization recipe. Dealing with Speech Recognition for the first time may cause some initial issues since kaldi has a very standardized way of organizing and producing data in form of its own file formats which may take some time to get used to.

We use the pre-trained PLDA backend of kaldi and use the pre-trained model for CALLHOME downloaded from the kaldi's website[1]. The data preparation and feature extraction steps are used as provided in the kaldi's *callhome_diarization* recipe. A sliding window of length 1.5s moves over audio with 50% overlap. After PLDA scoring and AHC clustering, VB resegmentation is

[1] kaldi's CALLHOME diarization pre-trained model.

used to refine the results. The original VB resegmentation script present in kaldi was erroneous and we had to go deeper into its implementation to find that *coo_matrix* from the SciPy library was missing a shape parameter. The same was fixed and a Pull Request (PR) was raised[2] which finally merged into kaldi's codebase.

LSTM-SC uses kaldi's front-end for feature extraction. Therefore, the steps until the generation of x-vectors are identical to the setup of kaldi pipeline above, and the same 1.5s sliding window with 50% overlap is used. The Bi-LSTM is trained for 100 epochs with an initial learning rate of 0.01. The authors have proposed a learning rate scheduler which sets the value of the rate to its 1/10th value every 40th epoch. The whole process takes place 5 times for the 5-fold valiation of the model. CALLHOME has 500 recordings and the data preparation script divides it into 5 folds each with 400 recordings for training and 100 for evaluation.

The collar value of 0.25 seconds is used throughout for DER evaluation.

3.3 Experimental Results

Table 3.1 and 3.2 lists the experimental results of kaldi's and LSTM-SC's diarization pipelines. In both cases, we evaluate the DER with and without considering overlapped speech. The results clearly show that simply by taking overlapped speech into account, the DER rises by $\sim 10\%$. Therefore, detection and management of overlapped speech in diarization is a separate research area in itself.

	DER(in %)	
No. of speakers	w/o overlap	w/ overlap
oracle	7.06	16.67
supervised calibration	8.03	17.5

TABLE 3.1: Diarization results produced by the kaldi x-vector diarization pipeline (2-fold CALLHOME). DER for with and without overlapped speech are shown for oracle number of speakers (row 1) as well as number of speakers found by supervised calibration (row 2).

	DER(in %)	
Resegmentation	w/o overlap	w/ overlap
N/A	8.91	18.09
VB resegmentation	7.12	16.65

TABLE 3.2: Diarization results produced by the LSTM-SC diarization pipeline (5-fold CALLHOME). DER for with and without overlapped speech are shown for the cases where VB resegmentation was (row 2) and was not (row 1) used.

In case of kaldi, there is an option to use the oracle number of speakers at the clustering step. If this is not provided, the model calibrates itself to find the optimum number of speakers and

[2]PR fixing the broken VB resegmentation in the kaldi codebase.

clusters using that. We provide the results for this option as well. As expected, the model performs better when it knows the correct number of speakers.
For LSTM-SC, we see the utility of VB resegmentation step. The results denote that about $\sim$ 1-2% improvement is seen when performing the resegmentation step.

3.4 Conclusion

We have discussed the two diarization pipelines (kaldi's x-vector and LSTM-SC) which achieve state-of-the-art DER on CALLHOME dataset. We saw their theoretical foundations and the experimental setup required to reproduce their results. The best DER of these pipelines are 7.06% and 7.12% respectively. We build upon LSTM-SC in the next chapter to construct our end-to-end pythonic pipeline.

Chapter 4

Fully Pythonic Pipeline for Speaker Diarization

4.1 Introduction

As we have seen in previous sections, the field of diarization is moving towards an end-to-end pipeline. The power of neural networks come from being able to optimize large architectures via Gradient Descent. As of today, most of the diarization pipelines depend on kaldi, which is written in C++, bash, Perl and Python. Even the ones that claim to be end-to-end, usually involve kaldi at some point in their processing or use the Permutation Invariant Training loss [23] which fails to scale in real world scenario.

We must note that as long as kaldi is a part of the speaker diarization pipeline, it cannot be made fully end-to-end. Therefore, it is necessary to remove kaldi from the process by substituting its utilities by their Pythonic counterparts. kaldi has been around for a long time and the diarization field might take some time to be completely kaldi-free.

This chapter will discuss our efforts to remove kaldi from the diarization pipeline of the LSTM-SC model discussed in previous chapter and to develop a fully pythonic pipeline.

We started our analysis by replacing the kaldi's x-vector extractor by a Pythonic TDNN (PyTDNN). This PyTDNN was trained on 16 kHz data from VoxCeleb [63] dataset and employed extensive augmentations in its training. The results of the experiment can be seen in Tables 4.2 and 4.3. These results are discussed in forthcoming sections.

The rest of this chapter is organised as follows.

In the section 4.2, we shall see our experimental setup and the models used in the experiments.

In section 4.3 we shall see the effects of data augmentation on x-vectors.

In section 4.4, we shall discuss the results of all of our experiments.

Lastly, section 4.5 will conclude the discussion.

4.2 Experimental Setup

The experimental setup in terms of code is similar to the previous chapter. Differences, wherever applicable will be mentioned appropriately. Code for all the experiments in this section can be found at the GitHub link[1].

4.2.1 Datasets

While CALLHOME stays as our diarization dataset of choice, we use the following two new datasets for specific purposes. The SWBD and SRE-databases, together referred to as the **telephony data**, comprise 8 kHz speech and are used to fine-tune the PyTDNN on 8 kHz speech.

SRE10 (8 kHz) and VoxCeleb1 (16 kHz) datasets are used for the development of alternate diarization datases which serve as an additional point of evaluation for our model.

In the next sections we delve into the specifics of these datasets.

4.2.1.1 Switchboard (SWBD) and SRE-databases

SWBD and SRE-databases were used by the authors of LSTM-SC [52] to train their x-vector extraction network. In particular, SRE 2004, 2005, 2006, 2008 were used.

Following the LSTM-SC authors, we also use the same databases to fine-tune our PyTDNN.

4.2.1.2 Synthetic Data from SRE10 and VoxCeleb1

To evaluate our model holistically, we generate synthetic data which alongside CALLHOME serves to evaluate our diarization model. Two synthetic datasets were prepared, one each from SRE10 and VoxCeleb1 [63]. The complete procedure followed for the generation of synthetic datasets is given in Algorithm 1.

Following are the inputs taken by the synthetic data generation algorithm 1:

[1] https://github.com/mdrpanwar/SpeakerDiarization

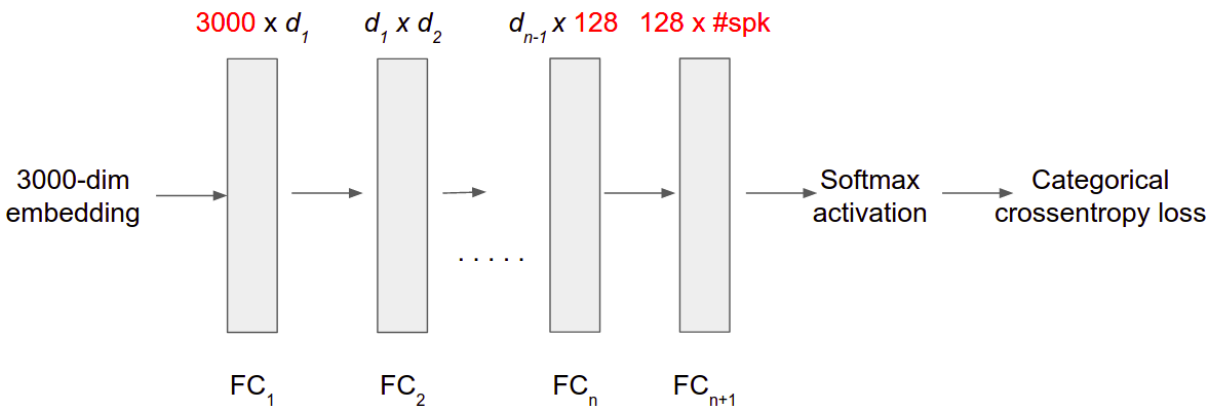


FIGURE 4.1: The architecture of the subsequent FC layers of PyTDNN during fine-tuning.

- *num_samples* - The number of synthetic samples to generate.
- *num_speakers* - The number of speakers to include in the synthetic samples.
- *max_num_seg* - Maximum number of segments per speaker.
- *min_olap_prob* and *max_olap_prob* - Minimum and Maximum overlap probabilities, taken into account while forming overlapping segments.
- *min_olap_len* and *max_olap_len* - Minimum and Maximum lengths desired for overlap (in seconds).
- *max_sil_len* - Maximun length of silence desired between speech segments.
- *sample_rate* - Sampling rate of the input base dataset from which the synthetic samples will be generated.
- *wav_dir* - Directory containing the base wav files.
- *normalise_wav* - flag denoting whether to normalise the synthetic samples in [-1, 1].
- *out_dir* - Directory where generated samples are saved.

In an effort to keep the statistics of the generated data in control, we kept the number of audio files for each speaker number identical to CALLHOME.

We also generate "simplified SRE10 data": a collection of 10 audio files generated from SRE10 without overlapping speech, with only 2 speakers, and containing at most 5 segments per speaker in one wav file.

4.2.2 Models

This section discusses the neural network models involved in our pipeline. These are the PyTDNN and the LSTM-SC model.

Algorithm 1 Algorithm for generating synthetic data from SRE10 and VoxCeleb1.

for each sample to be generated from *num_samples* **do**
  • Randomly choose *num_speakers* speakers and put in *chosen_spk*. These speakers will be used to form the synthetic audio.
  for each speaker S in *chosen_spk* **do**
    • Randomly choose a number *num_seg* between 1 and *max_num_seg* (both inclusive) to denote the number of speech segments from each speaker in the current sample.
    • Randomly choose *num_seg* segments from speaker S and append to list *syn_utt*.
  end for
  • Shuffle *syn_utt* and add silence of random duration in [0.001, *max_sil_len*) in between speech segments.
  • Randomly choose an overlap probability *oprob* from the user inputted bounds.
  • Randomly choose a maximum overlap length *olen* for the current synthetic sample from user inputted bounds.
  • Randomly remove silences with a probability of *oprob* and overlap the two adjacent speech segments with a random length in [*min_olap_len*, *olen*).
  if *normalise_wav* is ***true*** **then**
    • Normalise the synthetic wav in [-1, 1].
  end if
  • Generate the reference RTTM *syn_rttm* file for the syntehtic file.
  • Write the synthetic wav file and the *syn_rttm* to *out_dir*.
end for

4.2.2.1 Speaker Classification Model

This model is a softmax classification model developed to fine-tune the PyTDNN. The feature extractors used in diarization all come from the field of speaker verification. The training step of speaker verification systems is same as the softmax classification, except for the fact that for model selection, Equal Error Rate (EER) is monitored and not the classification accuracy.

We, however, monitor the classification accuracy only and notice that it also leads to a model which performs well. The architecture of this model can be seen in Figure 4.1. Note that the original PyTDNN has the same architecture as presented in the x-vector research paper [94], and it therefore generates 512-dimensional x-vector.

Following kaldi, we wanted to keep the x-vector dimension to be 128 and therefore we chopped off the final layers of the PyTDNN upto the 3000-dimensional embedding layer and and experimented with different architectures all of which had final layer dimension as 128. This is the interpretation of Figure 4.1.

We first extracted the 3000-dimensional embeddings for all the telephony recordings and then used this Multi-Layer Perceptron (MLP) model to learn the weights for the terminal layers. This fine-tuning classification model was written in tensorflow-keras [1] framework. Learning rates of 0.01 & 0.001 and various values of dropout were used with different architectures (based on the number of layers). The Table 4.4 shows the best results for each configuration.

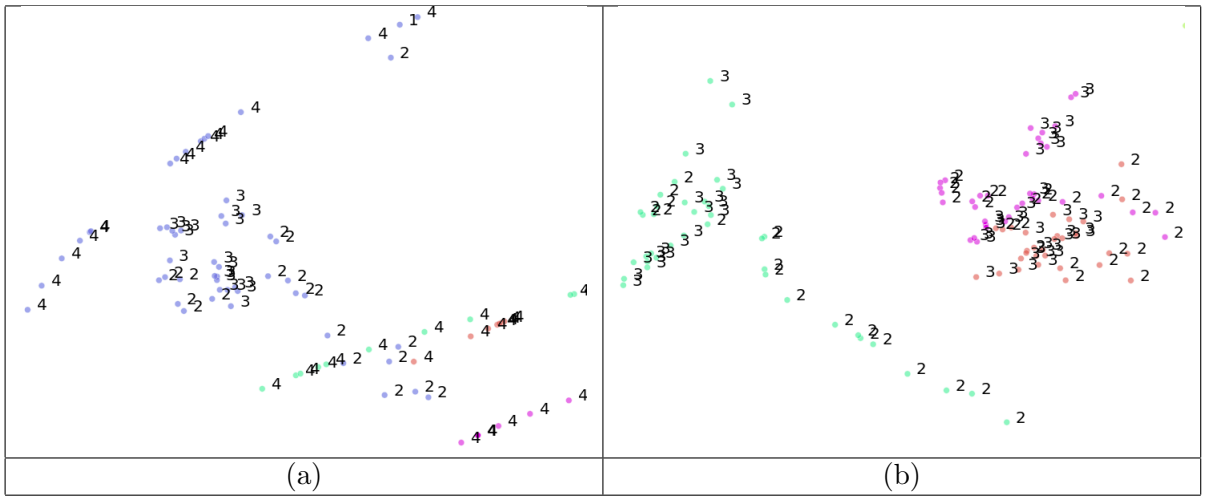


TABLE 4.1: Two sections (a) & (b) of the PCA visualization plot of x-vectors extracted from original PyTDNN for the 4 perturbation types from CALLHOME's utterances: **1** - original utterance, **2** - pitch shift perturbations, **3** - time stretch perturbations, **4** - amplitude perturbations. Different colours represent different utterances. It can be seen that amplitude perturbations are most diffcult for the network to cluster, followed by pitch shift and time stretch perturbations respectively.

To evaluate the effectiveness of the fine-tuned PyTDNN on CALLHOME, we extracted x-vectors by windowing over some of the CALLHOME's recordings and plotted them. The plots we shall see in Table 4.7 are for the window size of 5 cm and o.5 cm overlap. We experimented with plotting different segments from the following three types.

- **All segments**: Plotted all the segments that resulted from a sliding window.
- **Remove fully overlapping multiple speakers**: Removed those segments where for the entire duration multiple speakers spoke (since we cannot unambiguously assign these segments to a unique speaker). *The results in CALLHOME visualization plots in forthcoming sections correspond to this category of segments being plotted.*
- **Only single speaker**: Removed all segments with any overlap i.e. plotted only those segments where exactly one speaker spoke.

4.2.2.2 LSTM based Similarity Measurement Model

The setup of this model is identical to the one used in section 3.1.2, except for the few changes which are as follows.

- The input and hidden size of the Bi-LSTM depends on the size of the x-vector used. Therefore, in case of original PyTDNN, these sizes were changed to $512 \times 2 = 1024$.
- Instead of using the 5-fold validation, we just evaluate on 1-fold.

	LSTM-SC + kaldi x-vector (128-dim)	**LSTM-SC + original PyTDNN x-vector (512-dim)**
DER	9.29%	38.78%
Test Loss over 100 epochs	0.692 → 0.51 (dec. rapidly)	0.693 → 0.682 (dec. slowly)
Train Loss range after 100^{th} epoch	[0.10, 0.69]	[0.68, 0.70]

TABLE 4.2: **Experiment 1(a)**: Comparison of two LSTM-SC models: **1.** Using kaldi's 128-dim x-vector, **2.** Using original PyTDNN's 512-dim x-vector.

4.2.3 Evaluation Metric

For evaluating the Speaker Classification model, the classification accuracy was used as the metric, while for the LSTM based Similarity Measurement Model, DER was used, as defined in section 2.3.2.1.

4.3 Investigating the effects of Data Augmentation on Speaker Embeddings

We wanted to analyse the effect of different perturbations on the x-vectors extracted from original PyTDNN. This would tell us the proportions of different augmentations required while training our PyTDNN. Following steps were followed to generate the plots for investigating perturbations:

- Find 100 utterances of 10 seconds each using the segments file of CALLHOME dataset (therefore, all these utterances are sure to contain speech).
- Extract 61 x-vectors per utterance: 1 corresponding to original utterance, 20 pitch shifts uniformly sampled from [-4,4), 20 time stretches uniformly sampled from [0.8,1.2), 20 augmentations with gain sampled from [0.33,3). Thus, we get 6100 x-vectors in total.
- Fit t-SNE on these 6100 x-vectors of 512 dimensions each.
- Randomly sample 5 utterances to plot. 61 x 5 = 305 points will be present in each plot.

The next section describes the usage of librosa for data augmentations.

4.3.1 Data Augmentation using librosa

Librosa is a very popular framework used for feature extraction, plotting and exploring audio and music data. It provides extensive support for perturbing an audio using various built-in

S. No.	Experiment	DER (in %)
1.	Kaldi x-vector & LSTM-SC (5-folds)	8.91
2.	Kaldi x-vector & LSTM-SC (1-fold)	9.29
3.	PyTDNN & LSTM-SC (1-fold)	38.78
4.	Tuning the LR in 3. (8 LRs used in [0.01, 0.5])	31.71 (best value, occurring at LR = 0.4)
5.	Dim. reduction of Bi-LSTM input in 3. (from 1024 to 256)	38.78
6.	Upsampled (16 kHz) CALLHOME data in 3.	38.78

TABLE 4.3: **Experiment 1(b)**: Experimental results of LSTM-SC diarization experiments (on CALLHOME) after replacing the kaldi x-vector by PyTDNN. LR = learning rate. Dimensionality was reduced by means of FC layers.

methods. Below we see how pitch shift and time stretch perturbations can be applied using librosa.

- **Pitch Shift**:

```
# data contains original audio sample (for all examples),
# and sr is its sampling rate.
pitch_shifted = librosa.effects.pitch_shift(data, sr, n_steps=shift_step)
```

  We shift pitch by a random step sampled from [-4, 4).

- **Time Stretch**:

```
time_stretched = librosa.effects.time_stretch(data, stretch_step)
```

  We stretch time by a random step sampled from [0.8, 1.2).

Alongside pitch shift and time stretch, we also use the noise addition and amplitude perturbation. These augmentations do not require librosa and can be performed as follows.

- **Noise Addition**:

```
noise = numpy.random.randn(len(data))
noised_data =  data + noise_factor * noise
```

  The $noise\_factor$ used is randomly sampled from [0.002, 0.007).

- **Amplitude Perturbation**:

```
amp_data = data * gain_step
```

  The $gain\_step$ is randomly sampled from [0.33, 3).

n	**Architecture**	**Best train-test accuracies (in %)**	**Configuration**
2	d_1=128, d_2=128	82-76	LR=0.01, dp=0, b=32
3	d_1=128, d_2=128, d_3=128	77-71	LR=0.01, dp=0, b=64
3	d_1=512, d_2=256, d_3=128	68-77	LR=0.01, dp=0.1, b=64
3	**d_1=1024, d_2=256, d_3=128**	**73-79**	**LR=0.01, dp=0.1, b=32**
4	d_1=1024, d_2=512, d_3=256, d_4=128	79-72	LR=0.01, dp=0, b=64

TABLE 4.4: **Experiment 1(c)**: Experimental results for PyTDNN fine-tuning experiments on a subset of telephony data. (LR = learning rate, dp = dropout after each FC layer, b = batch size). n denotes the number of FC layers before softmax. Best configuration is boldfaced.

Regardless of the type of augmentation or even no augmentation, the x-vector embeddings that we need to plot are multi-dimensional. They cannot be visualized trivially on a 2-dimensional plane. Therefore, we need to resort to the dimensionality reduction techniques mentioned in the literature which efficiently bring the dimensions of x-vectors to 2 so that they can be plotted. PCA and t-SNE are two popular algorithms of choice for visualizing points in higher dimensional spaces by approximating their positions in the 2-dimensional space. In the next two sections we look at them in little more detail.

4.3.2 PCA

While visualizing these higher dimensional datasets, our goal is to retain as much variance as possible. It is the variability of the data that makes it what it is. Principal Component Analysis (PCA) is a standard tool in the field of exploratory data analysis. It is an orthogonal linear transformation which takes the data into a new coordinate system under the constraint that the most variance lies on the first coordinate (now called the first principal component). Similarly, second largest variance lies on the second coordinate and so on.

Let a $n \times p$ matrix X contain our data whose dimension we want to reduce. Here, n denotes the number of samples we have and p denotes their features. We also additionally assume that the mean of each column is zero i.e. the all the features averaged over the entire data result in zero mean. If this consition is not met, we can simply subtract the mean of the column from each entry and ensure that this condition is met [41].

The transformation is then given by a set of cardinality l of p-dimensional vectors, each of which denote weights. $w_{(k)} = (w_1, w_2, ..w_n)_{(k)}$. These weights map each row vector $x_{(i)}$ to a vector of scores for principal component $t_{(i)} = (t_1, t_2, ..t_l)_{(i)}$. The transformation is given by

$$t_{k(i)} = x_{(i)}.w_{(k)} \quad \forall\, i \in \{1, 2, ..n\} \quad \& \quad k \in \{1, 2, ..l\}$$

Model (Front-end + LSTM-SC)		
Front-end x-vector extractor	***Learning Rate***	***DER(in %)***
Kaldi (128-dim)	0.01	9.29
Original PyTDNN (512-dim)	0.01	38.78
	0.4	31.71
Fine-tuned PyTDNN (128-dim)	0.01	33.15
	0.4	31.37
	0.05	31.64

TABLE 4.5: **Experiment 1(d):** Results of LSTM-SC experiments (on CALLHOME) with different front-ends (100 epochs).

This is done such that each of the individual components of t retain maximum possible variance from the data, and w is constrained as a unit vector.

We use the implementation of PCA from the scikit-learn library's decomposition module.

4.3.3 t-SNE

t-distributed Stochastic Neighbor Embedding (or t-SNE) is another algorithm from the class of dimensionality reduction algorithms. However, it differs from PCA in that t-SNE is **nonlinear**. This means that while PCA can only efficiently work with a linearly separable data, t-SNE does not suffer from this restriction.

The essence of t-SNE is that it measures similarity between pair of points in higher as well as lower dimensional space. It then optimizes these similarity measures via a cost function. The authors Van der Maaten and Hinton explained: "The similarity of datapoint x_j to datapoint x_i, is the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i." [56]

Given N points x_1, x_2,..x_N, t-SNE computes the probabilities p_{ij} that are proportional to similarities of the data points. To do so, it first finds the conditional probabilities $p_{j|i}$ and then sets p_{ij} as follows.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

.

The aim of t-SNE is to learn a map $y_1, y_2, ..., y_N$ with $y_i \in \mathbb{R}^d$ which reflects p_{ij} similarities as well as possible. To do so, it measures similarities q_{ij} between points in map y_i and y_j using the approach similar to the computation of p_{ij}. Finally, the actual map y is determined by minimizing the Kullback-Leibler divergence between distributions P and Q using the Gradient descent algorithm.

Model [Front-end + LSTM-SC + specifics]			
Front-end x-vector extractor	***Specifics***	***DER(%) on SRE10***	***DER(%) on VoxCeleb1***
Original PyTDNN (512-dim)	synthetic data	35.55	36.85
Fine-tuned PyTDNN (128-dim)	synthetic data Pre-tr and simp-SRE10	32.97 36.76	35.82 N/A
Kaldi (128-dim)	Pre-tr and simp-SRE10	22.44	N/A

TABLE 4.6: **Experiment 2**: Results of LSTM-SC experiments with different front-ends (100 epochs, learning rate = 0.01) on synthetic diarization data prepared from SRE10 and VoxCeleb1. Pre-tr = LSTM-SC model pre-trained on CALLHOME (9.29% DER) and simp-SRE10 = the simplified version of synthetic data prepared from SRE10 as mentioned in section 4.2.1.2.

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \; log \; \frac{p_{ij}}{q_{ij}}$$

We use the implementation of t-SNE from the scikit-learn library's manifold module.

4.3.4 Results

The zoomed in sections from the PCA augmentation plots can be seen in Table 4.1. The perturbations of same utterance are all displayed in one colour and the type of perturbation is marked as mentioned in the caption. We make the following observations.

- The points corresponding to the same utterance tend to be close to each other.This is expected and desired.
- The time shift x-vectors corresponding to the same utterance tend to form one cluster, while the pitch shift points tend to split into multiple clusters.
- The points corresponding to the amplitude perturbations are spread all over the place. This tells us that the original PyTDNN cannot identify the amplitude augmented utterance effectively.

Overall, this proves that the original PyTDNN has trouble adapting to 8 kHz speech. The order of perturbations in which the PyTDNN faces difficulty in identifying the utterances is as follows:

$$time - stretch < pitch - shift << amplitude - perturbation$$

. One solution to alleviate this issue is to retrain the PyTDNN on 8 kHz speech with these augmentations being in the ratio 2:3:5.

FIGURE 4.2: **Experiment 3(a)**: x-vectors plotted for the 150 utterances each for the 10 out of 20 held-out speakers from the subset of telephony data. x-vectors were extracted from the model configuration giving best classification results.

4.4 Experimental Results

In this section, we discuss the results of the experiments performed above. The experiments we shall discuss are as follows.

Experiment 1:

- 1(a): Replacing the kaldi's x-vector network by PyTDNN: comparison of two LSTM-SC models.
- 1(b): Hyperparameter tuning of the original PyTDNN + LSTM-SC system.
- 1(c): Fine-tuning the PyTDNN on a subset of telephony data.
- 1(d): Evaluating the fine-tuned PyTDNN + LSTM-SC model on CALLHOME: experimenting with different front-end extractors.

Experiment 2: Evaluating the fine-tuned PyTDNN + LSTM-SC model on synthetic data: experimenting with different front-end extractors.

Experiment 3:

- 3(a): Visualization of x-vectors from fine-tuned PyTDNN for held-out set of speakers.
- 3(b): Visualization of x-vectors from fine-tuned PyTDNN for CALLHOME recordings.

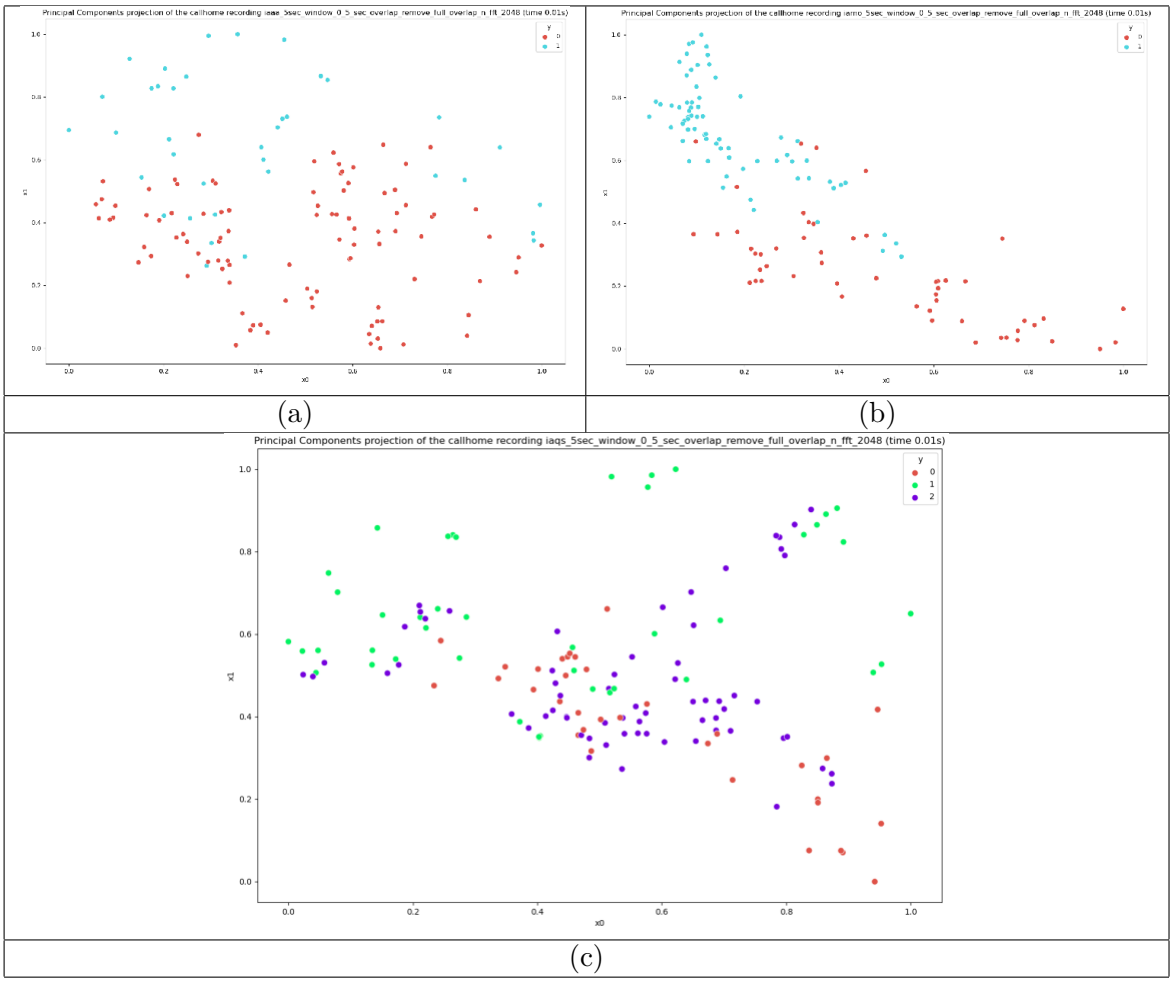


TABLE 4.7: **Experiment 3(b)**: Visualization of x-vectors from CALLHOME's recordings extracted using the PyTDNN fine-tuned on a subset of telephony data. (a) Recording **iaaa**, 2 speakers with 70%-30% speech duration. (b) Recording **iamo**, 2 speakers, 50%-50% speech duration. (c) Recording **iaqs**, 3 speakers, 20%-30%-50% speech duration.

4.4.1 Observations

We make the following **observations** from the results.

Experiment 1:

- 1(a): Table 4.2. Simply replacing the kaldi's TDNN by PyTDNN results in a high DER. The loss also tells that model is having difficulty to learn given the new x-vectors. We hypothesize that this is due to the mismatch in the sampling rates between the two x-vectors.
- 1(b): Table 4.3. Tuning the learning rate results in decreasing the DER by 7%. However, there is no effect of reducing the dimensionality of x-vectors or of upsampling the CALLHOME data.

- 1(c): Table 4.4. Of all the experiments performed for fine-tuning the PyTDNN, the best classification test accuracy is 79% .
- 1(d): Table 4.5. The fine-tuned PyTDNN + LSTM-SC when evaluated over CALLHOME, resulted in a decrease of about 5% in DER from 38.78% to 33.15%. This verifies our hypothesis about the mismatch in sampling rate. However, further tuning the LSTM-SC is not effective and DER does not go below 31%.

Experiment 2: Table 4.6. Fine-tuned PyTDNN + LSTM-SC evaluated on synthetic SRE10 data results in a DER decrease of about 3% from 35.55% to 32.97%. The same decrease is not seen with synthetic VoxCeleb1 data, because VoxCeleb is 16 kHz while the mismatch is between 8 kHz and 16 kHz. This further strengthens our hypothesis.

Experiment 3:

- 3(a): Figure 4.2. For this experiment, we held-out a set of 20 speakers from the training set and plotted the x-vectors for their utterances. 10 of those 20 speakers are shown for brevity. We note that fine-tuned PyTDNN very well clusters the utterances of the held-out speakers.
- 3(b): Table 4.7. For the recordings with number of speakers as 2, i.e. (a) and (b), the two speakers are well separated with some mixing of points. However, the x-vectors fail to discriminate between speakers when the number of speakers increases to 3 in (c). Also note that the assignment of speaker to a segment is based on majority voting (the speaker who speaks for the longest duration is assigned to the segment) which may not be accurate. Therefore, the plots are only an approximation of the reality.

4.5 Conclusion

We saw the fully Pythonic pipeline and accompanying experiments. We noted that simply replacing the kaldi's TDNN by PyTDNN results in a DER increment because of the mismatch in sampling rate between the two x-vector systems. To fix the same, we fine-tune the TDNN on a subset of 8 kHz telephony speech data, which decreases the DER. To restore the DER to the optimal value of 9.29%, we need to train the TDNN on complete telephony data.

Chapter 5

Conclusion

5.1 Discussion

In this thesis we developed a fully pythonic pipeline for speaker diarization and evaluated it qualitatively as well as quantitatively.

In Chapter 2, we reviewed the different components of Speaker Diarization systems. We witnessed the history and development of each component and saw various state-of-the-art diarization systems. We also noted the datasets used in dizarization and the evaluation metrics used to assess the performance of diarization systems. Lastly, we saw the various open source toolkits available for diarization which ease certain tasks in the pipeline.

In Chapter 3, we chose two state-of-the-art baselines on CALLHOME dataset and reproduced their results. The two approaches were the kaldi [70] x-vector diarization pipeline for CALLHOME and LSTM based Similarity Measurement [52] pipeline. We discussed their theoretical background and experimental details. We ended the Chapter with the discussion of reproduced results.

In Chapter 4, we discussed the development and analysis of our pythonic pipeline step by step. We started by replacing the kaldi's x-vector front-end from the LSTM-SC pipeline by a Pythonic TDNN (PyTDNN) which had the same base architecture as kaldi's TDNN. We investigated the reason for a rise in DER and hypothesized the reason to be the frequency mismatch in the x-vector due to difference in sampling rates of the data that PyTDNN was trained on (16 kHz) and of CALLHOME (8 kHz). We fine-tuned and evaluated PyTDNN on CALLHOME as well as on synthetic data to verify our hypothesis empirically. We also presented a qualitative analysis of our system by visualizing the x-vectors.

Overall, we established the effectiveness of the pythonic pipeline and its potential in being used as a baseline system for the development of robust speaker diarization models.

We also point out the following two observations:
1. The sampling rate is an important characteristic which is encoded along with the features used in speaker diarization systems. A change of sampling rate results in features which give very different results for diarization. Features trained on dataset with sampling rate s_1 cannot directly be used on dataset with samping rate s_2.

2. Let u_1 be an utterance with sampling rate s_1 and let perturbation p be applied on u_1 to produce u_2. Now, if an x-vector extraction network N, trained on data with sampling rate s_2 ($\neq s_1$), is used to extract x-vectors for u_1 & u_2, then representations of u_1 & u_2 may not be similar, even though N was trained explicitly for the perturbation p. This is apparent from the observation that x-vectors from PyTDNN have trouble distinguishing amplitude variants of one utterance from another even though PyTDNN was trained on extensive augmentations containing the amplitude augmentations as well.

5.2 Future Work

The field of Speaker Diarization is quite nuanced and there is a potential of development and progress at every stage of the pipeline. Based on our observations, we present some of the future directions where research can be carried out.

- **Domain adaptation for sampling rate**: Based on the first observation from section 5.1, domain adaptation of speaker diarization systems from one sampling rate to another can result in massive technological advancements. We would then be able to use one trained diarization system and adapt it to other sampling rate without changing the features or without any retraining.

- **Using Spectrograms as an additional feature for diarization**: Spectrograms are a visual representations of speech information and can serve as additional features which can be used alongside the d- or x-vectors. We have made several advancements in Computer Vision and employing all of it to speech processing can lead to state-of-the-art systems. The speech processing community is already working on the same and spectrograms have been put to use in various other speech related tasks, just not diarization yet. Detection and Classification of Acoustic Scenes and Events (DCASE) is an annual challenge for sound classification. The winning team of one of the DCASE 2020 tasks used an ensemble of CNNs over log Mel Spectrograms [97].

- **Multimodal Diarization**: Additional modalities are useful in gaining additional information about any problem. The effectiveness of interface between the different modalities determines the scope of the approach. Until now, only audio has been used to perform

diarization. However, video of human interactions contain viable cues e.g. movement of the mouth, body language, etc. which can be used to predict speaker turns and improve diarization. To date, no multimodal diarization dataset is available. Creation of one can be a great starting point in this direction.

- **Alternative to Permutation Invariant Loss**: The Permutation-free training loss presented in [23] is not scalable when the number of speakers is large. An alternate approach to deal with the Permutation invariant nature of diarization output is by bringing the model's prediction as well as the label to the lexicographically smallest permutation. If they correspond to same output, they must match. This removes the task of having to consider all the permutations. It could be an interesting exploration to see if this idea scales to large number of speakers.

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] J. Ajmera and Chuck Wooters. "A robust speaker clustering algorithm". In: *IEEE Workshop on Automatic Speech Recognition and Understanding* (2003), pp. 411–416.

[3] Xavier Anguera. "Robust Speaker Diarization for Meetings". English. PhD Thesis. Barcelona: Universitat Politècnica de Catalunya, Oct. 2006. URL: https://theses.eurasip.org/theses/333/robust-speaker-diarization-for-meetings/.

[4] Ashish Arora et al. "The JHU Multi-Microphone Multi-Speaker ASR System for the CHiME-6 Challenge". In: *ArXiv* abs/2006.07898 (2020).

[5] David M. Blei and P. Frazier. "Distance dependent Chinese restaurant processes". In: *Journal of Machine Learning Research* 12 (2010), pp. 2461–2488.

[6] Christoph Böddeker et al. "Front-end processing for the CHiME-5 dinner party scenario". In: 2018.

[7] Simon Bozonnet et al. "System output combination for improved speaker diarization". In: *INTERSPEECH*. 2010.

[8] Hervé Bredin et al. "Pyannote.Audio: Neural Building Blocks for Speaker Diarization". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. May 2020, pp. 7124–7128. DOI: 10.1109/ICASSP40776.2020.9052974.

[9] Niko Brümmer et al. "Fusion of Heterogeneous Speaker Recognition Systems in the STBU Submission for the NIST Speaker Recognition Evaluation 2006". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15 (2007), pp. 2072–2084.

[10] S. Chen. "Speaker, Environment and Channel Change Detection and Clustering via the Bayesian Information Criterion". In: 1998.

[11] Joon Son Chung et al. "Spot the conversation: speaker diarisation in the wild". In: *arXiv preprint arXiv:2007.01216* (2020).

[12] Najim Dehak et al. "Front-End Factor Analysis for Speaker Verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19 (2011), pp. 788–798.

[13] Perrine Delacourt and C. Wellekens. "DISTBIC: A speaker-based segmentation for audio data indexing". In: *Speech Communications* 32 (2000), pp. 111–126.

[14] M. Díez, L. Burget, and P. Matejka. "Speaker Diarization based on Bayesian HMM with Eigenvoice Priors". In: *Odyssey*. 2018.

[15] M. Díez et al. "Analysis of Speaker Diarization Based on Bayesian HMM With Eigenvoice Priors". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), pp. 355–368.

[16] M. Díez et al. "Bayesian HMM Based x-Vector Clustering for Speaker Diarization". In: *INTERSPEECH*. 2019.

[17] D. Dimitriadis. "Enhancements for Audio-only Diarization Systems". In: *ArXiv* abs/1909.00082 (2019).

[18] Lukas Drude et al. "NARA-WPE: A Python package for weighted prediction error dereverberation in Numpy and Tensorflow for online and offline processing". In: *ITG Symposium on Speech Communication*. 2018.

[19] Thomas Drugman et al. "Voice Activity Detection: Merging Source and Filter-based Information". In: *IEEE Signal Processing Letters* 23 (2016), pp. 252–256.

[20] G. Dupuy et al. "i-vectors and ILP clustering adapted to cross-show speaker diarization". In: *INTERSPEECH*. 2012.

[21] Hakan Erdogan et al. "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing* (2015), pp. 708–712.

[22] Enrico Fini and Alessio Brutti. "Supervised Online Diarization with Sample Mean Loss for Multi-Domain Data". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. May 2020, pp. 7134–7138. DOI: 10.1109/ICASSP40776.2020.9053477.

[23] Yusuke Fujita et al. "End-to-End Neural Speaker Diarization with Permutation-Free Objectives". en. In: *Interspeech 2019*. ISCA, Sept. 2019, pp. 4300–4304. DOI: 10.21437/Interspeech.2019-2899. URL: http://www.isca-speech.org/archive/Interspeech_2019/abstracts/2899.html.

[24] Yusuke Fujita et al. "End-to-End Neural Speaker Diarization with Self-Attention". en. In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. SG, Singapore: IEEE, Dec. 2019, pp. 296–303. ISBN: 978-1-72810-306-8. DOI: 10.1109/ASRU46091.2019.9003959. URL: https://ieeexplore.ieee.org/document/9003959/.

[25] J. Gauvain, L. Lamel, and G. Adda. "Partitioning and transcription of broadcast news data". In: *International Conference on Spoken Language Processing*. 1998.

[26] J. Gauvain et al. "The LIMSI 1998 Hub-4E Transcription System". In: 1997.

[27] H. Gish, M. Siu, and J. R. Rohlicek. "Segregation of speakers for speech recognition and speaker identification". In: *International Conference on Acoustics, Speech, and Signal Processing* (1991), 873–876 vol.2.

[28] Xifeng Guo et al. "Improved Deep Embedded Clustering with Local Structure Preservation". In: *International Joint Conference on Artificial Intelligence*. 2017.

[29] Kyu Jeong Han and Shrikanth S. Narayanan. "A robust stopping criterion for agglomerative hierarchical clustering in a speaker diarization system". In: *INTERSPEECH*. 2007.

[30] D. Haws et al. "On the importance of event detection for ASR". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2016), pp. 5705–5709.

[31] G. Heigold et al. "End-to-end text-dependent speaker verification". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2016), pp. 5115–5119.

[32] J. Hershey et al. "Deep clustering: Discriminative embeddings for segmentation and separation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2016), pp. 31–35.

[33] Shota Horiguchi et al. "End-to-End Speaker Diarization as Post-Processing". In: *ArXiv* abs/2012.10055 (2020).

[34] Shota Horiguchi et al. "End-to-End Speaker Diarization for an Unknown Number of Speakers with Encoder-Decoder Based Attractors". en. In: *Interspeech 2020*. ISCA, Oct. 2020, pp. 269–273. DOI: 10.21437/Interspeech.2020-1022. URL: http://www.isca-speech.org/archive/Interspeech_2020/abstracts/1022.html.

[35] Zili Huang et al. "Speaker Diarization with Region Proposal Network". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. May 2020, pp. 6514–6518. DOI: 10.1109/ICASSP40776.2020.9053760.

[36] Marijn Huijbregts, D. V. Leeuwen, and F. D. Jong. "The majority wins: a method for combining speaker diarization systems". In: *INTERSPEECH*. 2009.

[37] Sergey Ioffe. "Probabilistic Linear Discriminant Analysis". en. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 531–542. ISBN: 978-3-540-33839-0. DOI: 10.1007/11744085_41.

[38] D. Istrate et al. "NIST RT'05S Evaluation: Pre-processing Techniques and Speaker Diarization on Multiple Microphone Meetings". In: *MLMI*. 2005.

[39] Uday Jain et al. "Recognition of continuous broadcast news with multiple unknown speakers and environments". In: 1995.

[40] Qin Jin and Tanja Schultz. "Speaker segmentation and clustering in meetings". In: *INTERSPEECH*. 2004.

[41] Ian T. Jolliffe and Jorge Cadima. "Principal Component Analysis: A Review and Recent Developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (Apr. 2016). Publisher: Royal Society, p. 20150202. DOI: 10.1098/rsta.2015.0202. URL: https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202.

[42] P. Kenny, Gilles Boulianne, and P. Dumouchel. "Eigenvoice modeling with sparse training data". In: *IEEE Transactions on Speech and Audio Processing* 13 (2005), pp. 345–354.

[43] P. Kenny, D. Reynolds, and F. Castaldo. "Diarization of Telephone Conversations Using Factor Analysis". In: *IEEE Journal of Selected Topics in Signal Processing* 4 (2010), pp. 1059–1070.

[44] P. Kenny et al. "A Study of Interspeaker Variability in Speaker Verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 16 (2008), pp. 980–988.

[45] P. Kenny et al. "Joint Factor Analysis Versus Eigenchannels in Speaker Recognition". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15 (2007), pp. 1435–1447.

[46] P. Kenny et al. "Speaker and Session Variability in GMM-Based Speaker Verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15 (2007), pp. 1448–1460.

[47] Morten Kolbaek et al. "Multitalker Speech Separation With Utterance-Level Permutation Invariant Training of Deep Recurrent Neural Networks". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25 (2017), pp. 1901–1913.

[48] Federico Landini et al. "Analysis of the BUT Diarization System for VoxConverse Challenge". In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. June 2021, pp. 5819–5823. DOI: 10.1109/ICASSP39728.2021.9414315.

[49] Federico Landini et al. "BUT System Description for DIHARD Speech Diarization Challenge 2019". en. In: *arXiv:1910.08847 [eess]* (Oct. 2019). arXiv: 1910.08847. URL: http://arxiv.org/abs/1910.08847.

[50] Qiujia Li et al. "Discriminative Neural Clustering for Speaker Diarisation". In: *2021 IEEE Spoken Language Technology Workshop (SLT)*. Jan. 2021, pp. 574–581. DOI: 10.1109/SLT48900.2021.9383617.

[51] Qingjian Lin, Y. Hou, and M. Li. "Self-Attentive Similarity Measurement Strategies in Speaker Diarization". In: *INTERSPEECH*. 2020.

[52] Qingjian Lin et al. "LSTM Based Similarity Measurement with Spectral Clustering for Speaker Diarization". en. In: *Interspeech 2019*. ISCA, Sept. 2019, pp. 366–370. DOI: 10.21437/Interspeech.2019-1388. URL: http://www.isca-speech.org/archive/Interspeech_2019/abstracts/1388.html.

[53] Daben Liu and F. Kubala. "Fast speaker change detection for broadcast news transcription and indexing". In: *International Conference on Spoken Language Processing*. 1999, pp. 1031–1034.

[54] X. Lu et al. "Speech enhancement based on deep denoising autoencoder". In: *INTERSPEECH*. 2013.

[55] U. V. Luxburg. "A tutorial on spectral clustering". In: *Statistics and Computing* 17 (2007), pp. 395–416.

[56] L. V. D. Maaten and Geoffrey E. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[57] Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th python in science conference*. Vol. 8. 2015.

[58] I. Medennikov et al. "The STC System for the CHiME-6 Challenge". In: 2020.

[59] Ivan Medennikov et al. "Target-Speaker Voice Activity Detection: A Novel Approach for Multi-Speaker Diarization in a Dinner Party Scenario". en. In: *Interspeech 2020*. ISCA, Oct. 2020, pp. 274–278. DOI: 10.21437/Interspeech.2020-1602. URL: http://www.isca-speech.org/archive/Interspeech_2020/abstracts/1602.html.

[60] S. Meignier and T. Merlin. "LIUM SpkDiarization: An Open Source Toolkit For Diarization". In: *CMU SPUD Workshop, Dallas (Texas, USA)*. 2010.

[61] S. Meignier et al. "Step-by-step and integrated approaches in broadcast news speaker diarization". In: *Computer Speech Language* 20 (2006), pp. 303–330.

[62] Nikki Mirghafori and Chuck Wooters. "Nuts and Flakes: A Study of Data Characteristics in Speaker Diarization". In: *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings* 1 (2006), pp. I–I.

[63] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. "VoxCeleb: A Large-Scale Speaker Identification Dataset". In: *INTERSPEECH*. 2017.

[64] T. Nakatani et al. "Speech Dereverberation Based on Variance-Normalized Delayed Linear Prediction". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18 (2010), pp. 1717–1731.

[65] Tim Ng et al. "Developing a Speech Activity Detection System for the DARPA RATS Program". In: *INTERSPEECH*. 2012.

[66] S. Novoselov et al. "Speaker Diarization with Deep Speaker Embeddings for DIHARD Challenge II". In: *INTERSPEECH*. 2019.

[67] Tae Jin Park et al. "A Review of Speaker Diarization: Recent Advances with Deep Learning". en. In: *arXiv:2101.09624 [cs, eess]* (Jan. 2021). arXiv: 2101.09624. URL: http://arxiv.org/abs/2101.09624.

[68] Tae Jin Park et al. "Auto-Tuning Spectral Clustering for Speaker Diarization Using Normalized Maximum Eigengap". In: *IEEE Signal Processing Letters* 27 (2020), pp. 381–385.

[69] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[70] Daniel Povey et al. "The Kaldi Speech Recognition Toolkit". In: (2011). IEEE Catalog No.: CFP11SRW-USB. URL: http://infoscience.epfl.ch/record/192584.

[71] Desh Raj et al. "Integration of Speech Separation, Diarization, and Recognition for Multi-Speaker Meetings: System Description, Comparison, and Analysis". In: *IEEE Spoken Language Technology Workshop (SLT)* (2021), pp. 897–904.

[72] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2015), pp. 1137–1149.

[73] D. Reynolds, P. Kenny, and F. Castaldo. "A study of new approaches to speaker diarization". In: *INTERSPEECH*. 2009.

[74] D. Reynolds, T. Quatieri, and R. B. Dunn. "Speaker Verification Using Adapted Gaussian Mixture Models". In: *Digital Signal Processing* 10 (2000), pp. 19–41.

[75] D. Reynolds and P. Torres-Carrasquillo. "The MIT Lincoln Laboratory RT-04F Diarization Systems: Applications to Broadcast Audio and Telephone Conversations". In: 2004.

[76] J. R. Rohlicek et al. "Gisting conversational speech". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing* 2 (1992), 113–116 vol.2.

[77] J. E. Rougui et al. "Fast Incremental Clustering of Gaussian Mixture Speaker Models for Scaling up Retrieval In On-Line Broadcast". In: *IEEE International Conference on Acoustics Speech and Signal Processing* 5 (2006), pp. V–V.

[78] Neville Ryant. *Diarization Scoring Tools*. Feb. 2018. URL: https://github.com/nryant/dscore.

[79] Neville Ryant et al. "The Second DIHARD Diarization Challenge: Dataset, task, and baselines". In: *arXiv:1906.07839 [cs, eess]* (June 2019). arXiv: 1906.07839. URL: http://arxiv.org/abs/1906.07839.

[80] Neville Ryant et al. "The Second DIHARD Diarization Challenge: Dataset, task, and baselines". In: *INTERSPEECH*. 2019.

[81] Neville Ryant et al. "Third DIHARD Challenge Evaluation Plan". In: *ArXiv* abs/2006.05815 (2020).

[82] Itay Salmun et al. "PLDA-based mean shift speakers' short segments clustering". In: *Computer Speech Language* 45 (2017), pp. 411–436.

[83] R. Sarikaya and J. Hansen. "Robust speech activity detection in the presence of noise". In: *International Conference on Spoken Language Processing*. 1998.

[84] Gregory Sell and D. Garcia-Romero. "Diarization resegmentation in the factor analysis subspace". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2015), pp. 4794–4798.

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

[86] Gregory Sell et al. "Diarization is Hard: Some Experiences and Lessons Learned for the JHU Team in the Inaugural DIHARD Challenge". In: *INTERSPEECH*. 2018.

[87] Mohammed Senoussaoui et al. "A Study of the Cosine Distance-Based Mean Shift for Telephone Speech Diarization". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22 (2014), pp. 217–227.

[88] Mohammed Senoussaoui et al. "Efficient iterative mean shift based cosine dissimilarity for multi-recording speaker clustering". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), pp. 7712–7715.

[89] Yan Shi et al. "Addressing Text-Dependent Speaker Verification Using Singing Speech". In: *Applied Sciences* 9 (June 2019), p. 2636. DOI: 10.3390/app9132636.

[90] Stephen Shum, Najim Dehak, and J. Glass. "On the Use of Spectral and Iterative Methods for Speaker Diarization". In: *INTERSPEECH*. 2012.

[91] Stephen Shum et al. "Unsupervised Methods for Speaker Diarization: An Integrated and Iterative Approach". In: *IEEE Transactions on Audio, Speech, and Language Processing* 21 (2013), pp. 2015–2028.

[92] M. Siegler. "Automatic Segmentation, Classification and Clustering of Broadcast News Audio". In: 1997.

[93] David Snyder et al. "Deep Neural Network Embeddings for Text-Independent Speaker Verification". en. In: *Interspeech 2017*. ISCA, Aug. 2017, pp. 999–1003. DOI: 10.21437/Interspeech.2017-620. URL: http://www.isca-speech.org/archive/Interspeech_2017/abstracts/0620.html.

[94] David Snyder et al. "X-Vectors: Robust DNN Embeddings for Speaker Recognition". en. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, AB: IEEE, Apr. 2018, pp. 5329–5333. ISBN: 978-1-5386-4658-8. DOI: 10.1109/ICASSP.2018.8461375. URL: https://ieeexplore.ieee.org/document/8461375/.

[95] Themos Stafylakis, Vassilis Katsouros, and G. Carayannis. "Speaker clustering via the mean shift algorithm". In: *Odyssey*. 2010.

[96] A. Stolcke and T. Yoshioka. "DOVER: A Method for Combining Diarization Outputs". In: *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)* (2019), pp. 757–763.

[97] Sangwon Suh et al. *Designing Acoustic Scene Classification Models with CNN Variants*. Tech. rep. DCASE2020 Challenge, 2020.

[98] Y. Sun, Xiaogang Wang, and X. Tang. "Deep Learning Face Representation from Predicting 10,000 Classes". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1891–1898.

[99] Yaniv Taigman et al. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1701–1708.

[100] S. Tranter et al. "Generating and evaluating segmentations for automatic speech recognition of conversational telephone speech". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing* 1 (2004), pp. I–753.

[101] W. Tsai, Shih-Sian Cheng, and H. Wang. "Speaker clustering of speech utterances using a voice characteristic reference space". In: *INTERSPEECH*. 2004.

[102] F. Valente, P. Motlícek, and Deepu Vijayasenan. "Variational Bayesian speaker diarization of meeting recordings". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2010), pp. 4954–4957.

[103] Ehsan Variani et al. "Deep neural networks for small footprint text-dependent speaker verification". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2014), pp. 4052–4056.

[104] J. Villalba et al. "State-of-the-Art Speaker Recognition for Telephone and Video Speech: The JHU-MIT Submission for NIST SRE18". In: *INTERSPEECH*. 2019.

[105] Jixuan Wang et al. "Speaker Diarization with Session-Level Speaker Embedding Refinement Using Graph Neural Networks". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. May 2020, pp. 7109–7113. DOI: 10.1109/ICASSP40776.2020.9054176.

[106] Quan Wang et al. "Speaker Diarization with LSTM". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. Apr. 2018, pp. 5239–5243. DOI: 10.1109/ICASSP.2018.8462628.

[107] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. "Unsupervised Deep Embedding for Clustering Analysis". In: *ArXiv* abs/1511.06335 (2016).

[108] Yanchen Xu et al. "A Regression Approach to Speech Enhancement Based on Deep Neural Networks". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23 (2015), pp. 7–19.

[109] T. Yoshioka and T. Nakatani. "Generalization of Multi-Channel Linear Prediction Methods for Blind MIMO Impulse Response Shortening". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20 (2012), pp. 2707–2720.

[110] T. Yoshioka et al. "Recognizing Overlapped Speech in Meetings: A Multichannel Separation Approach Using Neural Networks". In: *ArXiv* abs/1810.03655 (2018).

[111] Aonan Zhang et al. "Fully Supervised Speaker Diarization". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. May 2019, pp. 6301–6305. DOI: 10.1109/ICASSP.2019.8683892.

[112] W. Zhu and Jason W. Pelecanos. "Online speaker diarization using adapted i-vector transforms". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2016), pp. 5045–5049.