UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AUDIO-VISUAL SPEAKER DIARIZATION IN CHALLENGING ENVIRONMENTS USING DEEP-LEARNING METHODS

Diploma Thesis

Konstantinos Fanaras

Supervisor: Gerasimos Potamianos

Volos, September 2021

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AUDIO-VISUAL SPEAKER DIARIZATION IN CHALLENGING ENVIRONMENTS USING DEEP-LEARNING METHODS

Diploma Thesis

Konstantinos Fanaras

Supervisor: Gerasimos Potamianos

Volos, September 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΟΠΤΙΚΟ-ΑΚΟΥΣΤΙΚΗ ΚΑΤΑΛΟΓΟΠΟΙΗΣΗ ΟΜΙΛΗΤΩΝ ΣΕ ΑΠΑΙΤΗΤΙΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ ΜΕ ΜΕΘΟΔΟΥΣ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ

Διπλωματική Εργασία

Κωνσταντίνος Φανάρας

Επιβλέπων: Γεράσιμος Ποταμιάνος

Βόλος, Σεπτέμβριος 2021

Approved by the Examination Committee:

Supervisor	**Gerasimos Potamianos**
	Associate Professor,
	Department of Electrical and Computer Engineering,
	University of Thessaly
Member	**Georgios Stamoulis**
	Professor,
	Department of Electrical and Computer Engineering,
	University of Thessaly
Member	**Nikolaos Bellas**
	Professor,
	Department of Electrical and Computer Engineering,
	University of Thessaly

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisor, Assoc. Prof. Gerasimos Potamianos for giving me the chance to work on an intriguing project that was a perfect match to my interests. His guidance and expertise was crucial for completing this thesis, while his immediate response to any arising issue saved me unproductive time and simultaneously inspired me to become more professional.

Additionally, I am more than thankful to Prof. Emeritus Elias N. Houstis for his invaluable support during these years and for the knowledge I acquired through my engagement into his courses. He was the one who inspired me to study the fascinating field of machine learning.

Last but not least, I would like to thank my family for their unconditional love, understanding, and unwavering belief in me throughout all those years.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Abstract

Speaker diarization is a task to identify “who spoke when”. With the emergence of deep learning technology, modern speaker diarization systems reach state-of-the-art performance. In this thesis, we employ deep learning in order to implement two speaker diarization systems. First of all, we develop an audio-only system. More specifically, we implement a pipeline, which consists of five modules that work cooperatively in order to perform speaker diarization. Furthermore, we utilize visual information to develop a multimodal system. The visual information is synchronized with audio data and as a result the performance of this system is significantly better than the performance of the audio-only system.

Περίληψη

Η καταλογοποίηση ομιλητών είναι μία διεργασία που προσδιορίζει ”ποιός μίλησε πότε”. Με την εμφάνιση της τεχνολογίας βαθιάς μάθησης, τα σύγχρονα συστήματα καταλογοποίησης ομιλητών πετυχαίνουν την καλύτερη δυνατή απόδοση. Στην παρούσα διπλωματική εργασία, χρησιμοποιούμε βαθιά μάθηση έτσι ώστε να υλοποιήσουμε δύο συστήματα καταλογοποίησης ομιλητών. Αρχικά, αναπτύσσουμε ένα σύστημα που χρησιμοποιεί μόνο δεδομένα ήχου. Πιο συγκεκριμένα, υλοποιούμε ένα pipeline που απαρτίζεται από πέντε τμήματα, τα οποία λειτουργούν συνεργατικά ώστε να επιτευχθεί η καταλογοποίηση ομιλητών. Επιπλέον αξιοποιούμε την οπτική πληροφορία ώστε να αναπτύξουμε ένα σύστημα καταλογοποίησης ομιλητών που χρησιμοποιεί και την πληροφορία ήχου και την πληροφρία εικόνας. Η οπτική πληροφορία συγχρονίζεται με τα δεδομένα ήχου και σαν αποτέλεσμα η απόδοση αυτού του συστήματος είναι σημαντικά καλύτερη του συστήματος που χρησιμοποιεί μόνο δεδομένα ήχου.

Table of contents

Acknowledgements **ix**

Abstract **xiii**

Περίληψη **xv**

Table of contents **xvii**

List of figures **xxi**

List of tables **xxiii**

Abbreviations **xxv**

1 Introduction **1**

1.1 The Speaker Diarization Task 1

1.2 Related Work 2

1.3 Thesis Contribution 3

1.4 Thesis Overview 3

2 Databases **5**

2.1 The AMI Database 5

2.2 The VoxCeleb2 Database 6

3 Audio-Only Speaker Diarization System **9**

3.1 Voice Activity Detection 9

3.1.1 SincNet Description 9

3.1.2 VAD Architecture 10

3.1.3 Implementation Details 11

3.2 Speaker Change Detection 12

3.2.1 SCD Architecture 13

3.2.2 Implementation Details 14

3.3 Speaker Embedding and Clustering 16

3.3.1 X-Vector Model Architecture 16

3.3.2 Implementation Details 18

3.3.3 Clustering 19

3.4 Resegmentation 19

3.4.1 Resegmentation Module Architecture 20

3.4.2 Implementation Details 21

3.5 Evaluation 21

4 Audio-Visual Speaker Diarization System **23**

4.1 Voice Activity Detection 23

4.2 Shot Detection 24

4.3 Face Detection and Tracking 24

4.3.1 Face Detection Architecture 24

4.3.2 Face Tracking 26

4.4 Face Recognition 27

4.4.1 ResNet-50 Architecture 27

4.4.2 Implementation Details 27

4.5 Audio-Visual Synchronization 29

4.5.1 System Architecture 29

4.5.2 Training Process 30

4.6 Off-Screen Speaker Recognition 31

4.7 Evaluation 31

5 Evaluation and Results **33**

5.1 Evaluation Metric 33

5.2 Data 33

5.3 Results 34

6 Conclusion and Future Work **35**

Bibliography **37**

List of figures

1.1 Traditional speaker diarization system architecture (figure from [1]) 2

2.1 AMI IS1008a closeup camera images of individual speakers 6

2.2 Ground-truth annotation of the AMI ES2004a MixHeadset recording, with four speakers (FEE013, FEE016, MEE014 and MEO015) active 6

2.3 Examples and statistics of the VoxCeleb2 dataset (figure from [2]) 7

3.1 Audio-only speaker diarization System 9

3.2 The SincNet architecture (Figure from [3]) 11

3.3 The voice activity detection module (figure adapted from [4]) 11

3.4 Speaker change detection schematic 13

3.5 Speaker change detection architecture 14

3.6 Speaker change detection: Dealing with the class imbalance problem, as in [5] 15

3.7 Speaker embedding system Architecture, based on [6] 17

3.8 The agglomerative hierarchical clustering procedure (figure from [1]) . . . 20

3.9 The resegmentation module architecture 21

4.1 The architecture of our proposed audio-visual speaker diarization system . . 23

4.2 Architecture of the single-shot scale-invariant face detector (S^{3}FD). It consists of base convolutional layers, extra convolutional layers, detection convolutional layers, normalization layers, predicted convolutional layers and a multi-task loss layer (Figure from [7]). 25

4.3 Example network architectures for ImageNet. Left: the VGG-19 model [8]. Middle: a plain network with 34 parameter layers. Right: a residual network (ResNet-34) with 34 parameter layers (Figure from [9]) 28

4.4 A deeper residual function. Left: a building block (on 56×56 feature maps) for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152 (Figure from [9]) 29
4.5 Audio-Visual Synchronization System as depicted in [10] 30
4.6 The training procedure of the audio-visual synchronization network (Figure from [10]) 31

List of tables

3.1 VAD Evaluation on the MixHeadset AMI data in terms of three metrics . . 12

3.2 SCD evaluation on the AMI MixHeadset data in terms of two metrics . . . 16

3.3 The x-vector speaker embedding DNN architecture. T is the number of frames in the input segment and N in the softmax layer is the number of training speakers 18

4.1 Stride size, anchor scale, and receptive field (RF) of the six detection layers of the architecture of Figure 4.2 26

5.1 Diarization results of [11], [12], compared to our developed audio-only and audio-visual systems in terms of DER % 34

5.2 VAD module performance of the two best-performing diarization systems of Table 5.1, in terms of false alarm rate 34

Abbreviations

MFCC	Mel-Frequency Cepstral Coefficient
NN	Neural Network
DNN	Deep Neural Network
FF	Feed-Forward
LSTM	Long Short-Term Memory
biLSTM	bi-directional Long-Short Term Memory
CNN	Convolutional Neural Network
VAD	Voice Activity Detection
SCD	Speaker Change Detection
FA	False Alarm
UBM	Universal Background Model
PLDA	Probabilistic Linear Discriminant Analysis
AHC	Agglomerative Hierarchical Clustering
SSL	Sound Source Localization
EER	Equal Error Rate
DER	Diarization Error Rate

Chapter 1

Introduction

1.1 The Speaker Diarization Task

Speaker diarization is a well-known problem in the field of speech processing. In short, it is the task that answers the question: ”who spoke when”. There are many use-cases of speaker diarization. For example, it can be used in broadcast news for record-keeping purposes, in marketing for recording meetings and interviews, in health care and medical services for recording medical conversations, such as between a doctor and patient or caregiver and in software development for integrating chatbots and home assistants, such as Alexa, Google Home, and Siri into existing technology. For many years, computer scientists and engineers have been trying to come up with novel ideas in order to solve the problem of speaker diarization in the most effective way, by processing audio and video data. Typical speaker diarization systems consist of many modules as depicted in Figure 1.1. Front-end processing techniques like speech enhancement or speech separation are first applied in order to mitigate any artifacts in the acoustic environment. Speech activity detection then finds speech and non-speech regions in a given audio stream. Speaker embeddings are some kind of features that are extracted from raw audio signals in order to be used for speaker recognition. Speaker embeddings are clustered into speaker clusters and in the post-processing step the clustering results are further refined. Except for audio-only pipelines like the one illustrated in Figure 1.1, modern approaches to the speaker diariation task exploit the visual information in order to boost performance, employing audio-visual fusion techniques in order to synchronize audio and visual information. In this thesis, utilizing deep learning, we implement two speaker diarization systems, an audio-only and an audio-visual one.

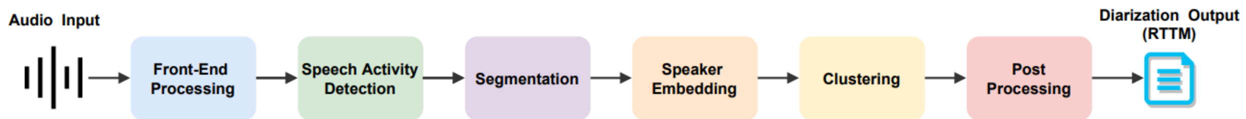

Figure 1.1: Traditional speaker diarization system architecture (figure from [1])

1.2 Related Work

Many systems that utilize audio or video information have been developed during the last two decades in order to deal with the speaker diarization problem. During the early years of speaker diarization, many organizations like the National Institute of Standards and Technology (NIST), developed speaker diarization systems that were applied in different domains like broadcast news [13], [14] and meetings [15], [16]. These systems led to new approaches in the speaker diarization domain, like information bottleneck clustering [17] and variational Bayesian approaches [18]. The advent of i-vectors [19] improved the performance of speaker recognition tasks and led the speaker diarization systems to adopt i-vectors in order to enhance the performance of the clustering step. The rise of deep learning in the 2010s, lead to further improvements in the speaker diarization task. For example, the advent of x-vectors [20] for speaker embeddings extraction led the x-vectors to replace i-vectors, because the former contributed to enhanced performance of the speaker recognition and the speaker diarization task. Also, many end-to-end systems [21], [22] yielding very good results in the speaker diarization task.

In addition to the development of audio-only speaker diarization systems, a modern approach to the diarization task is the implementation of multimodal speaker diarization systems. For example, [23] proposed the use of compressed domain video features for multimodal speaker diarization that comprises frame-based visual activity features. These features were computed as a motion vector magnitude. Multimodal fusion was applied to mel-frequency cepstral coefficient (MFCC) and video features by a weighted likelihood of a Gaussian mixture model (GMM). An agglomerative hierarchical clustering (AHC) technique was used, where each cluster was modelled by a joint audio and video GMM. In other work, [24] compares two audio-visual synchronization methods. The first method is based on canonical correlation analysis and the second one on mutual information, where MFCC features are used along with motion features from face tracks. Finally, many implementations employ sound source localization (SSL) techniques [25], which is shown to improve diarization performance [26].

1.3 Thesis Contribution

The main focus of this thesis is to utilize visual information in order to develop a multi-modal speaker diarization system with significantly better performance than an audio-only system. Furthermore, we show that working directly from the waveform, instead of using hand-crafted features (like MFCCs), for tasks like voice activity detection (VAD) and speaker change detection (SCD) enhances the overall performance of a speaker diarization system. First, we implement an audio-only system utilizing deep learning methods, and subsequently we develop an audio-visual model in order to exploit the visual information that will lead us to improve our diarization results. We evaluate our systems on a subset of the AMI database [27].

1.4 Thesis Overview

The remainder of the thesis is structured as follows:

- In Chapter 2, we describe the two databases that we used in order to train and evaluate every component of our speaker diarization pipelines.
- In Chapter 3, we describe in detail every component of our audio-only speaker diarization system.
- In Chapter 4, we describe in detail every component of our audio-visual speaker diarization system.
- In Chapter 5, we evaluate our systems and we compare them with two audio-visual systems in the literature.
- Finally, in Chapter 6 we provide our conclusions and discuss future extensions.

Chapter 2

Databases

In this thesis, we use two databases: The AMI Meeting Corpus [27] and the VoxCeleb2 [2]. We use the AMI database to train every component of our audio-only system except from the speaker embedding module. We also use it to evaluate both our audio-only and audio-visual systems. We use VoxCeleb2 to train an x-vector architecture for speaker embeddings extraction in the audio-only pipeline. Below, we present the two databases used in this thesis.

2.1 The AMI Database

As we already mentioned, in order to train the modules (except the speaker embedding module) of our audio-only speaker diarization system we utilized the AMI database. The AMI corpus consists of 100 hours of video recorded across a number of locations and has been used by many previous works on audio-only and audio-visual diarization. We use a popular subset of the AMI database, the MixHeadset subset in order to train our audio-only system and to evaluate both of our systems. This subset consists of 5.4 h audio-only recordings and 5.8 h of multimodal recordings. These meetings were recorded in English with mostly non-native speakers. Each clip is in a Matroska video container. The video is encoded as MPEG-4, and the audio is encoded as MPEG-4 AAC at 48000 samples per second (48 kHz). All video clips have 30 frames per second (FPS) and a low resolution of 288×352 pixels. The acoustic properties of the recordings vary due to the different room scenarios. The audio is recorded from an 8-element circular microphone array with a diameter of 20 cm. However, we only use one microphone from the array. The video recordings of the subset we use, consist of individual speakers and room view cameras. We use the close-up cameras of every speaker in our work. All recordings consist of four speakers, and each recording is labeled with and id and the lower-case letters a-c, which indicate the session of that particular recording. Each session has a duration of 15-35 minutes. In Figure 2.1, we depict some closeup camera samples from meeting IS1008a.

The AMI database contains manual transcriptions appropriately transformed to be used to train the speaker diarization system. More specifically, using the pyannote library [28], we convert the manual transcriptions provided by the AMI corpus to files in the rttm format in order to create the ground truth labels that are necessary in training our system assigning a speaker id to the corresponding speech segments, as shown in Figure 2.2.

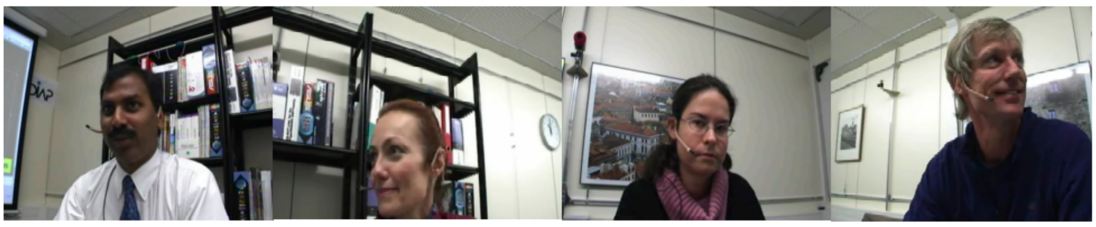

Figure 2.1: AMI IS1008a closeup camera images of individual speakers

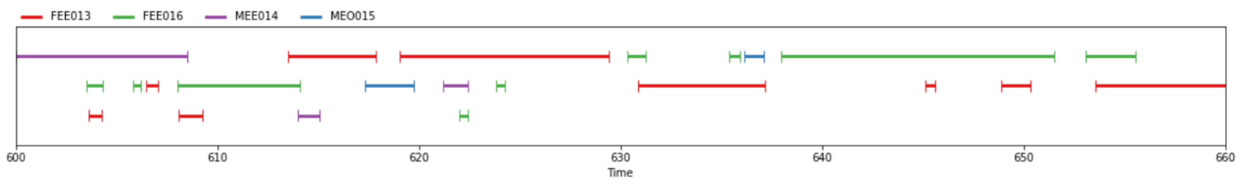

Figure 2.2: Ground-truth annotation of the AMI ES2004a MixHeadset recording, with four speakers (FEE013, FEE016, MEE014 and MEO015) active

2.2 The VoxCeleb2 Database

The second database we use is VoxCeleb2. We use this database in order to train an x-vector architecture for the speaker embedding module of our audio-only speaker diarization pipeline. VoxCeleb2 contains over a million utterances from over 6000 speakers. This is several times larger than any publicly available speaker recognition dataset. It's an audio-visual database with it's videos extracted from YouTube, mostly consisting of non-native speakers with different accents (61% male), shot in a variety of visual and auditory environments, making it a very challenging dataset. The data include red-carpet interviews, outdoor stadiums and quiet indoor studio environments, speeches given to large audiences, excerpts from professionally shot multimedia, and even crude videos shot by hand-held devices. Audio segments of this database contain different types of ”noise” like background chatter, laughter, overlapping speech, and varying room acoustics. An overview of the database is provided in Figure 2.3, showing example cropped faces with corresponding audio, as well as data duration, gender, and nationality distribution. Note that durations shorter than 20 sec are binned in 1 sec intervals and all utterances greater than 20 sec are binned together.

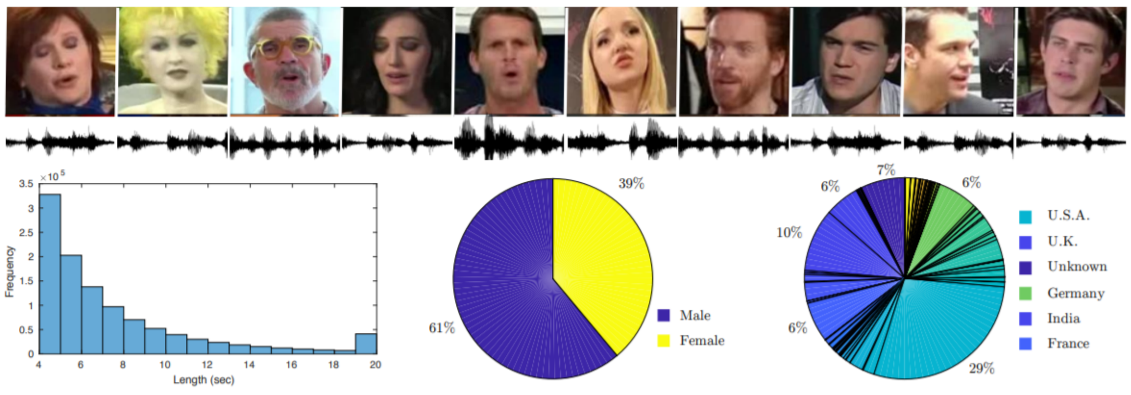

Figure 2.3: Examples and statistics of the VoxCeleb2 dataset (figure from [2])

Chapter 3

Audio-Only Speaker Diarization System

In this chapter we present our developed audio-only speaker diarization system. More specifically, we describe the implementation details of every component of the pipeline depicted in Figure 3.1.

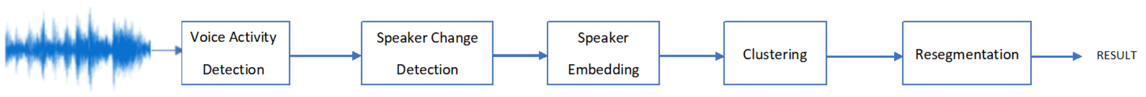

Figure 3.1: Audio-only speaker diarization System

3.1 Voice Activity Detection

The first component of the audio-only pipeline is the voice activity detection (VAD) module. Given an audio stream, the goal is to detect it's speech and non-speech parts. Traditional speaker diarization systems rely on MFCC features, which are the most widely used features in VAD, as well as speaker and speech recognition [29], [30].

3.1.1 SincNet Description

Based on [4], instead of using MFCCs, we choose to work directly from the waveform. For this purpose, we utilize SincNet [3], a convolutional neural network (CNN) architecture (Figure 3.2) that was proposed for speaker recognition. There are many advantages arising from the use of SincNet in order to find trainable instead of hand-crafted (e.g., MFCC) features. Exploiting the raw waveform, CNNs can learn low-level speech representations. The first layer of the SincNet (Figure 3.2) can find meaningful filters, in contrast to hand-crafted features that smooth the speech spectrum and prevent the extraction of narrow-band speaker characteristics. Based on parametrized sinc functions, which implement band-pass filters, SincNet learns low and high cutoff frequencies from the raw audio input.

As depicted in Figure 3.2, the SincNet architecture can be divided into:

- a sinc-based convolution that can be defined as follows:

$$y[n] = x[n] * g[n, f_1, f_2], \tag{3.1}$$

  where $x[n]$ is a chunk of the input (audio) signal and $g[n, f_1, f_2]$ is a function, where f_1 and f_2 are the learned low and high cutoff frequencies, respectively. This function is defined in a way that a filter-bank composed of rectangular bandpass filters is employed. So, in the frequency domain it can be written as follows:

$$G[f, f_1, f_2] = rect(\frac{f}{2f_2}) - rect(\frac{f}{2f_1}), \tag{3.2}$$

  where $rect(.)$ is the rectangular function. Using the inverse Fourier transform [31], in the time domain it can be written as:

$$g[n, f_1, f_2] = 2f_2 sinc(2\pi f_2 n) - 2f_1 sinc(2\pi f_1 n), \tag{3.3}$$

  where $sinc(x) = \frac{sin(x)}{x}$

- a CNN pipeline (pooling, normalization, activation functions, dropout layer)
- Convolutional, fully-connected or recurrent layers that perform speaker classification using a softmax classifier

3.1.2 VAD Architecture

We use the first layer of the SincNet architecture in order to extract trainable features from the raw audio. More specifically, we divide the speech waveform into 2-sec audio chunks and for every chunk we perform sinc-based convolutions (Equations 3.1, 3.3). After the learnt feature vectors are extracted, two bi-directional long short-term memory (biLSTM) models followed by three feed-forward (FF) layers are employed to define speech and non-speech in the given audio chunks (Figure 3.3).

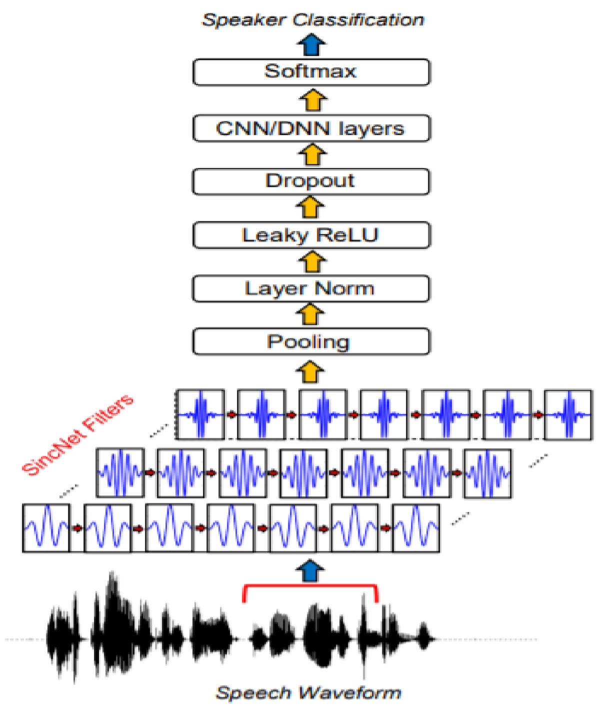

Figure 3.2: The SincNet architecture (Figure from [3])

3.1.3 Implementation Details

Training

For the training process, we use the training set of the MixHeadset of the AMI meeting corpus database, and we train our model for 200 epochs. For every epoch, we set a batch size of 64 2-sec audio chunks. During training, we also apply data augmentation by adding noise from the MUSAN database, with random signal-to-noise ratio between 5 and 20 dB. We work with a 16 kHz audio sampling rate. The two biLSTMs have 128 units (forward and backwards) with no temporal pooling. The first two FF layers have 128 units and use the tanh activation function, while the third FF layer is used for classification and has 2 units with the softmax activation function (see also Figure 3.3). Finally, we use 0.01 as the learning rate.

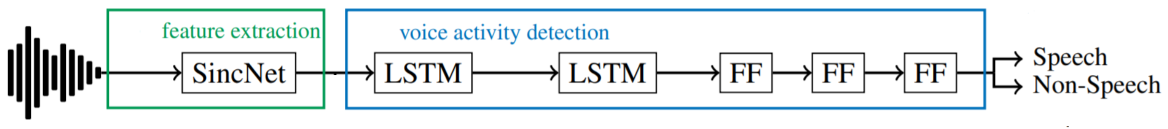

Figure 3.3: The voice activity detection module (figure adapted from [4])

Evaluation Metric

In order to evaluate our method we use the detection error rate from pyannote.metrics [28], defined as follows:

$$detection\ \ error\ \ rate = \frac{false\ \ alarm + missed\ \ detection}{total},$$

where ”false alarm” is the duration of audio segments that are non-speech and classified as speech, ”missed detection” is the duration of audio segments that are speech and classified as non-speech, and ”total” is the total duration of the speech.

Validation

In order to evaluate our model during the training phase, we utilize the development subset of the MixHeadset of the AMI meeting corpus. Every 10 epochs, we evaluate our model on the development set and keep the one that minimizes the detection error rate.

Testing

We evaluate our model on the test set of the MixHeadset of the AMI meeting corpus. As a baseline we use the audio-only speaker diarization system [32] that showed state-of-the-art performance in the diarization task in the DIHARD Challenge in 2018. The baseline used 13 MFCCs computed every 10 ms as features for VAD. As depicted in Table 3.1, utilizing SincNet in order to work direct from the waveform, our method performs better.

System	Detection Error Rate (%)	False Alarm (%)	Missed Detection (%)
Baseline [32]	8.2	6.1	5.8
Ours	**5.9**	**3.5**	**2.5**

Table 3.1: VAD Evaluation on the MixHeadset AMI data in terms of three metrics

3.2 Speaker Change Detection

The next component of our audio-only speaker diarization pipeline is the SCD module. This constitutes a significant component of a speaker diarization pipeline that helps improve system performance. SCD finds speaker change points in a given audio stream. We address this task by utilizing biLSTMs, similar to the VAD module (Section 3.1.2).

3.2.1 SCD Architecture

As in the VAD module (Section 3.1.2), we address this problem as a sequence labeling task. Using SincNet (Section 3.1.1) in order to work direct from the waveform (in exact the same way we did in Section 3.1), we give as input to the system a list of learnt feature vectors $x = [x_1, x_2, \dots, x_n]$ and we take as output another list of vectors $y = [y_1, y_2, \dots, y_n]$, where $y_i = 0$, when there is not a speaker change point in the audio recording, and $y_i = 1$, when there is a speaker change point (Figure 3.4). Having said that, our purpose is to find and model a function $f : x \to y$ that matches a feature sequence x to a label sequence y, as a recurrent neural network (RNN) trained using the binary cross-entropy loss function [5].

The SCD architecture is depicted in Figure 3.5. We use the first layer of the SincNet architecture in order to extract trainable features from the raw audio. More specifically, we divide the speech waveform into 2-sec audio chunks, and for every chunk we perform sinc-based convolutions (Equations 3.1, 3.3). After the learnt feature vectors are extracted, two biLSTMs followed by three FF layers are employed to define speaker change and no speaker change points in a given audio stream (Figure 3.5).

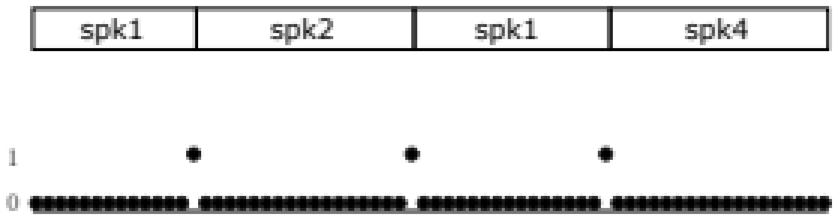

Figure 3.4: Speaker change detection schematic

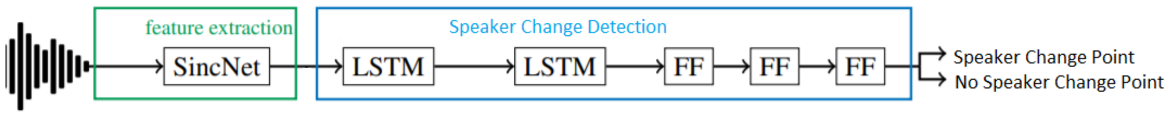

Figure 3.5: Speaker change detection architecture

3.2.2 Implementation Details

Training

At this point, it is noteworthy to mention that before the training phase of the SCD network, we annotate as speaker change points ($y_i = 1$) audio segments of 50-ms duration left and right of the manually annotated speaker change points (Figure 3.6). We do this in order to deal with the class imbalance problem [5]. More specifically, the number of speaker change points are limited in the audio files ($\approx 1\%$ of all frames in the MixHeadset of the AMI meeting corpus). This could be a serious problem for the training phase of our network. Many speaker diarization evaluation benchmarks [33], [34], [35] took into account the class imbalance problem. For the training process, we use the training set of the MixHeadset of the AMI meeting corpus, and we train our model for 1000 epochs. For every epoch, we set a batch size of 64 2-sec audio chunks. During training, we also apply data augmentation by adding noise from the MUSAN database, with random signal-to-noise ratio between 5 and 20 dB. We work with a 16kHz audio sampling rate. The two biLSTMs and the first two FF layers have exactly the same architecture as the corresponding ones in Section 3.1.3, while this time the third FF layer uses sigmoid activation function (see also Figure 3.5). Finally, we use 0.01 as the learning rate.

Evaluation Metric

Based on [5], in order to evaluate our method, we use purity [36] and coverage [37] from pyannote.metrics [28], defined as:

$$purity = \frac{\sum_{cluster} \max_{speaker} |cluster \cap speaker|}{\sum_{cluster} |cluster|}$$

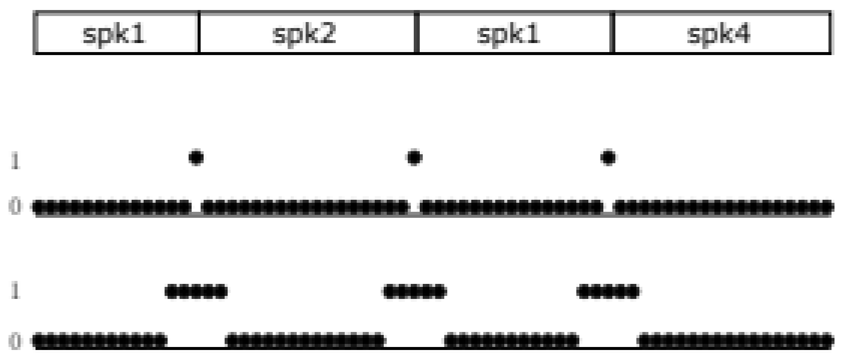

Figure 3.6: Speaker change detection: Dealing with the class imbalance problem, as in [5]

$$coverage = \frac{\sum_{speaker} \max_{cluster} |speaker \cap cluster|}{\sum_{speaker} |speaker|}$$

where $|speaker|$ (respectively $|cluster|$) is the speech duration of this particular reference speaker (respectively the hypothesized cluster), and $|speaker \cap cluster|$ is the duration of their intersection. Oversegmented results (when our system detects many speaker change points) tend to lead to high purity and low coverage, while under-segmented results (e.g. when two speakers are merged into one large cluster) lead to low purity and higher coverage.

Validation

In order to evaluate our model during the training phase, we utilize the development subset of the MixHeadset of the AMI meeting corpus. Every 100 epochs, we evaluate our model on the development set and keep the one with the best results in terms of purity and coverage.

Testing

We evaluate our model on the test set of the MixHeadset of the AMI meeting corpus. As a baseline we use an audio-only speaker diarization system [5] that shows state-of-the-art performance in the SCD task, employing 11 MFCCs extracted every 16 ms. As depicted in Table 3.2, utilizing SincNet in order to work direct from the waveform, our method yields a somewhat better purity, which is crucial for the next steps of our speaker diarization pipeline. When it comes to coverage, we managed to have a very good result as well, but because we have to do with speaker diarization, we care a little more about purity, which is very critical for the clustering (Section 3.3) that follows in the speaker diarization pipeline.

System	Purity (%)	Coverage (%)
Baseline [5]	89.7	80.2
Ours	**90.1**	**85.6**

Table 3.2: SCD evaluation on the AMI MixHeadset data in terms of two metrics

3.3 Speaker Embedding and Clustering

One of the most important steps in a speaker diarization pipeline is the speaker embedding extraction in order to feed them to the subsequent clustering step. Traditional speaker embedding extractor systems were based on i-vectors [38]. The standard approach consists of a universal background model (UBM) and a large projection matrix that are learned in an unsupervised way to maximize the data likelihood. The projection maps high-dimensional statistics from the UBM into a low-dimensional representation, known as an i-vector. A probabilistic linear discriminant analysis (PLDA) [39] classifier is used to compare i-vectors and enable same-or-different speaker decisions. One of the best performing speaker embedding architectures is the x-vectors one that we use in this Section.

3.3.1 X-Vector Model Architecture

In order to extract speaker embeddings, we utilize the DNN architecture that was proposed in [6], use SincNet (Section 3.1.1) for feature extraction, and apply data augmentation to further improve performance. The DNN architecture is illustrated in Figure 3.7.

As depicted in Figure 3.7, the DNN consists of nine layers. The architecture of the first five layers is based on [40]. They are time-delay neural networks (TDNNs), and they work at the frame-level as follows: Let t be the current time step. We take the frames at $[t-2, t-1, t, t+1, t+2]$ and splice them together. The two layers that follow take the output of the previous layers and splice it together at times $[t-2, t, t+2]$ and $[t-3, t, t+3]$ respectively. The next two layers also operate at frames, but without any added temporal context. The statistics

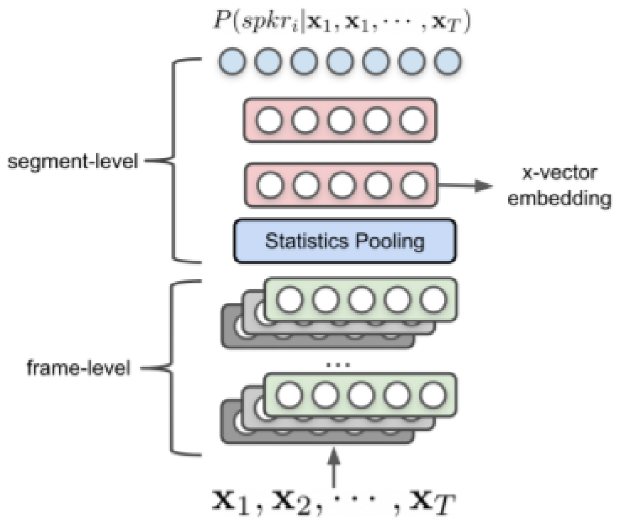
$P(spkr_i|\mathbf{x}_1, \mathbf{x}_1, \cdots, \mathbf{x}_T)$
segment-level
x-vector embedding
Statistics Pooling
frame-level
$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$

Figure 3.7: Speaker embedding system Architecture, based on [6]

pooling layer takes the output of the 5th frame-level layer, aggregates over the input segment, and computes its mean and standard deviation. These statistics are 1500-dimensional vectors, computed once for each input segment. All these are passed to two hidden layers that follow and to the softmax output layer. In Table 3.3, we summarize the layer temporal context, the total temporal context, and the dimension for each of the nine layers. X-vectors are extracted in segment 6 (Table 3.3). After the vectors are extracted, we use LDA [41] in order to reduce the dimensionality of embeddings and cosine distance [42] in order to compare the extracted embeddings.

The system is trained in order to predict a number of speakers from segments of variable length. To achieve this, the network uses a multi-class cross-entropy objective function that can be defined as follows:

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} \ln(P(spkr_k|x_{1:T}^{(n)})),$$

where K is the number of speakers in N training segments, $P(spkr_k|x_{1:T}^{(n)})$, is the probability of speaker k given T input frames $[x_1^{(n)}, x_2^{(n)}, \ldots, x_T^{(n)}]$ and d_{nk} is one if the speaker label for segment n is k and zero in any other case.

Layer	**Layer Context**	**Total Context**	**Input $\times$ Output**
frame 1	[t-2,t+2]	5	120 $\times$ 512
frame 2	[t-2,t, t+2]	9	1536 $\times$ 512
frame 3	[t-3,t,t+3]	15	1536 $\times$ 512
frame 4	[t]	15	512 $\times$ 512
frame 5	[t]	15	512 $\times$ 1500
Statistics Pooling	[0,T)	T	1500T $\times$ 3000
segment 6	[0]	T	3000 $\times$ 512
segment 7	[0]	T	512 $\times$ 512
softmax	[0]	T	512 $\times$ N

Table 3.3: The x-vector speaker embedding DNN architecture. T is the number of frames in the input segment and N in the softmax layer is the number of training speakers

3.3.2 Implementation Details

Training

For the training process, we use the VoxCeleb2 database after we remove the overlapping speakers for better performance, and we use short, 500-ms audio chunks. We work with a 16kHz audio sampling rate and apply data augmentation during the training phase by using background noise and music from the MUSAN database, with random signal-to-noise ratio between 5 and 15 dB. We train our model for 200 epochs. Finally, we use 0.01 as the learning rate.

Evaluation Metric

As a metric for the performance of our system we use the equal error rate (EER %).

Validation

In order to evaluate our model during the training phase, we utilize the test subset of the VoxCeleb2 database and evaluate our model every 5 epochs. We find the best EER ($\approx 4\%$) after 219 epochs.

Testing

We test our method on the test subset of the VoxCeleb2 database. The resulting EER is 7.1 %. Next, we apply our model is doing we can apply it on the MixHeadset of the AMI database.

3.3.3 Clustering

A clustering algorithm is applied to create clusters of the speech segments after generating the speaker embedding for each segment. So, after we apply our x-vector model on the MixHeadset of the AMI database, we employ AHC [43] as the clustering method on the previously learned embeddings.

AHC Description

AHC is a well-known method that has been applied in many speaker diarization pipelines with different distance metrics like the Bayesian information criterion [44], Kullback-Leibler distance [45], PLDA [39], and cosine distance [42]. AHC starts with a predefined number of clusters and merges them iteratively until a criterion is met. More specifically, the method starts by calculating the similarity between N singleton clusters. At each step, a pair of clusters that has the highest similarity is merged. The process is depicted in Figure 3.8. One of the most important aspects of AHC is the stopping criterion. The merging process can be stopped by using either a threshold or a target number of clusters. In our system, we use a threshold as a stopping criterion, which is defined by using the cosine distance as the similarity metric between pairs of clusters.

3.4 Resegmentation

The final part of our audio-only pipeline is resegmentation. This is a post-processing step, and is actually a process of refining the speaker boundaries that are roughly estimated using the clustering procedure. Thus, it is very crucial for the overall performance of our diarization system.

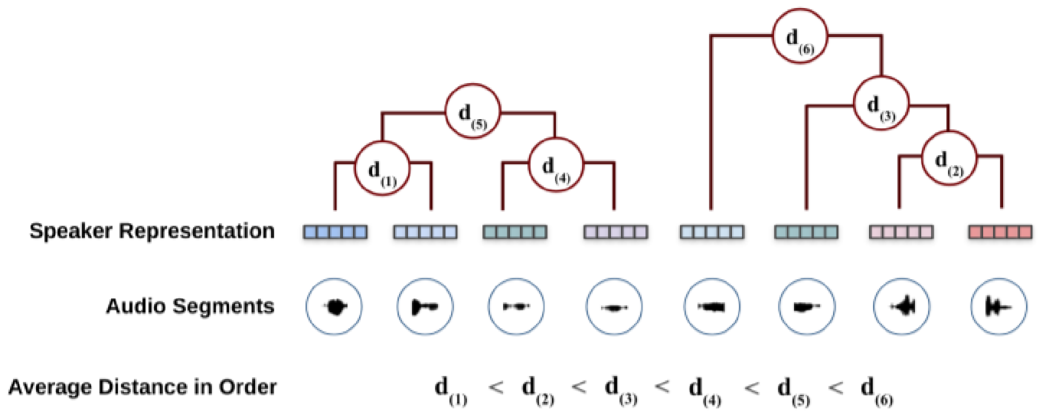
Speaker Representation
Audio Segments
Average Distance in Order
d(1) < d(2) < d(3) < d(4) < d(5) < d(6)
d(1)
d(2)
d(3)
d(4)
d(5)
d(6)

Figure 3.8: The agglomerative hierarchical clustering procedure (figure from [1])

3.4.1 Resegmentation Module Architecture

Viewing resegmentation as an unsupervised task, we address it similarly to the VAD and SCD modules, as a sequence labelling task. Our architecture is based in [46]. More specifically, let $K = k + 1$, where k is the number of speakers found in the clustering step. Also, let $y_t = 0$ if no speaker is active at time step t, and $y_t = s$ if speaker s ($s \in [1, k]$) is active at time step t. Because resegmentation is an unsupervised task, we cannot pre-train such models. So, for each audio file a new resegmentation model is trained, using the output of the clustering step. After we train the model, we apply it on the same audio file we used to train it, making this whole procedure unsupervised. Each time step is assigned to the class (non-speech or one of the k speakers) with the highest prediction scores. While this re-segmentation step does improve the labeling of speech regions, it also has the side effect of increasing false alarms (i.e. non-speech regions classified as speech). Therefore, its output is further post-processed to revert speech/non-speech regions back to the original VAD output.

The resegmentation module architecture is similar to the ones we used in VAD and SCD modules (shown in Figure 3.9), containing two biLSTMs and a multi-layer perceptron (MLP). The main difference is that here the MLP has two layers instead of three and the dimension of the second FF layer (final layer) depends on the task, because, as mentioned, for each audio file a new resegmentation model is trained.

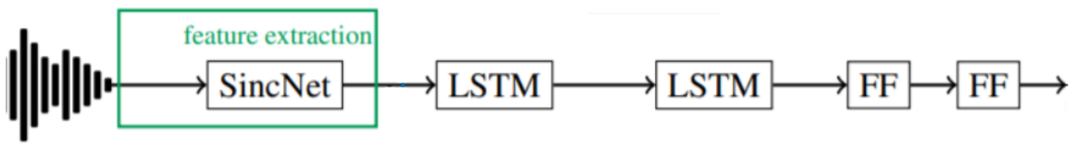

Figure 3.9: The resegmentation module architecture

3.4.2 Implementation Details

We train every model for 20 epochs. The two biLSTMs have 32 outputs (16 forward and 16 backward), the first FF layer is 16-dimensional, and the dimension of the second FF layer (final layer) depends on the task.

3.5 Evaluation

The results of our audio-only pipeline are examined in Chapter 5.

Chapter 4

Audio-Visual Speaker Diarization System

In this chapter we analyze the audio-visual speaker diarization system. More specifically, we describe the implementation details of every component of the pipeline illustrated in Figure 4.1.

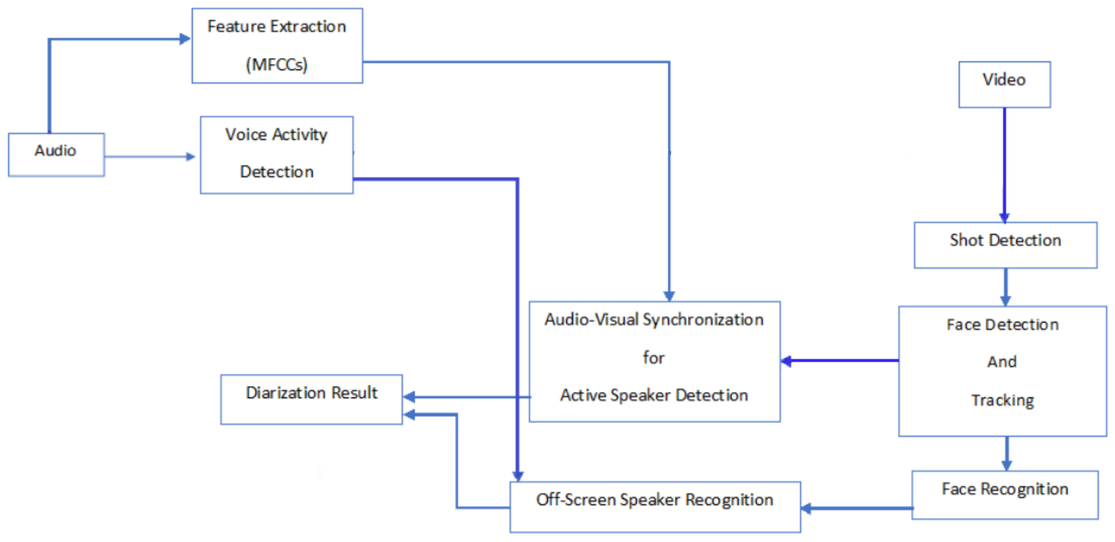

Figure 4.1: The architecture of our proposed audio-visual speaker diarization system

4.1 Voice Activity Detection

This component is exactly the same as the one we used in the audio-only system (see Section 3.1). We use this component in order to perform off-screen speaker recognition as described in Section 4.6.

4.2 Shot Detection

Shot detection is the process where we determine shot boundaries in order to find within-shot frames for which the subsequent face tracking is to be run. In order to detect these boundaries we utilize the open source implementation of PySceneDetect [47].

PySceneDetect is a command-line application and a Python library for detecting scene changes in videos and automatically splitting the video into separate clips. Not only this software is free and open-source, but it also contains several detection methods, from simple threshold-based fade in/out detection to advanced content-aware fast-cut detection of each shot. PySceneDetect can be used on its own as a stand-alone executable, with other applications as part of a video processing pipeline, or integrated directly into other programs/scripts via its Python API. PySceneDetect is written in Python, and it requires the OpenCV and Numpy software libraries.

4.3 Face Detection and Tracking

In this section we describe the two pre-trained models we use in order to perform face detection and face tracking.

4.3.1 Face Detection Architecture

After we define the frames in which the face tracing is to be run, we perform face detection using the CNN face detector described in [7]. The architecture of [7] is illustrated in Figure 4.2. As depicted there, the architecture of [7] is based on VGG16 [48] with some auxiliary structures. More specifically, it contains:

- Base convolutional layers (VGG16 through pool5 layer): All the layers of VGG16 from conv1_1 to pool5 are kept, and all the other layers are removed.
- Extra convolutional layers: The fully-connected layers fc6 and fc7 are converted to convolutional layers by subsampling their parameters [49]. After that, extra convolutional layers are added behind them. These layers decrease in size progressively and form the multi-scale feature maps.

- Detection layers: Conv3_3, Conv4_3, Conv5_3, Conv_fc7, Conv6_2, and Conv7_2 are selected as the detection layers.

- Normalization layers: Conv3_3, Conv4_3, and Conv5_3 have different feature scales compared to the other detection layers. So, L2 normalization [50] is applied to these 3 layers in order to rescale their norm to 10, 8, and 5 respectively, and subsequently learn these scales during the back-propagation.

- Predicted convolutional layers: Let p and q be the input and output number of channels, respectively, and 3×3 be the kernel size. Each detection layer is followed by a $p \times 3 \times 3 \times q$ convolutional layer. For each anchor, four offsets relative to its coordinates and N_s scores for classification are predicted, where $N_s = N_m + 1$ (N_m is the max-out background label) for the conv3_3 detection layer and N_s = 2 for the other detection layers.

- Multi-task loss layer: Softmax loss for classification and L1 loss for regression are used.

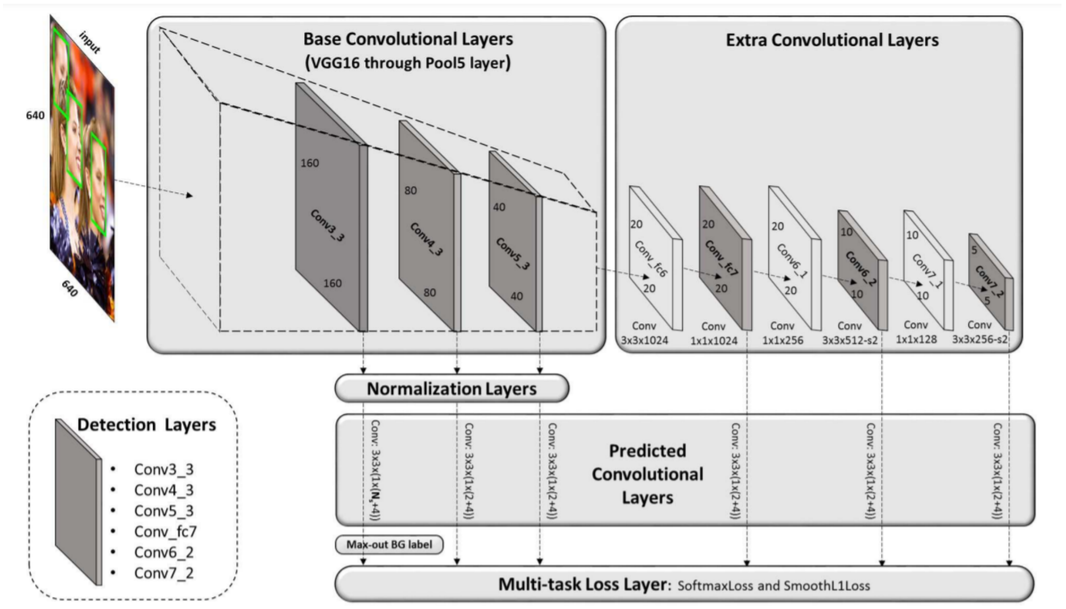
input
640
640
Base Convolutional Layers
(VGG16 through Pool5 layer)
Extra Convolutional Layers
160
Conv3_3
160
80
Conv4_3
80
40
Conv5_3
40
20 Conv_fc6 20 Conv 3x3x1024
20 Conv_fc7 20 Conv 1x1x1024
20 Conv6_1 20 Conv 1x1x256
10 Conv6_2 10 Conv 3x3x512-s2
10 Conv7_1 10 Conv 1x1x128
5 Conv7_2 5 Conv 3x3x256-s2
Normalization Layers
Detection Layers
Conv3_3
Conv4_3
Conv5_3
Conv_fc7
Conv6_2
Conv7_2
Conv: 3x3x(1x(Ns+4))
Conv: 3x3x(1x(2+4))
Predicted Convolutional Layers
Max-out BG label
Multi-task Loss Layer: SoftmaxLoss and SmoothL1Loss

Figure 4.2: Architecture of the single-shot scale-invariant face detector (S^{3}FD). It consists of base convolutional layers, extra convolutional layers, detection convolutional layers, normalization layers, predicted convolutional layers and a multi-task loss layer (Figure from [7]).

Table 4.1 provides details of the six detection layers.

Detection Layers			
Detection Layer	Stride	Anchor	RF
conv3_3 [32]	4	16	48
conv4_3	8	32	108
conv5_3	16	64	228
conv_fc7	32	128	340
conv6_2	64	256	468
conv7_2	128	512	724

Table 4.1: Stride size, anchor scale, and receptive field (RF) of the six detection layers of the architecture of Figure 4.2

The face detector is trained on 12880 images of the WIDER FACE [51] training set and employs the multi-task loss of [52]

Implementation Details

The convolutional layers Conv3_3, Conv4_3, and Conv5_3 are have exactly the same architecture as the corresponding ones in [48]. The parameters of fc6 and fc7 as we mentioned before, are subsampled in order to initialize the parameters of Conv_fc6 and Conv_fc7. The other additional layers are initialized with the ”Xavier” method [53]. Standard gradient descent is used with 0.9 momentum and 0.0005 weight decay. The batch size is 32, and the learning rate starts at 0.001.

4.3.2 Face Tracking

For each shot detected (Section 4.2), we perform face detection, and all detected faces within each shot are grouped to form face tracks. A face track represents the face of a single person in different video frames. In order to form a face track, we have to establish that two detected faces in different video frames correspond to the same person. So, for each detected shot, we apply the KLT tracker proposed in [54]. This algorithm detects interest points in the first frame of the shot and propagates them to succeeding frames based on local appearance matching. Points that cannot be reliably propagated from one frame to the next are discarded and replaced with new points. The output is a set of point tracks, starting at some frame in the shot and continuing until some later frames [55].

4.4 Face Recognition

After the face tracks are formed, for each face track, a face recognition CNN is employed in order to extract embeddings. Instead of employing the x-vector architecture described in Section 3.3.1, we utilize the visual information and the tracks that formed before, and we use the ResNet-50 architecture [9].

4.4.1 ResNet-50 Architecture

ResNet stands for Residual Network, and it was first proposed in 2015 [9]. There exist various ResNet architectures, one of which is ResNet-50 that consists of 50 layers. The ResNet-50 architecture is based on ResNet-34 that is illustrated in Figure 4.3. ResNet-34 (Figure 4.3, Right) inserts shortcut connections in order to turn a plain network (like the architecture in Figure 4.3 Middle, which is based in VGG-19 architecture in Figure 4.3 Left) into its residual counterpart. Comparing the plain architecture to its residual counterpart, we notice that the residual network has fewer filters and lower complexity.

While ResNet-50 is based on ResNet-34, there is an important difference. In the ResNet-50 architecture, the building block was modified into a bottleneck design (Figure 4.4) due to concerns over the time taken to train the layers. ResNet-50 used a stack of 3 layers instead of the ResNet-34's 2. So, each block in ResNet-34 that had 2 blocks was replaced by a 3-layer bottleneck block. That's how the ResNet-50 architecture was created, which led to a higher accuracy than the ResNet-34 network. Thus, we selected ResNet-50 over any other architecture.

4.4.2 Implementation Details

We use the pre-trained ResNet-50 trained on the VGGFace2 database [56]. During inference, we extract face embeddings 5 times per face track by computing features with a 1.5-sec window and moving the window by 0.75 sec at time. Then, we average the embeddings.

Finally, as in the audio-only pipeline (Section 3.3.3), the embeddings are clustered utilizing AHC [43].

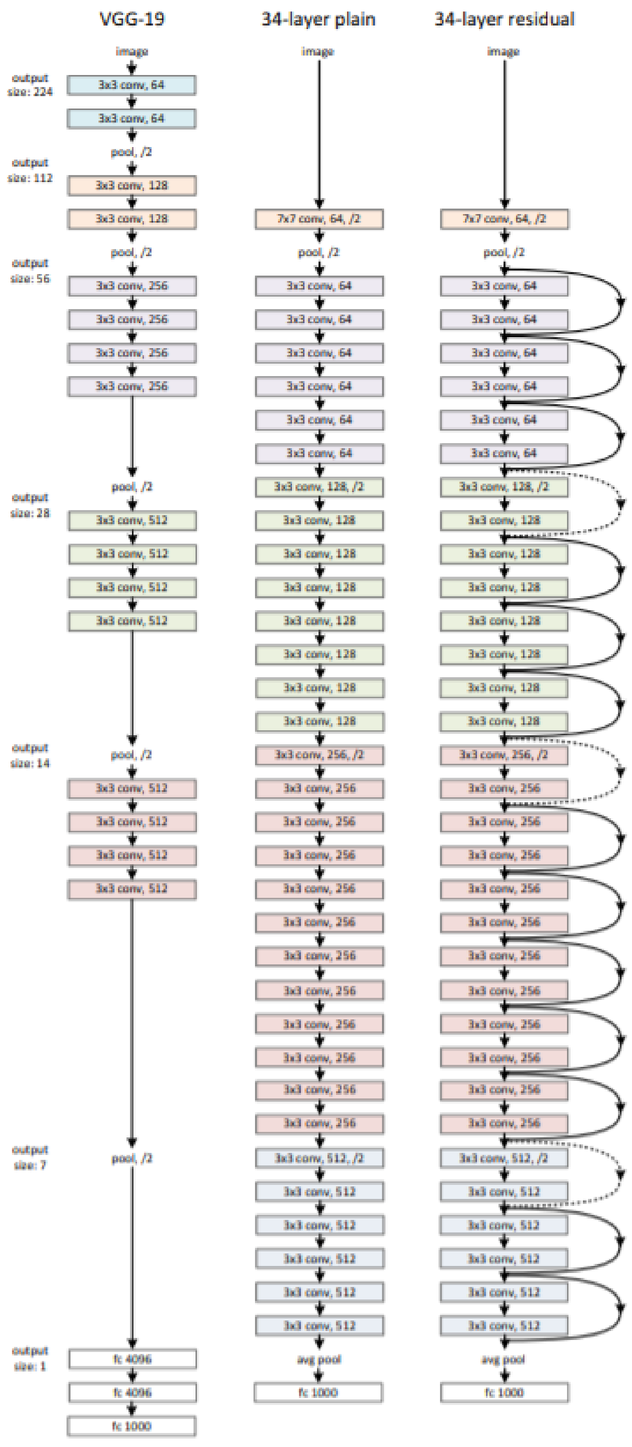

Figure 4.3: Example network architectures for ImageNet. Left: the VGG-19 model [8]. Middle: a plain network with 34 parameter layers. Right: a residual network (ResNet-34) with 34 parameter layers (Figure from [9])

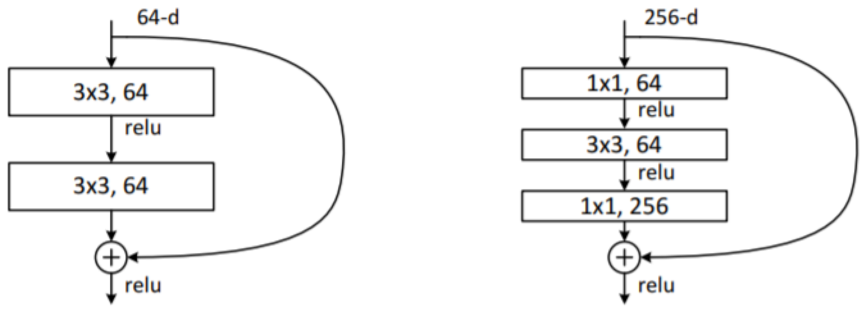

Figure 4.4: A deeper residual function. Left: a building block (on 56×56 feature maps) for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152 (Figure from [9])

4.5 Audio-Visual Synchronization

One of the most important components of the audio-visual pipeline is to synchronize audio and visual information in order to detect the active speaker among other speakers in a video recording.

4.5.1 System Architecture

To address the problem of audio-visual synchronization, we utilize a variant [10] of SyncNet [57]. The architecture of the system is illustrated in Figure 4.5. Like SyncNet [57], the adopted architecture of [10] has exactly the same layer configurations and the same input format. More details follow.

Audio Stream

The input to the audio part of the network are 13-dimensional MFCCs [29], [30]. To extract them, we use librosa [58], a python package for music and audio analysis. MFCCs are extracted every 10 ms using 25-ms frames. Because we have 20 audio frames and 13-dimensional MFCCs, the input to the network has a dimension of 13×20. The network was designed based on the architecture of the VGG-M CNN [59], but as we can see in Figure 4.5, the filter sizes are modified for the audio input size.

Visual Stream

The input to the visual part of the network is a video with resolution of 224×224 and a frame rate of 25 FPS. The network processes 5 RGB frames of faces detected in Section 4.3 at once, corresponding to 0.2 sec. This network is also based on VGG-M CNN [59], but as we can see in Figure 4.5, the first layer has a filter size of $5 \times 7 \times 7$ (instead of 7×7 [59]), in order to capture the motion information over the 5 frames.

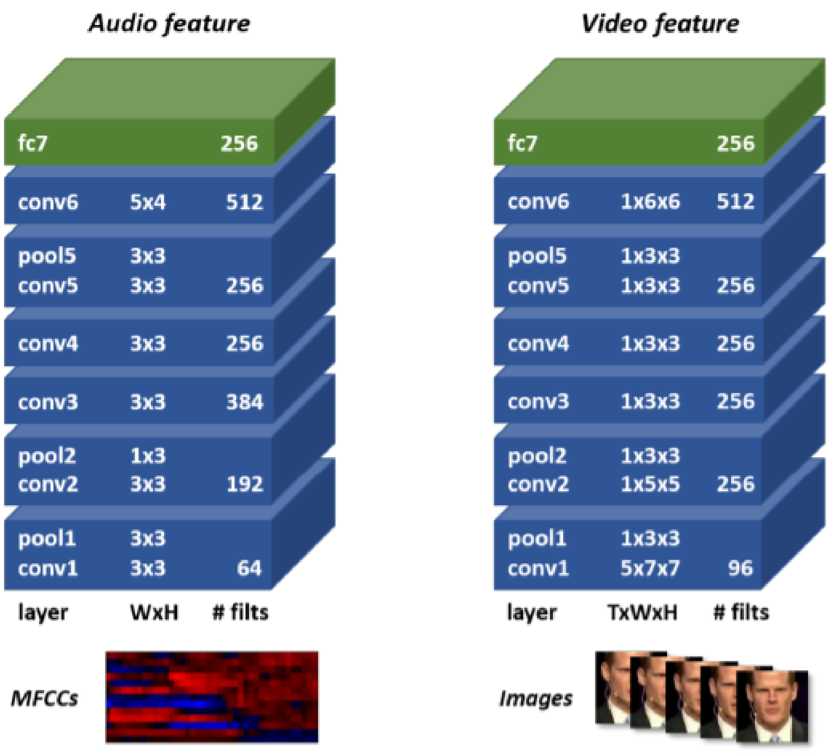

Figure 4.5: Audio-Visual Synchronization System as depicted in [10]

4.5.2 Training Process

The training process of the system we use is illustrated in Figure 4.6. In total, N L2 (Euclidean) distances are calculated between one video feature and N audio features. In order to find matching and non-matching pairs, the system uses a softmax layer for the computed distances and in the final step the system is trained with multi-way cross-entropy loss on the inverse of the N distances. The audio and video streams are synchronized when the distances between features are minimized.

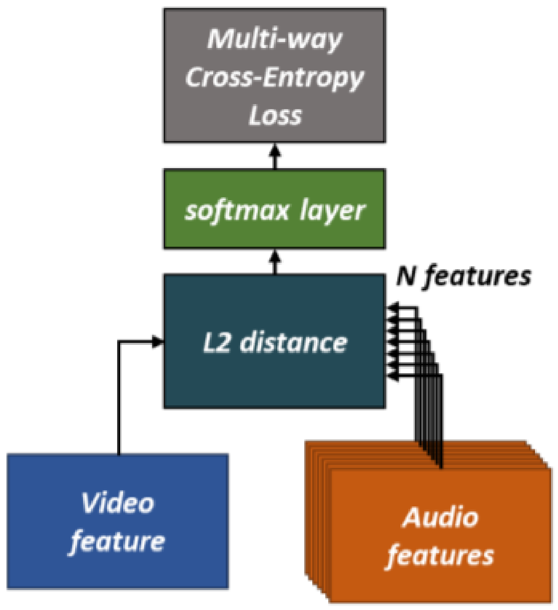

Figure 4.6: The training procedure of the audio-visual synchronization network (Figure from [10])

4.6 Off-Screen Speaker Recognition

In this section, we try to handle speech segments that don't correspond to visible faces in a video stream. In other words, we try to find the identity of speech that comes from off-screen speakers. To achieve this, we utilize the face detector we used in Section 4.3, the speaker embeddings architecture described in Section 3.3.1, the VAD module we used in Section 3.1, and the cross-modal embeddings extracted in Section 4.5. When the output of the VAD module is 1 and the face detector does not detect any faces, we assume that we are dealing with off-screen speech. Then, embeddings are extracted from this audio region using the x-vector architecture described in Section 3.3.1 and are compared to the cross-modal embeddings of every visible speaker in the video. We use the cosine distance in order to achieve this, and if this distance is below a tunable threshold, we assign the speech region to a known speaker. Otherwise, we leave this region as unknown.

4.7 Evaluation

The results of our audio-visual pipeline are examined in Chapter 5.

Chapter 5

Evaluation and Results

In this chapter we evaluate both the audio-only and audio-visual systems. First of all, we compare the two systems to each other and after that, as a baseline, we use the audio-visual system of [11] that employs SSL in order to boost its performance according to [26] and reaches state-of-the-art results in the diarization task. We also compare our audio-visual system to a more traditional multimodal system [12] that employs support vector machines [60] for VAD and hidden Markov models [61] with GMMs for clustering.

5.1 Evaluation Metric

In the evaluation we use the Diarization Error Rate (DER) from pyannote.metrics [28], defined as follows:

$$DER = \frac{false\ \ alarm + missed\ \ detection + confusion}{total},$$

where ”false alarm” is the duration of non-speech incorrectly classified as speech, ”missed detection” is the duration of speech incorrectly classified as non-speech, ”confusion” is the duration of speaker confusion, and ”total” is the total duration of speech in the reference.

5.2 Data

We use the AMI meeting corpus and more specifically the ES (Edinburgh) and IS (IDIAP) meetings. Note that videos IS1002a, IS1003b, IS1005d and IS1007d were not used in the experiments because of missing data.

5.3 Results

Table 5.1 aggregates the DER results of all the systems we compared. As we can see, the best-performing system is the one described in [11] making it clear that the integration of SSL in the diarization pipeline helps the system to improve its performance. Despite the fact that we didn't apply SSL, our results come close to the ones of [11]. This is because our VAD architecture leads to better False Alarm scores as we can see in Table 5.2.

Comparing our two systems, it is obvious that the use of the visual information boosts performance in terms of DER (Table 5.1). Finally, it is also clear that deep learning - based components in a speaker diarization pipeline lead to better results in terms of DER (see DER of [12] in Table 5.1).

System	IS (IDIAP)	ES (Edinburgh)
[11]	**20**	**19.9**
[12]	30.5	-
Ours, audio-only	32.2	32.6
Ours, audio-visual	21.32	22.45

Table 5.1: Diarization results of [11], [12], compared to our developed audio-only and audio-visual systems in terms of DER %

System	IS (IDIAP)	ES (Edinburgh)
VAD Module of [11]	4	6.6
Ours VAD Module	**3.2**	**4.7**

Table 5.2: VAD module performance of the two best-performing diarization systems of Table 5.1, in terms of false alarm rate

Chapter 6

Conclusion and Future Work

In this thesis we address the problem of speaker diarization. For this purpose, we implemented two speaker diarization systems. The first one (Chapter 3) is an audio-only speaker diarization system and the second one (Chapter 4) is an audio-visual System. In particular, we followed a deep learning - based approach. The architecture of every component of our two diarization pipelines is based on DNNs. As a result, both of our systems reach state-of-the-art results in the diarization task, in terms of DER. It is also noteworthy to mention that working directly on the audio waveform and utilizing visual information led to a significant performance improvement in terms of DER.

Nevertheless, our systems do not deal with overlapping speakers, as we do not exploit SSL techniques. Incorporating SSL techniques, based on the microphone array streams already present in the AMI meeting corpus, would allow handling overlapping speakers and potentially improve diarization results.

Bibliography

[1] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan. A review of speaker diarization: Recent advances with deep learning. *arXiv preprint arXiv:2101.09624*, 2021.

[2] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv:1806.05622v2 [cs.SD]*, 2018.

[3] M. Ravanelli and Y.Bengio. Speaker Recognition from raw waveform with SincNet. In *Proc. of SLT*, 2018.

[4] M. Lavechin, M.-P. Gill, R. Bousbib, H. Bredin, and L. P. Garcia-Perera. End-to-end Domain-Adversarial Voice Activity Detection. In *ICASSP 2020, IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[5] R. Yin, H. Bredin, and C. Barras. Speaker change detection in broadcast TV using bidirectional long short-term memory networks. In *Proc. Interspeech Conf*, pages 3827 – 3831, 2017.

[6] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur. Deep neural network embeddings for text-independent speaker verification. In *Proc. Interspeech*, pages 999 – 1003, 2017.

[7] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. S^3FD: Single shot scale-invariant face detector. In *ICCV*, pages 192–201, 2017.

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[10] Soo-Whan Chung, Joon Son Chung, and Hong-Goo Kang. Perfect match: Improved cross-modal embeddings for audio-visual synchronisation. In *ICASSP*, 2019.

[11] J. S. Chung, B.-J. Lee, and I. Han. Who said that?: Audio-visual speaker diarisation of real-world meetings. In *Interspeech*, pages 371–375, 2019.

[12] Ahmad, Zubair, Alquhayz, and Ditta. Multimodal speaker diarization using a pre-trained audio-visual synchronization model. *Sensors*, 19(23):5163, 2019.

[13] S. E. Tranter and D. A. Reynolds. Speaker diarization for broadcast news. In *Proc. Odyssey Speaker and Language Recognition Workshop*, pages 337–344, 2004.

[14] S. Meignier, D. Moraru, C. Fredouille, J.-F. Bonastre, and L. Besacier. Step-by-step and integrated approaches in broadcast news speaker diarization. *Computer, Speech and Language*, pages 303–330, 2006.

[15] J. Ajmera, G. Lathoud, and L. McCowan. Clustering and segmenting speakers and their locations in meetings. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 605–608, 2004.

[16] X. Anguera, C. Wooters, and J. Hernando. Purity algorithms for speaker diarization of meetings data. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1025–1028, 2006.

[17] D. Vijayasenan, F. Valente, and H. Bourlard. An information theoretic approach to speaker diarization of meeting data. *IEEE Transactions on Audio, Speech, and Language Processing*, pages 1382–1393, 2009.

[18] F. Valente, P. Motlicek, and D. Vijayasenan. Variational Bayesian speaker diarization of meeting recordings. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4954–4957, 2010.

[19] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.

[20] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur. X-vectors: Robust DNN embeddings for speaker recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5329 – 5333, 2018.

[21] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe. End-to-end neural speaker diarization with permutation-free objectives. In *Proceedings of the Annual Conference of the International Speech Communication Association*, pages 4300 – 4304, 2019.

[22] Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe. End-to-end neural speaker diarization with self-attention. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 296–303, 2019.

[23] G. Friedland, H. Hung, and C. Yeo. Multimodal speaker diarization of real-world meetings using compressed-domain video features. In *Proc. ICASSP*, pages 4069–4072, 2009.

[24] G. Garau, A. Dielmann, and H. Bourlard. Audio-visual synchronisation for speaker diarisation. In *Proc. Interspeech*, pages 2654–2657, 2010.

[25] J. Schmalenstroeer, M. Kelling, V. Leutnant, and R. Haeb-Umbach. Fusing audio and video information for online speaker diarization. In *Proc. Interspeech*, 2009.

[26] V. Rozgic, K. J. Han, P. G. Georgiou, and S. S. Narayanan. Multimodal speaker segmentation and identification in presence of overlapped speech segments. *Journal of Multimedia*, 5(4):322–331, 2010.

[27] J. Carletta et al. The AMI meeting corpus: A pre-announcement. In *Proc. Int. Conf. Methods and Techn. Behavioral Res.*, pages 28–39, 2005.

[28] H. Bredin. pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems. In *Interspeech*, 2017.

[29] A. Srinivasan. Speaker Identification and Verification using Vector Quantization and Mel Frequency Cepstral Coefficients. *Research Journal of Applied Sciences, Engineering and Technology*, 4(1):33–40, 2012.

[30] V. Tiwari. Mfcc and its applications in speaker recognition. *International Journal on Emerging Technologies*, 2010.

[31] L. R. Rabiner and R. W. Schafer. Theory and Applications of Digital Speech Processing. *Prentice Hall*, 2011.

[32] G. Sell, D. Snyder, A. Mccree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe, and S. Khudanpur. Diarization is Hard: Some Experiences and Lessons Learned for the JHU Team in the Inaugural DIHARD Challenge. In *Proceedings of the 19th Annual Conference of the International Speech Communication Association, INTERSPEECH 2018*, pages 2808–2812, 2018.

[33] J. G. Fiscus, N. Radde, J. S. Garofolo, A. Le, J. Ajot, and C. Laprun. The Rich Transcription 2005 spring meeting recognition evaluation. In *Proc. Machine Learning for Multimodal Interaction Workshop (MLMI)*, pages 369–389, 2005.

[34] O. Galibert, J. Leixa, G. Adda, K. Choukri, and G. Gravier. The ETAPE speech processing evaluation. In *LREC*, pages 3995–3999, 2014.

[35] O. Galibert. Methodologies for the evaluation of Speaker Diarization and Automatic Speech Recognition in the presence of overlapping speech. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association*, pages 1131–1134, 2013.

[36] M. Cettolo. Segmentation, classification, and clustering of an Italian broadcast news corpus. In *Proc. Content-Based Multimedia Inf. Access Conf*, pages 372–381, 2000.

[37] J.-L. Gauvain, L. Lamel, and G. Adda. Partitioning and transcription of broadcast news data. In *Proc. Int. Conf. Spoken Lang. Process*, pages 1335–1338, 1998.

[38] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

[39] S. Ioffe. Probabilistic linear discriminant analysis. *Computer Vision–ECCV 2006*, pages 531–542, 2006.

[40] V. Peddinti, D. Povey, and S. Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Proceedings of INTERSPEECH*, pages 3214–3218, 2015.

[41] Linear discriminant analysis. `https://en.wikipedia.org/wiki/Linear_discriminant_analysis`. Access Date: 17-09-2021.

[42] Measuring similarity from embeddings. `https://developers.google.com/machine-learning/clustering/similarity/measuring-similarity`. Access Date: 17-09-2021.

[43] Hierarchical clustering. `https://en.wikipedia.org/wiki/Hierarchical_clustering`. Access Date: 17-09-2021.

[44] S. S. Chen and P. S. Gopalakrishnam. Speaker, environment and channel change detection and clustering via the Bayesian information criterion. In *Proc. 1998 DARPA Broadcast News Transcription and Understanding Workshop*, pages 127–132, 1998.

[45] M. A. Siegler, U. Jain, B. Raj, and R. M. Stern. Automatic segmentation, classification and clustering of broadcast news. In *Proc. DARPA Speech Recognition Workshop*, pages 97–99, 1997.

[46] R. Yin, H. Bredin, and C. Barras. "Neural speech turn segmentation and affinity propagation for speaker diarization. In *Proc. Interspeech*, pages 1393–1397, 2018.

[47] B. Castellano. Pyscenedetec. `https://github.com/Breakthrough/PySceneDetec`, 2020.

[48] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[49] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *Proc. Int. Conf. Learn. Representations*, 2015.

[50] W. Liu, A. Rabinovich, , and A. C. Berg. Parsenet: Looking wider to see better. *arXiv:1506.04579*, 2015.

[51] S. Yang, P. Luo, C. C. Loy, , and X. Tang. WIDER FACE: A Face detection benchmark. *arXiv:1511.06523*, 2016.

[52] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. Neural Inf. Process. Syst.*, pages 91–99, 2015.

[53] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, pages 249–256, 2010.

[54] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[55] M. Everingham, J. Sivic, , and A. Zisserman. Taking the bite out of automatic naming of characters in tv video. *Image and Vision Computing*, 2009.

[56] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. VGGFace2: a dataset for recognising faces across pose and age. *arXiv preprint arXiv:1710.08092*, 2017.

[57] J. S. Chung and A. Zisserman. Out of time: automated lip sync in the wild. In *Workshop on Multi-view Lip-reading, ACCV*, 2016.

[58] B. McFee and et al. librosa: Audio and music signal analysis in Python. In *Proc. 14th Python Sci. Conf*, 2015.

[59] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. BMVC*, 2014.

[60] J. Dey, M. S. B. Hossain, and M. A. Haque. An Ensemble SVM-based Approach for Voice Activity Detection. In *Proc. ICECE. IEEE*, pages 297–300, 2018.

[61] Lin Lin and Jia Li. Clustering with Hidden Markov Model on Variable Blocks. *Journal of Machine Learning Research*, pages 1–49, 2017.