University of Thessaly
Greece, Winter 2020

Audio-Visual Speaker Diarization in Broadcast News

Οπτικο-Ακουστική Καταλογοποίηση Ομιλητή σε Δελτία Ειδήσεων

Charalampos Vossos

Supervisor: Gerasimos Potamianos

Committee Members: Nikolaos Bellas, Georgios Stamoulis

Diploma Thesis

Department of Electrical and Computer Engineering

University of Thesssaly

Volos, Greece

This Thesis is submitted to the University of Thessaly as part of the requirements for the Diploma of Electrical and Computer Engineering.

Περίληψη

Ένα πολύ διαδεδομένο πρόβλημα στον τομέα της Μηχανικής Μάθησης είναι η καταλογόποιηση ομιλητή. Πρόκειται για τη διαδικασία κατά την οποία απαντάται το ερώτημα του "ποιός μιλάει και πότε" σε δεδομένα ήχου ή βίντεο χωρίς να γνωρίζουμε εκ των προτέρων τη διάρκεια ομιλίας και τον αριθμό των ομιλητών. Η συνήθης διαδικασία με την οποία αυτό επιτυγχάνεται χωρίζεται σε τρία μέρη: στην εξαγωγή χρήσιμων χαρακτηριστικών απο τα δεδομένα ήχου και εικόνας, στην κατηγοριοποίηση της ομιλίας και των ομιλητών και τέλος στην αξιολόγηση του συστήματος.

Στην παρούσα εργασία αναπτύξαμε συστήματα καταλογοποίησης ομιλητή, στα οποία χρησιμοποιούμε δεδομένα είτε μόνο ήχου, είτε μόνο εικόνας, είτε συνδυασμού αυτών. Για την προσέγγιση με τα ηχητικά δεδομένα χρησιμοποίησαμε ενα εργαλείο καταλογοποίσης ομιλητή που αναπτύχθηκε από την εργαστηριακή ομάδα του Πανεπιστημίου στο Le Mans της Γαλλίας και λέγεται LIUM SpkDiarization toolkit [1]. Ως ηχητικά χαρακτηριστικά χρησιμοποιήθηκαν 13 συντελεστές MFCC.

Όσον αφορά την οπτική προσέγγιση δημιουργήσαμε δύο διαφορετικά συστήματα καταλογοποίησης ομιλητή. Και στα δύο συστήματα ως περιοχή ενδιαφέροντος επιλέχθηκε η περιοχή γύρω από το στόμα. Για το πρώτο σύστημα εξετάσαμε καρέ-καρέ τις μετατοπίσεις που σημειώνονται στα pixels της περιοχής ενδιαφέροντος εφαρμόζοντας τεχνική οπτικής ροής (optical flow). Αυτές οι μεταβολές (μέτρο και γωνία) αποτελούν τα οπτικά χαρακτηριστικά του συστήματος. Για την ταξινόμηση σε ομιλία ή μη χρησιμοποιήσαμε Μηχανές Διανυσματικής Στήριξης (SVM). Για το δεύτερο σύστημα ενώσαμε τις περιοχές του στόματος για κάθε ομιλητή, σε διαδοχικά στιγμιότυπα του βίντεο, σε μεγαλύτερες εικόνες. Στη συνέχεια χρησιμοποιήσαμε αυτές τις εικόνες σαν είσοδο σε ένα Συνελικτικό Νυρωνικό Δίκτυο για να ταξινομήσουμε τα δείγματα στις κλάσεις "ομιλία" ή "μη ομιλία".

Για τον συνδυασμό της οπτικής και της ακουστικής πληροφορίας αναπτύξαμε ένα σύστημα καταλογοποίησης ομιλητή σε δύο στάδια. Στο πρώτο στάδιο κρατήσαμε σαν βάση την ακουστική μέθοδο και προσθέσαμε πληροφορία για τον αριθμό των ομιλητών που φαίνονται στο βίντεο εφαρμόζοντας αλγόριθμο ομαδοποίησης των προσώπων.

Στο δεύτερο στάδιο προσθέσαμε επιπλέον πληροφορία για τα διαστήματα επικαλυπτόμενης

ομιλίας. Για να το επιτύχουμε αυτό όπως και στην οπτική προσέγγιση εντοπίσαμε την περιοχή του στόματος και εφαρμόσαμε τεχνική οπτικής ροής. Αθροίζοντας τα μέτρα των μετατοπίσεων των pixels καταλήξαμε σε 2 πιθανούς επικαλυπτόμενους ομιλητές με τις μεγαλύτερες βαθμολογίες εντός παραθύρου διάρκειας 2 δευτερολέπτων. Οι αντίστοιχες βαθμολογίες των ομιλητών αυτών αποτέλεσαν το δίανυσμα χαρακτηριστικών του συστήματος. Για την ανίχνευση της επικαλυπτόμενης ομιλίας χρησιμοποιήθηκαν Μηχανές Διανυσματικής Στήριξης. Στη συνέχεια τα αποτελέσματα του ταξινομητή φιλτραρίστηκαν με εμπειρικά επιλεγμένο κατώφλι και με τον αλγόριθμο ομαδοποίησης Κ-μέσων.

Τέλος και τα πέντε συστήματα καταλογοποίησης αξιολογήθηκαν με βαση τη μέθοδο μέτρησης του Ρυθμού Σφάλματος Καταλογοποίησης (DER) σε τηλεπαράθυρα δελτίων ειδήσεων της Ελληνικής τηλεόρασης απο τη βάση δεδομένων Gridnews [2].

Abstract

A very common problem in the field of Machine Learning is speaker diarization. This is the process of answering the question of "who is talking and when" in audio or video data without knowing in advance the duration of the speech and the number of speakers. The usual process by which this is accomplished is divided into three parts: the extraction of useful features from audio and video data, the classification of speech and speakers, and finally the evaluation of the system.

In this Thesis we have developed speaker diarization systems in which we use either audio-only data, video-only data, or their combination. In the audio-only approach we used a speaker diarization toolkit developed by the Le Mans University Laboratory of Informatics team in France, called LIUM SpkDiarization toolkit [1]. 13 MFCC coefficients were used as audio features.

For the visual-only diarization we implemented two different systems. In both systems the mouth area was chosen as the region-of-interest. For the first one, we examined frame-by-frame the displacements in pixels in the region-of-interest using an optical flow technique. These changes (magnitude and angle) are the visual features of the system. We used Support Vector Machines (SVMs) in order to classify the speakers into "speech" or "non speech" classes. For the second system, we concatenated the mouth areas for each speaker, in consecutive frames, into larger concatenated images. Then, we used these images as input to a Convolutional Neural Network to classify the samples into the "speech" or "non-speech" classes.

To combine visual and audio information we developed a two-stage speaker diarization system. In the first approach we kept the audio method as a basis and added information about the number of speakers shown in the video by applying a face clustering algorithm.

In the second step we inserted additional information about overlapping speech intervals. To achieve this, as in the visual approach, we located the mouth area and applied an optical flow technique. Summing up the pixel displacement magnitudes resulted in 2 potentially overlapping speakers with the highest scores within a 2-second window. The corresponding scores of these speakers constituted the system feature vectors. Support

Vector Machine classifiers were used for speech overlap detection. Then, the classifier results were filtered using an experimentally selected threshold and the k-means clustering algorithm.

Finally, all five diarization systems were evaluated by measuring the Diarization Error Rate (DER) in multi-speaker conversation panels in Greek Broadcast News from the Gridnews database [2].

Acknowledgments

First of all, I would like to thank my thesis advisor Associate Professor Gerasimos Potamianos for all his help, observations, and time spent developing this diploma thesis. His guidance and support have helped me succeed in this task. I would also like to thank my fellow student and friend Georgios Gkountouras for the great collaboration we had during the past year. Last but not least, I would like to thank my family for all their support throughout these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Speaker Diarization Task

The Speaker Diarization (SD) problem is a crucial problem in pattern recognition. Many different approaches to this task have been implemented in the last decades. The main goal of SD is to find out who speaks and when in an audio or video stream. The continually increasing reach of technology, high tech cameras, and mobile phones, as well as the large amount of multimedia data necessitates robust speaker diarizartion. The common steps followed for building speaker diarization systems are the following:

- The audio signal is classified into speech and non speech segments.
- The segments that belong to the same speaker are grouped in the same cluster.

However, many studies have proven that the role of visual information is very important in speaker diarization as well. So, in the last few years many approaches have been implemented that combine both audio and visual information in several ways. When only one modality is used, the speaker diarization task may face challenges that make diarization quite difficult.

For the audio-only diarization task, there are several problems that make the diarization harder. In particular, the acoustic signal may come from several speakers, or it may come from other sources that exist in the background or have some noise or overlapping speech. On the other hand, the video-only approach is very challenging as well, since it only

focuses on the lips and facial motion detection. Hence, in cases where the face is not in a frontal pose or the camera isn't close enough to capture lip movements, speaker detection becomes really hard.

As a way to address the challenges of the two individual modalities, new methods were implemented that combine useful information from each modality to create an audio-visual approach. Feature extraction is a very important part for every speaker diarization task. Fusion can then occur at the feature extraction stage (called early fusion), or after the separate decisions of each modality (called late fusion). In this thesis, we implemented a late fusion method.

1.2 Thesis Contribution

The purpose of this thesis is to explore the idea of audio-visual fusion for speaker diarization ("who speaks and when"). We study how to achieve better results in speaker diarization by combining audio and visual cues. Then, we compare the resulting approach to single-modality speaker diarization. In the audio case, an open-source toolkit that was developed by Le Mans University (Laboratoire d'Informatique de l'Université du Mans) called LIUM SpkDiarization toolkit [1] was used to find the speaker identities. In the visual part, the motion of lips was employed as an indicator of speech. The audio-visual fusion method was implemented in two stages. First, the audio-only approach was combined with information about the number of speakers that is revovered from the video sequence. Additionally, the key observation that this method cannot handle more than one speaker leads to an extended version that detects speech overlaps through tracking concurrent multi-speaker lip motion. Finally, all systems were evaluated in discussion panels of Greek broadcast news from the Gridnews database [2].

1.3 Related Work

In the speaker diarization task the main purpose is to find speech segments and to cluster segments that belong to the same speaker. In order to achieve that, we can use audio,

visual, or a combination of audio and visual information. In the audio-only case, the Mel Frequency Cepstral Coefficients (MFCCs) are a common choice of audio features when each speech segment corresponds to a single speaker. One of these methods was implemented in [6], where the authors extracted the MFCC feature vector for each frame and employed agglomerative clustering, so that each generated cluster corresponds to a different speaker. Subsequently, consecutive speech frames were grouped either into segments of the same speaker, or into another speaker cluster by using a Hidden Markov Model (HMM).

Among others, visual speech diarization was addressed in [7], where the authors focused on lip activity detection. The solution introduced in this paper was to detect the movement of the lips and then classify faces into the "speech" or "non speech" classes for TV data. For facial feature detection they used a detector based on the Active Shape Model (ASM) [8]. For facial feature extraction, faces were detected by the OpenCV library implementation of the Viola-Jones algorithm. After that, the pose of the faces was estimated by matching them to a 68-landmark location model. Then, the authors used the lips region in order to measure lip motion. By using optical flow techniques, they calculated pixel displacement between the mouth regions of two consecutive frames. Afterwards, they computed the entropy of pixel displacement of a mouth region and summed them. For classifying the faces as speaking or not, they implemented one thresholding method and one method with mean squared difference between pixels in consecutive mouth regions.

Another interesting approach of audio-visual diarization was implemented in [9]. There, the authors implemented two methods in order to synchronize the audio and visual features. Then, they compared the results of the two methods separately. One was mutual information and the other was Canonical Correlation Analysis (CCA). For audio features, they used MFCCs and for visual features they implemented the Kanade–Lucas–Tomasi (KLT) Optical Flow algorithm with skin color detection techniques in order to find the motion in the lips region. Because they collected some useful parts of the image, like the lips region, and not the whole image, the computation of synchrony became computationally tractable. Afterwards, they used vertical and horizontal movements and they asserted that the audio features are more correlated with the vertical motion of lips. Finally, their algorithm was evaluated by using mutual information and CCA.

Furthermore, a multimodal diarization system on talk-shows was proposed in [10]. The system was divided into feature extraction both in the visual and audio domains, model creation, and classification of the show speech segments. MFCCs were used as audio features. In the visual approach, the authors considered features that characterize the participants' clothing. For the training patterns, the authors collected shots that are long enough and contain faces in the foreground and performed lip activity detection. Then the data were clustered, so that each cluster corresponds to a different speaker. Finally, for the remaining video (parts of the talk show that weren't selected in the previous stage), an SVM classifier was used for the speaker classification.

An additional multimodal speaker diarization approach was proposed in [11]. There, the authors used meeting videos for training and testing their system. In this dataset videos of all the participants are seen in full body motion and with non-frontal face view. The extracted features in the audio domain were MFCCs. On the other hand, the visual features were extracted in grey-scale images by the image difference method. Subsequently, the audio and visual features were concatenated. Then, the authors employed an agglomerative clustering algorithm in order to group the different speakers. Finally, the system was evaluated on 2-person meeting data.

1.4 Thesis Overview

This thesis is divided into 7 chapters. After this introductory chapter, the remainder has the following structure:

- In **chapter 2**, we describe the Greek public broadcast news dataset in detail and the form of its accompanying transcription files. Furthermore, we explain how we created ground-truth annotations from these files.
- In **chapter 3**, we analyze the audio-only approach. We present the subsystems of the SpkDiarization toolkit by LIUM and their interactions. We explain the steps for audio feature extraction, the probabilistic models used, as well as the clustering algorithms, along with their corresponding theoretical formulation.
- In **chapter 4**, we describe the two video-only approaches. We begin with

an introduction to Support Vector Machines (SVMs) and Convolutional Neural Networks used as part of our methods. Afterwards, we offer a complete description of the algorithms and libraries that we used (face detection through the dlib library, centroid tracking, and optical flow). Lastly, we describe in detail our two implemented diarization systems.

- In **chapter 5**, we introduce the combination of audio and visual modalities. Specifically, we augment the audio-only approach with face clustering information and the detection of overlapping speech segments.
- In **chapter 6**, we present the results of our implemented diarization systems (five in total), comparing and commenting on the results, also providing characteristic examples.
- In **chapter 7**, we conclude the thesis by summarizing our findings. Finally, we provide some ideas for future work on the speaker diarization task.

Chapter 2

Dataset

2.1 Dataset Description

For developing and evaluating our speaker diarization systems we used part of a corpus of videos with multi-speaker conversations in Broadcast News of Greek television [2]. The dataset contains 14 clips with a total duration of 1 hour and 36 minutes. These clips were extracted from 11 Broadcast News videos having a duration of about one hour per Broadcast with a resolution of 288×352 pixels, 25 fps frame rate and encoded as MPEG-4. The number of frames in each video ranges from 5200 to 19200, and the total number of frames for the whole dataset is about 121570. Each clip contains a single discussion topic and has a duration of 4-12 minutes. In order to cut the dataset videos we used the free, open source avidemux video editing program [12].

In each video we see the main speaker that directs the conversation and the panel that takes part in the discussion. Each of the participants (host and guests) is in a single window and their number is typically between 2 and 5. All speak at least once during the video and there are no silent segments. In each clip the number of participants is constant, that means that no one enters or leaves. All the participants are visible in frontal face pose and we can see only their upper body. The speakers don't move during the video, and the camera records from a constant position. Moreover, there are some cases where the camera zooms in at the current speaker (see Figure 2.1), but that doesn't last a long time before the camera goes back to the entire panel. A crucial characteristic

(a) **(b)**

Figure 2.1: Two cases of video frames. In (a) there is a frame with the participants and a highlight window and in (b) a frame where the camera zooms in at the speaker.

of the dataset videos is that while the speakers talk to each other, there is sometimes a different window featuring some highlights (see Figure 2.1) that are related to the subject of discussion but doesn't include any speakers. Note that highlight windows may contain incidental (transient) people.

In this thesis, for the visual diarization system implementation we are only interested in the mouth region of every speaker. One additional consideration which is very challenging for the diarization system implementation is that there are some segments in the clips where two or more speakers speak at the same time (overlapping speech). A significant portion of this thesis is devoted to developing a method for detecting this special case.

For the audio part, we transformed the extracted clips into WAV format using the VLC media player software. The audio files have 44100 samples per second (44.1 kHz) and 2 channels (stereo information).

2.2 Transcription Files

Each of the 11 Broadcast News videos is accompanied by one transcription file. These files contain the words of the speakers from Broadcast News data in the Greek language. They also include some other information about turns, speakers, sections, acoustic conditions, and other events. The data are stored in an XML file. In particular, the transcription

contains sections, which are divided into turns. The turns are divided into segments. Changes in acoustic background conditions can occur independently from sections and turns.

At the beginning of the file, there is a header that contains metadata information about the file format, such as encoding and XML version. In our case the encoding is Windows-1253 (Greek). This is followed by a list with all the names of the speakers, their genders, their dialect, and their ids (every speaker has a unique id). Furthermore, the file contains a list of events. In our transcription files there are four different kinds of events:

- **noise** - Speech with noise in the background.
- **music** - Speech with music in the background.
- **non Greek** - Speech in a language other than Greek.
- **advertisements**

This is followed by the main part of the transcription with the speakers' words and headers, which contains the section and the turn. Each section includes the duration of speech (start time, end time) and the section type. There are two section types:

- **report** - Clearly articulated speech in the studio without noise.
- **non-trans** - There is no speech.

Each turn contains the speaker id, the mode, the audio fidelity, the channel, as well as the turn start-time and end-time. In this thesis we are only interested in the speaker ids and the speech duration and not in the contents of speech (words). In case of overlapping speech, the only change is that in the header there are two speaker ids and one time period during which both speak. There aren't any cases where more than two participants speak in the transcription files. Of course, it is possible in a video to have more than 2 overlapping speakers. In this situation, the transcription file contains two different sections with small, consecutive time periods of speech and two overlapping speakers each.

An example of a transcription file follows:

```
<Episode>
</Turn>
</Section>
<Section type="report" startTime="21.012" endTime="82.711">
<Turn speaker="spk0" mode="planned" fidelity="high"
channel="studio" startTime="21.012" endTime="82.711">
<Sync time="21.012"/>
<Event desc="music" type="noise" extent="begin"/>
Κυρίες και κύριοι καλησπέρα σας, ας δούμε την επικαιρότητα
<Event desc="music" type="noise" extent="end"/>
με τίτλους.
<Sync time="24.222"/>
Από κόσκινο
<Event desc="music" type="noise" extent="begin"/>
όλες οι συναλλαγές
<Sync time="26.229"/>
χρηματιστηριακών εταιριών με ασφαλιστικά ταμεία, την τελευταία
εξαετία.
</Turn>
</Section>
</Episode>
```

2.3 Ground Truth

In order to get the ground truth for the dataset, we utilised a python script for parsing the transcription files. It was very fortunate that the files were XML documents, because it means that all the information was stored as text in a tree-based format. This parsing was implemented only for the duration of the cropped video clip, not the entire transcription file. To do that, we found the start and end-times of the clip and we parsed only between those two boundaries. From this, we extracted for all speech sections the duration of speech and the speaker ids. We ignored all other types of events that are included in the transcription file such as music, noise, and advertisements.

Figure 2.2: 3 face tracks in 8 consecutive frames. The 1st face track contains a speaker, therefore all the target labels for it are 1. All other face target labels are 0.

The purpose of a supervised learning algorithm is to find a function that maps the relationships between input training samples and targets (ground truth). So it is crucial to have well-defined target values. In this speaker diarization task, the target is a binary label (speaking or not) for each participant in a frame. In order to create the target data, we used the extracted values from the transcription (speech duration in frames, speaker ids) in the following way:

- We set a numpy vector with size equal to the number of speakers multiplied by the total number of video frames and initialized it with zeros.
- For every frame included in the extracted duration of speech, we set the speaker targets for the current frame to 1 and we didn't make any changes for any other participant targets in the frame.

Obviously, in overlapping speech cases we set the target values to 1 for all overlapping speakers in the frame.

2.4 Reference Annotations

In the next step, we inserted the parsed values (speech durations, speaker ids) into an annotation (see Figure 2.3) by assigning the speaker id to the corresponding time of speech. Note that the speech duration was converted from frames to time by dividing the frame number by 25 (number of frames per second). In order to create the annotation, the

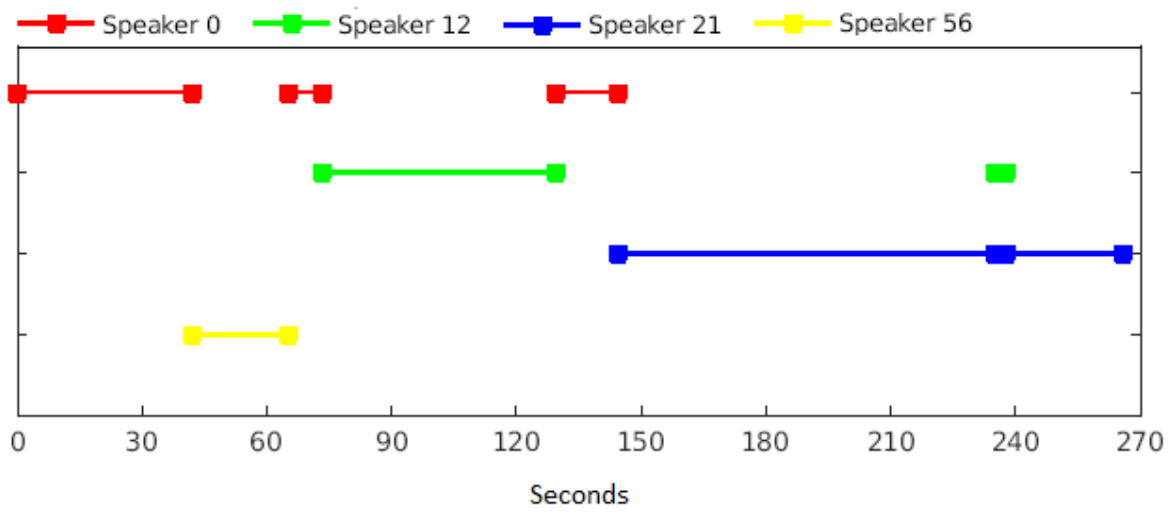


Figure 2.3: A visualization of an annotation.

python pyannote library was used [13]. In particular, the speech duration was converted into an appropriate type in order to be inserted into the annotation. To achieve that, we used the pyannote.core.Segment class, which functions as a description of temporal fragments with the following form:

[00:00:00.000 – 00:01:30.425]

Above, the time is formatted as hours:minutes:seconds.milliseconds.

Afterwards, in order to create the reference annotations we used the pyannote.core.Annotation class. Some characteristics of that class are the following:

- The inserted segments are sorted by start-time or, in cases of tie, by end-time.
- It's not possible to add the same track twice.
- Only non-empty tracks can be inserted.

A track refers to a pair (support, name), where support is the speech segment duration and name is the speaker id. Furthermore, in overlapping speech situations it is possible to add multiple different names (speaker ids) to the same support (speech segment). This is a part of a reference annotation file:

[00:00:00.000 – 00:00:41.847] - spk0
[00:00:41.847 – 00:01:05.100] - spk56
[00:01:05.100 – 00:01:13.041] - spk0
[00:01:13.041 – 00:02:09.172] - spk12
[00:02:09.172 – 00:02:24.267] - spk0
[00:02:24.267 – 00:03:54.692] - spk21
[00:03:54.692 – 00:03:57.578] 0 spk21
[00:03:54.692 – 00:03:57.578] 1 spk12
[00:03:57.578 – 00:04:25.656] - spk21

The reference annotation is used for system evaluation by comparing it to the hypothesis annotation produced by our diarization system. Both the reference and the hypothesis annotations were inserted into python metrics tools in order to measure system accuracy. In this thesis, the Diarization Error Rate (DER) metric was computed by python library "pyannote metrics" [14].

Chapter 3

Audio Speaker Diarization

In this chapter we present our audio-only approach using the LIUM speaker diarization toolkit [1]. Specifically, we provide an overview of the toolkit and the possibilities it provides for speaker diarization in Broadcast News. In addition, we describe the stages of feature extraction (MFCC), segmentation, clustering, and the mathematical background of the statistical models.

3.1 Preliminaries

3.1.1 Gaussian Mixture Models

A Gaussian Mixture is a probability function comprised of several Gaussian distributions, each identified by $k \in \{1, ..., K\}$, where K is the number of mixtures. Each Gaussian k in the mixture is fully determined by the following parameters:

- A mean μ_k that defines its center.
- A covariance Σ_k that defines its spread.
- A mixing probability π_k that defines the contributions of the Gaussian function to the mixture.

The mixing coefficients satisfy:

$$\sum_{k=1}^{K} \pi_k = 1. \tag{3.1}$$

The general equation of a single Gaussian density function is:

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right), \tag{3.2}$$

where $\mathbf{x}$ represents the data points, D is the number of dimensions of each data point, μ is the mean vector and Σ is the covariance matrix. The natural logarithm of this equation is:

$$\ln \mathcal{N}(\mathbf{x}|\mu, \Sigma) = -\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu). \tag{3.3}$$

Let:

$$p(z_{nk} = 1|\mathbf{x}_n), \tag{3.4}$$

be the probability that data point $\mathbf{x}_n$ comes from Gaussian k, where z is a latent variable with the value of one when $\mathbf{x}_n$ comes from Gaussian k, and zero otherwise. Then:

$$\pi_k = p(z_k = 1). \tag{3.5}$$

This means the probability of observing a point from Gaussian k is equivalent to that Gaussian mixing coefficient. Let $\mathbf{z}$ be the set of all latent variables:

$$\mathbf{z} = \{z_1, \cdots, z_K\}. \tag{3.6}$$

Each $\mathbf{z}$ is independent and can only have a value of one when the point comes from k.

This implies:

$$p(\mathbf{z}) = p(z_1 = 1)^{z_1} p(z_2 = 1)^{z_2} \cdots p(z_K = 1)^{z_K} = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{3.7}$$

The probability of observing $\mathbf{x}_n$ conditioned on the event that it came from Gaussian k is the Gaussian itself, so:

$$p(\mathbf{x}_n|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)^{z_k}. \tag{3.8}$$

The Bayesian rule states that:

$$p(\mathbf{x}_n, \mathbf{z}) = p(\mathbf{x}_n|\mathbf{z})p(\mathbf{z}). \tag{3.9}$$

We marginalize by summing up the terms on $\mathbf{z}$:

$$p(\mathbf{x}_n) = \sum_{k=1}^{K} p(\mathbf{x}_n|\mathbf{z})p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k). \tag{3.10}$$

We need to find the maximum likelihood of the model in order to determine the best values for the GMM parameters. The likelihood is the joint probability of all observations $\mathbf{x}_n$:

$$p(\mathbf{X}) = \prod_{n=1}^{N} p(\mathbf{x}_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k). \tag{3.11}$$

Taking the natural logarithm of each side:

$$\ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k). \tag{3.12}$$

From Bayes rule (3.9):

$$p(z_k = 1|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|z_k = 1)p(z_k = 1)}{\sum_{j=1}^{K} p(\mathbf{x}_n|z_j = 1)p(z_j = 1)}. \tag{3.13}$$

From (3.5), (3.8), and (3.13):

$$p(z_k = 1|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)} = \gamma(z_{nk}). \tag{3.14}$$

3.1.2 Expectation - Maximization Algorithm

Evaluating (3.12) directly is hard, so we use an iterative method to find the best parameters for the Gaussians in the mixture. Those parameters are:

$$\theta = \{\pi, \mu, \Sigma\}, \tag{3.15}$$

- **Step 1: Initialize** θ. We can use a different algorithm for this part.
- **Step 2: Expectation**. Evaluate:

$$p(\mathbf{Z}|\mathbf{X}, \theta). \tag{3.16}$$

  The expectation of z_{nk} is:

$$p(z_{nk}|\mathbf{X}, \theta) = \mathbb{E}[z_{nk}] = \sum_{j=1}^{K} z_{nj}\gamma(z_{nj}) = \gamma(z_{nk}), \tag{3.17}$$

  which is the same result we found in (3.14).

- **Step 3: Maximization**. Find the new parameters θ^* using:

$$\theta^* = \arg\max_{\theta} \mathcal{Q}(\theta^*, \theta), \tag{3.18}$$

  where:

$$\mathcal{Q}(\theta^*, \theta) = \mathbb{E}[\ln p(\mathbf{X}, \mathbf{Z}|\theta^*)] = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta) \ln p(\mathbf{X}, \mathbf{Z}|\theta^*), \tag{3.19}$$

where $p(\mathbf{Z}|\mathbf{X}, \theta)$ is known from the Expectation step. In (3.19), $p(\mathbf{X}, \mathbf{Z}|\theta^*)$ is the complete likelihood of the model:

$$p(\mathbf{X}, \mathbf{Z}|\theta^*) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi^{z_{nk}} \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)^{z_{nk}}, \tag{3.20}$$

which is derived from calculating the joint probability of all observations $\mathbf{x}_n$ and latent variables z_{nk} and an extension of (3.11).

Taking the natural logarithm of each side:

$$\ln p(\mathbf{x}, \mathbf{z}|\theta^*) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk}[\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\mu_k, \sigma_k)]. \tag{3.21}$$

Replacing (3.17) and (3.21) in (3.19) and applying a Lagrangian multiplier for the mixing coefficients π:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk})[\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\mu_k, \sigma_k)] - \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right). \tag{3.22}$$

In order to find the parameters that maximize the likelihood, we take the derivative of $\mathcal{Q}$ with respect to π and set it to zero:

$$\frac{\partial \mathcal{Q}(\theta^*, \theta)}{\partial \pi_k} = \sum_{n=1}^{N} \frac{\gamma(z_{nk})}{\pi_k} - \lambda = 0. \tag{3.23}$$

Rearranging and summing over k:

$$\sum_{n=1}^{N} \gamma(z_{nk}) = \pi_k \lambda \Longrightarrow \sum_{k=1}^{K} \sum_{n=1}^{N} \gamma(z_{nk}) = \sum_{k=1}^{K} \pi_k \lambda. \tag{3.24}$$

From (3.1), the sum of all mixing coefficients π is 1. We know that the sum of probabilities γ over k is also 1. Thus, $\lambda = N$, and solving for π:

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N}. \tag{3.25}$$

In the same manner, we can differentiate $\mathcal{Q}$ with respect to μ and Σ, set the derivative to zero and solve for the parameters using (3.3):

$$\mu_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})}, \tag{3.26}$$

$$\Sigma_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T}{\sum_{n=1}^{N} \gamma(z_{nk})}. \tag{3.27}$$

We can use these revised values to determine γ for the next EM iteration, until the likelihood value converges. We are guaranteed to reach a local maximum of the log-likelihood.

3.1.3 Hidden Markov Models

A Hidden Markov Model [15] is a statistical model in which the system is assumed to be a Markov process with hidden states. Hidden Markov Models have many applications, including temporal pattern recognition such as speech, handwriting, gesture recognition, tagging, musical score following, and bioinformatics. They can be considered an extension of Markov Chains where the states cannot be observed and only the system outputs are visible. A brief introduction to Markov Chains follows.

Markov Chains are a type of a random process. The chain has a set of states $\{S_1, \ldots, S_k\}$. The defining property of a Markov Chain is:

$$\mathbb{P}(X_t = j | X_1 = i_1, \ldots, X_{t-1} = i_{t-1}) = \mathbb{P}(X_t = j | X_{t-1} = i_{t-1}). \tag{3.28}$$

This means that the probability of being in a state j depends only on the immediately previous state. A Markov Chain can be described by a transition matrix P whose element $p_{i,j}$ is the probability of moving from state i to state j. The sum of each row of this matrix is 1, since it's the probability that *any* state will be next after state i.

Extending the above to a Hidden Markov Model means we assume the existence of an invisible Markov Chain that we cannot observe, but each state can generate one of K

observations randomly, which can be measured.

3.1.4 Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm for the efficient calculation of the most probable sequence of hidden states that corresponds to a series of known observations. That sequence is the Viterbi path. Let $o_1, \ldots, o_t$ be a sequence of observations. For each state i and $t = 1, \ldots, T$ we define:

$$n_t(i) = \max_{i_1,\ldots,i_{t-1}} \mathbb{P}\{x_1 = i_1, \ldots, x_{t-1} = i_{t-1}, x_t = i, o_1, \ldots, o_t\}, \tag{3.29}$$

as the maximum probability of a path which ends at time t at state i given those observations. According to the Markov property, if the most likely path that ends with state i at time t includes some state i^* at time $t-1$, then i^* is the last state of the most likely path which ends at time $t-1$. This means the maximum probability at time t is defined by the recursive function:

$$n_t(i) = \max_i p_{i,j} a_j(o_t) n_{t-1}(i), \tag{3.30}$$

where $a_j(o_t)$ denotes the probability that the output is o_t at hidden Markov state j.

3.2 LIUM SpkDiarization

LIUM SpkDiarization (originally [1], improved in [16]) is an open-source set of Java tools for speaker diarization by the Laboratoire d'Informatique de l'Université du Mans. It can be used to target multimedia applications in radio, TV shows, and Broadcast News environments. It can find the speaker identity and gender, the channel type (narrow bandwidth vs. wide bandwidth), and the nature of the background (quiet vs. music). In its default settings, it is appropriate for speech recognition: short segments (less than 20 seconds) featuring a single speaker in a single channel. Its modular design (segmentation, classification, Viterbi decoder, GMM, etc.) facilitates the development

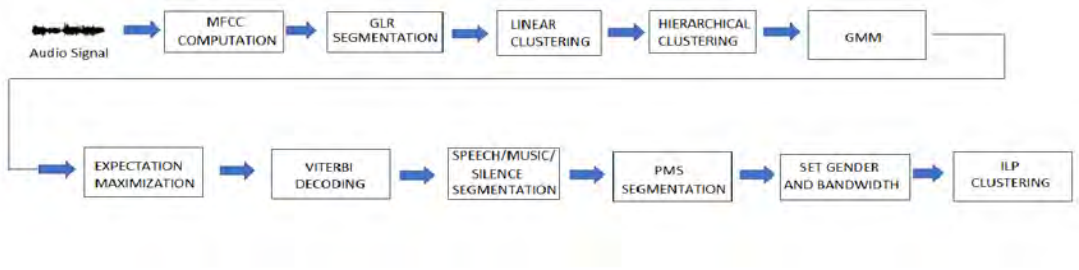


Figure 3.1: The complete audio diarization method of LIUM.

of extended diarization systems. The toolkit has evolved from earlier research efforts by LIUM for the ESTER 2 campaign [17]. In this thesis we employed LIUM SpkDiarization in a configuration oriented for Broadcast News.

3.3 Extraction of MFCC features

Many speech recognition and diarization systems, including the LIUM SpkDiarization toolkit, use MFCCs (Mel Frequency Cepstral Coeeficients) as features [18, 19]. The steps for calculating the MFCC features are:

- **Framing**: The signal is split to small segments called frames. Each frame is a window of 25ms with a step of 10ms and an overlap of 15ms. Shorter frames lead to unreliable spectrum estimates, while longer frames change too much within the window. The Java speech recognition library Sphinx4 in its 16kHz mode is used for the extraction, which means each window has a length of 0.025 * 16000 = 400 samples.

- **Discrete Fourier Transform (DFT)**: For each frame, the *periodogram estimate* of the power spectrum is calculated. This is achieved by computing the DFT on the windowed signal of that frame. The Fast Fourier Transform (FFT) is an $O(Nlog_2N)$ algorithm that computes the DFT faster than the naive $O(N^2)$ approach.

- **Mel filtering**: The Mel filterbank is applied to the frequency spectra. Since closely spaced frequencies are difficult for humans to discern, the frequency bins inside a frequency region are summed to compute the energy inside that region. The filters

are wider at higher frequencies and spaced according to the Mel scale.

- **Logarithm of filterbank energies**: Next, the logarithm of each filterbank energy is taken. This transformation is also motivated by human hearing: loudness is perceived in a logarithmic fashion. Therefore, large variations in energy may lead to small changes in perception. Note that although using the logarithm allows us to employ Cepstral Mean Subtraction (CMS) as a normalization technique, no such step is included here.

- **Discrete Cosine Transform (DCT)**: In the final step, the DCT of the log- -filterbank energies is computed. Because the filterbanks are overlapping, their energies are correlated. The DCT can help decorrelate the energies, which means diagonal covariance matrices can be used to model the features in e.g. an HMM classifier. The DCT can also be used to reduce the dimensionality of the features by dropping the higher coefficients (fast changes), although this does not happen here.

- **Deltas calculation**: Deltas and delta-deltas are the first and second derivatives of the MFCCs respectively. They are also known as velocity and acceleration coefficients. It is reasonable to assume that information exists not just in the power spectrum of a single frame but also in the change of the energies over time. The computed deltas and delta-deltas are appended to the MFCCs to create the final audio feature vector.

For the first three steps of the LIUM speaker diarization toolkit (BIC segmentation, BIC clustering and Viterbi decoding segmentation), the features include 13 MFCCs (with computed coefficient C_0 as energy), without using any delta or delta-delta information.

3.4 Generalized Likelihood Ratio Segmentation

Segmentation helps detect the speaker changes in the signal and uses them to create segments. Each segment is assumed to contain exactly one speaker. Later, segments that belong to the same speaker are grouped.

The first segmentation stage includes two passes. The first pass is based on the Generalized Likelihood Ratio (GLR) [20] distance measure between adjacent speech segments.

The reasoning behind this step is that segments from different sources will have a greater distance between segments than if they came from the same source. The algorithm finds the instantaneous change points and splits the signal into segments. The GLR is computed using Gaussians with full covariance matrices calculated over a sliding window of 5 seconds. When the GLR peaks locally, a segment boundary is detected in the middle of the window.

3.5 Segmentation Based on BIC

The second pass is used to fuse consecutive segments that belong to the same speaker. It is based on the ΔBIC [21] measure and uses full-covariance Gaussians. ΔBIC is defined as:

$$\Delta BIC_{i,j} = \frac{n_i + n_j}{2} \log|\Sigma| - \frac{n_i}{2} \log|\Sigma_i| - \frac{n_j}{2} \log|\Sigma_j| - \lambda P, \tag{3.31}$$

where $|\Sigma_i|$, $|\Sigma_j|$, and $|\Sigma|$ are the determinants of the Gaussians associated with the clusters i, j and $i \cup j$, λ is a parameter, and P is a penalty factor defined as:

$$P = \frac{1}{2}\left(d + \frac{d(d+1)}{2}\right) + \log(n_i + n_j). \tag{3.32}$$

P is dependent on the number of dimensions d of the features and the length of cluster i and cluster j, n_i and n_j respectively. Note that this penalty in speaker diarization doesn't take into account the length of the whole data as defined originally in [21]. Instead, only the length of the candidate clusters to merge, i and j is used, since later experiments [22, 23] show that better results are obtained that way.

3.6 Hierarchical Clustering

The second segmentation stage involves a Hierarchical Agglomerative Clustering. It starts with segments from the previous stage, assigning each segment to its own cluster. Each cluster is modeled by a Gaussian with a full-covariance matrix. The $\Delta BIC_{i,j}$ of (3.31) is used to choose which clusters to group, as well as a criterion for stopping the merging process. At each iteration the two closest clusters, i and j, are merged, until the best BIC distance is positive ($\Delta BIC_{i,j} > 0$).

3.7 Viterbi Decoding

At the next stage, 12 MFCCs with deltas are used as features. Note that the energy (which is the first coefficient) is ignored here. A Viterbi decoding (section 3.1.4) is employed to create a new segmentation. Each cluster is modeled by an HMM (section 3.1.3) with one hidden state which is modeled by a GMM (section 3.1.1). The GMM has a diagonal covariance matrix and 8 components. It is trained via the EM algorithm (section 3.1.2) on the segments of the previous stage (section 3.6). The HMM log-penalty is set experimentally.

3.8 Speech/Music/Silence Segmentation and Filtering

3.8.1 Speech/Music/Silence Segmentation

This stage also uses 12 MFCCs with deltas. Another Viterbi decoding using 8 HMMs with one state each is applied to classify segments into speech/non-speech categories (PMS segmentation). The 8 models feature 2 silence models (wideband and narrowband), 3 wideband speech models (clean, with noise, and with music), 1 model of narrowband speech, 1 model of jingles, as well as 1 model of music. Each hidden state of these HMMs

is represented by a mixture of 64 Gaussians, and each Gaussian has a diagonal covariance matrix.

Note that, unlike most other diarization systems, this stage comes after the segmentation of section 3.7. This leads to better results. It is hypothesized that this is because the speaker segmentation cannot make correct decisions when it comes to the boundaries at the end of each segment. This problem becomes worse when the speech/non-speech segmentation is done first, presumably because it generates many segments to cut.

3.8.2 Segmentation Filtering

At the end of this stage, the segments are filtered according to the PMS segmentation. This utilises 12 MFCCs with deltas as well. The minimum length of a silence segment is set to 1 MFCC window (25 ms) and the same applies for the segment padding. Meanwhile, the minimum length of a speech segment is set to 6 MFCC windows (150 ms).

3.9 Gender and Bandwidth Detection

This is another stage that uses 12 MFCCs with delta coefficients and no energy as features. There are 4 different combinations of gender and bandwidth:

- **Male Narrowband**
- **Male Wideband**
- **Female Narrowband**
- **Female Wideband**

Each combination is detected using a GMM with 128 components and a diagonal covariance matrix. The GMM that maximizes the likelihood of the cluster features gives the cluster its label.

These GMMs have been already trained with speech from the ESTER training dataset [17]. This stage includes two additional steps:

- **Feature warping**: The features are warped with a 3-second window as in [24] to deal with variabilities in the input.
- **Normalizing**: The features are centered and reduced.

At the end of this stage, each segment belongs to a single speaker and it has a bandwidth and gender associated with it.

3.10 Integer Linear Programming Clustering

3.10.1 Introduction

The final stage combines 12 MFCCs with deltas, and the gender and bandwidth data of the previous stage to produce a final hypothesis annotation for audio diarization. The original LIUM SpkDiarization project [1] employed a Universal Background Model (UBM) that resulted from the fusion of the 4 gender- and bandwidth-dependent GMMs from ESTER, with mean normalization. The clustering criteria were the Cross Likelihood Ratio (CLR) and the Cross Entropy in addition to Normalized CLR (CE/NCLR). However, we decided to use the improved SpkDiarization from [16] that formulates clustering as an Integer Linear Programming (ILP) problem and uses a generic solver.

3.10.2 I-Vectors Clustering Method

The new clustering method [25] uses i-vectors to both model the clusters and measure their similarity. The i-vector algorithm reduces the dimensionality of the speaker's audio in order to keep only the input significant contribution.

This audio-only diarization solution doesn't rely on knowing the number of speakers beforehand. After a first segmentation of speakers, each cluster is mapped to an i-vector using 12 MFCCs without energy or delta information and a GMM-UBM with 1024 components. In the next step, the N i-vectors are normalized [26]. Clustering needs to minimize the number K of resulting clusters as well as the intra-cluster distribution. In

order to achieve that, clustering is formulated as an ILP problem where the goal is to minimize:

$$\sum_{k=1}^{N} x_{k,k} + \frac{1}{D} \sum_{k=1}^{N} \sum_{j=1}^{N} d(k,j) x_{k,j}, \tag{3.33}$$

subject to the constraints:

$$x_{k,j} \in \{0,1\} \qquad \forall k, \forall j, \tag{3.34}$$

$$\sum_{k=1}^{N} x_{k,j} = 1 \qquad \forall j, \tag{3.35}$$

$$d(k,j) x_{k,j} \leq \delta \qquad \forall k, \forall j, \tag{3.36}$$

$$x_{k,j} - x_{k,k} \leq 0 \qquad \forall j, \tag{3.37}$$

where $x_{k,k}$ (3.33) and $x_{k,j}$ (3.34) are binary labels (1 for true, 0 for false) indicating whether i-vector k is the center of a cluster and whether i-vector j belongs to the cluster with center k, respectively. Thus, the first term in (3.33) is equal to the number of clusters. The Mahalanobis distance between 2 i-vectors, $d(k,j)$, is used as a similarity measure. The second term in (3.33) includes normalization by a factor D, the greatest distance among all k,j pairs. It is evident that an i-vector can only be assigned to a single cluster (3.35). The i-vector j that belongs in the cluster with center k must have a distance $d(k,j)$ that is less than an experimentally set threshold δ (3.36). Equation (3.37) handles the assignment of i-vector j to cluster k, when the distance from the i-vector to the cluster's center is the shortest among all centers.

Chapter 4

Visual Speaker Diarization

In this chapter we present our visual speaker diarization approaches. At first, we introduce the Support Vector Machines classifier and the Convolutional Neural Networks and their theoretical background. Then, we describe the dlib library and how it achieves face and mouth detection. In addition, we explain the centroid algorithm for face tracking as well as the theoretical formulation of the optical flow technique. Finally, we describe in detail all the steps that we made in order to implement our two visual speaker diarization systems.

4.1 Preliminaries

4.1.1 Support Vector Machines

Support Vector Machines (SVMs) are a supervised learning model that can be used for pattern classification and non-linear regression pioneered by Vapnik [27]. The main idea of SVMs is to construct a hyperplane to separate the data points into two classes. Obviously, there are many hyperplanes that can separate the data points, but the SVM classifier manages to find the hyperplane that maximizes the margin between the separated examples.

For the linearly separable patterns in 2 dimensions [28] the separation hyperplane is just a line. In this case, a set of 2 dimensional features x_i is used for training, each of

them corresponding to an output target $d_i = +1$ for one class and $d_i = -1$ for the other, $i = 1, \ldots, N$, where N is the total number of training examples. This separating line has the following form:

$$\mathbf{w}^T\mathbf{x} + b = 0, \tag{4.1}$$

where $\mathbf{x}$ is an input vector, $\mathbf{w}$ is the weight vector, and b is a bias factor.

Let ρ be the margin of separation, which is defined as the distance between the hyperplane and its closest data point. The SVM algorithm finds the hyperplane in which the margin ρ is maximized, with $\mathbf{w}_0, b_0$ the optimum values of the weight vector and bias respectively.

The discriminant function:

$$g(\mathbf{x}) = \mathbf{w}_0^T\mathbf{x} + b_0, \tag{4.2}$$

gives an algebraic measure of distance between $\mathbf{x}$ and the hyperplane. An easy way to express it is:

$$\mathbf{x} = \mathbf{x}_p + \mathbf{r}\frac{\mathbf{w}_0}{||\mathbf{w}_0||}, \tag{4.3}$$

where $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto the optimal hyperplane and $\mathbf{r}$ is the algebraic distance. If $\mathbf{x}$ is on the positive side of the hyperplane, then $\mathbf{r} > 0$ and if it is on the negative side of the hyperplane, then $\mathbf{r} < 0$.

Because $g(\mathbf{x}_p) = 0$ by definition, we have:

$$g(\mathbf{x}) = \mathbf{w}_0^T + b_0 = \mathbf{r}||\mathbf{w}_0||, \tag{4.4}$$

or

$$\mathbf{r} = \frac{g(\mathbf{x})}{||\mathbf{w}_0||}. \tag{4.5}$$

That is, the distance $\mathbf{r}$ between the optimal hyperplane and the origin (i.e. $\mathbf{x} = \mathbf{0}$) is

computed as follows:

$$\mathbf{r} = \frac{b_0}{||\mathbf{w}_0||}. \tag{4.6}$$

When $b_0 > 0$, $\mathbf{x}$ is on the positive side of hyperplane and when $b_0 < 0$, $\mathbf{x}$ is on its negative side.

The main idea of non linear SVMs [28] is to assume a mapping of an input vector $\mathbf{x}$ of dimension m_0 into a feature space of dimension m_1 through a non linear transformation $\{\phi_j(x)\}_{j=1}^{m_1}$, defined *a priori*.

We may then define a hyperplane that acts as the decision surface:

$$\sum_{j=1}^{m_1} w_j \phi_j(\mathbf{x}) + b = 0, \tag{4.7}$$

where $\{w_j\}_{j=1}^{m_1}$ are the linear weights that connect the feature space to the output space and b is a bias factor. We simplify by writing:

$$\sum_{j=0}^{m_1} w_j \phi_j(\mathbf{x}) = 0, \tag{4.8}$$

and assuming that $\phi_0(\mathbf{x}) = 1$ for all $\mathbf{x}$, so that w_0 denotes the bias b, where $\phi_j(\mathbf{x})$ represents the input to the weight w_j via the feature space. We then define the vector:

$$\boldsymbol{\phi}(\mathbf{x}) = [\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \ldots, \phi_{m_1}(\mathbf{x})]^T, \tag{4.9}$$

where, by definition:

$$\phi_0(\mathbf{x}) = 1. \tag{4.10}$$

In that case, $\boldsymbol{\phi}(\mathbf{x})$ represents the image in the feature space due to input $\mathbf{x}$. Therefore:

$$\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) = 0, \tag{4.11}$$

for all $\mathbf{x}$. Applying the optimality condition to the Lagrangian function (the interested reader can find the full derivations in the literature [28]):

$$\mathbf{w} = \sum_{i=1}^{N} a_i d_i \boldsymbol{\phi}(\mathbf{x}_i), \tag{4.12}$$

where *feature vector* $\phi_0(\mathbf{x}_i)$ corresponds to the input pattern $\mathbf{x}_i$ in the ith data point. Substituting (4.12) in (4.11) the decision surface in the feature space is:

$$\sum_{i=1}^{N} a_i d_i \boldsymbol{\phi}^T(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}) = 0. \tag{4.13}$$

We can define the *inner-product kernel* as:

$$K(\mathbf{x}, \mathbf{x}_i) = \boldsymbol{\phi}^T(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}_i) = \sum_{j=0}^{m_i} \phi_j(\mathbf{x})\phi_j(\mathbf{x}_i), \tag{4.14}$$

for $i = 1, 2, \ldots, N$. The inner-product kernel is a *symmetric function* of its arguments, since:

$$K(\mathbf{x}, \mathbf{x}_i) = K(\mathbf{x}_i, \mathbf{x}), \tag{4.15}$$

for all i. The inner-product kernel can be used to construct the optimal hyperplane in the feature space by substituting (4.14) in (4.13):

$$\sum_{i=1}^{N} a_i d_i K(\mathbf{x}, \mathbf{x}_i) = 0. \tag{4.16}$$

Three common types of SVM kernels are:

- **Polynomial learning machine**: Power p is specified *a priori* by the user.

$$(\mathbf{x}^T\mathbf{x}_i + 1)^p. \tag{4.17}$$

- **Radial-basis function network**: The width σ^2 is shared among all kernels and

specified *a priori*.

$$\exp\left(-\frac{1}{2\sigma^2}||\mathbf{x}-\mathbf{x}_i||^2\right). \tag{4.18}$$

- **Two-layer perceptron**: Only some β_0, β_1 satisfy Mercer's theorem.

$$\tanh(\beta_0 \mathbf{x}^T \mathbf{x}_i + \beta_1), \tag{4.19}$$

In this thesis, the Radial-Basis Function network (RBF) kernel was employed in the task of visual speaker diarization.

4.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a branch of *Deep Neural Networks* (DNNs). The purpose of DNNs is to model a high degree of abstraction in the input samples using multiple-level non-linear transform architectures. A CNN is a feed forward neural network. It is a state-of-the-art solution that is generally used in image, video, and sound recognition tasks. The network consists of the following basic components:

1. Convolution layer.
2. Activation function.
3. Pooling-sub sampling.
4. Fully-connected (dense) layer.

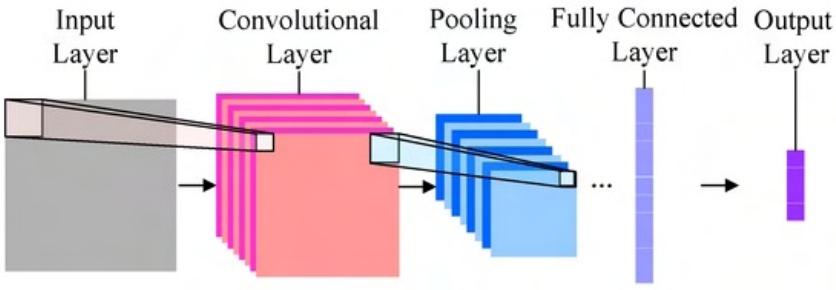


Figure 4.1: A typical Convolutional Neural Network architecture (Figure from [3]).

In CNNs, a filter is represented by a convolutional kernel that is much smaller spatially than the input. Each convolution layer consists of several filters, arranged in such a way that

their output response corresponds to the same input area (which is also called receptive field). The filters scan the input, so that the receptive fields consist of overlapping areas of the input in order to extract a smoother representation of the inserted image-sample (Figure 4.2). This method is called *weight sharing*.

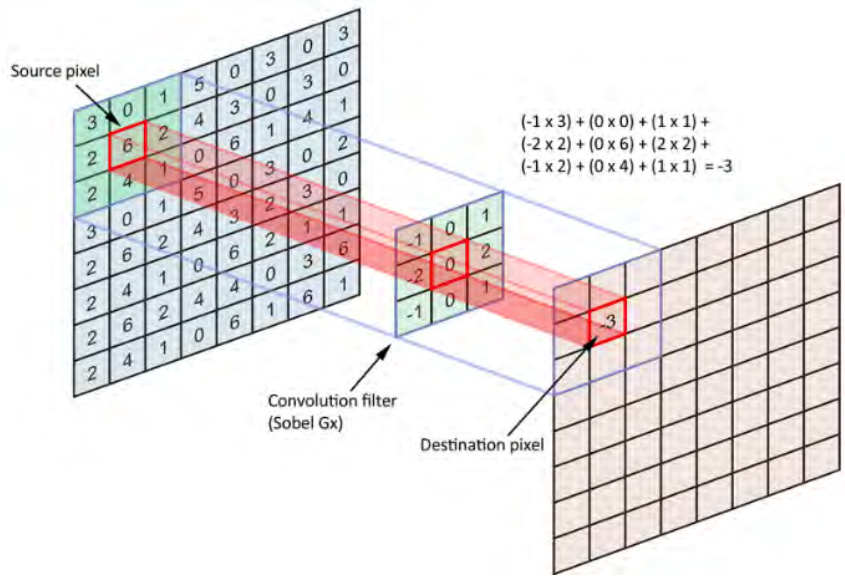


Figure 4.2: Convolution operation (Figure from [4]).

Each convolution layer is followed by an activation function. Some of the most common activation functions in neural networks are the following:

1. Binary step

$$y = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{4.20}$$

2. Sigmoid

$$y = \frac{1}{1 + e^{-x}} \tag{4.21}$$

3. Identity

$$y = x \tag{4.22}$$

4. ReLU

$$y = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{4.23}$$

The pooling layer is used to reduce the number of parameters and consequently the

computations of the network. It also decreases the possibility of overfitting. The most common functions of this layer are *Max Pooling* and *Average Pooling* (Figure 4.3).

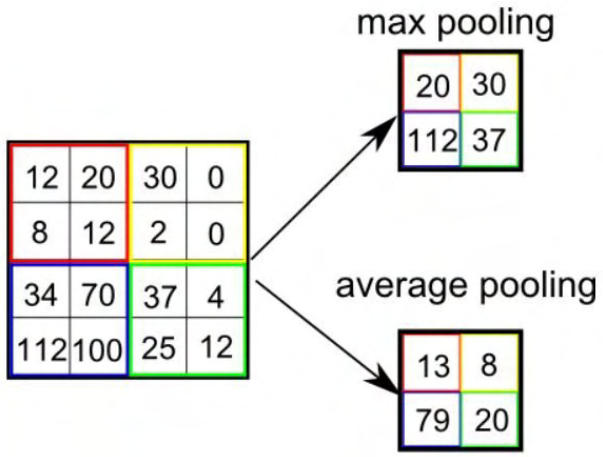


Figure 4.3: Pooling operation (Figure from [5]).

Finally, the fully connected layers are used to classify the input samples to one of the existing classes, by using the extracted features of the previous layers.

4.2 Dlib Library

4.2.1 Overview

Dlib is a C++ toolkit with machine learning algorithm implementations. It's used in industrial as well as academic applications. We've chosen to use it because of its performant face detection algorithm, its permissive licensing, and its convenient python API.

4.2.2 Dlib Face Detection

The dlib face detector finds human faces and returns bounding boxes for each face in an image. These include points on the eyes, mouth, nose, eyebrows, and along the face edges. The face detector is pre-trained with the Histogram of Oriented Gradients (HOG) algorithm [29], combined with a linear classifier, an image pyramid, and a sliding window detection scheme.

In particular, an image pyramid is a series of subdivisions of the original image into $2^{2\times l}$ subregions, with levels $l \in [1...L]$. A detector slides over each subregion and the results are concatenated into a feature vector.

In the original HOG method, the image is divided into cells. The input is colour-normalized in larger regions called blocks. The detector is initially trained with only a portion of the negative examples. Then, false positives are used to re-train the algorithm, which significantly improves performance.

Gradients are computed using a 1-D centered point derivative $[-1, 0, +1]$ and its transpose, without any smoothing. Then, each pixel within a cell votes depending on the orientation of its gradient. The vote is the gradient magnitude, and nearby votes are bilinearly interpolated. Orientation is provided via nine unsigned bins over [0°, 180°].

Better performance is achieved by normalizing in overlapping blocks, with each cell contributing to many different blocks. A block can be rectangular with rectangular cells, or circular with polar cells. In rectangular blocks, applying a Gaussian spatial window helps by weighting more the pixels near the block center. Vertical blocks (1×2) outperform horizontal blocks (2×1) when it comes to human detection. In circular blocks, a central cell is surrounded by one layer of four angular sectors. Normalization on the block level can involve the L2-norm, the L2-norm followed by clipping, or the square root of the L1-norm.

Context around the object helps in its detection, even more than increasing the image resolution. The features are classified via a linear SVM. Note that the performance relies on including information at different normalization levels.

Rather than the original HOG algorithm [29], dlib implements an improved variation found in [30], where a convolutional kernel that computes gradients is applied at all positions and scales of an image. The new algorithm includes a "root" filter that detects the entire shape and additional "part" filters that detect parts of the shape (e.g. for a human detector, each filter might detect a limb). Part detections are penalized for deviating from the ideal locations relative to the root. Part filters operate at double resolution (l levels down the pyramid) compared to the root filter. The algorithm finds the root location by independently placing each part, then places the parts at their optimal

locations relative to the fixed root.

The classifier is a latent SVM (LSVM) trained via gradient descent. Each example has a set of possible latent values. A binary label for the example can be obtained by choosing the set of latent values that maximize the classifier and thresholding the result. The model parameters are trained by minimizing the standard hinge loss $l(y) = \max(0, 1 - ty)$ where t is the target output. Note that linear SVMs are a special case of latent SVMs where there is one possible latent value for each example.

A complexity arises when considering the minimization of the loss function: in general LSVMs the hinge loss is not convex for positive examples, since it is the maximum of a convex and a concave function. The hinge loss is convex when considering only negative examples. This property is called *semi-convexity*. For an LSVM with a single latent value for each positive example, the classifier is linear and the loss is convex. Combined with semi-convexity, the objective function is convex.

In practice, a round of LSVM training is split into two phases. In the first phase, the positive example loss is minimized by selecting the highest scoring latent value for each positive example. In the second phase, the entire model is optimized by solving a convex optimization problem via gradient descent, implicitly considering latent values for negative examples. This is repeated until convergence.

The object detection model is trained with "hard" rather than "easy" examples. The hard examples are either incorrectly classified or inside the classifier's margin. The method starts with a cache of examples, then alternates between training a model and updating the cache (removing easy examples and adding hard examples).

As in [29], HOG features are computed from centered 1-D finite difference kernels, $[-1, 0, +1]$ and its transpose. The feature map includes a sparse histogram of oriented magnitudes, defined for each pixel. The orientation of the gradient magnitudes is discretized with either a contrast-sensitive or a contrast-insensitive definition. Edges are expected to have a higher magnitude. Pixel-level feature maps are aggregated into "cell"-level feature maps, where each pixel map contributes to four nearby rectangular cells via interpolation, as in [29]. Normalization helps make the gradients invariant to changes in gain. The 36-dimensional features of [29] are reduced to 13-dimensional (or 27-dimensional, including

the contrast-sensitive orientations) features, corresponding to 9 (or 9 + 18) quantized orientation features and 4 gradient energy features.

4.2.3 Dlib Pose Estimation

The pose estimator is created with the dlib implementation of an Ensemble of Regression Trees [31] and trained on the iBUG 300 face landmark dataset [32]. Given an image and a face bounding box, it matches the pose using 68 facial landmarks.

Pose estimation and alignment of facial landmarks can be achieved in milliseconds, using a cascade of regression functions. A circular problem arises when matching pixel intensities to shape estimates. We need reliable features in order to accurately find the shape, but an accurate shape is needed in order to extract reliable features. An iterative approach with two steps is used to solve this problem. At the first step, the image is transformed into a normalized coordinate system based on the current shape estimate. At the second step, new landmarks are computed in order to update the shape. The process is repeated until convergence. Another problem is that the shape is a high-dimensional vector, and optimizers can get stuck in local optima. This is solved by assuming that the global solution lies in a linear subspace that can be discovered through Principal Component Analysis (PCA) of the inputs and using regressors with solutions in that linear subspace. Input features are selected via a combination of gradient boosting the regressors with a squared loss function and a prior probability in the algorithm initialization that describes the mean face pose. A learning rate lower than 1 helps prevent overfitting.

Each regression tree node decision is based on the difference of intensity between two pixels. We need to use the same points in the current shape as the reference points in the mean shape. Instead of warping the image to match the points, it's more efficient to change the point locations to fit the image shape. This is achieved by a global similarity transform, once per cascade iteration. Each node threshold is chosen greedily from a set of randomly generated splits. Note that comparisons at the pixel level are more robust than comparing average intensities over a range, at the expense of more computational cost. The number of pairwise comparisons is quadratic on the number of pixels, but using an exponential prior over pixel distance encourages choosing closer pairs. An extension to

Figure 4.4: Face numbers change in consecutive frames.

the objective function allows it to handle missing or incomplete labels.

The original Ensemble of Regression Trees [31] used the HELEN [33] face dataset for training.

4.3 Centroid Tracking

After computing the bounding box ((x, y)-coordinates of faces in each frame) we faced the problem that the id of the same detected participant changed frame by frame (Figure 4.4). To solve that we implemented the centroid tracking algorithm.

The centroid tracking algorithm is a multi-step process. In this section we will describe every step of the algorithm:

- Step 1: Having all bounding boxes for each detected face in every single frame, we compute the corresponding centroid, i.e. the center of the bounding box. In the first frame we initialize the face ids as unique ids.
- Step 2: For every subsequent frame, we compute the Euclidean distance between every detected face centroid in this frame and all the detected face centroids in the previous frame.
- Step 3: We associate the faces that have the minimum Euclidean distance.
- Step 4: If there are more input faces than existing faces, we assign a new id for this

Figure 4.5: Face numbers remain constant with centroid tracking.

face and we compute the centroid of its bounding box. Subsequently, we repeat the algorithm from step 2.

- Step 5: At this step we consider the case in which one face disappears from the frame. Here, the face deregisters from the object tracker, if it cannot match any existing face for L subsequent frames.

After implementing centroid tracking, the face numbers remain constant in consecutive frames (Figure 4.5).

For the problem of visual speaker diarization, the useful information about speaking or not was extracted from the mouth of each face in every single frame. More specifically, the dlib pose estimator was used in order to detect the mouth of each participant in each frame. Then every mouth (ROI - region of interest) was rescaled and normalized into a 32×32 image.

4.4 Optical Flow

Optical flow [34] is an algorithm with which the movement of an object in consecutive frames can be estimated. For this problem, this technique was used in order to capture the movement of the lips frame by frame, in order to make a binary decision about speaking. The purpose of optical flow is to estimate the motion or displacement of pixels between two consecutive frames. To do that, each of the consecutive frames has to be

expressed as a polynomial expansion. In this approach quadratic polynomials were used. Furthermore the optical flow technique is based on the assumption that neighboring pixels have similar motion, therefore the whole image can be split into smaller neighborhoods, each of which can be expressed as a polynomial quadratic function:

$$f(\mathbf{x}) \sim \mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{b}^T \mathbf{x} + c,$$

where **A** is a matrix, **b** is a vector, and c is scalar. The ideal scenario is that every real signal has an exact translation to a quadratic polynomial, but in fact that is quite unrealistic. To solve this problem and keep the error small enough the global polynomial can be replaced with a local polynomial. Afterwards for each image the coefficient values were estimated by performing polynomial expansion.

Because of the assumption that in every neighborhood the displacement field is slowly varying, the information about each pixel is integrated over a neighborhood. Each pixel of the neighborhood corresponds to a weight. The main pixel has the largest weight, and the other pixels of the neighborhood have weight values that are radially decreased.

In order to improve robustness, the displacement field was parameterized according to the eight-parameter model in two dimensions.

$$\begin{aligned} d_x(x,y) &= a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy, \\ d_y(x,y) &= a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2. \end{aligned} \tag{4.24}$$

More formally:

$$\mathbf{d} = \mathbf{S}\mathbf{p}, \tag{4.25}$$

$$\mathbf{S} = \begin{pmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{pmatrix}, \tag{4.26}$$

$$\mathbf{p} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \end{pmatrix}^T. \tag{4.27}$$

This is the weighted square error:

$$\sum_i w_i \left\| \mathbf{A}_i \mathbf{S}_i \mathbf{p} - \Delta \mathbf{b}_i \right\|^2, \tag{4.28}$$

where i iterates over the neighborhood pixels. Its minimization leads to the solution:

$$\mathbf{p} = (\sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{A}_i \mathbf{S}_i)^{-1} \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \Delta \mathbf{b}_i. \tag{4.29}$$

4.5 Optical Flow and SVM Visual Diarization Description

4.5.1 Speech Detection Features

First, the mouth of each participant is detected in every frame and compared with the corresponding mouth of the same participant at the next frame. The mouth regions, as mentioned before (section 4.5), have been resized and normalized into a 32×32 image. The displacement of every pixel in the 32×32 mouth region (ROI) is computed as magnitude and orientation (optical flow features). Subsequently, the dimensionality of these features is reduced by computing the average value of each four pixels in the same 2×2 square neighborhood. The new dimensionality of optical flow features is $16 \times 16 = 256$.

4.5.2 Speech Detection SVM Training

These optical flow features were used as the training examples of an SVM classifier. In total, nearly 25000 reduced feature vectors from videos with multi-speaker conversation in Broadcast News of Greek television were used to train the algorithm.

For training, python's sklearn library was used, with an RBF as the SVM kernel function.

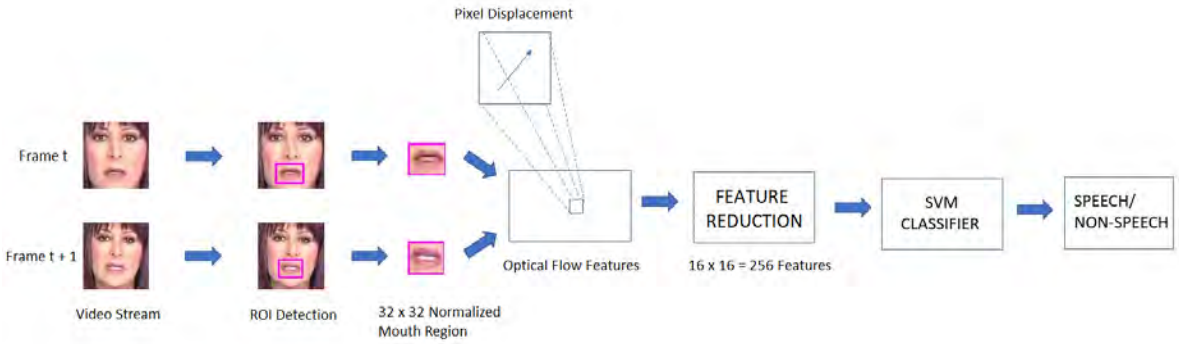


Figure 4.6: The optical flow with SVM visual diarization method.

4.5.3 Speech Detection SVM Testing

During testing, we used 5 new Broadcast videos from the dataset (section 2) with a total duration of 35 minutes. For each participant (mouth region) in every frame we used the SVM-trained model to predict whether that participant speaks or not. Afterwards, we summed the predictions for each participant separately inside a window of N frames. The participant with the maximum value in that window was considered the speaker of the window. Subsequently, we created some hypothesis annotations using videos where every speaker detected speaks. The hypothesis annotation was then used to measure the diarization error rate (DER) (section 6.3).

4.6 CNN Visual Diarization Description

4.6.1 Training and Testing Dataset

As with the optical flow and SVM method (section 4.5), we begin by detecting the faces and mouths of each participant in each frame using the dlib library. In this approach too, we consider the mouth area as the region-of-interest (ROI). As before, these ROIs have been resized and normalized to $32 \times 32 \times 3$ (RGB) images. Then, inspired by the problem of lip-reading discussed in [35], we concatenated the mouths of each speaker separately for k continuous frames in one larger concatenated image. In this thesis we examined 3

different values of k:

1. Concatenate $k = 20$ ROIs (Figure 4.7a).
2. Concatenate $k = 30$ ROIs (Figure 4.7b).
3. Concatenate $k = 49$ ROIs (Figure 4.7c).

(a) 20 Mouths **(b)** 30 Mouths **(c)** 49 Mouths

Figure 4.7: Example of concatenated images

In the first image we have concatenated 20 mouths in a 4×5 arrangement, so it has $128 \times 160 \times 3$ pixels. In the second image, the arrangement is 5×6 so its dimensions are $160 \times 192 \times 3$, and in the last one, it is 7×7, so the concatenated image has $224 \times 224 \times 3$ pixels.

In this approach too, the data was collected from the 14 clips of the Gridnews dataset (section 2.1), with 9 of them used as the training set, and the other 5 as the test set. Finally, each sample was assigned a binary label (0 for "non-speech", 1 for "speech").

4.6.2 CNN Model Training

During network training we used 3 different input sizes, corresponding to the 3 concatenated image arrangements. For each k (20, 30, 49) we used 7814, 5032, 3171 concatenated image samples for training, and 1095, 715, 445 for validation respectively. There was no overlap of mouths in consecutive concatenated images. All training and testing samples were associated with a binary label. Our goal was to classify the concatenated images into "speech" and "non speech" classes. A very important step before the model training was data normalization. Since we were using image data, the maximum and minimum pixel values were known (255 and 0, respectively). For that reason, we could normalize the training and testing data by dividing all the pixel values by 255.

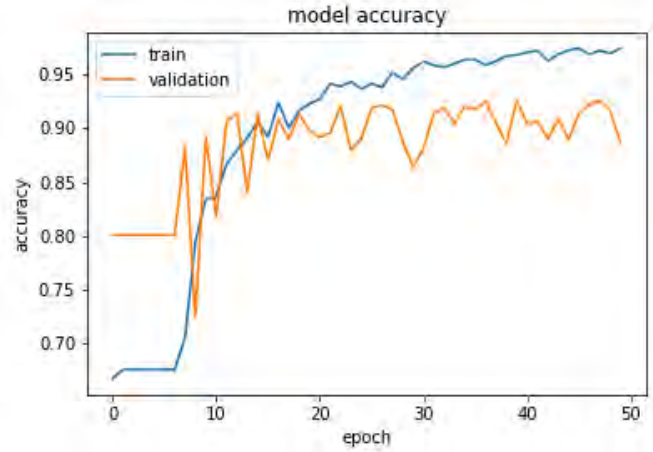
After preparing the training and testing data, we built the Convolutional Neural Network model (CNN). In order to achieve that, we employed the tensorflow open source platform for machine learning (version 1.15.0) [36]. Our model consists of 5 convolutional layers, each with 64, 128, 256, 256, and 256 filters respectively. Each filter has a size of 3×3. The convolution is performed using the "same" padding option and a stride of 1×1 and ReLU activation function . Each convolutional layer is followed by a max pooling layer with a pool size of 2×2 and dropout in 30% of the neurons. This is followed by a flatten layer and 3 dense layers with 256, 512, and 1024 neurons respectively. Each dense layer has ReLU activation. Finally, the output layer has one neuron with a sigmoid activation function (binary decision). After experimental efforts, the Adam optimizer with a learning rate of 0.001 and a batch size of 256 were selected. The network was trained on the free tier of cloud service Google Colab [37] which provides a GPU.

Figure 4.8: CNN model architecture.

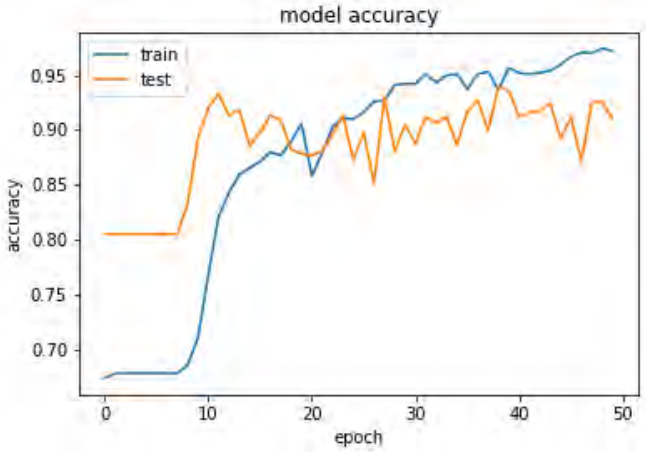
The model was trained for the 3 different arrangements of concatenated images. In Figure 4.9 we see the training and validation accuracies of the model over 50, 50, and 70 epochs respectively. In the case of $k = 49$, the model was trained for more epochs because it converged slower.

4.6.3 CNN Model Testing

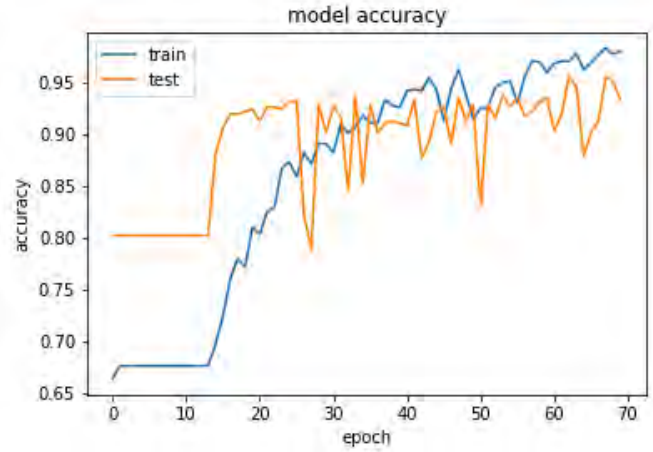
For each sample (concatenated image) in the test set (section 4.6.1) and each speaker at the same frame period, we predicted "speech" or "non speech" by using our trained model (section 4.6.2). A frame period is the number of mouth regions in a concatenated image (20, 30, or 49). The possible speaker with the highest "speech" probability result in a frame period, is considered the speaker for that period. These predictions were converted to hypothesis annotations using the pyannote python library. Finally, these annotations were compared to the reference annotations in order to measure the diarization error rate



(a) $k = 20$



(b) $k = 30$



(c) $k = 49$

Figure 4.9: Training and validation set CNN model accuracy over training epochs.

(section 6.3.2).

Chapter 5

Audio-Visual Speaker Diarization

Our audio-visual approach is based on the audio only method and using additional visual information. The proposed system was implemented in two steps. At the first step, we added information about the number of speakers by applying a face clustering algorithm whose theoretical background is explained in detail. At the second step, we added extra information about overlapping speech segments. In this chapter, we describe all the steps of both proposed methods, such as feature extraction, training and testing the SVM classifier. Finally, the results of the classifier were filtered by empirically selected thresholds and K-means clustering, whose algorithmic implementation is described thoroughly.

5.1 Preliminaries

5.1.1 K-Means Clustering

K-means clustering is one of the most popular unsupervised clustering algorithms, i.e. when the data points are unlabeled. The purpose of the algorithm is to cluster the data points into k different groups. The features with the greatest similarity between them are classified into the same cluster. The K-means algorithm has the following steps:

- Step 1: For each of the k clusters, its centroid is computed as the mean value of the data points inside the cluster.

- Step 2: Each new pattern is assigned to the cluster with the minimum Euclidean distance (L_2) from the equivalent centroid point:

$$\arg\min_{c_i \in \mathcal{C}} dist(c_i, x)^2. \tag{5.1}$$

  Let the set of examples assigned to the ith centroid be S_i.

- Step 3: The centroid of the cluster is recomputed as the mean value of all data points assigned to it:

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i. \tag{5.2}$$

The algorithm iterates between these steps and stops in the following cases:

- No data points change cluster in an entire iteration.
- The sum of the distances reaches the minimum value.
- The iteration count has reached a maximum value that has been set at the algorithm initialization.

5.2 Motivation for Audio-Visual Fusion

During our experiments with the dataset, we noticed that for each modality separately we got some very useful information that could be used to achieve better results. The audio-only solution was very good in general, but it had the disadvantage that it couldn't detect overlapping segments due to the way it was implemented. Furthermore, in many cases it wasn't able to find the correct number of speakers. In order to overcome these problems, we added visual information.

5.3 Face Clustering

At first, we included a clustering algorithm from the python dlib library in order to find the number of faces in the video. It was fortunate that all videos that we used had conversation between multiple speakers in Broadcast News. In such videos, all people that

appear speak at least once. Therefore, the number of face clusters is equal to the number of speakers. In case that a cluster exists with less than 50 appearances in the whole video, we ignored that face (it is considered incidental). Then, we use this information to augment the LIUM SpkDiarization toolkit by setting the toolkit's clustering thresholds (sections 3.5 and 3.6) automatically (via an iterative search) in order to achieve the same number of faces that was suggested by dlib clustering.

Speaker face clustering is achieved via a *deep residual convolution network* (ResNet) with 29 layers. A quick introduction to the theory of deep residual networks follows.

5.3.1 Deep Residual Networks

Deep residual networks [38] are an extension to deep neural networks that allows training networks with many more layers than before without degradation in performance. They achieve that by learning residual functions instead of arbitrary mappings. Residual networks (ResNets) have achieved remarkable results in image detection and classification tasks, even though their computational cost is comparable to regular convolutional deep neural networks. This is due to their depth, a critical part of neural network performance.

There is a *degradation* of results when adding more layers, even as techniques such as normalized initialization can combat the vanishing/exploding gradients issue of networks with many layers (20+). This degradation cannot be explained by the overfitting problem, because it also leads to higher training error.

It would be trivial to extend a shallow network with more layers that don't do anything (identity mappings). Such an experiment shows that a deep network should not have more training error than the shallow equivalent. Indeed, the problem is likely not that a solution doesn't exist, but that the gradient descent solvers cannot find it in reasonable time using reasonable computing power.

Deep residual neural networks solve this problem by using the layers to fit a residual function rather than an arbitrary mapping. We denote the desired function as $\mathcal{H}(\mathbf{x})$.

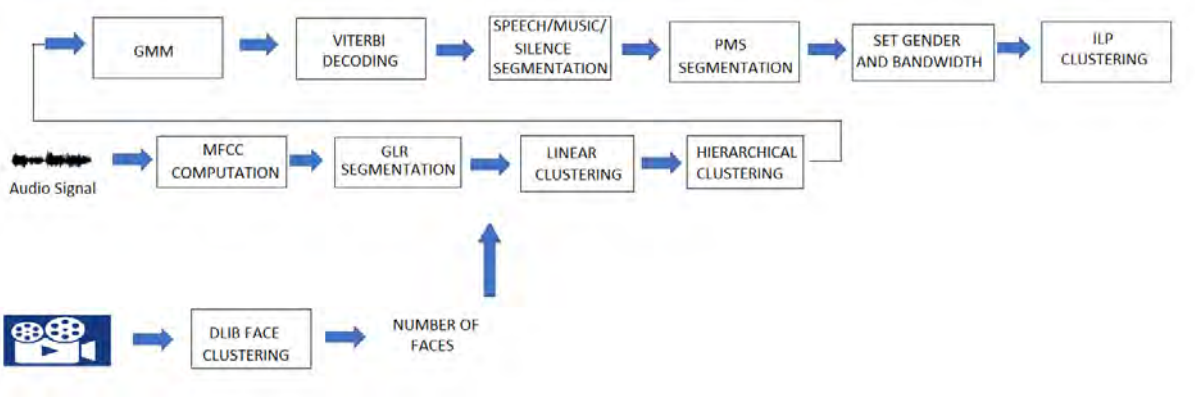


Figure 5.1: Audio-visual fusion using face clustering information.

Then the residual mapping is:

$$\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}. \tag{5.3}$$

Then the initial function can be rewritten as:

$$\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}, \tag{5.4}$$

which is easier to learn than $\mathcal{H}(\mathbf{x})$. Finally, if an identity mapping is actually optimal, the network can make the residual function zero.

In this case, $\mathcal{F}(\mathbf{x})$ is approximated by two convolutional filter weight layers with a Rectified Linear Unit (ReLU) activation function in between. An identity shortcut connects the input directly to the output by skipping layers. The $\mathcal{H}(\mathbf{x})$ primary building block is activated by an additional ReLU. The ResNet can be trained via Stochastic Gradient Descent (SGD) and backpropagation. Note that the identity connections have no weights. In other words, the ResNet has no additional parameters over a plain convolutional network and can be trained without any overhead (other than a trivial vector addition).

The ResNet achieves remarkable results compared to regular convolutional networks, taking advantage of up to 150 layers.

5.4 Speech Overlap Detection

5.4.1 Overlap detection features

For extraction of features, 11 videos from the dataset were used. From these videos all the overlapping and some non-overlapping speech segments were collected. Their duration was 2-6 seconds. In these segments, the mouth of each participant is detected in every frame and is compared with the corresponding mouth of the same participant at the next frame. Then the displacement of every pixel in the 32×32 mouth region (ROI) is computed as magnitude and orientation. For this area of displacements the magnitudes are summed. Now each face in a frame corresponds to an intensity (sum of pixel displacement magnitudes). Subsequently, for $L = 50$ consecutive frames ($L/25$ seconds), for each participant separately the intensities are summed. We normalize the sum, dividing it by the number of dlib face detections for that window. Now each face in a window of L frames corresponds to this normalized value. From the normalized scores of a window, the 2 greatest values are assumed to be possible overlapping speakers and are employed as features. Note that overlaps of more than 2 speakers don't appear in the reference annotations (e.g. a 2-second segment with 3 overlapping speakers appears as 2 1-second segments of 2 speakers each).

Let $x_i \in \mathbb{R}^2$ be the x, y-coordinates of the ith pixel in an image I. Then $d_{i,j} \in \mathbb{C}$ denotes the displacement of the ith pixel in ROI j. Assume $\mathbf{R}_f$ is a vector of ROI detections in frame f, with each ROI having 32×32 pixels. We use $IN_{f,j}$ to denote the intensity of ROI j in frame f. Note that ROI detections can be reliably indexed by j across frames (see section 4.3).

$$IN_{f,j} = \sum_{i \in \mathbf{R}_{f,j}} \left\| d_{i,j} \right\|. \tag{5.5}$$

Then the normalized intensity NIN for ROI detection j across window W can be computed

as:

$$NIN_{W,j} = \frac{\sum_{f \in W} IN_{f,j}}{N_{W,j}}. \tag{5.6}$$

The length of W is $L = 50$ frames. ROI detection j need not appear in every frame of W. Indeed, we assume the number of detections of ROI j in W is $N_{W,j}$. The 2 greatest $NIN_{W,j}$ in a window W are considered to be possible overlapping speaker features.

5.4.2 Overlap Detection SVM Training

In total, 201 feature vectors ($l = 2$, for two possible overlapping speakers) were used for training: 148 of those were labeled as overlapping and the rest were labeled as non-overlapping.

For the SVM classifier, we used the python sklearn library, as before (section 4.5.2), but we employed a linear kernel function this time.

5.4.3 Overlap Detection SVM Testing

In order to test the algorithm, we used 5 videos from the dataset. Each of those videos contains at least one overlapping speech segment. As in the corresponding training stage (section 5.4.2), the two speakers with the greatest scores are found and then these two scores (of $speaker_1$ and $speaker_2$ respectively) are used as input for the SVM-trained model in order to predict whether there is overlapping speech or not in a window of $L = 50$ frames. After that, all the overlap-predicting scores are collected and filtered via an experimentally determined threshold. The goals were:

- To keep the greatest scores.
- To keep the features where the score of the second speaker was at least equal to 85% of the first speaker score.

An extra filtering stage was applied in post-processing that employs the k-means clustering algorithm in order to separate the largest from the smallest values. We consider all segments

that exist in the k-means cluster with the largest values as segments of overlapping speech.

In the audio-visual fusion stage, we augmented the audio hypothesis annotations with these overlapping segments hypothesis and improved the DER (section 6.4).

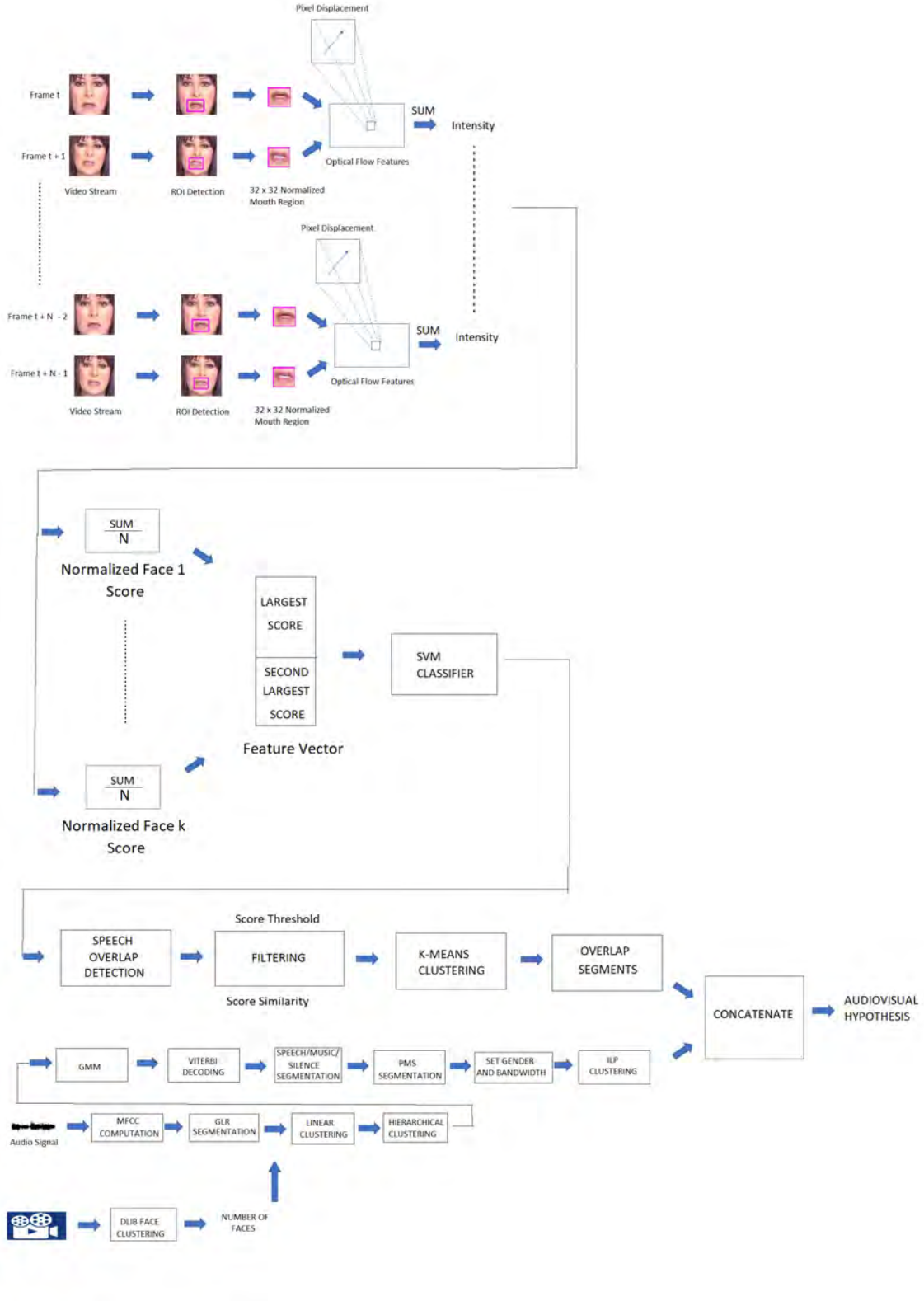


Figure 5.2: The complete audio-visual diarization method.

Chapter 6

Evaluation

6.1 Diarization Error Rate

The Broadcast News dataset of chapter 2 was used for the evaluation of the visual speaker diarization. In order to measure the error rate of our systems, we used the Diarization Error Rate (DER) [13] metric, which is very common in diarization tasks and is defined as follows:

$$\text{DER} = \frac{\text{false alarm} + \text{missed detection} + \text{confusion}}{\text{total}}, \tag{6.1}$$

where a false alarm is the duration when a person was labeled as speaker, but wasn't speaking in these frames, a missed detection is the duration when speech exists but the diarization system labeled it as non-speech, and confusion is the duration when the diarization system labeled a speaker incorrectly.

6.2 Audio-Only Evaluation

For the audio evaluation we selected 5 audio files from the dataset. In order to measure the audio-only algorithm accuracy, we computed the diarization error rate. The LIUM toolkit results were converted to hypothesis annotations and compared to the reference annotations using the python DiarizationErrorRate metric function to get the diarization

error rate. In the following table we can see the diarization error rate results for the audio-only approach:

audio 1	7.63 %
audio 2	16.13 %
audio 3	4.56 %
audio 4	12.50 %
audio 5	3.85 %
average	8.93 %

Table 6.1: Results of audio-only diarization.

The error is affected by various factors. For example, the system cannot detect the number of speakers in an audio clip. Consequently, the system may create more or less speaker clusters than the real number of speakers in the audio. As we have seen in the diarization error rate introduction (section 6.1) the error increases because of incorrect labeling of speakers (*confusion*).

The DER can also be affected by overlapping speech in the audio. Our audio-only approach is unable to detect overlapping segments. For this reason, audio clips without many overlapping segments exhibit a lower error rate than audio files with more overlapping segments. In particular, in audio files 1, 3 and 5, the system finds the correct number of speakers. However, in audio file 1, there are more overlapping segments than in the other two clips, so the algorithm achieves worse results. Note that the duration of overlapping speech in these videos is relatively small (about 1 to 2 seconds per overlap), so this has little effect on overall results.

6.3 Visual-Only Evaluation

6.3.1 Optical flow and SVM Method Evaluation

In order to evaluate the optical flow with SVM method we used the DER metric, computed via the DiarizationErrorRate python function. The hypothesis annotation that was extracted from the implemented system (section 4.5) and the reference annotation were compared. For the evaluation we used 5 videos from the dataset (section 2.1).

In order to better measure the system accuracy, we computed the DER of each video for five different windows. These windows had a size of 50, 100, 150, 200, and 250 frames (2, 4, 6, 8, and 10 seconds respectively). Then for each window we computed the average DER.

In the following table we can see the diarization error rate results:

	Window Length				
DER (%)	**50**	**100**	**150**	**200**	**250**
video 1	18.20	19.50	20.54	19.12	21.29
video 2	9.02	8.96	10.35	7.29	8.73
video 3	22.13	20.04	14.57	17.80	18.44
video 4	16.95	15.89	16.56	18.17	17.63
video 5	20.59	17.80	18.08	18.76	19.92
average	17.38	16.44	**16.02**	16.23	17.20

Table 6.2: Results of visual-only diarization.

It appears that the 150-frame window achieves better results than the other four cases (L=50, L=100, L=200, L=250). At first, we assume that the 150-frame window is better than the 50 and 100-frame windows because it examines a larger region of speech and that makes it more robust. On the other hand, the 200 and 250-frame windows are worse than the 150-frame window because they examine a quite large region of speech and that makes their estimations of speech duration less accurate when speakers change.

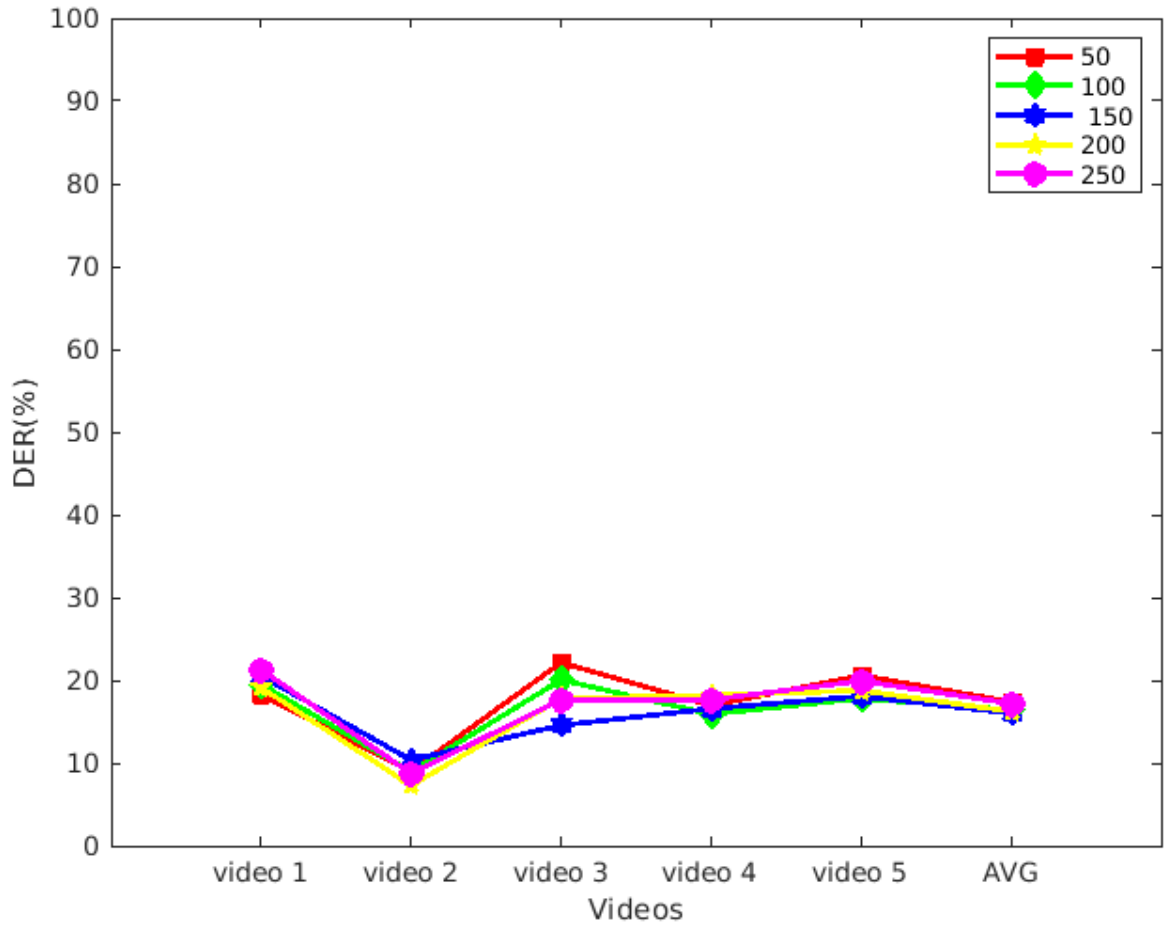


Figure 6.1: Comparison between DERs for each window size for different videos.

6.3.2 CNN Method Evaluation

As with the optical flow and SVM method, we measured the Diarization Error Rate (DER) by comparing the hypothesis annotations to the reference annotations for the same 5 videos of the dataset (section 2.1). In table 6.3 we show the DER results for the 3 different arrangements of concatenated image (k = 20, 30, 49).

	k		
DER (%)	**20**	**30**	**49**
video 1	21.74	22.35	18.76
video 2	8.10	6.67	4.55
video 3	16.53	18.92	12.70
video 4	11.11	9.75	12.00
video 5	23.93	18.37	20.73
average	16.28	15.21	**13.75**

Table 6.3: Results of visual-only diarization.

It appears that the concatenated image of 49 mouth regions had better results than the other 2 cases (Figure 6.2). We assume this is because the longer the frame prediod (k), the more lip movements are made. So the network is able to detect the "speech" concatenated images more accurately.

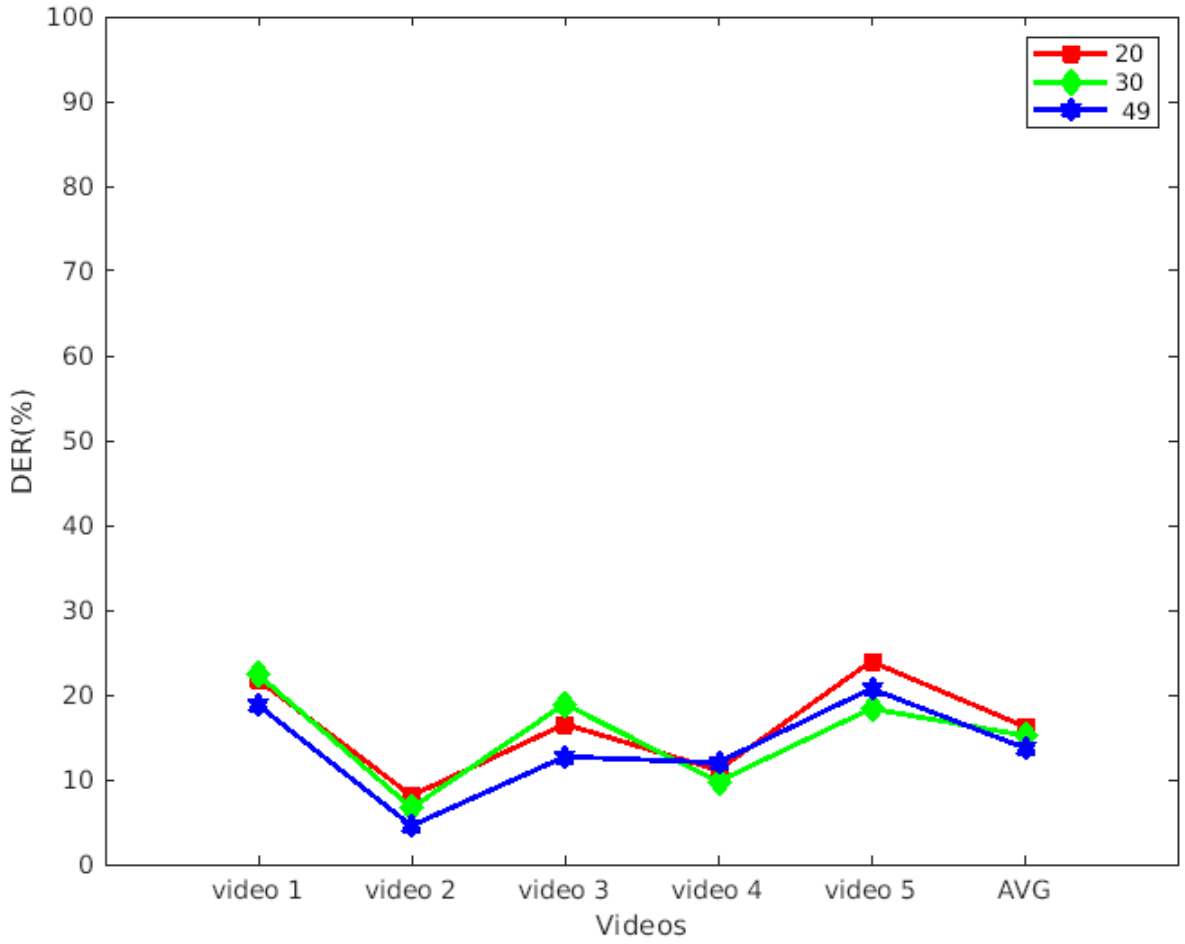


Figure 6.2: Comparison between DERs for each concatenated image size for different videos.

The comparison of this CNN model with the optical flow with SVM model is particularly interesting. In 2 of 3 cases the CNN achieved better DER results on average than the optical flow with SVM method in its most efficient 150-frame window. Only in case of k = 20 the optical flow with SVM achieved lower DER on average.

6.4 Audio-Visual Evaluation

For the audio-visual fusion approach, we measured the diarization error rate for each of the two implemented methods (section 5). For comparison purposes, we used the same five video clips as in the audio-only evaluation (section 6.2).

In the first method, we combined the information from the audio-only algorithm with additional information about the number of speakers that we discovered through face clustering. This method had the following results:

video 1	7.86 %
video 2	7.04 %
video 3	4.56 %
video 4	5.31 %
video 5	3.92 %
average	5.74 %

Table 6.4: Results of audio-visual diarization with face clustering.

We can see that the diarization error rate is overall better than in the audio-only evaluation. In some cases, the DER is almost equal to the audio-only error. That happens because the audio-only method was able to find the correct number of speakers without the extra visual information from face clustering. Note that the error never gets significantly worse. It might get trivially worse because the automatically discovered clustering thresholds of this method are different than the audio-only method thresholds, even though both indicate the same number of speakers.

On the other hand, in cases where the audio-only method didn't find the correct number of speakers and visual information was needed, a significant improvement was observed. Such cases include videos 2 and 4.

In the final, complete system we add information about overlapping speech segments to the previous method. In the following table 6.5 we see the DER results. It seems obvious from the results that this method achieved an important improvement compared to the previous audio-visual approach. As we have already mentioned, the overlap segments have small durations and we don't observe large changes between the two audio-visual fusion methods but it seems clear that the results were improved. In particular, for all 5 test files the diarization error rate decreased.

video 1	7.05 %
video 2	6.32 %
video 3	4.08 %
video 4	5.24 %
video 5	2.99 %
average	5.14 %

Table 6.5: Results of the complete audio-visual diarization method.

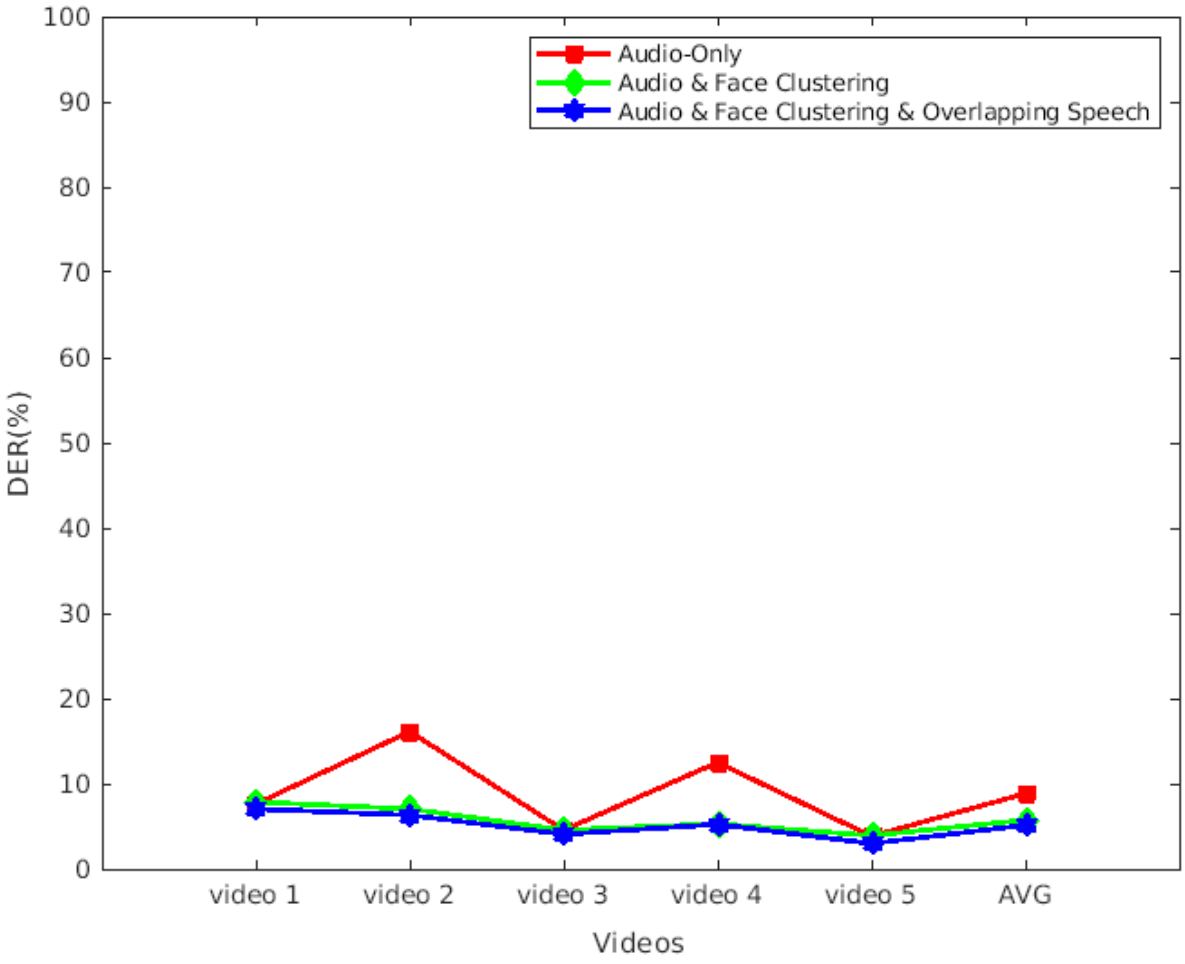


Figure 6.3: Comparison between audio and audio-visual approaches for different videos.

Chapter 7

Conclusion and Future Work

In this thesis we implemented an audio-visual speaker diarization system that is robust against multi-speaker scenarios in Broadcast News environments. At first, we investigated audio and visual information separately. As far as the audio modality is concerned, we used the LIUM speaker diarization toolkit that is based on MFCC feature extraction and GMM classification. In the visual approach we employed the dlib library for face and mouth region detection. Subsequently, we developed two visual diarization systems. The first was an optical flow technique for feature extraction with an SVM classifier and the second was a Convolutional Neural Network approach. The audio-visual fusion evolved from the audio approach in two steps. At the first step we combined the audio modality with extra information about the number of speakers that was discovered through a face clustering method. At the second step we added to the previous method overlapping speech information. This final iteration is the complete audio visual speaker diarization system. The accuracy of the system was measured by using the diarization error rate (DER) metric, and the results of each method were compared. From this comparison we concluded that the audio-visual fusion had better performance than the individual modalities.

Some ideas for further investigation are associated with neural networks. In particular, a combination of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) is an approach that has led to very good accuracy in speaker diarization in the literature.

Bibliography

[1] S. Meignier and T. Merlin, "LIUM spkdiarization: an open source toolkit for diarization," in *Proc. CMU SPUD Workshop*, 2010.

[2] D. Dimitriadis, A. Metallinou, I. Konstantinou, G. Goumas, P. Maragos, and N. Koziris, "Gridnews: A distributed automatic Greek broadcast transcription system," in *Proc. 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 1917–1920.

[3] M. Peng, C. Wang, T. Chen, and G. Liu, "NIRFaceNet: A Convolutional Neural Network for Near-Infrared Face Identification," *Information*, vol. 7, no. 4, 2016.

[4] https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/.

[5] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53/.

[6] C. Wooters and M. Huijbregts, "The ICSI RT07s speaker diarization system," in *Proc. Multimodal Technologies for Perception of Humans*, 2008, pp. 509–519.

[7] M. Bendris, D. Charlet, and G. Chollet, "Lip activity detection for talking faces classification in TV-content," *Proceedings of International Conference on Machine Vision*, pp. 187–190, 2010.

[8] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models - their training and application," *Comput. Vis. Image Underst.*, vol. 61, no. 1, pp. 38–59, 1995.

[9] G. Garau, A. Dielmann, and H. Bourlard, "Audio-visual synchronisation for speaker diarisation," in *Proc. INTERSPEECH*, 2010.

[10] F. Vallet, S. Essid, and J. Carrive, "A multimodal approach to speaker diarization on TV talk-shows," *IEEE Transactions on Multimedia*, vol. 15, no. 3, pp. 509–520, 2013.

[11] H. Vajaria, T. Islam, S. Sarkar, R. Sankar, and R. Kasturi, "Audio segmentation and speaker localization in meeting videos," in *Proc. 18th International Conference on Pattern Recognition*, 2006, pp. 1150–1153.

[12] http://avidemux.sourceforge.net/.

[13] H. Bredin, "pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems," in *Proc. INTERSPEECH*, 2017, pp. 3587–3591.

[14] https://github.com/pyannote/pyannote-metrics.

[15] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Statist.*, vol. 37, no. 6, pp. 1554–1563, 1966.

[16] M. Rouvier, G. Dupuy, P. Gay, E. el Khoury, T. Merlin, and S. Meignier, "An open-source state-of-the-art toolbox for broadcast news diarization," in *Proc. INTERSPEECH*, 2013.

[17] S. Galliano, G. Gravier, and L. Chaubard, "The ESTER 2 evaluation campaign for the rich transcription of French radio broadcasts," in *Proc. INTERSPEECH*, 2009, pp. 2583–2586.

[18] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[19] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[20] A. G. Adam, S. S. Kajarekar, and H. Hermansky, "A new speaker change detection method for two-speaker segmentation," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, 2002, pp. 3908–3911.

[21] S. S. Chen and P. S. Gopalakrishnan, "Speaker, environment and channel change

detection and clustering via the Bayesian information criterion," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 127–132.

[22] C. Barras, X. Zhu, S. Meignier, and J. L. Gauvain, "Multistage speaker diarization of broadcast news," *Trans. Audio, Speech and Lang. Proc.*, vol. 14, no. 5, pp. 1505–1512, 2006.

[23] S. Tranter and D. A. Reynolds, "Speaker diarisation for broadcast news," in *Proc. Odyssey Speaker and Language Recognition Workshop*, 2004, pp. 337–344.

[24] J. Pelecanos and S. Sridharan, "Feature warping for robust speaker verification," in *Proc. IEEE Odyssey: The Speaker and Language Recognition Workshop*, 2001, pp. 213–218.

[25] M. Rouvier and S. Meignier, "A global optimization framework for speaker diarization," in *Proc. Odyssey Workshop*, 2012.

[26] P.-M. Bousquet, M. Driss, and J.-F. Bonastre, "Intersession compensation and scoring methods in the i-vectors space for speaker recognition." in *Proc. INTERSPEECH*, 2011, pp. 485–488.

[27] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[28] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[29] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 886–893.

[30] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, 2010.

[31] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867–1874.

[32] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: Database and results," *Image and Vision Computing (IMAVIS)*, 2016.

[33] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, "Interactive facial feature localization," in *Proc. of the 12th European Conference on Computer Vision - Volume Part III*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 679–692.

[34] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Proc. Image Analysis*, J. Bigun and T. Gustavsson, Eds. Berlin, Heidelberg: Springer, 2003, pp. 363–370.

[35] A. G. Amit, J. N. Jnoyola, and S. B. Sameepb, "Lip reading using CNN and LSTM," Stanford University, Tech. Rep., 2016.

[36] https://www.tensorflow.org/.

[37] https://colab.research.google.com/notebooks/intro.ipynb/.

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