ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ARTIFICIAL INTELLIGENCE**

**MASTER THESIS**

**in**

**NATURAL LANGUAGE PROCESSING**

# DISRUPTIVE SITUATIONS DETECTION ON PUBLIC TRANSPORTS THROUGH SPEECH EMOTION RECOGNITION

**CANDIDATE:**
**Eleonora Mancini**

**SUPERVISOR:**
**Prof. Paolo Torroni**

**CO-SUPERVISORS:**
**PhD. Andrea Galassi**
**Dr. Federico Ruggeri**
**Dr. Josep Escrig Escrig**
**Dr. Ivan Huerta Casado**

Academic year 2020/21

Session 3rd

# Abstract

In this thesis, we describe a study on the application of Machine Learning and Deep Learning methods for Voice Activity Detection (VAD) and Speech Emotion Recognition (SER). The study is in the context of a European project whose objective is to detect disruptive situations in public transports. To this end, we developed an architecture, implemented a prototype and ran validation tests on a variety of options.

The architecture consists of several modules. The denoising module was realized through the use of a filter and the VAD module through an open-source toolkit, while the SER system was entirely developed in this thesis. For SER architecture we adopted the use of two audio features (MFCC and RMS) and two kind of classifiers, namely CNN and SVM, to detect emotions indicative of disruptive situations such as fighting or shouting. We aggregated several models through ensemble learning. The ensemble was evaluated on several datasets and showed encouraging experimental results, even compared to the baselines of the state-of the-art.

The code is available at: `https://github.com/helemanc/ambient-intelligence`

# Contents

# List of Figures

# List of Tables

# Introduction

In recent years, thanks to the strong development of Artificial Intelligence techniques, we have witnessed the spread of numerous technologies that accompany us in everyday life. In many cases, the introduction of these technologies in the social context has led to a marked improvement in the quality of life. Just think of the voice assistants installed on our smartphones, which allow us to use our mobile, while driving, without having to take our hands off the steering wheel; or, we could take the example of speech-to-text systems which, in their noblest function, can be used by people suffering from degenerative diseases, such as Parkinson's, in order to type messages with the sole use of their own voice. These are just some of the examples of technologies that, thanks to the union of Artificial Intelligence and voice analysis, can bring extraordinary results in our lives, and this is exactly the context that this thesis fits in.

The work carried out for this thesis is part of the European research project H2020 - 5GMED [1, 2]. 5GMED aims to demonstrate advanced Cooperative Connected and Automated Mobility (CCAM) and Future Railway Mobile Communications System services (FRMCS) along the "Figueres – Perpignan" cross-border corridor between Spain and France. It is a very broad project with a lot of partners and many objectives. Among the various objectives of 5GMED, the project of this thesis pursues the development of an AI application for the recognition of disruptive situations on public transports. Disruptive situations include, for example, people fighting or screaming.
The development of such a technology would be of great importance, since it could really have a strong social impact. However, these kinds of systems have never been introduced in real scenarios, because they still present many scientific and technological challenges.

The main challenge is to build a model that is:

- speaker-independent
- gender-independent
- cross-linguistic
- robust to environmental noise

Having an architecture that respects all the aforementioned conditions requires an in-depth scientific and technological study.
In particular, the problems listed above are widespread in all Speech Emotion Recognition systems, and not only among applications that must work in our reference context. Public transport is just an example of context in which all these issues arise at the same time, because transports are very noisy environments frequented by people of different gender and ethnicity.
Achieving a good performance for SER is always a challenge because of the variability in signals of speech emotion and speaker-dependent features. A lot of research works aimed to extract typical features for speech emotions, such as INTERSPEECH 2009 Emotion Challenge and the INTERSPEECH 2013 computational paralinguistics challenge [3]. Although SER models can achieve a good performance no matter which type of input is used, they still treat these features as general representations of emotions and ignoring personalized differences.
In order to address this problem, we performed a meticulous work on the datasets that we had available. The datasets used for this thesis project are RAVDESS, TESS, SAVEE and CREMA-D.
In particular, we carefully balanced the number of speakers, of female and male voices and of files per emotion when splitting the datasets into training, validation and test sets. Unfortunately, as these datasets are all in English (mainly British or North American accent), we were unable to construct a cross-linguistic model.
A good part of the work was spent on the data, not only to carry out this division, but also to make them homogeneous and as realistic as possible due to the lack of real data. In order to make the data realistic, a particular Data Augmentation strategy has been developed.

Thanks to a careful study of the recent literature, we identified which features we could use to build the input of our models (MFCCs and RMS) and which classifiers (CNNs and SVMs).
Several experiments have been performed to analyze the change in performances linked to a combination of the following aspects:

- dataset
- features
- binary classifier

To do so, we built 96 different models; among them, we selected the 19 models that led to the best performances.
The best models were integrated through ensemble learning and several aggregation strategies were implemented.
Even though the application is not yet ready to be used in the real scenario (i.e. public transports), we have obtained very satisfactory results.

The core of our work is the construction of the SER system. However, the overall architecture is also composed of a denoising and a Voice Activity Detection module; the latter has been realized through an open-source toolkit.

The main contributions introduced by this work are:

- the integration of the three modules to realize a working prototype (i.e. a self-contained Python application) for disruptive situations detection
- the modeling of emotions in two classes: *disruptive*, *non-disruptive*
- the construction of a speaker-independent and gender-independent SER model

The thesis is organized as follows:

- *Chapter 1*: presents the context in which this thesis project is inserted, that is the European project H2020-5GMED.
- *Chapter 2*: provides the necessary background. The focus is on audio features, architectures for speech recognition and literature review for VAD and SER.
- *Chapter 3*: outlines the proposed architecture, showing its building blocks and how they are linked one to the other.
- *Chapter 4*: describes the open-source toolkit that we used to address VAD task and its validation.
- *Chapter 5*: discusses the development of the speech emotion recognition system, from the analysis of available datasets to the construction of the ensemble.
- *Chapter 6*: offers an analysis of the critical issues and possible margins for improvement of the application and discusses the challenges encountered during the implementation of the SER system and possible future developments.
- *Chapter 7*: concludes and suggests future extensions of this work.
- *Appendices 1-5*: provide further details on experimental setups and results and on the developed application code.

# Chapter 1

# 5GMED: Disruptive Situations Detection in Public Transports

The work carried out for this thesis project is part of the European research project H2020 - 5GMED [1, 2].
The 5GMED project is a very broad project that aims to demonstrate advanced Cooperative Connected and Automated Mobility (CCAM) and Future Railway Mobile Communications System services (FRMCS) along the "Figueres – Perpignan" cross-border corridor between Spain and France. The value proposition for this project is to deploy a single infrastructure to be used by multi-stakeholder.
The project is enabled by a multi-stakeholder compute and network infrastructure deployed by MNOs[1], neutral hosts, and road and rail operators, based on 5G and offering support for Artificial Intelligence functions.
The consortium coordinated by Cellnex Telecom includes 21 partners from 7 countries, including the i2CAT Foundation of Barcelona, which is the research center where this thesis project was carried out.
The main topics of 5GMED are:

- Cross-operator service orchestration
- Innovations in multi-connectivity supporting high-speed vehicles and trains
- Self-sustainable 5G access network infrastructure that can be deployed when power and backhauling resources are scarce

[1]Mobile Network Operators

- Enhancements to speed up roaming transitions across MNOs and neutral hosts
- Novel high-speed access network architectures for railways
- The ability to support AI-enabled functions executing at the edge of the network

Among the various project objectives listed above, our project pursues the latter, namely: "The ability to support AI-enabled functions executing at the edge of the network".
In particular, among the functions managed by the i2CAT Foundation there is the creation of a system capable of identifying disruptive situations in public transports.
The great successes of Artificial Intelligence and recent discoveries in this field have allowed the realization of the function mentioned above.
Artificial Intelligence will be used to detect situations inside the train that can be considered disruptive or risky, such as people shouting, fighting or talking in the silence car. Also emergency words like "help" may be recognised by a Keyword Spotting (KWS) algorithm. For this purpose, CCTV cameras will send live stream video and audio to the AI Module located at the Edge Server. One or more Natural Language Processing (NLP) architectures and techniques will classify the audio between normal and disruptive, triggering an alarm when needed and allowing the Train Control Centre to remotely manouver the camera in order to verify the alarm. Additionally, video stream may be analysed by a Computer Vision model to confirm the detection of unexpected behaviours that can indicate a disruptive situation.
In Figure 1.1 we can find the sequence diagram of a disruptive situation detected by the AI Module and trasmitted to the Control Center to be analyzed.

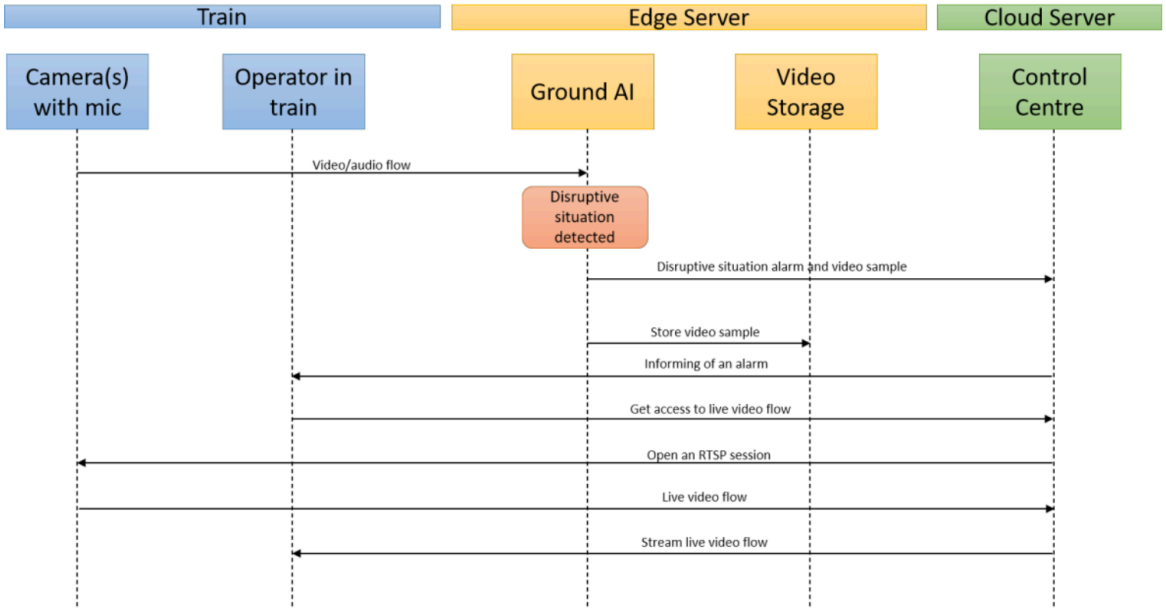


Figure 1.1: Sequence diagram of a disruptive situation detected by the AI Module and transmitted to the Control Centre

Let's describe in details the workflow showed in the figure above:

- The camera or cameras with microphones, located in the passengers car, send the video/audio flow to the AI module (Ground AI) running on the Edge Server

If a disruptive situation is detected on the video/audio flow:

- An alarm is sent to the Control Center
- A sample of the video in which the disruptive situation was occurred is stored in the Video Storage module
- The Control Centre informs to the Operator in train that a Disruptive situation has been detected by the IA
- The Control Centre gets access to the cameras
- A RTSP[2] session is opened with the cameras
- The cameras might be operated from the Control Centre to inspect the interior of the passengers car and verify if a disruptive situation really exists
- The Control Center can forward the Stream live video to the Operator in train

[2]Real Time Streaming Protocol

The architecture developed in this thesis project is part of the ”Ground AI” module in Figure 1.1. In particular, we have been involved in creating an architecture capable of analyzing the flow of audio arriving from the microphones placed on the train and understanding whether continuous portions of audio can refer to disruptive situations or not; the focus was placed on the analysis of the emotions within the portions of audio which contain speech. Please refer to *Section 3* for further details about the whole architecture.

# Chapter 2

# Background

## 2.1 Audio Features

### 2.1.1 MFCC

Mel Frequency Cepstral Coefficients (MFCC) based features are very common and are used in a lot of SER models, such as [4] and [5].

MFCC is the classic, efficient and successful approach used for speech-related tasks such as gender identification by voice, speech recognition, speech emotion recognition and many more. MFCC coefficients are successful because they are derived from human speech patterns. [6]

The MFCC attempts to mimic the human ear, where the audio frequency is determined as an Asymmetric Spectrum. The main steps of MFCC feature extraction are pre-emphasis, frame-blocking, fast-Fourier transform (FFT), Mel frequency warping, and discrete cosine transform (DCT) [7].

Figure 2.1 explains the MFCC extraction process from an audio signal.

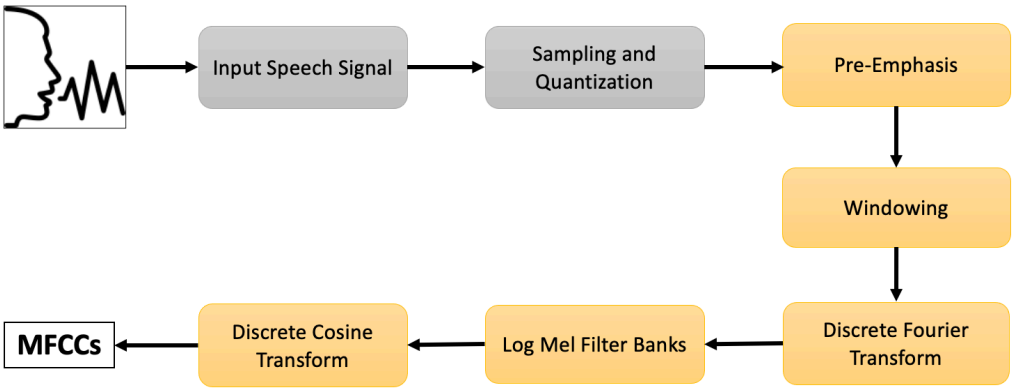


Figure 2.1: Block diagram for MFCC

The detailed description of various steps involved in the MFCC feature extraction is explained below. Most of the details reported below have been taken from [8] and [9].

- ***Sampling and Quantization***: even if this step is not strictly related to the MFCCs computation pipeline, it is mandatory to perform it before starting the process. As we know the audio wave is a continuous signal while the computer is a digital machine that cannot directly represent the continuous signals, so we have to convert these continuous signals into a discrete finite (i.e. digital) set of information; this analog-to-digital conversion has two steps: sampling and quantization [9].
  A signal is *sampled* by measuring its amplitude at a particular time; the sampling rate is the number of samples taken per second. To accurately measure a wave, we must have at least two samples in each cycle: one measuring the positive part of the wave and one measuring the negative part. More than two samples per cycle increases the amplitude accuracy, but less than two samples will cause the frequency of the wave to be completely missed [9]. Thus, the maximum frequency wave that can be measured is one whose frequency is half the sample rate (since every cycle needs two samples) [9]. This maximum frequency for a given sampling rate is called the Nyquist frequency. Although using higher sampling rates produces higher ASR accuracy, we cannot combine different sampling rates for training and testing ASR systems. Thus if we are testing on a telephone corpus like Switchboard (8 KHz sampling), we must downsample our training corpus to 8 KHz. Similarly, if we are training on multiple corpora and one of them includes telephone speech, we downsample all the wideband corpora to 8 KHz [9]. In this thesis project we

have decided to standardize all data to 16 KHz.

Amplitude measurements are stored as integers, either 8 bit (values from -128–127) or 16 bit (values from -32768–32767). This process of representing real-valued numbers as integers is called *quantization*; all values that are closer together than the minimum granularity (the quantum size) are represented identically.

*Sampling and Quantization* correspond to what we will call later as *Data Loading*. For further details please refer to *Section 5.3.4*.

- ***Pre-emphasis***: is a filtering process that is used to process a signal before performing feature extraction on it. The spectrum of speech has higher energy at low frequencies compared to high frequencies. Through pre-emphasis, energies are increased to high-frequency levels, thereby balancing the level of energies in the spectrum. From a more technical point of view the purpose of pre-emphasis is to balance the spectrum of voiced sounds that have a steep roll-off in the high-frequency region. For voiced sounds, the glottal source has an approximately $-12$ dB/octave slope [10]. However, when the acoustic energy radiates from the lips, this causes a roughly +6 dB/octave boost to the spectrum. As a result, a speech signal when recorded with a microphone from a distance has approximately a $-6$ dB/octave slope downward compared to the true spectrum of the vocal tract. Therefore, pre-emphasis removes some of the glottal effects from the vocal tract parameters. The most commonly used pre-emphasis filter is given by the following transfer function:

$$H(z) = 1 - bz^{-1} \tag{2.1}$$

  where the value of $b$ controls the slope of the filter and is usually between 0.4 and 1.0 [10]. Pre-emphasis is considered a noise reduction tool because it reduces the power of the noise without affecting the rest of the signal. [11]

- ***Framing***: or *Frame blocking and Windowing*. It consists of splitting the signal into several frames. The speech signal is a slowly time-varying or quasi-stationary signal. For stable acoustic characteristics, speech needs to be examined over a sufficiently short period of time. Therefore, speech analysis must always be carried out on short segments across which the speech signal is assumed to be stationary. These short segments are called *windows*.

Inside this small window, we can roughly think of the signal as stationary [1], that is, its statistical properties are constant within this region [9]. We extract this roughly stationary portion of speech by using a window which is non-zero inside a region and zero elsewhere, running this window across the speech signal and multiplying it by the input waveform to produce a windowed waveform [9]. The speech extracted from each window is called a frame. The windowing is characterized by three parameters: the *window size* or *frame size* of the window (its width in milliseconds), the *frame stride*, (also called shift or offset) between successive windows, and the *shape of the window* [9]. To extract the signal we multiply the value of the signal at time *n*, *s[n]*, by the value of the window at time *n*, *w[n]*:

$$y[n] = w[n]s[n] \tag{2.2}$$

Short-term spectral measurements are typically carried out over 20ms windows, and advanced every 10ms [8]. Advancing the time window every 10 ms enables the temporal characteristics of individual speech sounds to be tracked, and the 20 ms analysis window is usually sufficient to provide good spectral resolution of these sounds, and at the same time short enough to resolve significant temporal characteristics. The purpose of the overlapping analysis (i.e. windowing) is that each speech sound of the input sequence would be approximately centered. On each frame, a window is applied to taper the signal towards the frame boundaries. Generally, Hanning or Hamming windows are used [10], instead of a rectangular one. This is done to enhance the harmonics, smooth the edges, and to reduce the edge effect while taking the DFT on the signal.

- ***DFT spectrum***: each windowed frame is converted into magnitude spectrum by applying DFT:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nk}{N}} \quad 0 \leq k \leq N-1 \tag{2.3}$$

where N is the number of points used to compute the DFT. This operation has the objective of understanding how much energy the signal contains at different fre-

[1] By contrast, in general, speech is a non-stationary signal, meaning that its statistical properties are not constant over time.

quency bands [9]. The equation above exploits the FFT (Fast Fourier Transform). This implementation of the DFT is very efficient, but only works for values of N that are powers of 2 [9].

- ***Mel Spectrum***: Mel spectrum is computed by passing the Fourier transformed signal through a set of band-pass filters known as Mel-filter bank. A *mel*[2] is a unit of measure based on the human ears perceived frequency; a *mel* is commonly described as unit of pitch [9]. It does not correspond linearly to the physical frequency of the tone, as the human auditory system apparently does not perceive pitch linearly.

  Mel scaling is performed as shown in Equation 2.4

$$mel(f) = \begin{cases} f & \text{if } f \leq 1kHz \\ C \cdot log(1 + \frac{f}{f_0}) & \text{if } f > 1kHz \end{cases} \tag{2.4}$$

  where $f$ denotes the physical frequency in Hz, and *mel(f)* denotes the perceived frequency. Below the frequency $f_0$ the Mel scale changes approximately linearly with frequency, whereas above $f_0$ it changes logarithmically; in particular the Mel scale is approximately a linear frequency spacing below 1 kHz and a logarithmic spacing above 1 kHz [8]. This is the result of measurements. The difference between most formulas is the choice of the corner frequency $f_0$, which is usually chosen somewhere between 600 *Hz* and 1000 *Hz*; the most common value for $f_0$ is 700. The constant $C$ in 2.4 is normally chosen such that 1000 *Hz* correspond to 1000 *mel*:

$$C = \frac{1000}{log(1 + \frac{1000}{f_0})} \tag{2.5}$$

  - if we take the natural logarithm and $f_0 = 700$ we get $C = 1127$ from Eq. 2.5
  - if, instead, we use the logarithm with base 10, we obtain the other well-known constant $C = 2595$

  Since the logarithm is applied in 2.4, they are also called *Log-Mel Filter Banks*.

  Filter banks can be implemented in both time domain and frequency domain. For MFCC computation, filter banks are generally implemented in frequency domain.

[2]The name *mel* comes from the word *melody* to indicate that the scale is based on pitch comparisons.

The center frequencies of the filters are normally evenly spaced on the frequency axis. However, in order to mimic the human ears perception, the warped axis, according to the nonlinear function given in Eq. 2.4, is implemented. The most commonly used filter shaper is triangular, and in some cases the Hanning filter can be found [10]. The triangular filter banks with Mel frequency warping is given in Fig. 2.2. The Mel spectrum of the magnitude spectrum X(k) is computed by multiplying the magnitude spectrum by each of the of the triangular Mel weighting filters:

$$s(m) = \sum_{k=0}^{N-1} [|X(k)|^2 H_m(k)] \quad 0 \leq m \leq M-1 \tag{2.6}$$

where M is total number of triangular Mel weighting filters [12] [13]. $H_m(k)$ is the weight given to the $k^{th}$ energy spectrum bin contributing to the $m^{th}$ output band and is expressed as:

$$H_m(k) = \begin{cases} 0 & \text{if } k < f(m-1) \\ \frac{2(k-f(m-1))}{f(m)-f(m-1)} & \text{if } f(m-1) \leq k \leq f(m) \\ \frac{2(f(m+1)-k)}{f(m+1)-f(m)} & \text{if } f(m) < k \leq f(m+1) \\ 0 & \text{if } k > f(m+1) \end{cases} \tag{2.7}$$

with m ranging from 0 to M − 1.

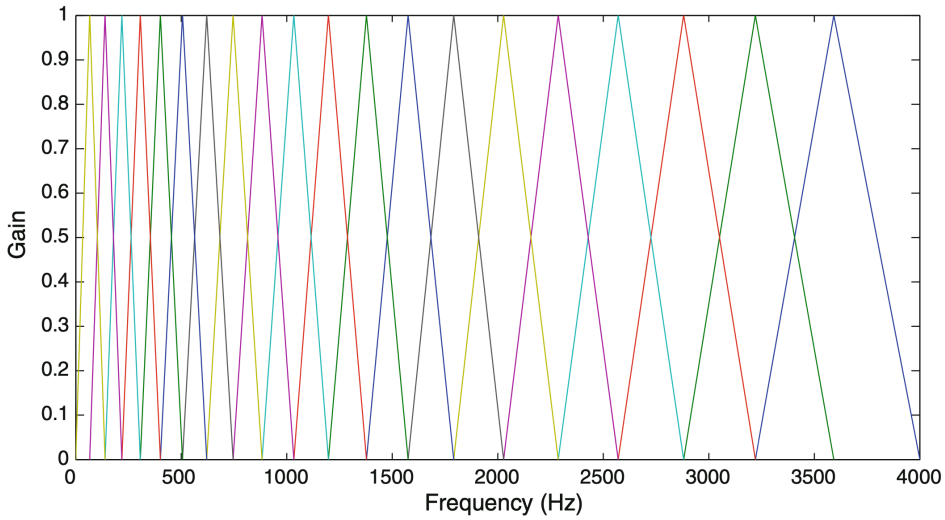


Figure 2.2: Mel-filter bank

If we do not continue with the DCT calculation, we can stop here and use Log-Mel Filter Banks as audio features; we will find them in our Voice Activity Detection system (*Section 4*).

- ***DCT***: since the vocal tract is smooth, the energy levels in adjacent bands tend to be correlated. In the discrete cosine transform (DCT) stage, the mel spectrum coefficient is converted into the time domain; in particular the DCT is applied to the transformed Mel frequency coefficients and produces a set of cepstral[3] coefficients (MFCCs). Prior to computing DCT, the Mel spectrum is usually represented on a log scale. This results in a signal in the cepstral domain with a quefrequency peak corresponding to the pitch of the signal and a number of formants representing low quefrequency peaks. Since most of the signal information is represented by the first few MFCC coefficients, the system can be made robust by extracting only those coefficients, ignoring or truncating higher order DCT components[4] [10]. Finally, MFCC is calculated as:

$$c(n) = \sum_{m=0}^{M-1} log_{10}(s(m))cos\Big(\frac{\pi n(m-0.5)}{M}\Big) \quad \text{n=0,1,2,...,C-1} \tag{2.8}$$

where *c(n)* are the cepstral coefficients, and $C$ is the number of MFCCs.

[3]In Fourier analysis, the *cepstrum* is the result of computing the inverse Fourier transform (IFT) of the logarithm of the estimated signal spectrum.

[4]That's why we will limit our exploration to a range of MFCCs values between 13 and 26.

Traditional MFCC systems use only 8–13 cepstral coefficients. The zero-th coefficient is often excluded since it represents the average log-energy of the input signal, which only carries little speaker-specific information [8]. Precisely for this reason we will exclude the zero-th coefficient in the Feature Extraction phase of our Speech Emotion Recognition architecture.

MFCC has numerous advantages like simple calculation, better ability of distinction and high robustness to noise. We have used MFCC features to represent audio samples in the Speech Emotion Recognition System.
All the steps for the calculation of the MFCCs are wrapped inside the `librosa.feature.mfcc` [14], the function that we used in the Feature Extraction phase for the Speech Emotion Recognition system.

### 2.1.2 RMS

RMS stands for Root-Mean-Square-Energy per frame. RMS is the measure of the loudness of an audio signal; it is found by calculating the square root of the sum of the mean squares of the amplitudes of the sound samples [15].
RMS formula is given in Equation 2.9:

$$x_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + ... + x_n^2}{n}} \tag{2.9}$$

We used this feature in our Speech Emotion Recognition system. The RMS feature has been computed through Librosa's `librosa.feature.rms` [16] function.

## 2.2 Architectures for Speech Recognition

In this section we illustrate the main architectures used in our Voice Activity Detection and Speech Emotion Recognition models. In particular, the key element of the Voice Activty Detection system is a Convolutional Neural Network, while the main element of the Speech Emotion Recognition system is an ensemble of Convolutional Neural Networks and SVM classifiers.

### 2.2.1 Convolutional Neural Networks

Convolutional Neural Network is one of the most popular deep learning methods manifested in areas of face recognition, handwriting recognition, and many other processing and recognition problems. Convolutional Neural Networks were initially implemented for computer vision (CV) tasks. In recent years, Convolutional Neural Networks (CNNs) have also been widely applied in the field of natural language processing (NLP), due to their good generation, and discrimination capability.
The CNN can be regarded as a variant of the standard neural network; instead of using fully connected hidden layers, the CNN introduces a special network structure, which consists of alternating so-called convolution and pooling layers [17].

In general, the mathematical operation that characterizes convolutional neural networks is, as it is possible to guess from the name itself, convolution. Convolution is an important operation in signal and image processing. Convolution operates on two signals (in 1D) or two images (in 2D): we can think of one as the "input" signal (or image), and the other (called the kernel) as a "filter" on the input signal (or image), producing an output signal (or image). So convolution takes two signals (or images) as input and produces a third as output. Convolution is an incredibly important concept in many areas of math and engineering.

In particular, a convolutional layer is composed by kernels that are convolved with the input. A convolutional kernel divides the input signal into smaller parts, namely the *receptive field* of the kernel. Furthermore, the convolution operation is performed by multiplying the kernel with the corresponding parts of the input that are into the

receptive field [18]. After this general introduction to Convolutional Neural Networks, we analyze more in details their functioning and their use in Automatic Speech Recognition. In carrying out the discussion of the next paragraphs, we will follow the outline and contents presented in [17].

### Organization of the Input Data to the CNN

When using the CNN for pattern recognition, the input data must be organized as a set of feature maps that are fed into the CNN. This notion comes from image processing, where it is intuitive to organize the input data as a two-dimensional (2-D) array, consisting of the pixel values at the $x$ and $y$ coordinates (horizontal and vertical). For color images, the RGB values (red, green, blue) can be considered as three different 2-D feature maps.

CNNs pass a small window over the input image at both training and testing time, so that the weights of the network looking through this window can learn from different features of the input data, regardless of their absolute position within the input.

*Weight sharing*, or more specifically in our case, *full weight sharing*, refers to the decision to use the same weights at each positioning of the window.

CNNs are also often referred to as *local* because the individual units computed at a particular position of the window depend on features of the local region of the image that the window is currently looking at.

When we consider 2D CNNs, the input image in the context of ASR can be roughly thought of as a spectrogram, with static, delta, and delta-delta features (i.e., first and second temporal derivatives) taking on the roles of red, green, and blue, or organized as acoustic features (such as MFCC features) in a two-dimensional feature map where one axis represents the frequency domain and the other the time domain. If we consider the case of the spectogram, the features can be arranged as three 2-D feature maps distributed along both frequency (using the frequency band index) and time (using the frame number within each context window). In this case, a two-dimensional convolution is performed (see below) to normalize both frequency and time variations simultaneously.

When we consider 1D-CNNs, in the context of ASR, the input signal can be considered as a vector of shape (`time_steps, number_of_features`), where each time step corresponds to a single window with a large amount of audio context (9-15 frames). In this

case, the features are organized as a number of 1-dimensional feature maps. As a result, a one-dimensional convolution is performed along the time axis.

Once the input feature maps are formed, the convolution layer and the pooling layer apply their respective operations to generate the activations of the units in these layers in sequence, as shown in Figure 2.3.

Similar to the units in the input layer, the units in the convolutional layer and the pooling layer can also be organized into maps. In CNN terminology, a pair of convolutional layers and pooling layers in sequence, as in Figure 2.3, is usually referred to as a CNN "layer". Thus, a deep CNN consists of two or more of these pairs in sequence. To avoid confusion, we will refer to convolutional layers and pooling layers as convolutional layers and pooling *plies*, respectively.

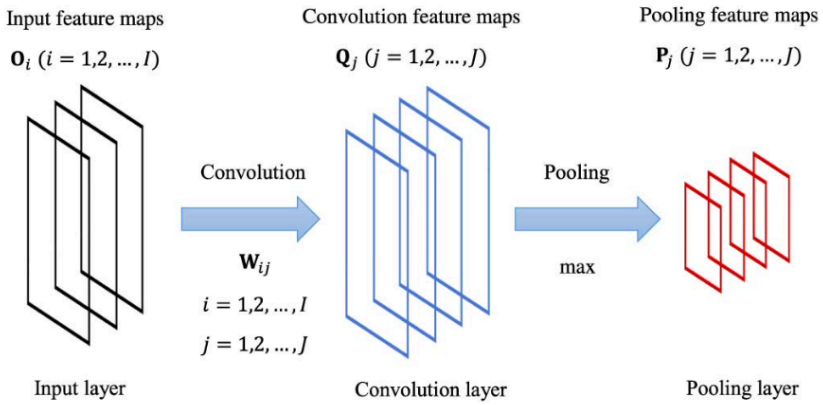


Figure 2.3: An illustration of one CNN layer ply and a pooling ply in succession, where mapping from either the input layer or a pooling ply to a convolution ply is based on eq. (2.11) and mapping from a convolution ply to a pooling ply is based on eq. (2.12) [17]

### Convolution Ply

As shown in Figure 2.3, every input feature map (assume $I$ total number), $O_{\mathrm{i}(i=1\ldots I)}$ is connected to many feature maps (assume $J$ is the total number), $Q_{\mathrm{j}}(j = 1\ldots J)$, in the convolution ply based on a number of local weight matrices ( $I \times J$ in total), $w_{\mathrm{i,j}}(i = 1\ldots I; j = 1\ldots J)$. The mapping can be represented as the well-known convolution operation in signal processing. Assuming input feature maps are all one dimensional,

each unit of one feature map in the convolution ply can be computed as:

$$q_{j,m} = \sigma(\sum_{i=1}^{I}\sum_{n=1}^{F} o_{i,n+m-1} w_{i,j,n} + w_{0,j}), \quad \text{j} = 1,...,\text{J} \tag{2.10}$$

where $o_{\text{i,m}}$ is the $m$-th unit of the $i$-th input feature map $O_{\text{i}}$, $q_{\text{i,m}}$ is the $m$-th unit of the $j$-th input feature map $Q_{\text{j}}$ in the convolution ply, $w_{\text{i,j,m}}$ is the $n$-th element of the weight vector $w_{\text{i,j}}$ which connects the $i$-th input feature map to the $j$-th feature map of the convolution ply.

$F$ is called the *filter size*, which determines the number of frequency bands (or time steps) in each input feature map that each unit in the convolution ply receives as input. Equation 2.10 can be written in a more concise matrix form using the convolution operator $\circledast$:

$$Q_j = \sigma(\sum_{i=1}^{I} O_i \circledast w_{i,j}), \quad \text{j} = 1,...,\text{J} \tag{2.11}$$

where $O_{\text{i}}$ represents the $i$-th input feature map and $w_{\text{i,j}}$ represents each local weight matrix, flipped to adhere to the convolution's operation definition.

Both $O_{\text{i}}$ and $w_{\text{i,j}}$ are vectors if one dimensional feature maps are used, and are matrices if 2-dimensional feature maps are used (where 2-D convolution is applied to the above equation), as described in the previous section.

Figure 2.4 shows a representation of a 1-D convolutional kernel, considering as input a vector of MFCCs features with shape (`time_steps, number_of_features`):

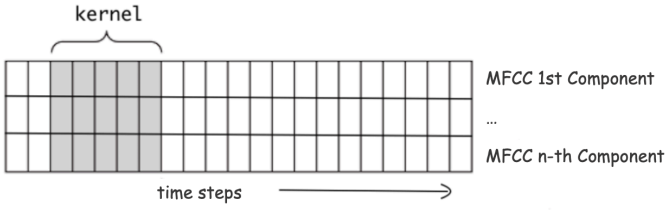


Figure 2.4: Visual representation of 1D convolutional kernel with MFCCs feature vector as input

The convolution operation itself produces lower-dimensional data; each dimension decreases by the filter size minus one, but we can pad the input with dummy values (both dummy time frames and dummy frequency bands) to preserve the size of the feature maps. Consequently, there could in principle be as many locations in the feature map of the convolution ply as there are in the input.

### Pooling Ply

As shown in Figure 2.3, a pooling operation is applied to the convolutional ply to produce the corresponding pooling ply. The pooling ply is also divided into feature maps and has the same number of feature maps as the number of feature maps in its convolution ply, but each map is smaller.

The purpose of the pooling ply is to reduce the resolution of the feature maps. That is, the entities in this ply serve as generalizations about the features in the lower convolutional ply. Since these generalizations are in turn spatially localized in frequency, they are also invariant to small variations in location. This reduction is achieved by applying a pooling function to multiple units in a local region whose size is determined by a parameter called *pooling size*.

This is usually a simple function such as *maximization* or *averaging*. The pooling function is applied to each convolutional feature map independently. When the *max-pooling* function is used, the pooling ply is defined as:

$$p_{i,m} = \max_{n=1}^{G} q_{i,(m-1)\times s+n} \tag{2.12}$$

where $G$ is the pooling size, and $s$, the *shift size*, determines the overlap of adjacent pooling windows.

Similarly, if the average function is used, the output is calculated as:

$$p_{i,m} = r \sum_{n=1}^{G} q_{i,(m-1)\times s+n} \tag{2.13}$$

where $r$ is a scaling factor that can be learned.

### 2.2.2 Support Vector Machine

SVM is usually used as a binary classifier, but it can also be used as a multi-class classifier. It is a highly effective tool for computing machine learning algorithms and is widely used in all kinds of pattern recognition problems. In particular, it is known to outperform other classifiers in cases where limited training data is available.

SVM is essentially based on using kernel features to nonlinearly map the original features to a high-dimensional space, where the data is then well classified using a linear manifold. SVM has been used extensively for classification (especially image classification). It has been successful in applications such as thyroid disease detection [19], mammogram classification in breast cancer detection [20], and even in poverty determination [21]. SVM has been shown to be particularly powerful in emotion detection compared to linear discriminant classifiers and nearest neighbor classifiers. It has been used as a classifier for sound-based emotion recognition [22] and showed high accuracy [23]. Moreover, Deep Support Vector Machines were tested for speech emotion recognition in [24] and achieved better results than previous studies. A decision tree SVM model with Fisher feature selection for speech emotion recognition was also implemented and achieved 98.29% accuracy [25]. Finally, an SVM classifier was also used in [23], which performed very well on a speech emotion recognition task. Therefore, in this work, we decided to use SVM as a classifier for our speech emotion recognition architecture, since it can be considered as one of the most successful classifiers.

#### How SVM classifiers work

An SVM is one example of a classifier that estimates decision surfaces directly, rather than modeling a probability distribution across the training data [26].
SVMs have demonstrated good performance on several classic pattern recognition problems [26]. Figure 2.5 shows a typical two-class problem in which the examples are perfectly separable using a linear decision region. H1 and H2 define two hyperplanes. The distance separating these hyperplanes is called the *margin*. The closest in-class and out-of-class examples lying on these two hyperplanes are called the *support vectors*.

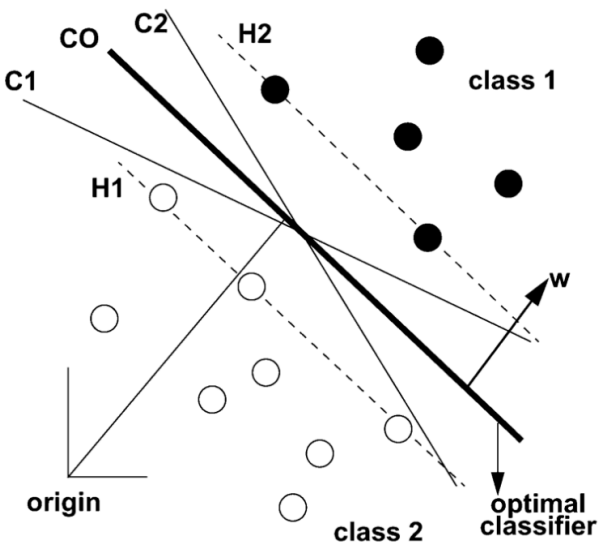


Figure 2.5: C0 is the optimal hyperplane because it maximizes the margin, i.e. the distance between the hyperplanes H1 and H2. Maximizing the margin indirectly results in better generalization [26].

An SVM classifier is defined in terms of the training examples.

Real-world classification problems typically involve data that can only be separated using a nonlinear decision surface. Optimization on the input data in this case involves the use of a kernel-base transformation:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \tag{2.14}$$

Kernels allow a dot product to be computed in a higher dimensional space without explicitly mapping the data into these spaces [26].

A kernel-based decision function has the form:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b \tag{2.15}$$

where $\alpha_\mathrm{i}$ are the weights for the training examples, as determined by the learning algorithm, $y_\mathrm{i}$ is a value in the range $-1, 1$ and $K$ is the kernel function.

The most widely used kernel functions are: [27]

- the simple *linear* kernel

$$K_L(x_i, x_j) = x_i^T \cdot x_j \tag{2.16}$$

- the *radial basis function kernel (RBF kernel)*

$$K_{RBF}(x_i, x_j) = exp(-\gamma||x_i - x_j||^2) \tag{2.17}$$

  where $\gamma$ is proportional to the inverse of the variance of the Gaussian function and whose associated feature space is of infinite dimensionality

- the *polynomial* kernel

$$K_P(x_i, x_j) = (1 + x_i^T \cdot x_j)^p \tag{2.18}$$

  whose associated feature space are polynomials up to grade $p$

- the *sigmoid* kernel

$$K_{SIG}(x_i, x_j) = tanh(ax_i^T \cdot x_j + b) \tag{2.19}$$

## 2.3 Related Work

In the following paragraphs we will analyze the latest works that have been published in the areas of Voice Activity Detection and Speech Emotion Recognition. The greatest criticality has been identified among the research works in Speech Emotion Recognition and concerns the way in which the data is processed. It emerged that in most cases the authors divide the datasets randomly; the main problem with this type of approach is that files referring to the same actor are both in the training and in the validation set, leading to very optimistic evaluations.

### 2.3.1 Voice Activity Detection

As will be mentioned in *Section 4*, current VAD techniques typically use a classifier to make speech/non-speech predictions about each audio frame independently, together with a temporal smoothing scheme to reduce noise in the classifier's output.

One way to build a VAD system involves two GMMs, one trained on speech frames and the other on non-speech frames, to predict the per-frame probability of speech, followed by an ergodic hidden Markov model (HMM) that penalizes transitions between speech and non-speech states to give temporal continuity to the prediction [28].

With the increasing availability of computing power, and the recent prominence of Deep Learning, the use of Artificial Neural Networks (ANNs) for VAD has become more popular than GMMs.
Su et al. had success using a 2D Convolutional Neural Network (CNN) based on spectral features to detect possible speech segments for pitch classification [29]. The CNN was used to exploit the highly shift-invariant structure of the harmonics in the spectrogram. Although this was not a VAD task, it was shown that CNNs can be used to find precises regions of an audio signal that contain harmonics, similar to how periodicity or harmonicity can be used to find candidate regions for detecting speech [30, 31] [32].

The use of Fully Connected Neural Networks (DNNs) has also been explored for VAD.

Bai et al. investigated variants of an architecture with 3-hidden layers [33]. The small variant had 128 nodes per hidden layer, while the large variant had 512 nodes per hidden layer. Similar to Sohn et al. [34], Bai et al. also overlaid an HMM on top of the DNN to capture temporal context and smooth out fluctuations in decisions. Although the large variant was more informative due to greater non-linearity and trainable parameters, the performance improvement over the small variant was negligible [32]. However, the large variant was over 64 times slower in making decisions.

The underlying models and algorithms discussed so far have made a largely incorrect assumption about a noisy speech signal: the existence of speech within a discrete segment is independent of other segments [32]. The use of HMMs in [33, 34] attempts to improve this unrealistic assumption by making decisions not only the output of the underlying model, but also on the decision of the previous frame. This has the effect of penalizing transitions between the speech and non-speech states, resulting in much smoother decision sequences.
In [35], Hughes et al. point out that while HMMs help to provide contextual information when making decisions, they still have some fundamental shortcomings [32]:

1. Audio frame labels are not conditionally independent. HMMs assume that the label for a given frame is conditionally independent of all other labels given the labels of its neighboring frames. This is a simplification that is not true in practice.

2. For VAD tasks, the hidden state space of the HMM is often finite and binary: speech and non-speech. This is a fundamental limitation on the information that a given state can convey. More states with richer information could improve decision making when conditioning on these states. However, developing more states, is largely a heuristic approach, and it is not clear how these can be reconciled with an underlying model that simply outputs the probability that a frame contains speech.

3. HMMs and underlying models cannot be trained together. First, the underlying model must be optimized, and then the HMM can be trained

4. Hughes et al. uses recurrent neural networks (RNNs) for VAD, which solve many of the problems associated with the HMM-based architectures. RNNs are advanta-

geous because they can be trained to jointly minimize frame-level errors while also learning a continuous hidden state space that captures useful temporal context going back arbitrarily far. The architecture in [35] consists of multiple recurrent cells and feed-forward components, and also used an unconventional quadratic activation scheme. It was shown to significantly outperform a GMM-State Machine model, while requiring only tenth of the parameters [35].

More recent works than the above mentioned show the great advantage of using Convolutional Neural Networks for VAD.

Shuo et al. proposed an alternative architecture that does not suffer from saturation problems by modeling temporal variations through a stateless dilated convolution neural network (CNN). The proposed architecture differs from conventional CNNs in three ways: it uses dilated causal convolution, gated activations and residual connections [36].

Jia et al. proposed an end-to-end neural network for Voice Activity Detection (VAD): MarbleNet. MarbleNet is a deep residual network consisting of blocks of 1D time-channel separable convolution, batch-normalization, ReLU and dropout layers. According to the authors, when compared to a state-of-the-art VAD model, MarbleNet is able to achieve similar performance with about 1/10 the parameter cost [37].

In conclusion, following are some of the more recent approaches which have been developed as open-source toolkits:

- *Pyannote Audio* [38], having a Neural Network as underlying discrimination model.
- *Ina Speech Segmenter* [39] [40], as a CNN-based audio segmentation toolkit.
- *WebrtcVAD [41]*, with a Gaussian Mixture Model (GMM) approach.

As specified in the Introduction, the creation of the architecture for VAD was not the subject of this thesis, therefore we opted for the use of one of the open-source toolkits listed above: Ina Speech Segmenter.
We will analyze in *Section 4* the reasons that led us to choose this model and its main features.

### 2.3.2 Speech Emotion Recognition

A substantial amount of research has been carried out in the field of Speech Emotion Recognition (SER). In this section, we present a brief review of the work done on emotion detection from audio. Many of the current research methodologies are based on two different classification approaches. The first is the use of classical classifers such as SVM and shallow artificial neural networks (ANN) and the second is the use of classifers based on deep learning approaches such as Convolutional Neural Networks (CNN) and deep neural networks (DNN).

The research frequently presents innovative techniques in this era in order to increase the performance of the SER and reduce the complexity of the overall system. Usually, a SER system has two core parts, which have challenges that need to be solved for an efficient emotion recognition system that include (i) the selection of the robust, discriminative, and salient features of the speech signals and (ii) the classification methods in order to accurately classify them accordingly [42].
In recent years, many researchers have focused on the use of Convolutional Neural Networks for speech recognition and, in particular, for SER. In [17] the use and impact of Convolutional Networks for speech recognition has been analyzed in details. In [43] an extensive comparison of various approaches to speech based emotion recognition systems has been carried out. In particular, the researchers compared the use of different features such as Log-Mel Spectrogram, Mel-Frequency Cepstral Coefficients (MFCCs), pitch and energy in combination with the use of different neural architectures such as Long Short Term Memory (LSTM), Convolutional Neural Networks (CNNs), Hidden Markov Models (HMMs) and Deep Neural Networks (DNNs). The dataset used in [43] is RAVDESS and the models were built to address both binary and multiclass classification. Moreover, in [43] it has been pointed out that the choice of audio features impacts the results much more than the model complexity. Similarly, some researchers in [44] used several techniques with conventional vocal feature extraction (MFCC, STFT), along with deep-learning approaches such as 2D-CNN, and also context-level analysis, by providing the textual data, and combining different approaches for improved emotion-level classification. Here, the datasets used are RAVDESS and TESS and the classifier were built to address multiclass classification tasks. Furthermore, also in [45] researchers

used approaches based on CNN architectures and MFCC features; also in this work, the authors focused on the audio recordings available in RAVDESS dataset. The model has been trained to classify eight different emotions and the authors claim to have beaten the 2020 state-of-the-art score on RAVDESS, achieving an average accuracy of 91%. Researchers of [46] proposed a very interesting approach based on the use of different features (MFCC, ZCR, HNR, TEO) and two different architectures: an autoencoder to perform additional feature extraction and SVM as a classifier. Later, in [11] it has been presented a classification of emotions using SVM and MFCCs; the authors achieved 97% accuracy with TESS and 86% with IEMOCAP datasets, respectively. In [47] the authors have used Linear Predictive Coding (LPC) apart from the MFCC feature extraction method, before feature merging. Besides, they have performed a novel application of Manta Ray optimization in speech emotion recognition tasks that resulted in a state-of-the-art result in this field. Performances of this model have been evaluated using SAVEE and EMO-DB, two publicly available datasets. The method proposed in [47] outperformed all the existing methods in speech emotion analysis and, stating on what has been reported in [47], it resulted in a decent result in these two datasets with a classification accuracy of 97.49% and 97.68% respectively.

During the current year many other research works have been published. In [48] the authors developed a 1D-CNN based approach using MFCC features; the approach has been evaluated on RAVDESS dataset and it was reported an average accuracy which is better than as compared to the existing SER models, with reduced computation cost. Similarly, in [49] a 1D-CNN classifier with MFCC feature has been developed; also in this case the performances were tested on RAVDESS dataset and the results achieved in terms of accuracy are quite similar to [48]. In [50] the researchers built two parallel CNNs to extract spatial features and a transformer encoder network to extract temporal features, classifying emotions from one of 8 classes; they also performed a very interesting data augmentation on RAVDESS dataset. Another interesting approach has been presented in [51], in which an attention-based CNN-BLSTM model with the end-to-end (E2E) learning method was proposed. The proposed method seems not to have achieved as good results as those of the other works in terms of accuracy, but in reality it is a truly avant-garde work since it is one of the only works that has dealt with a cross-language architecture. In [23], researchers studied the impact of autoencoder based compact

representation on emotion detection from audio, achieving very good performances on TESS dataset. Finally, in [52] the authors performed many experiments on RAVDESS, TESS and SAVEE dataset using CNN and MLP classifiers.

All the works mentioned above have been a source of inspiration for the construction of the baselines of the Speech Emotion Recognition architectures of this thesis project. Unfortunately, it has not been possible in many cases to faithfully reproduce these research works since they often lack either implementation details or details regarding the training parameters. Furthermore, all the works, except for [43], presented solutions to solve the problem of multi-class classification, while we, taking a cue from [43], had the need to make a strong assumption and split the emotions in two categories: *disurptive* and *non-disruptive*. Please refer to *Section 5.3.1* for further details.
A very important thing to point out is that the main problem with these approaches is that in most cases [11, 23, 44, 45, 47, 48, 50, 52] the division between training, validation and test set is done randomly. This type of division does not allow to achieve the objectives mentioned above, since files of the same actor are both in the training set and in the test set, leading to very optimistic evaluations.
We will address this problem by implementing a different splitting of data, so as to take into account the different actors and genders in the datasets. This will allow us to achieve the objectives set initially: to have a model gender-independent and speaker-independent. Please refer to *Section 5.3* for further details.

# Chapter 3

# Architecture

In this section we are going to present the architecture that has been built to address the problem of "Disruptive Situations Detection in Public Transports".
The system consists of three main blocks designed to perform three different functions:

- Denoising
- Voice Activity Detection
- Speech Emotion Recognition

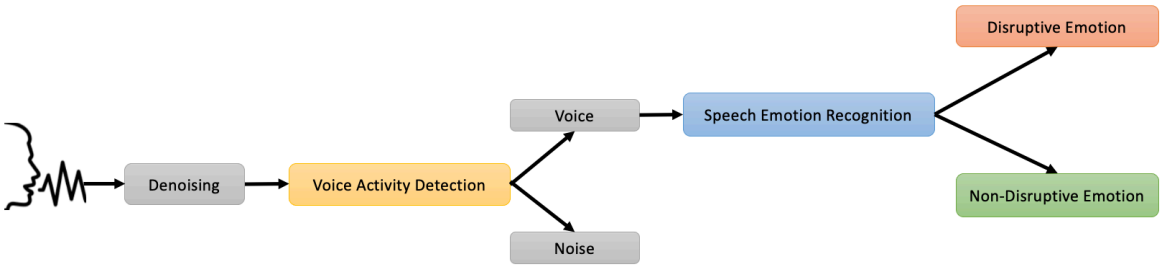


Figure 3.1: Proposed Architecture

In Figure 3.1 we can understand the general functioning of architecture. In the next paragraphs we are going to describe in detail what are the main steps involved within the system.

### Denoising

The first step is the denosing of the input signal. The application of a denoising filter is fundamental considering the reference context (i.e. public transports). This function is achieved by applying a Wiener filter to the samples of the audio signal. In signal processing, the Wiener filter is a filter used to produce an estimate of a desired or target random process by linear time-invariant (LTI) filtering of an observed noisy process, assuming known stationary signal and noise spectra, and additive noise. In particular, Wiener filter minimizes the Mean Square Error between the estimated random process and the desired process [53].
For example, the known signal might consist of an unknown signal of interest that has been corrupted by additive noise. The Wiener filter can be used to filter out the noise from the corrupted signal to provide an estimate of the underlying signal of interest.
Summarizing, Wiener filters are characterized by the following:

- *Assumption*: signal and (additive) noise are stationary linear stochastic processes with known spectral characteristics or known auto-correlation and cross-correlation
- *Requirement*: the filter must be physically realizable/causal (this requirement can be dropped, resulting in a non-causal solution)
- *Performance criterion*: minimum mean-square error (MMSE) [53]

The method provided by SciPy [54] was used for the application of this filter.

### Voice Activity Detection

The system is designed to perform continuous and real-time "Disruptive Situations Detection"; the incoming audio is divided into frames lasting 5 seconds. Each audio fragment is initially analyzed through the Voice Activity Detection system. In particular, in order to be analyzed, the audio must be written in a temporary *.wav* file, which is used by the network to perform the prediction. In order to realize the functionality in question, a pre-existing model called *inaSpeechSegmenter* was used and validated. The authors of this model have made public a Python API in order to be able to use the architecture in inference [39, 40]; the API was used within this project in order to invoke the model and perform segmentation of the incoming audio. The model returns a sequence containing

information about the content of the audio track; in particular, the output allows you to understand which fragments of the track contain speech and the duration of each of them. Anything that is not recognized as a voice is classified as noise. [1] Our contribution consisted in carrying out the analysis of this output, performing the classification of the audio file as "Voice" or "Noise"; this analysis was implemented through a rule-based system based on the fact that if the audio track contains at least one segment classified as "Voice", then the audio is passed to the Speech Emotion Recognition system, otherwise we passed to the analysis of the next snippet of incoming audio. Further details about the architecture just described, and the validation of it, will be provided in *Section 4.*

**Speech Emotion Recognition**

As mentioned in the previous section, if the incoming audio contains voice, the audio file is analyzed through the Speech Emotion Recognition (SER) system. The Speech Emotion Recognition system was completely realized within the project of this thesis. This architecture consists of an ensemble of classifiers (SVMs and Convolutional Neural Networks) capable of distinguishing whether the incoming audio contains a "disruptive" or "non-disruptive" type of emotion. Before being analyzed by the SER system, the audio must be sampled, resampled at the sample rate of the audio files used to train the model and the features described later in *Section 5.3.6* must be extracted in order to be used as model input. Further details regarding the Speech Emotion Recognition system and the division of emotions into two classes can be found in *Section 5.*

In the following sections we will describe in detail each of the two blocks (VAD and SER) and, at the end, we will provide a detailed description of the functioning of the overall application, with the addition of the parts of the code which are necessary for its functioning.

[1] The system also allows to distinguish between "music", "voice" and "noise" and to distinguish the gender of the speaker. These features have not been exploited within this project.

# Chapter 4

# Voice Activity Detection

Voice activity detection (VAD), also known as speech activity detection or speech detection, is the detection of the presence or absence of human speech, used in speech processing [55]. The main uses of VAD are in speech coding and speech recognition.

VAD is an important enabling technology for a variety of speech-based applications. Therefore, various VAD algorithms have been developed that provide varying features and compromises between latency, sensitivity, accuracy and computational cost. Some VAD algorithms also provide further analysis, for example whether the speech contains male or female voice [39]. Voice activity detection is usually independent of language [55].

The typical design of a VAD algorithm is as follows:[55]

1. There may first be a noise reduction stage, e.g. via spectral subtraction. In our case, as mentioned in *Section 3*, we decided to apply a Wiener filter on the input signal to address the noise reduction task.

2. Then some features or quantities are calculated from a section of the input signal.

3. A classification rule (or a classifier[1]) is applied to classify the section of audio as *speech* or *non-speech*; often this classification rule finds when a value exceeds a certain threshold.

Figure 4.1 summarizes the steps listed above.

[1]Here we mean a classifier realized through a Machine Learning algorithm or a Deep Neural Network.

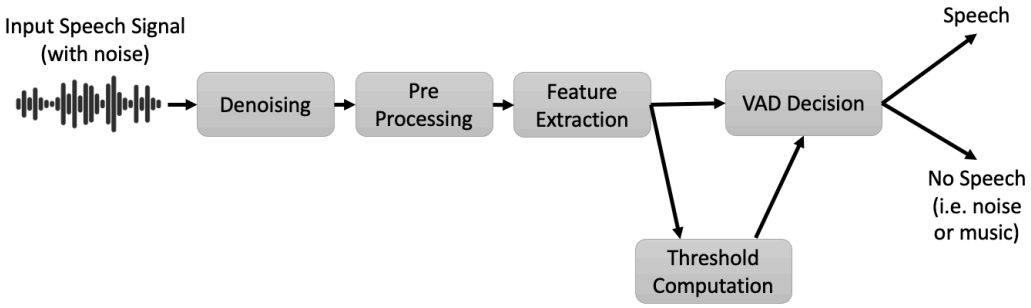


Figure 4.1: A typical VAD scheme (prototype) for speech processing application

Independently from the choice of the VAD algorithm, a compromise must be made between having voice detected as noise, or noise detected as voice (i.e. between false positive and false negative) [55]; in particular, this kind of compromise is what we needed to take into account during the validation phase of our VAD system. A VAD operating in a mobile phone must be able to detect speech in the presence of a range of very diverse types of acoustic background noise. In these difficult detection conditions, as well as in public transports context, it is often preferable that a VAD should fail-safe, indicating speech detected when the decision is in doubt, to lower the chance of losing speech segments.
The biggest difficulty in the detection of speech in this environment is the very low signal-to-noise ratios (SNRs) that are encountered. It may be impossible to distinguish between speech and noise using simple level detection techniques when parts of the speech utterance are buried below the noise.

The realization of the architecture for VAD was not the subject of the project of this thesis, but a pre-existing model [39] was used and validated; this model proved to be very effective. As mentioned in *Section 2.3.1* we chose Ina Speech Segmenter model to realize the VAD module of our architecture.
In the next section we are going to describe in details the model that we used.

## 4.1 Ina Speech Segmenter

As mentioned in the previous section, we decided to use Ina Speech Segmenter to build the VAD module of our application.
Before committing to this model, we analyzed other solutions such as WebrtcVAD[56] and we were not satisfied. In particular, we found that WebrtcVAD lacks comprehensive documentation, and the experiments we run on the data described in *Section 4.1.2* showed a much lower accuracy compare to Ina Speech Segmenter. This tallies with the comparative analysis carried out in [57].

Ina Speech Segmenter [39] is a CNN-based audio segmentation toolkit, designed for conducting gender equality studies. It splits audio signals into homogeneous zones of speech, music and noise. Moreover, this toolkit, in addition to VAD, provides the possibility to tag signals by speaker gender (male or female). In the experiments, zones corresponding to speech over music or speech over noise are tagged as speech.
The system uses log-mel filterbank features (2.1.1), together with 4 convolutional and 4 dense layers.
In Figure 4.2 we can see the architecture implemented in Ina Speech Segmenter.

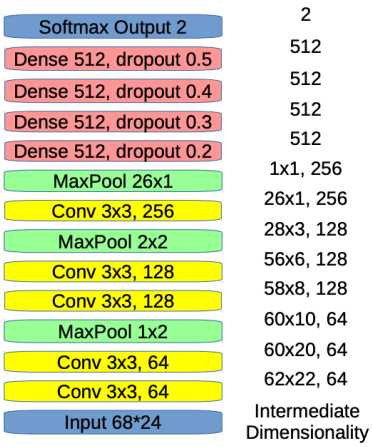


Figure 4.2: CNN gender detection architecture, using input features of 68 concatenated 24-dimension filter banks frames [39].

As described in [39], the CNN architecture was implemented using Keras. Two 3x3 convolutional layers of 64 neurons are first used, followed by a 1x2 max-pooling layer, aimed at providing invariance in the frequency domain. Two additional 3x3 layers of 128 neurons are then followed by a 2x2 max-pooling layer and a last 3x3 convolutional

layer of 256 neurons. A maximal temporal pooling layer, in charge of selecting the most discriminating pattern found in this relatively large time interval is finally used, and followed by 4 dense layers of 512 neurons, associated to increasing dropout rates, before the last softmax layer outputting 1 probability per gender. Each layer of the network is followed by batch normalizations and ReLU activations.

As for the features used by the authors of [39] to represent the audio entering the convolutional network, 24 Mel Scaled Filter Banks coefficients were used. As for energy detection, a simple energy threshold is used to discard frames associated with low energy [39].

Practically speaking, in order to use this architecture we needed to exploit the Python API of Ina Speech Segmenter [58]. The instatiation of the Convolutional Model is made through the intialization of the `Segmenter` object provided by the API.

```
segmenter = Segmenter(vad_engine='smn', detect_gender=False)
```

Figure 4.3: Instatiation of Ina Speech Segmenter.

As can be seen in Figure 4.3 we needed to specify the following parameters:

- `vad_engine='smn'`: it is the more recent engine and splits signal into speech, music and noise segments.
- `detect_gender=False`: if set to True, performs gender segmentation on speech segment and outputs labels ”female” or ”male”. Otherwise, outputs labels ”speech”. Since we were not interested in gender segmentation, we have set it to `False`.

When you pass an input file path to the segmenter after it has been instantiated, it performs the segmentation and returns a list of tuples, each containing the following elements:

- `label`: it is a string which can match with `speech`, `music` or `noEnergy`
- `start`: the time in which the segment is a associated to different label than that of the previous segment
- `end`: the time in which it is recognized that the assigned label no longer corresponds to the current one

The starting time instant of a segment corresponds to the ending time instant of the previous segment.

The discussion of the characteristics of Ina Speech Segmenter that have been used for the realization of the Voice Activity Detection system ends here, since we did not need to use the other features made available by the API. In the next section we will describe how we managed to use the information provided by the segmenter in order to achieve our goals.

### 4.1.1 Perform VAD with Ina Speech Segmenter

As explained in the previous paragraph, we can use the Ina Speech Segmenter to obtain a list of tuples characterized by different information.
In order to be able to use this information for a Voice Activity Detection architecture, we decided to use a *rule-based* approach. In particular, we analyzed the labels that the segment associates with the audio part coming into the system: if the number of segments associated with the label `speech` is greater than zero, then we considered the part of the input audio as containing speech, otherwise as noise.

### 4.1.2 Validation

In this section we will analyze the way in which the validation process of the Voice Activity Detection system was conducted. This process was aimed at evaluating the use of this Voice Acitivity Detection architecture within the global architecture proposed in this thesis.

#### Datasets

A combination of different audio sources was used to validate Ina Speech Segmenter. We are going to list the data that were used:

- Two datasets of those that were used for the training and validation of the Speech Emotion Recognition system (please refere to S*ection 5.2* for a detailed description of the datasets):
  - RAVDESS
  - TESS

  These two datasets were included because it is critical that the system recognizes the presence of the voice in at least two of the datasets that were used for the recognition of emotions within the voice.

- EMO-DB: the EMO-DB database [59] is a freely available German emotional database. The database is created by the Institute of Communication Science, Technical University, Berlin, Germany. Ten professional speakers (five males and five females) participated in data recording. The database contains a total of 535 utterances. The EMODB database comprises of seven emotions: 1) anger; 2) boredom; 3) anxiety; 4) happiness; 5) sadness; 6) disgust; and 7) neutral. The data was recorded at a 48-KHz sampling rate and then downsampled to 16-KHz. We downloaded the dataset from Kaggle [60]. The inclusion of this dataset served to assess that the Voice Activity Detection system is not dependent on the language of the incoming audio; this is a fundamental aspect to consider in the context of public transports, since they are places frequented by a great variety of ethnic groups.

- A custom dataset made up of the union of the three datasets listed above, for which all files have been modified by superimposing white noise on the original file. Even if the white noise is not comparable to the noise present in the context of public transports, it is essential to verify that the voice activity detection system is able to work discreetly even in the presence of audio files whose sound is not completely clean.

- A custom dataset consisting of some audio files extracted from the Google AudioSet ontology [61]. The audio files that we have labeled as *speech* refer to video contents present under the following AudioSet categories:

  - Speech
  - Laugh
  - Scream
  - Fear
  - Monologue

  The audio files that we have labeled as *noise* refer to video contents present under the following AudioSet categories:

  - Music Instruments (2 videos)
  - Guitar (2 videos)
  - Strange Instrument (1 video)
  - Classic music concert (1 video)
  - Car race noise (3 videos)
  - Rural sounds (3 videos)
  - Engine (2 videos)
  - Dog (2 videos)
  - Train (4 videos)

We ended up having 9506 audio files.

There was no need to worry about the length of the audio files as the segmenter accepts files of any length; moreover, we did not have to implement any resampling strategy

since the segmenter takes care of transforming all the input files into a temporary file converted at 16 KHZ[2] of sample rate before extracting the features.

**Results**

Evaluation was made on all datasets stated in (4.1.2). What we have achieved is a very satisfying result. In particular, what emerged from the validation is the following:

- Accuracy: 98.002%
- Error Rate: 1.998% (i.e. percentage of misclassified files)
- None of the files on which we manually injected white noise was misclassified; this means that all the speech files corrupted by noise have been classified as *speech*.
- Among the misclassified files, 108 files labelled as *speech* were labelled as *noise*.
- Among the misclassified files, 3 files labelled as *noise* were labelled as *speech*.
- Among the 108 misclassified files, 102 files belong to RAVDESS dataset and all those files belong to female actresses. Most of these files refer to "angry" and "fear" emotions.

The amount of misclassified files is so small that we are unable to generalize any of the considerations made regarding misclassified files.
Given the high accuracy, we decided to include the Voice Activity Detection system thus built within the overall architecture of this thesis project.
The only criticality identified is the time taken to instantiate the model, but it must be said that once the Segmenter object is instantiated (i.e. when the system is started), the prediction takes place in a very short time (around $1s$).

[2]This is the sample rate used by the authors of Ina Speech Segmenter while building the architecture.

# Chapter 5

# Speech Emotion Recognition

In this chapter we address the task of Speech Emotion Recognition (SER), which consists in detecting emotions from speech signals.

Speech is a rich, dense form of communication that can convey information effectively. It contains two types of information, namely linguistic and paralinguistic. The former refers to the verbal content, the underlying language code, while the latter refers to the implicit information such as body language, gestures, facial expressions, tone, pitch, emotion etc. Para-linguistic characteristics can help in understanding the mental state of the person (emotion), gender, attitude, dialect, and more [43].
Recorded speech has key features that can be leveraged to extract information, such as emotion, in a structured way.
There are two widely used representations of emotion: continuous and discrete.

In the continuous representation, the emotion of an utterance can be expressed as continuous values along multiple psychological dimensions. According to Ayadi, Kamel, Karray (2011) [62], "emotion can be characterized in two dimensions: activation and valence."

- *Activation* is the "amount of energy required to express a certain emotion" (p. 573) and research has shown that joy, anger, and fear can be linked to high energy and pitch in speech, whereas sadness can be linked to low energy and slow speech [43].
- *Valence* gives more nuance and helps distinguish between emotions like being angry and happy since increased activation can indicate both (p. 573) [43].

In the discrete representation, emotions can be discretely expressed as specific categories, such as angry, sad, happy, etc. We are going to rely on this kind of representation.

Speech emotion recognition systems are pattern recognition systems, and are generally composed of three parts: (1) speech signal acquisition, (2) feature extraction, and (3) emotion recognition through the use of classifiers [23].
Figure 5.1 shows the steps involved in every Speech Emotion Recognition process.

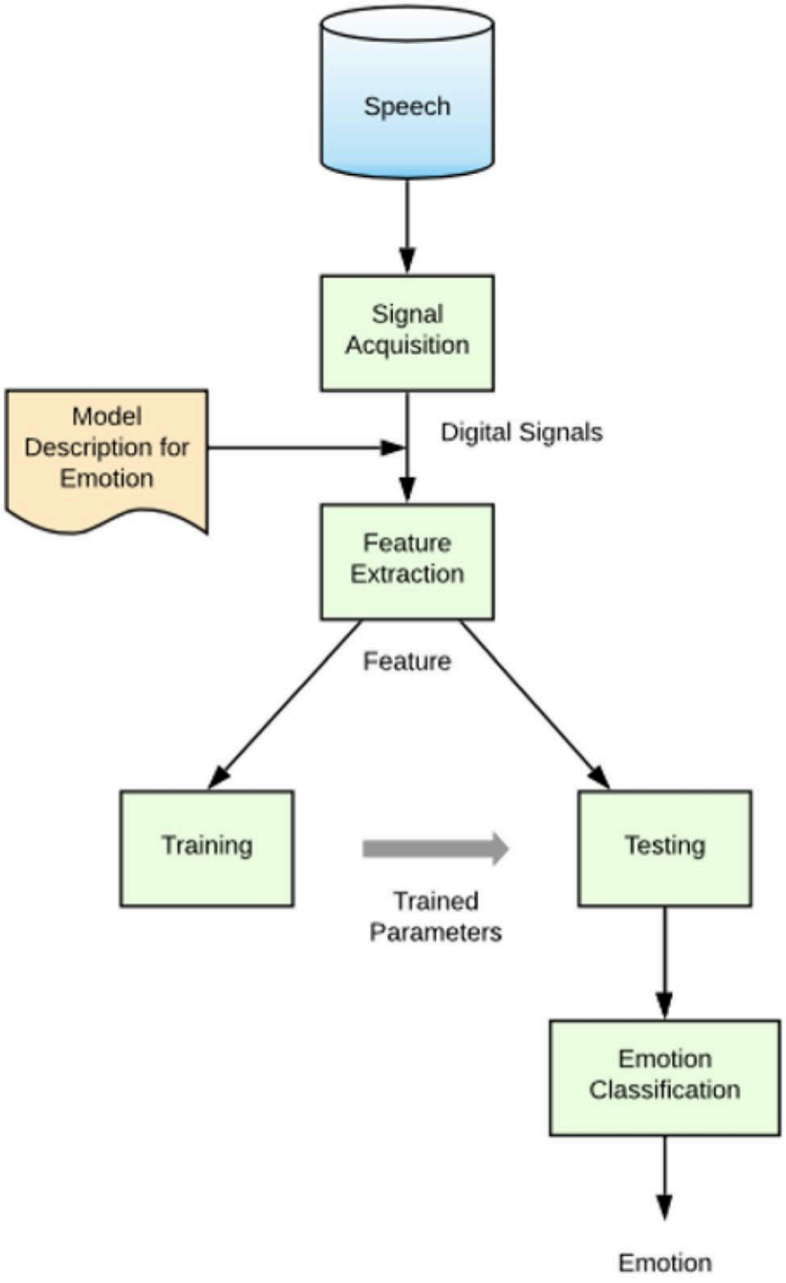


Figure 5.1: Block diagram of a general Speech Emotion Recognition system [23]

The performance of an emotion recognition system purely relies on feature extracted from the audio. They are broadly classified into time-based and frequency-based features. Extensive research has been carried out to weigh in the pros and cons of these features. There is no one particular sound feature which can perform well across all the sound signal processing tasks. Additionally, features are hand-crafted to suit the requirements of the problem in hand. With the advent of deep learning techniques, we have been successful in extracting the hierarchical representation of the speech from these features and identifying the underlying emotion in the speech. Hence, the performance of the model in a particular speech recognition task is much more dependent on the choice of the feature than the model architecture.[43]
Our emotion contains multiple features like Energy, Pitch, Rhythm, Loudness; these all come under acoustical information. We need to extract all these features from the given audio file using various pre-processing techniques known as feature extraction. By using the Librosa module, we can convert our audio files to digital data by using features in it like:

- *Mel feature*, which will be used to capture characteristic of the frequency of the signal represented on the Mel scale
- *MFCC feature*, which is used to describe the spectrum of the spectrum i.e. short term power spectrum of the given input audio file
- *Chroma feature* which is used to capture melodic and harmonic characteristics of sound based on pitch of the given input audio file
- *ZCR featur*e, which is used to specify the rate of sign changes of the particular signal during the duration of the particular frame
- *RMS feature*, which is used to analyse the loudness in the given input audio file since changes in loudness are important for extracting features in new input file [52]

After extracting all these features, we can apply various classifiers for matching these features with corresponding emotions. As specified in Section 2, here we are going to use the MFCCs and RMS features.

Despite extensive research in emotion recognition from speech, there are still several challenges such as imperfect databases, low quality of recorded utterances, cross-database performance, and difficulties when it comes to speaker independent recognition as each person has a different way of speaking [23].

One biggest challenge in this project is that many of the audio files may contain various disturbances like background noise, low voice, etc., which will affect the accuracy of the system; in particular, we are considering RAVDESS dataset, TESS dataset SAVEE dataset and CREMA-D. The four datasets just mentioned are publicly available. An ensemble of different classifiers has been built to address emotions as *disruptive* or *non-disruptive*.

In this chapter we are going to analyze in details the whole process that made possible the realization of the ensemble. This process includes a lot of steps that we are going to summarize in the following section (5.1). After having schematically mentioned the steps that allowed the construction of the architecture, we will proceed with the analysis of the datasets used.

## 5.1 Methodology

The flow diagram shown in Figure 5.2 explains the steps that led us from the selection of the data to the selection of the models for the construction of the ensemble.

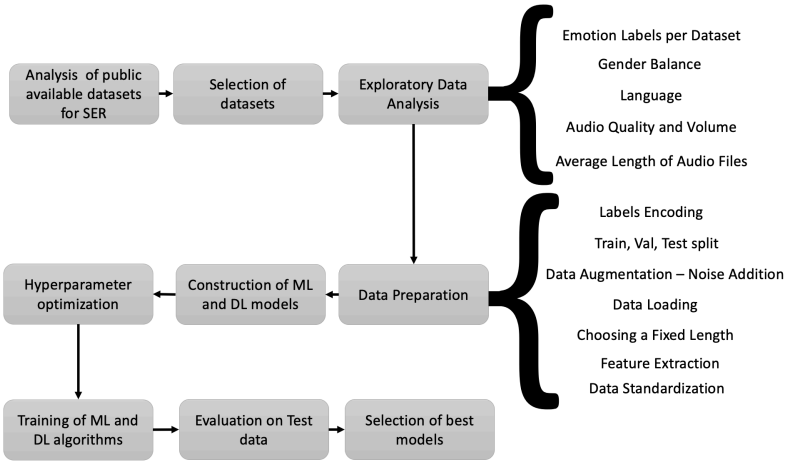


Figure 5.2: Flow diagram showing the steps of dataset selection, data preparation, models training and hyperparameters tuning

At first, researches in recent literature were analyzed in order to select the best datasets for our purpose. After selecting the datasets (RAVDESS, SAVEE, TESS, CREMA - please refer to *Section 5.2* for further details), we conducted an in-depth exploratory analysis, following the steps mentioned in the figure above. We then carried out the preparation of the data in order to be used as input for our classification models. The data preparation phase included all the steps illustrated in Figure 5.2. Subsequently, we have built our classifiers and we have carried out the optimization of some parameters and hyperparameters. After selecting the best parameters we proceeded by training our algorithms and we performed the evaluation on Test data. Finally, we selected the models which resulted in achieving good results on the test set.

Figure 5.3 summarizes the features that have been extracted in the Data Preparation phase, the classifiers that have been built and the Datasets that we built to carry out this analysis. It must be specified that even if we refer to them with the *noise* identifier, data augmentation has been applied only on the training sets.

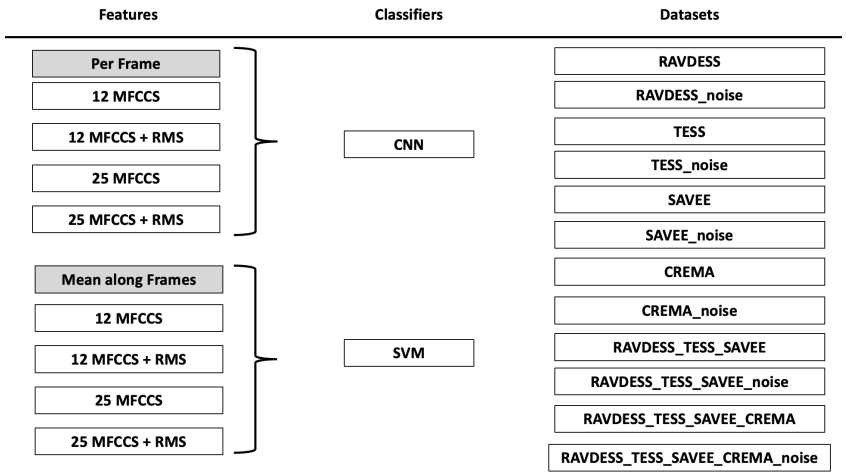


Figure 5.3: The 96 different experiments conducted for building the speech emotion classifier

Figure 5.4 explains the steps that have been followed in order to realize the last part of the architecture: the ensemble of the best models.

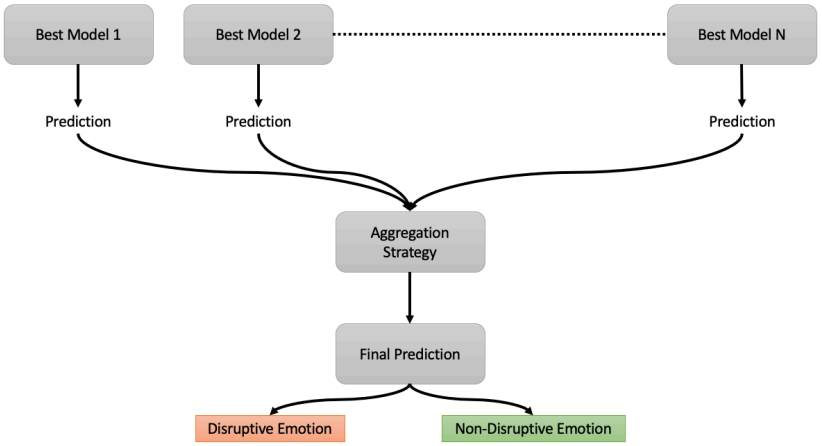


Figure 5.4: Flow diagram showing the steps of the creation of the ensemble

As can be seen above, the architecture of the ensemble is made up of several classifiers (the best models that we selected at the end of the flow in Figure 5.2) which are called upon to make a prediction on the entry fragment of the voice. Each of them produces a prediction and all these predictions are aggregated through an *Aggregation Strategy*. As we will see later *(Section 5.7)*, various strategies have been tried and evaluated. In the end, the final prediction is produced and the incoming audio segment is classified as *disruptive* or *non-disruptive*.

## 5.2 Datasets

Several speech corpora have been created in a wide variety of languages for developing emotional systems that work on audio [63]. They can be divided into three categories [64]:

- *Acted (or simulated)*, namely recorded by actors or trained volunteers. These are the most straightforward datasets to train a model, but the further from real world scenarios. As matter of fact, here the acoustic features of the utterances may be exaggerated, while more subtle features may be ignored.
- *Invoked*, i.e., obtained provoking an emotional reaction using staged situations. As compared to acted datasets, these are more naturalistic.
- *Spontaneous*, namely obtained by recording speakers in natural situations or extracting speech from movies, TV programs, call center conversations, radio talks. These datasets are hard to obtain due to the difficulty to classify emotions in "wild" situations, but are the most realistic.

From all available databases, we have selected the following:

- RAVDESS
- TESS
- SAVEE
- CREMA-D

The datasets listed above belong all to the *acted* category.
In the following sections we will analyse in detail the datasets that were used to train and test the emotion recognition model developed in this thesis. We will provide an in-depth description of the datasets and their main characteristics since nowadays the quantity and quality of the data used to build Speech Emotion Recognition Systems represent the biggest problem for the construction of robust models. Since our system is designed to work within a real scenario we needed to pay attention to each of these aspects in order to evaluate the possible performances of the model in the reference environment.
In particular, in the reference scenario, i.e. that of public transports, it is crucial to use

a noise robust, gender-independent, speaker-independent, cross-linguistic system and to do so requires meticulous data work.

### 5.2.1 Ryerson Audio-Visual Database of Emotional Speech and Songs

The Ryerson Audio-Visual Database of Emotional Speech and Songs (RAVDESS) [65] is a simulated and validated multimodal[1] database of emotional speech and song; it was released in 2018 by researchers of the SMART lab at Ryerson University in Toronto, Ontario, Canada. For the audio data, the actors recorded the sentences both as normal speech and as songs. The song data was not considered for this thesis. The audio files were recorded at 16 bits per sample, at a sampling rate of 48 KHz and saved in uncompressed wave format.

The database is gender balanced consisting of 24 professional actors (12 females and 12 males) of age range 21-33 year, speaking in a neutral North American accent. Each actor recorded two lexically matched sentences in eight different emotions. The two sentences are “Kids are talking by the door” and “Dogs are sitting by the door.” Moreover, the eight emotions are neutral, calm, happy, sad, anger, fear, disgust, and surprise. The emotion labels are included in the WAV audio file names. Out of the eight emotions, seven of them were recorded twice per sentence – once with normal intensity and the other time with stronger intensity. There was only one recording per sentence for the neutral emotion since there is no strong intensity for this emotion.

All information regarding the specifics of each audio file is reported in the filename identifiers as follows:

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong). There is no strong intensity for the ’neutral’ emotion.

[1] It contains both video and audio data.

- Statement (01 = ”Kids are talking by the door”, 02 = ”Dogs are sitting by the door”).
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

So, here is an example of an audio filename: *02-01-06-01-02-01-12.mp4*
This means the meta data for the audio file is:

- Video-only (02)
- Speech (01)
- Fearful (06)
- Normal intensity (01)
- Statement ”dogs” (02)
- 1st Repetition (01)
- 12th Actor (12) - Female (as the actor ID number is even)

We ended up having a total of 1,440 recording samples, with 24 actors x 2 sentences x 8 emotions x 2 repetitions x 2 emotional intensity with the exception of the neutral emotion. Thus, seven emotions have 192 data samples, and the neutral class has 96 data samples. The other datasets considered for the project of this thesis do not contain the ”calm” class emotion and we also observe that human raters found it difficult to distinguish ”neutral” and ”calm” emotions [43]. We manually listened to the audio files from RAVDESS and also felt that emotions ”calm” and ”neutral” sounded very similar to each other. For the two reasons listed above, we have decided to merge the ”neutral” class and the ”calm” class under the ”neutral” label. So, in our dataset, six emotions have 192 data samples and neutral class has 285 samples.
Although the union of these two classes led to an imbalance of the dataset, we decided not to implement any resampling techniques as we experimentally proved that the use of this technique led to an increase in the overfitting of the model. The motivation linked to this behavior is that, being the amount of data limited to be able to train a neural model, the replication of the data leads to an increase in variance.

Even though the dataset can be downloaded directly from the authors' site [65], for the purpose of this project it has proved useful to download the dataset from Kaggle [66], as it contains only the 1440 speech files that we needed[2].

[2]The original datasets contains 7356 recordings including audio/song/video contents.

### 5.2.2 Toronto Emotional Speech Set

The second dataset we considered is Toronto Emotional Speech Set (TESS)[67].
TESS is a simulated dataset created in 2010 by researchers from the University of Toronto Psychology Department. In this dataset, there are 2800 unique speech files recorded by two actresses aged 26 and 64 years, both of them speak a set of 200 target words in the carrier phrase "Say the word _____."; so, just like RAVDESS, the sentences spoken by the actors are lexically similar. Both actresses are from the Toronto area, speak English as their first language, are university educated, and have musical training. For 26 years aged person, there are about 1401 unique speech files, and for 64 years aged person, there are about 1399 unique speech files. Combining both these speech files, we ended up having 2800 unique speech files and seven different emotions for each of them: anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral. All audio files were recorded at 16-bits per sample, at a sampling rate of 24414 Hz, and saved in WAV audio file format.[68] The emotion labels were extracted from the file names. The TESS dataset can be downloaded from [69]. Just like RAVDESS, we preferred downloading the dataset from Kaggle website [70].

### 5.2.3 Surrey Audio-Visual Expressed Emotion dataset

The third dataset we considered is Surrey Audio-Visual Expressed Emotion (SAVEE) dataset [71].

In the SAVEE dataset, there are seven different emotions recorded by four native English male speakers DC, JE, JK, and KL. They are postgraduate students and researchers at the University of Surrey aged from 27 to 31 years. The database consists of 480 British English utterances in total. The sentences were chosen from the standard TIMIT corpus [72] and phonetically-balanced for each emotion. The emotions present in SAVEE are seven and they have been described psychologically in discrete categories: anger, disgust, fear, happiness, sadness and surprise. This categorization of emotions is supported by the cross-cultural studies of Ekman [73] and studies of automatic emotion recognition tended to focus on recognizing these [74]. The authors of the dataset did not report on the recording characteristics (bit-per-sample, sample rate and audio file format).

Just like 5.2.1 and 5.2.2, we preferred downloading the dataset from Kaggle website [75].

### 5.2.4 Crowd-sourced Emotional Multimodal Actors Dataset

The fourth dataset used to develop the speech model in this thesis is Crowd-sourced Emotional Multimodal Actors Dataset (CREMA-D). Just like RAVDESS (5.2.1) and TESS (5.2.2), this is a simulated speech corpus. Like RAVDESS, CREMA-D is a multimodal dataset.

Actors who participated had ages ranging from 20 to 74 years old. There were 48 male actors and 43 female actresses with a total of 91, and even though they came from different ethnic backgrounds (African America, Asian, Caucasian, Hispanic, and Unspecified), they were all English speakers. The speech recordings were done at 16-bits per sample, at a sampling rate of 16 KHz, and saved in the WAV audio format. The actors recorded twelve different emotionally neutral sentences:

- It's eleven o'clock.
- That is exactly what happened.
- I'm on my way to the meeting.
- I wonder what this is about.
- The airplane is almost full.
- Maybe tomorrow it will be cold.
- I would like a new alarm clock
- I think I have a doctor's appointment.
- Don't forget a jacket.
- I think I've seen this before.
- The surface is slick.
- We'll stop in a couple of minutes.

The actors recorded the above listed sentences in six different emotions of anger, disgust, fear, happy, neutral, and sad, at four different intensity levels of low, medium, high, and unspecified. Each emotion class has 1271 data samples, except for the neutral class,

which has 1087 data samples, with a total of 7442 samples.

As will be recalled later in the exploratory data analysis (*Section 5.2.5*), we decided to keep only the files marked with a *high level* of intensity in order to insert as little noise as possible in the classification. We have experimentally verified that due to the poor audio quality [68] of the files present in CREMA-D, keeping the files at any intensity level cheated the classifier. Therefore, unfortunately, we had to drastically reduce the number of samples in order to obtain a less robust, but more accurate classification. It is no coincidence that this dataset is hardly ever taken into consideration in the literature [76]. It is a pity to have to carry out this elimination of samples from such a large dataset because, if the quality of the audio files were better, it would be possible to obtain robust models with respect to the gender and accent independence from the speaker.

The dataset is available online at [77], but, just like for the previous mentioned datasets, we downloaded the dataset from Kaggle [78].

### 5.2.5 Exploratory Data Analysis

In order to carry out a comprehensive study, to correctly prepare the data and to correctly interpret/justify the results obtained experimentally, it was strategic to carry out a thorough exploratory analysis of the data. We will therefore list below the analysed aspects and the considerations made on the four datasets described in the previous sections. Most of the aspects we are going to analyse are those that define the so-called *scope* [3] of the dataset [63].

#### Emotion Labels per Dataset

| | RAVDESS | TESS | SAVEE | CREMA-D |
|---|---|---|---|---|
| **fear** | • | • | • | • |
| **disgust** | • | • | • | • |
| **neutral** | • | • | • | |
| **calm** | • | | | |
| **happy** | • | • | • | • |
| **sadness** | • | • | • | • |
| **surprise** | • | • | • | |
| **angry** | • | • | • | • |

Table 5.1: Emotions per Dataset

As can be seen from Table 5.1, not all datasets contain the same categories of emotions. In particular, it can be noted that the CREMA-D dataset contains neither "neutral" nor "calm" or "surprise". As stated in the description of the dataset in *Section 5.2.4*, the original dataset contains files in the "neutral" category, but having selected only audio files recorded with a *high level* of emotional intensity, this category was naturally excluded. In particular, this label was excluded because, as repeatedly stated in the literature, it is not possible to express "neutral" emotions at different levels of intensity without changing the intention and therefore the category of the emotion itself.
It would certainly be interesting in the future to carry out experiments involving CREMA-D again, by including the files that refer to the "neutral" emotion.
Moreover, as can be seen from the Table 5.1, no dataset, except for RAVDESS, contains files labelled with the emotion "calm".

[3] The scope of a database design consists of several kinds of variation in a database like number of speakers, speaker gender, type of emotions, number of dialects, type of language and age.[63]

For this reason, as previously specified in *Section 5.2.1*, we merged "neutral" and "calm" in RAVDESS, without implementing resampling strategies to rebalance the dataset.

### Gender Balance

The RAVDESS dataset is very rich in nature given that it does not suffer from gender bias; moreover, it consists of wide range of emotions and at different level of emotional intensity [43].
Other datasets, such as Surrey Audio-Visual Expressed Emotion (SAVEE) and Toronto Emotional Speech Set (TESS), consist of audios from only male and female actors respectively, and therefore they present a strong gender imbalance.
As regards CREMA-D we can say that the dataset is quite gender-balanced since it contains 48 male actors and 43 female actresses.

### Language

| | **Language** | **Accent** |
|---|---|---|
| **RAVDESS** | English | North American |
| **TESS** | English | North American |
| **SAVEE** | English | British |
| **CREMA-D** | English | Mixture of Accents |

Table 5.2: Languages of the Datasets

As can be seen from the Table 5.2 all the datasets we used refer to the English language; the only thing that varies between the various datasets in this case is the accent of the actors. It would certainly be interesting to expand this research by including datasets referring to different languages in order to implement a cross-linguistic model, like the one carried out in [3].
It might also be interesting to observe how much the generalisation ability of the model for the emotion recognition task changes as the language varies and to see how strong the correlation is between the language spoken and the emotion expressed.

### Audio Quality and Volume

In general, the aspects that characterise audio files are as follows:

- **Bit-depth**: the higher the bit-depth, the more dynamic range can be captured. Dynamic range is the difference between the quietest and loudest volume of an instrument, part or piece of music. A typical value seems to be 16 bit or 24 bit. A bit-depth of 16 bit has a theoretical dynamic range of 96 dB, whereas 24 bit has a dynamic range of 144 dB.
- **Channels**: usually 2, meaning you have one left speaker and one right speaker.
- **Sample rate**: audio signals are analog, but we want to represent them digitally; this means that we want to discretize them in value and in time. The sample rate gives how many times per second we get a value. The unit is $Hz$. The sample rate needs to be at least double of the highest frequency in the original sound, otherwise we get aliasing. Human hearing range goes from ∼20Hz to ∼20KHz, so we can cut off anything above 20KHz; this means that the use of a sample rate of more than 40KHz is useless.

We could retrieve the information listed above by looking at the `subtype` attribute of the `SoundFile` [79] object in Python. In reality, we discovered that the information retrieved by the use of attributes and methods of `SoundFile` do not refer to the original characteristic of the audio, but to characteristics of the SoundFile object, which applies some modification while reading audio files. So, to retrieve the exact information about it, it is better to refer directly to the description of the datasets provided by the authors. Table 5.3 shows the difference between the characteristics retrieved through the `SoundFile` object of Python and the characteristics listed by the authors of the datasets:

| | SoundFile | | | Authors' Description | | |
|---|---|---|---|---|---|---|
| | **Bit Depth** | **Channels** | **Sample Rate (Hz)** | **Bit Depth** | **Channels** | **Sample Rate (Hz)** |
| **RAVDESS** | 16 | 1 | 16000 | 16 | 1 | 48000 |
| **TESS** | 16 | 1 | 44100 | 16 | 1 | 24414 |
| **SAVEE** | 16 | 1 | 16000 | - | 1 | - |
| **CREMA-D** | 16 | 1 | 16000 | 16 | 1 | 16000 |

Table 5.3: Audio Characteristics of audio files for each dataset. In orange, the audio characteristics retrieved through SoundFile. In yellow, the audio characteristics provided by the authors of the datasets

We decided to take a closer look at the characteristics of the audio files because we noticed, through listening to the files, that the quality and volume of the recordings appears to differ between datasets. In particular, it is possible to notice a huge difference between the quality and clarity of the audio of RAVDESS and the characteristics of CREMA-D.
A special note should be made of the quality of the RAVDESS dataset, which is the most widely used in the literature, not least because of the excellent quality of the recordings. Extensive validation and reliability tests have been performed by the creators of RAVDESS dataset. From a "pseudo-randomly chosen set of 298 stimuli, consisting of 174 speech and 124 song presentations," 247 naive participants were asked to make three judgements on three classes: "category of the emotion, strength of the emotion, and genuineness of the emotion" [65]. In [43], the authors observed that approximately 73 % of the rater-chosen emotion were well-acted by the actors, ensuring the reliability of the classification of the emotions and the audio content.
The difference between the quality of the audio files of RAVDESS and CREMA-D was also measured quantitatively by the authors themselves: the CREMA-D dataset has fewer categories for the classification tasks, but CREMA-D holds the accuracy of 63.6% from human recognition for six categories which are less than RAVDESS at 72.3% for eight categories. [76].

Starting from these considerations present in the literature, we decided to carry out an in-depth analysis of this issue to avoid misclassification problems due to the merging of the datasets. In particular, from a qualitative analysis it was possible to notice that audio files referring to the emotion "angry" were clearly distinguishable in RAVDESS, SAVEE and TESS, while they were practically assimilable to "normal" in CREMA-D. Therefore, after this qualitative analysis, we wanted to verify quantitatively what was previously described. To do so we checked the mean of the amplitude in dB and the median of the Root Mean Squared Error (RMS), for each dataset and for each emotion. Both the amplitude in dB and the RMS are two measures to understand the loudness of an audio file.
As can be seen from Tables 5.4 and 5.5 the level of loudness for the same emotion varies between the different datasets.

| | RAVDESS | TESS | SAVEE | CREMA-D |
|---|---|---|---|---|
| **fear** | -46.64 | -26.96 | -30.69 | -27.40 |
| **disgust** | -51.79 | -34.38 | -37.01 | -32.24 |
| **neutral** | -61.54 | -38.74 | -42.40 | - |
| **calm** | -59.53 | - | - | - |
| **happy** | -51.27 | -35.07 | -28.84 | -27.47 |
| **sadness** | -56.89 | -39.51 | -38.81 | -36.95 |
| **surprise** | -54.04 | -29.67 | -29.04 | - |
| **angry** | -43.45 | -25.70 | -29.30 | -16.91 |

Table 5.4: Median of the Amplitude in dB for each dataset, for each emotion.

| | RAVDESS | TESS | SAVEE | CREMA-D |
|---|---|---|---|---|
| **fear** | 0.010 | 0.113 | 0.049 | 0.054 |
| **disgust** | 0.051 | 0.079 | 0.020 | 0.021 |
| **neutral** | 0.025 | 0.062 | 0.027 | - |
| **calm** | 0.021 | - | - | - |
| **happy** | 0.009 | 0.130 | 0.030 | 0.045 |
| **sadness** | 0.003 | 0.079 | 0.034 | 0.008 |
| **surprise** | 0.007 | 0.124 | 0.029 | - |
| **angry** | 0.018 | 0.152 | 0.059 | 0.124 |

Table 5.5: Mean of the Median of the RMS for each dataset, for each emotion.

Since this could introduce noise during the classification task, we tried to equalise the loudness level of the different datasets through volume normalisation.

We have performed volume normalisation through the use of `ffmpeg-normalize` [80]. The normalization is performed with the `loudnorm` filter from `ffmpeg`, which was originally written by Kyle Swanson. It brings the audio to a specified target level. This ensures that multiple files normalized with this filter will have the same perceived loudness.

After performing the normalisation process on all files in each dataset, we performed a qualitative analysis of the generated files by listening to some of them.

The qualitative analysis showed that the intention of the normalised files is different to that of the original files. This means that after normalisation, for example, files labelled as ”angry” seem to belong to the ”neutral” category. In order to verify quantitatively what emerged from the qualitative analysis, we carried out some experiments using the convolutional model that we will describe later. The results and settings of the various experiments are reported in the *Appendix A.1*. As is evident from the results, using the audio files normalized according to the previously described strategy does not lead to

improvements in the results, rather to a worsening.
The results therefore confirm what is possible to understand by listening to normalized files, that is, after normalization, the volume of the different audio files is equalized but the intention of the voice present in the audio is ruined; therefore it is possible that the labels are no longer characteristic of the file after normalization.
In conclusion, normalization is not a viable path in order to be able to match the quality of the different datasets.

### Average length

The average length of the audio files ranges from 2 to 5 seconds for all datasets considered [81]. As will be described later in the data preparation section, a fixed length (5 seconds) was chosen and the longest and shortest files were cut and padded respectively.
As for the padding, used the *median* value of the representation of the sampled audio file.

## 5.3 Data Preparation

In the following sections we will describe all the steps involved in the preaparation of the data to train our classification models. Specifically, the phases we are going to describe are the following:

- Labels Encoding
- Train, Val, Test Split
- Data Augmentation
- Data Loading
- Choosing a fixed length
- Feature Extraction
- Data Standardization

Our objective was to analyze the change in performances linked to a combination of the following variables:

- dataset
- features
- classification model

To pursue this objective we ended up having 96 different models to test.
In the next sections we will describe how we defined the input data of our models and what differentiate one model from another.

### 5.3.1 Labels Encoding

For the purpose of this research we have developed a model capable of classifying an emotion either as *non-disruptive* or as *disruptive*. In order to divide emotions into these two categories we have taken our cue from the division made in [43] between *positive* (*non-disruptive* in our case) and *negative* (*disruptive* in our case) emotions.
In particular, we decided to split the emotions as follows:

- *Disruptive Emotions* (`label` = 1): anger, sadness, fear, disgust
- *Non-Disruptive Emotions* (`label` = 0): neutral, happiness, surprise

In order to devise this division, it was important to think about what could be the "disruptive" situations in the context of public transports and consequently make assumptions to state which emotions (among those available) could characterise the situations identified.

Within the vast universe of possible "disruptive situations in public transports" we decided to consider:

- disputes between passengers
- possible cases of physical violence
- possible cases of non-compliance with rules

Following common sense we came to the conclusion that the emotions that could be encountered in the situations identified are:

- *anger*: possible emotion expressed by passengers during disputes with other passengers/controller or during telephone conflicts
- *sadness*: possible emotion expressed by passengers before/during/after a conflict
- *fear*: possible emotion expressed by the attacked passenger or by the frightened crowd if involved in a disruptive situation
- *disgust*: possible emotion of displeasure expressed by passengers directly involved in a conflict or observing the conflict; possible emotion of displeasure expressed by passengers criticising other passengers who violate the regulation

The division of emotions into *disruptive* and *non-disruptive* and the description of disruptive situations would require the development of a topology of emotions and situations by a heterogeneous team of experts. Not having the necessary professional figures available, we reasoned on the problem based on common sense and our previous knowledge, but in order to be able to use the model in the real scenario it would undoubtedly be necessary to refine this part of problem-design as previously mentioned.

### 5.3.2 Train, Val, Test Split: the need of a Gender-Independent and Speaker-Independent model

In the initial experiments we used a random division of the dataset into training, validation and test sets, replicating the approach of some of the state-of-the-art articles [23, 52, 76, 82–84].

At an early stage we decided to use RAVDESS, TESS and SAVEE for training and CREMA-D for final validation/testing. This choice was dictated by the fact that we wanted to expand RAVDESS consistently and the only way to do this without unbalancing the genders was to work with TESS and SAVEE together; recall that TESS contains only female voices, while SAVEE contains only male voices. We therefore combined the three datasets to construct and randomly split the training, validation and test set. As evident from the experimental results in the *Appendix A.2*, the convolutional model trained on these data was affected by significant overfitting. After some verifications we came to the conclusion that the cause of the overfitting did not reside only in the architecture used, but mainly in the data used for the training; that is, we realised that by randomly dividing the dataset, many files referring to the same actor were present both in the training set and in the validation set, leading to overfitting on the actor itself and therefore to a non-objective evaluation of the model.

Given the use case, we thought that in order to build a robust architecture it was necessary to have a model:

- gender-independent
- speaker-independent

To do so, we divided each dataset in a way that:

- each set had the same male and female voices
- the same actor would not repeat in the training, validation and test set

In realizing this dataset splitting we took a cue from [43].
After splitting the datasets we obtained what is shown in Figure 5.5:

| | Train | Val | Test |
|---|---|---|---|
| **RAVDESS** | Actors: 20 (10 females, 10 males) — Data Samples: 1200 | Actors: 2 (1 female 1 male) — Data Samples: 120 | Actors: 2 (1 female 1 male) — Data Samples: 120 |
| **SAVEE** | Actors: 2 (2 males ) — Data Samples: 240 | Actors: 2 (2 males) — Data Samples: 120 | Actors: 2 (2 males) — Data Samples: 120 |
| **TESS** | Actors: 1 (1 female) — Data Samples: 1400 | Actors: ------- — Data Samples: ------- | Actors: 1 (1 female) — Data Samples: 1400 |
| **CREMA-D** | Actors: 64 (32 females 32 males) — Data Samples: 320 | Actors: 12 (6 female 6 male) — Data Samples: 60 | Actors: 12 (6 females 6 males) — Data Samples: 60 |
| **Summary** | **Total number of actors**: 87<br>**Total number of samples**: 3160 | **Total number of actors**: 16<br>**Total number of samples**: 300 | **Total number of actors**: 17<br>**Total number of samples**: 1700 |

Figure 5.5: Distribution of genders and actors among the available datasets after carrying out the splitting

We need to specify few things about Figure 5.5:

- RAVDESS: after the splitting, each set is gender balanced.
- SAVEE: since the dataset contains only male actors, the sets are not gender balanced.
- TESS: since the dataset contains only female actresses, the sets are not gender balanced. Moreover, since the datasets is made up of only two actresses, in order to build a consistent division, we needed to get rid of the validation set.
- CREMA-D: after the splitting, each set is gender balanced. In particular, in order to obtain that division we proceeded as follows: after taking only the audio files with *high intensity* and checked which audio files belong to males or females (manually), each actor had 5 audio files associated. Since we had 235 males files and 220 females files, we decided to randomly remove 3 males actors from this list (3 actors * 5 files). In the end we ended up having 32 actors-per-gender in the training set, 6 actors-per-gender in validation set and 6 actors-per-gender in the test set.

After that splitting, the distribution of samples-per-class in the Training Set, before and after the Labels Encoding, is the following (Table 5.6):

| | **RAVDESS** | **TESS** | **SAVEE** | **CREMA** | **RAVDESS TESS SAVEE** | **RAVDESS TESS SAVEE CREMA** |
|---|---|---|---|---|---|---|
| **disgust** | 160 | 200 | 30 | 64 | 390 | 454 |
| **surprise** | 160 | 200 | 30 | - | 390 | 390 |
| **sadness** | 160 | 200 | 30 | 64 | 390 | 454 |
| **angry** | 160 | 400 | 30 | 64 | 590 | 654 |
| **fear** | 160 | 200 | 30 | 64 | 390 | 454 |
| **happy** | 160 | - | 30 | 64 | 190 | 254 |
| **neutral** | 240 | 200 | 60 | - | 500 | 500 |
| **disruptive (1)** | 640 | 1000 | 120 | 256 | 1760 | 2016 |
| **non-disruptive (0)** | 560 | 400 | 120 | 64 | 1080 | 1144 |

Table 5.6: Distribution of samples-per-class in the Training Sets. In orange the count of emotions before performing Labels Encoding, in green the count of emotions after performing Labels Encoding. Where a value is not indicated, it means that the emotion is not present in the dataset. For the TESS dataset the *happy* emotion is not present because the actress who acted for this emotion happened by chance in the Test Set.

### 5.3.3 Data Augmentation: Noise Addition

All speech recordings in the above-mentioned datasets were done in the absence of background noise, in a "noiseless" environment. The only noise that is audible in these recordings is a combination of the inherent static noise created by the recording equipment, and the echoes coming from the surroundings. In real-world applications, however, conversations in a "noiseless" environment are very rare because there is always some noise around people during a conversation [68].
Furthermore, in our specific use case, that of railway transport, we must consider that there are certain environmental noises typical of the reference context; these noises must necessarily be taken into account in order to ensure that the final speech emotion recognition model is robust even in the case of malfunctioning/absence of a denoising module. A new set of data was created from the clean speech recordings by introducing some background noise to them.
It is impossible to consider all possible types of background noises when creating the new dataset with added background noise. Therefore, three noise samples were selected for the purpose of this study [68]. The audio files were downloaded from [85] and [86]. The noise samples selected are:

- *Inside Train Sound*: a recording of inside of a train.
- *Freight Train*: a recording of a freight-train passing by with squeaky wheels.
- *Small Crowd*: a recording of children playing in a playground.

The "Small Crowd" sample is under an attribution 3.0 license, the "Freight Train" sample is of public domain and the "Inside Train Sound" is under an attribution of Standard YouTube license. The reason behind selecting these specific noise samples was to cover some of the general properties of common background noises in our context. For example, the playground background noise is a variation of the cocktail party noise where the noise created is from people talking in the background; this is a very common situation in a train. The other two kind of noises represent very typical sound that can be heard inside a moving train. In particular, "Inside Train Sound" refers to a type of noise that can be detected by the microphone if it is placed near the passengers, while the other refers to a type of noise that the microphone can detect if it is placed near the mechanical parts

of the train or near the passageways between carriages.
After these three noise samples were selected, the noise addition procedure was carried out using the PyDub library of Python [87].
The pipeline to add the noise to the audio files includes the following steps:

1. randomly pick one of the 3 noise samples
2. check duration of voice audio file from the dataset
3. compute chunks of the selected noise sample with the same duration of the voice audio file
4. pick a random chunk
5. compute the loudness of both noise and voice file in dB
6. compute the absolute difference in dB
7. lower the volume of the noise file of a quantity equal to the absolute difference minus 2 dB; this was done to make the voice sound a bit stronger than the background noise
8. add noise to the audio sample of the training dataset by overlaying the chunk with the voice audio file
9. export the new file (i.e. audio file affected by noise)

The pipeline showed above has been followed for each sample of each Training Set; for Training Set here we refer to the sets computed for each dataset along with the speaker-gender splitting (please refer to *Section 5.3.2*).
As can be seen from the steps listed above, to prevent the machine from learning the noise features, two precautions were taken [68]:

1. First, each noise sample is more than fifteen seconds long, with the "Inside Train Sound" 2-minutes long, "Freight Train" noise 16-seconds long, and "Small Crowd" noise 19-seconds long. The clean speech samples were roughly two to five seconds long for all datasets. The Python code randomly took chunks of noise audio samples of the size of the clean speech samples and added both together to create

the noise-added samples. This ensured that the same part of the noise samples was not being added to the clean speech data.

2. Second, also, for each clean speech sample, only one noise-added sample was generated, where noise type was randomly chosen as discussed above. Thus, the final dataset contained the same amounts of clean speech and noise-added samples.

Figure 5.6 summarizes the details of the final datasets. The numerical identifiers in the Figure 5.6 will be used later to identify the individual experiments carried out.

| Identifier | Name | Description | Data Samples | Training Samples |
|---|---|---|---|---|
| 1 | **RAVDESS** | Oiriginal RAVDESS corpus | 1440 | 1200 |
| 2 | **RAVDESS_noise** | Original RAVDESS corpus along with the noise-added versions | 2640 | 2400 |
| 3 | **TESS** | Original TESS corpus | 2800 | 1400 |
| 4 | **TESS_noise** | Original TESS corpus along with the noise-added version | 4200 | 2800 |
| 5 | **SAVEE** | Original SAVEE corpus | 480 | 240 |
| 6 | **SAVEE_noise** | Original SAVEE corpus along with the noise-added versions | 520 | 480 |
| 7 | **CREMA** | Original CREMA corpus | 440 | 320 |
| 8 | **CREMA_noise** | Original CREMA corpus along with the noise-added versions | 760 | 640 |
| 9 | **RAVDESS_TESS_SAVEE** | Original RAVDESS, TESS and SAVEE corpora | 4720 | 2840 |
| 10 | **RAVDESS_TESS_SAVEE_noise** | Original RAVDESS, TESS and SAVEE corpora along with the noise-added versions | 7650 | 5680 |
| 11 | **RAVDESS_TESS_SAVEE_CREMA** | Original RAVDESS, TESS, SAVEE and CREMA-D corpora | 5160 | 3160 |
| 12 | **RAVDESS_TESS_CREMA_noise** | Original RAVDESS, TESS, SAVEE and CREMA-D corpora along with the noise-added versions | 8320 | 6320 |

Figure 5.6: Dataset naming convention followed in this project together with the number of samples.

### 5.3.4 Data Loading

In order to work with audio files stored in a *.wav* format, we needed to transform them into a useful represention; to do so, we decided to read the audio files using the `librosa.load` [88] function. The function just mentioned loads an audio file as a floating point time series. The target sampling rate has been set to 16 KHZ, resampling all audio files which have a sample rate different from it (please refer to *Section 5.2.5* to retrieve information about the sample rate of the original audio files).
We must specify that the function just mentioned will be the same one used in the Final Architecture to process the input audio.

### 5.3.5 Choosing a fixed length

Since our training models work with fixed size inputs, and considering that all the audio samples of our datasets have different duration (between 2 and 5 seconds), we needed to make all the input audios into a fixed length (duration) by trimming or padding them appropriately.
Signals shorter than the chosen-length were padded using the median value of the audio representation in samples, while signals longer than the chosen-length were truncated.
We fixed the audio duration to 5 seconds; since 1 second of audio at a sample rate of 16 KHZ is represented by 16000 samples, each audio file padded/truncated at 5 seconds of duration is represented by 80000 samples (i.e. `chosen_length` = 80000).
In order to select the best value for the `chosen_length` parameter we ran many times the *Experiment 1.1* by varying the audio length between 3 and 5 seconds.
We observed an interesting pattern: by increasing the length of audio samples, the accuracy of the model increased from a value of 81 % to 88 %. This behaviour lead us thinking that by padding the shorter files with the median, we were adding useful information for the purpose of classification.

### 5.3.6 Feature Extraction

The task of emotion classification involves two stages. The first stage is feature extraction followed by classification. As previously mentioned in *Chapter 2*, here MFCC and RMS are the speech featured considered. In particular, our objective was to analyze the change in performances linked to a combination of the following variables:

- dataset
- features
- classification model

In order to pursue that objective, we decided to perform different experiments:

- Convolutional Model
  - **Experiment 1**: 12 MFCCs
  - **Experiment 2**: 12 MFCCs + RMS
  - **Experiment 3**: 25 MFCCs
  - **Experiment 4**: 25 MFCCs + RMS
- SVM Model
  - **Experiment 5**: 12 MFCCs (mean)
  - **Experiment 6**: 12 MFCCs (mean) + RMS (mean)
  - **Experiment 7**: 25 MFCCs (mean)
  - **Experiment 8**: 25 MFCCs (mean) + RMS (mean)

As mentioned in *Section 5.3.4*, we first load the data, resampling each audio file at 16 KHZ. Then we proceed with the extraction of the MFCCs features through the Librosa's function `librosa.feature.mfcc` [14]. We left all the parameters at their default value, except for `n_mfcc` and `sr`, i.e. the parameters to specify the desired number of MFCCs and the sampling rate respectively. Librosa's `librosa.feature.mfcc` function really just acts as a wrapper to `librosa's librosa.feature.melspectrogram()` [89] function (which is a wrapper to `librosa.core.stft` [90] and `librosa.filters.mel functions` [91]).

All of the parameters pertaining to segementation of the audio signal - namely the frame and overlap values - are specified in the Mel-Scaled Power Spectrogram function (with other tuneable parameters specified for nested core functions).
We can specify these parameters as keyword arguments in the `librosa.feature.mfcc()` function; all extra `**kwargs` parameters are fed to Librosa's `librosa.feature.melspectrogram()`) and subsequently to `librosa.filters.mel functions`. By default, the Mel-Scaled Power Spectrogram window and hop length parameters are:

- `n_fft` = 2048 (i.e. length of the FFT window)
- `hop_length` = 512 (i.e. number of samples between successive frames)

The output of the MFCC function can be computed as follows:

$$output\_length = \frac{(seconds) \cdot (sample\_rate)}{hop\_length} \tag{5.1}$$

We used `sr` = 16000 (i.e. 16 kHZ sample rate), and, as specified in *Section 5.3.5*, we decided to cut and pad all audio files to 5 seconds. So, the output lenght of our MFCCs function was:

$$output\_length = \frac{(5s) \cdot (16000Hz)}{512} = round(156.25) = 157 \tag{5.2}$$

The MFCCs function output is an array of shape (`n_mfccs, output_length`); the meaning is that there are *n_mfccs* over *output_length* audio samples. We transpose the array to match the input requirements of the 1D Convolutional Neural Network. In particular, the 1D convolutional Neural Network takes as input arrays of shape (`batch_size, time_steps, features`), where `time_steps` is represented by what we called `output_length`.
For *Experiment 1* and *Experiment 2* the input feature vector has the following shapes respectively: (`batch_size`, 157, 12) and (`batch_size`, 157, 25)[4].
For all the experiments that include the use of the SVM Model (so experiments from 5 to 8) we needed to have a 1-dimensional feature vector. To do so, we decided to use a

[4]This means that we have an array representing a temporal series of 157 time steps, each of which characterized by 25 features.

summary statistic of the MFCCs: the average value for each MFCC coefficient.
We ended up having a number of features per audio file equal to the number of MFCC components and each of this number represents the average value of the component along all the audio samples.
So, for *Experiment 5* and *Experiment 7* the input feature vector has the following shapes respectively: (`12,`) and (`25,`).
For *Experiment 2*, *Experiment 4*, *Experiment 6* and *Experiment 8* the RMS feature has been added to the MFCCs feature vector. As specified in *Section 2*, the RMS is root-mean-square (RMS) value for each frame; the RMS feature is computed through Librosa's `librosa.feature.rms` [16] function. For *Experiment 2* and *Experiment 4* we concatenated the RMS values to the feature vectors computed for *Experiment 1* and *Experiment 3* respectively; we ended up having feature vectors with the following shapes resepctively: (`batch_size, 157, 13`) and (`batch_size, 157, 26`).
For *Experiment 6* and *Experiment 8* we computed the average RMS value along all the frames of an audio file and we concatenated this value to the feature vectors previously obtained for *Experiment 5* and *Experiment 6*. We ended up having feature vectors with the following shapes resepctively: (`13,`) and (`26,`).

We need to specify that in reality we always computed 13 or 26 MFCCs, but we have always get rid of the first MFCC component (C_0) (obtaining 12 or 25 MFCCs) since it represents the mean value of the input signal, which carries little signal specific information. [92]

From now on we will identify the single experiments using the following naming convention:

*ID_experiment.ID_dataset*

where *ID_dataset* refers to the identifier shown in Figure 5.6.
For example *Experiment 1.1* has the following characteristics: 12 MFCCs (Experiment 1) + RAVDESS dataset (Dataset 1). Combining all the datasets and all the experiments we ended up testing 96 different models.

### 5.3.7 Data Standardization

After performing feature extraction we scaled data for each of our experiments. We tried both normalization and standardization in *Experiment 1.1* settings and we achieved better performances with data standardization. Standardization is performed by subtracting the mean of a feature from that feature value and then dividing it by the feature's standard deviation. This ensures that the feature values have a mean of zero and has a standard deviation of one. Machine learning algorithms like SVM are sensitive to unscaled data [68].

For each experiment we scaled our data using the `StandardScaler` function provided by scikit-learn [93]. In particular, we fitted the `StandardScaler` on each of our features-training-sets and we transformed the validation and the test set according to that scaler.

## 5.4 Architectures

In this section we will present the characteristics of the architectures used in this project for the Speech Emotion Recognition module. In particular, two approaches were tested: a convolutional model and an SVM classifier. In both cases the objective was the realisation of a binary classifier able to distinguish *disruptive* and *non-disruptive* emotions within a segment of speech. The Convolutional Neural network was implemented using Tensorwflow Keras and the SVM classifier was created through one of its implementations present in scikit-learn [94]. We will first analyse the characteristics of the convolutional model and then of the SVM classifier.

### 5.4.1 Convolutional Model

The model of classification of emotions here proposed is based on a deep learning strategy based on Convolutional Neural Networks (CNN) and Dense layers.[95] The deep neural network designed for the classification task is reported operationally in Table 5.7.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv1d (Conv1D) | (None, 157, 256) | 15616 |
| activation (Activation) | (None, 157, 256) | 0 |
| max_pooling1d (MaxPooling1D) | (None, 39, 256) | 0 |
| dropout (Dropout) | (None, 39, 256) | 0 |
| conv1d_1 (Conv1D) | (None, 39, 128) | 163968 |
| activation_1 (Activation) | (None, 39, 128) | 0 |
| max_pooling1d_1 (MaxPooling1D) | (None, 9, 128) | 0 |
| dropout_1 (Dropout) | (None, 9, 128) | 0 |
| flatten (Flatten) | (None, 1152) | 0 |
| dense (Dense) | (None, 64) | 73792 |
| dense_1 (Dense) | (None, 1) | 65 |
| activation_2 (Activation) | (None, 1) | 0 |

Total params: 253,441
Trainable params: 253,441
Non-trainable params: 0

Table 5.7: Model architecture used for CNN Model

Our CNN network consists of two blocks, each built of a 1-dimensional Convolution layer, Activation function (ReLu), 1-dimensional Pooling layer $4 \times 4$ and Dropout of 60% in the first block and 50% in the second one. The two blocks were followed by two fully connected dense layers and a Sigmoid activation function, as we are dealing with a binary-classification problem.
To determine the class of emotions we rounded the prediction made by the network to the nearest integer, using 0.5 as the threshold value.
The Rectified Linear Unit (ReLu) allows us to obtain a large value in case of activation by applying this function, as a good choice to represent hidden units [45]. Pooling can, in this case, help the model to focus only on principal characteristics of every portion of data, making them invariant by their position [45].
The architecture just described was built starting from the convolutional approaches mentioned in *Section 2.3.2*.
Before obtaining the the model architecture described in Figure 5.7, we have tested the model on *Experiment 1.1* settings, by changing the kernel sizes. After few attempts we manually selected the configuration that lead to best performances.
Then, we tested the model on *Experiment 1.1* settings one time more, by changing the amount of Dropout. After few attempts we manually selected the amount of dropout that led to best perfomances.
In the end, we tested the model another time by changing the number of neurons of the firs Fully Connected Layer. As in the other trials, we manually selected the number of units that led to the best performances.

We built 4 different networks in order to work with the different feature vectors described in *Section 5.3.6*. These networks differ only for the input shape; the models were able to work with (`n_samples, n_mfccs`) input shapes, where `n_samples` is always equal to 157 and `n_mfccs` can be 12,13,25,26 depending on the experiment.
Being a binary classification task, we used Binary Crossentropy as loss function. Moreover, we used Adam as optimizer.

For each of our experiments we performed a Randomized Search, with 3 k-folds, on the following parameters:

- *kernel initializer*: uniform, lecun uniform, glorot uniform, glorot normal, he normal, he uniform
- *bath size*: 4, 8, 16
- *learning rate for Adam Optimizer*: 0.001, 0.0001, 0.00005

To implement the Randomized Search we exploited its scikit-learn implementation [96]. For sake of reproducibility we set the random seed always equal to 7.
After running the Randomized Search, we selected the best configuration of parameters for each experiment; the evaluation was carried out taking into account the *accuracy* metric.
Then, we ran the best model of each experiment using the following Keras callback strategies:

- ***Reduce Learning Rate on Plateau***: reduce learning rate when a metric (accuracy in our case) has stopped improving. We left all the parameters at their default value except for:
  - *monitor*: quantity to be monitored. We set this value to 'training_accuracy' during the *Hypeparameter Optimization* phase and to 'val_accuracy' during the *Training* phase.
  - *factor*: factor by which the learning rate will be reduced. We have set this value to 0.5
  - *patience*: number of epochs with no improvement after which learning rate will be reduced. After some experiments, we have set this value to 4.
  - *mode*: one of 'auto', 'min', 'max'. In 'min' mode, the learning rate will be reduced when the quantity monitored has stopped decreasing; in 'max' mode it will be reduced when the quantity monitored has stopped increasing; in 'auto' mode, the direction is automatically inferred from the name of the monitored quantity. We have set this value to 'max'.
  - *min_lr*: lower bound on the learning rate. After some experiments, we have set this value to 0.000001

- ***Early Stopping***: stop training when a monitored metric has stopped improving. We left all the parameters at their default value except for:
  - *monitor*: quantity to be monitored. We set this value to 'training_loss' during the *Hypeparameter Optimization* phase and to 'val_loss' during the *Training* phase.
  - *patience*: number of epochs with no improvement after which training will be stopped. We have set this value to 45
  - *restore_best_weights*: whether to restore model weights from the epoch with the best value of the monitored quantity. If False, the model weights obtained at the last step of training are used. An epoch will be restored regardless of the performance relative to the baseline. If no epoch improves on baseline, training will run for patience epochs and restore weights from the best epoch in that set. We have set this value to *True*.
- ***Class Weight***: estimate class weights for unbalanced datasets. The use of *Class Weight* was necessary in order to deal with the slight imbalance of classes present in our datasets. The datasets are slightly unbalanced due to the fact that to take into account the gender, the speaker, and the binary division of the lables we have not been able to maintain a perfect balance between the classes. To rebalance the classes after splitting we would have had to use oversampling techniques, which caused overfitting on the model, or undersampling which led to a too significant loss of data, since the datasets were already very limited. For more details about the distribution of classes, please refer to *Section 5.3.2*.

The results and model validation for each experiment will be presented in *Section 5.5*.

### 5.4.2 SVM Model

The second model that we used was SVM. In order to use the SVM model we exploited the implementation provided by scikit-learn [94].
For each of our experiments we performed a Randomized Search with 3 k-folds on the following parameters:

- *kernel*: RBF, linear
- $C$[5]: 0.1, 1, 10, 100

We left all the other parameters at their default values.
After running the Randomized Search, we saved the best configuration of parameters and so the best classifier for each experiment. As for the Convolutional Model, also in this case it was necessary to use *Class Weight*, for the same reasons mentioned in the previous paragraph.
Further details about the models selected will be presented in *Section 5.5*.

[5]Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

## 5.5 Experiments

As previously mentioned in *Section 5.3*, we have optimized, trained and validated 96 different models. These models are the result of the combination of all the datasets we have built (please refer to the *Fig. 5.6* for more details) and all the features we have used (please refer to *Section 5.3.6* for more details).
Here are the two lists of datasets and features used, for which the number in the number-list corresponds to the number used to build the experiment identifier.
Datasets:

1. RAVDESS
2. RAVDESS_noise
3. TESS
4. TESS_noise
5. SAVEE
6. SAVEE_noise
7. CREMA
8. CREMA_noise
9. RAVDESS_TESS_SAVEE
10. RAVDESS_TESS_SAVEE_noise
11. RAVDESS_TESS_SAVEE_CREMA
12. RAVDESS_TESS_SAVEE_CREMA_noise

Features:

1. 12 MFCCs
2. 12 MFCCs + RMS energy
3. 25 MFCCs

4. 25 MFCCs + RMS energy
5. 12 MFCCs (mean along audio frames)
6. 12 MFCCs (mean along audio frames) + RMS (mean along audio frames)
7. 25 MFCCs (mean along audio frames)
8. 25 MFCCs mean along audio frames) + RMS mean along audio frames)

Architectures:

- Convolutional Neural Network
- SVM

So, recalling what has been stated in *Section 5.3.6*, the single experiments are identified using the following naming convention:

*ID_experiment.ID_dataset*

where *ID_dataset* refers to the identifier shown in Figure 5.6.
All experiments and introduced datasets are available at `https://github.com/helemanc/ambient-intelligence`.
It must be specified that, as stated in *Section 5.3.6*, features 1 to 4 were used for the Convolutional Model, while features 5 to 8 were used for the SVM model.
As already mentioned in *Section 5.4*, for each of the models we have performed an optimization phase of some parameters and hyperparameters and we have saved the best configuration for each model.
The best model for each experiment was re-trained on the experiment reference training set and was subsequently tested on the experiment reference test set. What has just been said means that, for example, the model for *Experiment 1.1*, which refers to the RAVDESS dataset, was tested on the test set extracted from RAVDESS, while the model for *Experiment 1.9*, which refers to the combination of the RAVDESS-SAVEE-TESS datasets was tested on a test set which is the concatenation of the three respective test sets.
For each experiment, we proceeded to save the learning curves and the classification report in order to analyze the results obtained and to be able to select the best models

to be used later in the model ensemble.

It should be noted that, in order to obtain the predictions of the Convolutional model, and therefore to be able to obtain results in terms of metrics for the comparison of the models, the output value[6] of the network has been rounded using 0.5 as threshold value. This means that all the predictions of the network with a value greater than or equal to 0.5 have been rounded to 1 (label corresponding to the *disruptive class*); vice versa for the other class. Different strategies for the treatment of predictions were tested later, in the validation phase of the ensemble (*Section 5.7*). Undoubtedly, it would be interesting to repeat the experiments and try different threshold values. These considerations are not necessary for SVM, which by its nature does not work with and does not return probability values; so, for the evaluation of the single models we exploited the predictions computed through the `predict` method of the scikit-learn SVM implementation. Later, for the evaluation of the aggregation strategies of the ensemble, we will use another approach that allows us to work with the probabilities associated by the SVM classifier to each of the two classes. Please refer to *Section 5.7* for further details.

In order to reuse the best models within the ensemble, it was necessary to save the different data-scalers trained on the training sets of each experiment. Please, refer to *Section 5.3.7* for further details about Data Standardization.

In the next section we will show the results of the experiments.

[6]Result of the application of a sigmoid function

## 5.6 Results

In this section we are going to discuss our results.
Comparisons are based on the following indicators:

- ***Learning Curve***: we exploited the observation of the learning curve in order to evaluate the amount of overfitting of the model on the training data.
- ***Accuracy***: we used this metric only to evaluate the models of the experiments for which the test sets are class-balanced. Accuracy is defined as follows: $\frac{TP+TN}{TP+FP+TN+FN}$
- ***F1 score***: we used this metric to evaluate all models and especially those with unbalanced test sets.
  F1 score is defined as the harmonic mean of Precision and Recall. Before giving the formula to compute F1 score, let's define what Precision and Recall are:
  - Precision: $\frac{TP}{TP+FP}$ of each class
  - Recall: $\frac{TP}{TP+FN}$ of each class

  Then, let's define F1 score:
  - F1 score: $\frac{2 \cdot P \cdot R}{P+R}$ where P stands for Precision and R for Recall

  Since the F1 score is an average of Precision and Recall, it means that the F1 score gives equal weight to Precision and Recall:
  - A model will obtain a high F1 score if both Precision and Recall are high
  - A model will obtain a low F1 score if both Precision and Recall are low
  - A model will obtain a medium F1 score if one of Precision and Recall is low and the other is high

  F1 score allows us to make interesting assessments, since in our system we prefer that the amount of *false positive*s (i.e. *non-disruptive* situations classified as *disruptive*) is greater than the amount of false negatives (i.e. *non-disruptive* situations classified as *disruptive*). In particular, in our context it is better that there are false alarms rather than undetected alarms. Therefore, for the purposes of

the assessment, we took into account the F1 score on the positive class (i.e. 1 = *disruptive*). High F1 score values on the positive class, in all cases in which it is not possible to use accuracy, are a symptom, for us, of the goodness of the model.

We are going to list the characteristics of the models that, in our opinion, have led to the best performances. They are 19 models in total, out of 96.
Table 5.8 reports the experiment identifier, the type of the classifier and the performance of the models in terms of accuracy and F1 score. The table containing the results of all models can be found in *Appendix A.3*.

| **Experiment ID** | **Classifier** | **Accuracy** | **F1 score (positive class - 1)** | **F1 score (negative class - 0)** |
|---|---|---|---|---|
| 1.1 | CNN | 0.88 | 0.88 | 0.87 |
| 1.2 | CNN | 0.85 | 0.85 | 0.85 |
| 1.6 | CNN | 0.85 | 0.85 | 0.85 |
| 1.8 | CNN | 0.87 | 0.87 | 0.67 |
| 2.1 | CNN | 0.82 | 0.81 | 0.82 |
| 2.2 | CNN | 0.80 | 0.79 | 0.81 |
| 2.8 | CNN | 0.87 | 0.92 | 0.64 |
| 2.10 | CNN | 0.85 | 0.85 | 0.86 |
| 2.12 | CNN | 0.73 | 0.75 | 0.70 |
| 3.1 | CNN | 0.89 | 0.90 | 0.89 |
| 3.2 | CNN | 0.90 | 0.91 | 0.89 |
| 3.12 | CNN | 0.70 | 0.74 | 0.64 |
| 4.1 | CNN | 0.89 | 0.90 | 0.89 |
| 4.2 | CNN | 0.85 | 0.85 | 0.85 |
| 6.8 | SVM | 0.83 | 0.90 | 0.58 |
| 7.8 | SVM | 0.82 | 0.89 | 0.42 |
| 7.10 | SVM | 0.85 | 0.91 | 0.53 |
| 8.8 | SVM | 0.85 | 0.91 | 0.53 |
| 8.10 | SVM | 0.88 | 0.88 | 0.87 |

Table 5.8: Results achieved by the models that showed best performance among all the experiments; they are 19 out of 96. Models have been evaluated in terms of a accuracy and F1 score. In green, the CNN and SVM model that that showed best results among all CNN and SVM models respectively.

In the table above we have also highlighted the best convolutional model and the best SVM model: 3.2 and 8.10 respectively.
From the learning curves of the convolutional models in *Appendix A.3*, it emerges that in all cases there is an overfitting problem. We show below the learning curve of model 3.2 as an example:

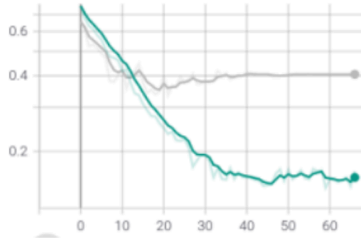


Figure 5.7: Learning curve for *Experiment 3.2*

We believe that this problem can only be solved by working on the architecture itself. In particular, we believe that this may be the only solution to the problem because:

- Data Augmentation has already been implemented by extending the datasets with noisy files
- The unbalancing of the classes occurred after the division of the dataset by gender is managed through the `class_weight` in the training phase
  - Undersampling was avoided in order not to reduce the size of the dataset too much.
  - Oversampling has been avoided in order not to generate overfitting since the audio files do not differ too much from each other

Furthermore, a careful observation of the results obtained allows us to conclude that:

- **Experiment 1.1**: the training dataset is quiet balanced and the model is robust since it is gender-independent and speaker-independent (thanks to the strategy followed for the construction of the dataset).
- **Experiment 2.2**: the training dataset is quiet balanced and the model is robust since it is gender-independent and speaker-independent. This model is more robust than *Experiment 1.1*, since the training set contains noisy file too. So, this model is more robust to noise than the one of *Experiment 1.1*.

- **Experiment 1.6**: the dataset is quite balanced and the model is quite robust since is speaker independent, but we cannot conclude if the model is gender-independent or not because the training set and the test set contain only male actors.

- **Experiment 1.8**: the training is not balanced; the elements fo class 0 are a way less than elements of class 1 (50 vs 25). So it is not surprising that the model classifies correctly element of class 1 and uncorrectly elements of class 0. We can think to use this model in the ensemble because the F1 score for the positive class is high, and as was stated above, we prefer having more false positives than false negatives.

- **Experiment 2.1**: the considerations that we can make about this model are not so distant from what we stated for *Experiment 1.1*. Comparing this model with the one of *Experiment 1.1* we can say that the energy feature is misleading for the purpose of classification, because performances of *Experiment 2.1* are worse than performances of *Experiment 1.1*.

- **Experiment 2.2**: the considerations that we can make about this model are not so distant from what we stated for *Experiment 1.2*. Comparing this model with the one of *Experiment 1.2* we can say that the energy feature is misleading for the purpose of classification, because performances of *Experiment 2.2* are worse than performances of *Experiment 1.2*.

- **Experiment 2.8**: for this model the same considerations made for *Experiment 1.8* hold; this means that **by adding the energy (RMS) feature we are not adding a useful source of information for the purpose of classification**.

- **Experiment 2.10**: This is one of the **more robust models** since we use 3 different datasets for training; so, we can conclude that this model is quite robust w.r.t. gender, speaker and noise.

- **Experiment 2.12**: the accuracy of the model is lower than *Experiment 2.11* because in the training set we are mixing files which have a very different audio-quality. Despite the low accuracy, we can still say that the model learns something, so it could be interesting to not exclude it in the ensemble; it could be useful having it and see how it weights the overall decision in the ensemble.

- **Experiment 3.1**: for this model the same considerations made for *Experiment 1.1* hold. In addition to that, we can observe that **using 25 MFCCs as input features the perfomances are better than with 12 MFCCs**; so, maybe, in this case the use of a greater amount of MFCC components help the classifier to achieve its task.

- **Experiment 3.2**: for this model the same considerations made for *Experiment 1.2* hold. In addition to that, we can observe that **using 25 MFCCs as input features the perfomances are better than with 12 MFCCs**; so, maybe, in this case the use of a greater amount of MFCC components help the classifier to achieve its task. Moreover, this is the **best convolutional model** in terms of F1 score and accuracy.

- **Experiment 3.12**: for this model the same considerations made for *Experiment 2.12* hold.

- **Experiment 4.1**: the performances of this model are better than those of its mirror convolutional model, i.e. *Experiment 1.1*. This let us think in this case that the use of a greater number of MFCC components represents much more significant information for the network than using a smaller number of components. This trend could allow us to conclude that, for the purpose of solving our problem, **the use of a higher number of features than the standard (13 MFCCs) is more adequate**.

- **Experiment 4.2**: the performances of this model are quite comparable to those of *Experiment 1.1*, so the same considerations hold.

- **Experiment 6.8**: the test set on which the analysis was carried out is rather unbalanced, as is the training set. We decided to keep this model simply because of the high F1 score for the positive class, but surely all convolutional models mentioned so far can be considered more reliable than this one.

- **Experiment 7.8**: the same considerations of *Experiment 6.8* hold.

- **Experiment 7.10**: this model is definitely **one of the best** in terms of F1 score on the positive class. The architecture of the classifier (SVM) is certainly lighter

and faster than a convolutional model. The performances are decidedly good and, moreover, we can consider this model robust towards noise, gender and speaker, given the composition of the training set of the reference experiment.

- **Experiment 8.8**: as for all the other models trained on the CREMA dataset, the same considerations apply.
- **Experiment 8.10**: this is the **best SVM model**. The same considerations of *Experiment 7.10* hold in this case. Therefore, we can say that for the SVM model we do not notice any difference as the features used vary.

In conclusion, what emerges from the results of the experiments is that:

- **Features**:
  - for the convolutional model, using 25 MFCCs features instead of 12 leads in any case to better performance.
  - both for the convolutional model and for the SVM model, the best performances are obtained without using the RMS feature. This means that the use of RMS does not add significant information to the classification task and can therefore be eliminated.
- **Best Models**:
  - Models 3.2 and 8.10 are the best two convolutional and SVM models respectively. Both are robust to noise having been trained on noise-affected files. The 8.10 model, however, could have greater generalization capabilities, having been trained on the union of 3 different datasets (RAVDESS, TESS, SAVEE).
- **Models Hyperparameters**:
  - As it emerges from the results reported in the *Appendix A.3*, it is not possible to identify a valid trend for all convolutional models in terms of model and training hyperparameters. On the other hand, it is possible to conclude that, as regards the SVM model, in any case the use of the *rbf kernel* was successful.

- **Datasets**:
  - Among the experiments that led to better performance, there is no model that has been trained on the union of the 4 datasets, but only at most on the union of RAVDESS, SAVEE and TESS. This confirms that CREMA is an unreliable dataset for experimentation, despite the meticulous data preparation work that has been carried out.

## 5.7 Ensemble of Best Models

The best models mentioned in the previous section were combined using *ensemble learning*. In supervised machine learning, ensemble learning is the procedure of combining multiple machine learning algorithms to classify the same data. In particular, after generating a set of base learners, rather than trying to find the best single learner, ensemble methods resort to combination to achieve a strong generalization ability, where the combination method plays a crucial role [97].
What we intend to do in the ensemble is to pass the audio segment entering the system as input to all classifiers; at this point, collect the predictions made by the individual models and use a strategy to combine the predictions.
There are different strategies that can be followed when combining predictions:

- **Voting**: voting is the most popular and fundamental combination method for nominal outputs [97]. Voting generally involves each model that makes a prediction assigning a vote for the class that was predicted. There exist many types of voting, such as:
  - *plurality voting*: it selects the class label with the most votes [98]. In the case of a binary classifier, this means that if more models predict class 1, then class 1 will be associated with the input data.
  - *majority voting*: selects the class label that has more than half the votes [98]. If no class has more than half the votes, then a "no prediction" is made.
  - *unanimous voting*: is related to majority voting in that instead of requiring half the votes, the method requires all models to predict the same value, otherwise, no prediction is made.
  - *weighted voting*: weighs the prediction made by each model in some way [98]. One example would be to weigh predictions based on the average performance of the model, such as classification accuracy (i.e. the weight of each classifier can be set proportional to its accuracy performance on a validation set [99]). Assigning weights to classifiers can become a project in and of itself and could involve using an optimization algorithm and a holdout dataset, a linear model, or even another machine learning model entirely [98]. The idea of weighted

voting is that some classifiers are more likely to be accurate than others and we should reward them by giving them a larger share of the votes [98].

For the purpose of this thesis we have decided to use *plurality voting*.

- **Combine Predicted Class Probabilities**: probabilities summarize the likelihood of an event as a numerical value between 0.0 and 1.0. When predicted for class membership, it involves a probability assigned for each class. The most common approach is to use voting, where the predicted probabilities represent the vote made by each model for each class. Votes are then summed and a voting method from the previous section can be used, such as selecting the label with the largest summed probabilities or the largest mean probability. As in the case of voting, there are several ways to deal with the probabilities assigned by the models to each class:

  - *Vote Using Mean Probabilities*: votes are summed and the label with the largest mean probability is selected.
  - *Vote Using Sum Probabilities*: votes are summed and the label with the largest summed probabilities is selected.
  - *Vote Using Weighted Sum Probabilities*: a weighted sum of the votes is computed and the label with the largest summed probabilities is selected.

  For the purpose of this thesis we have decided to use four strategies to deal with probabilities. These four strategies are variations of "voting using mean probabilities". We will analyze them in details in the next section.

After this overview of the strategies that can be used to aggregate the predictions of the individual classifiers, we are going to describe the customized-strategies we have built. Next we will present the details of the ensemble validation.

### 5.7.1 Proposed Aggregation Strategies

In order to combine the predictions of our models we decided to use variations of the strategies presented in the previous paragraph.
As previously mentioned, we have experimented with two plurality voting strategies and four voting strategies based on the use of probabilities.
We provide a detailed description below.

#### Customized Voting Strategy

In order to realize this strategy we exploited the predictions of the convolutional and SVM models. While the SVM output corresponds directly to the predicted class (0 or 1), the convolutional model output is provided by a sigmoid neuron and it is a value between 0 and 1, that can be interpreted in terms of probability.
Therefore, in order to obtain a class value predicted by the convolutional network, and so to be able to use the voting scheme, we experimented with two different threshold values: 0.5 and 0.7. If the value supplied in output by the convolutional network is above the threshold value, the predicted label is 1, otherwise 0.
After obtaining both the predictions of the convolutional model and the SVM model, the final prediction of the ensemble is calculated through the *plurality voting* strategy. Therefore, if the number of predictions referring to the 1 class is greater than the number of predictions referring to the 0 class, the final prediction will correspond to the 1 class, 0 otherwise.

#### Customized Voting Using Mean Probabilities Strategies

Unlike what was said in the previous section, in this case we find ourselves in the opposite situation: the SVM model provides us with a numerical prediction corresponding to the reference class, but we need a confidence measure that can be interpreted in terms of probability. As mentioned previously, convolutional models themselves provide a value that can be interpreted as a probability. Instead, we need to make some considerations for SVM models.
In order to obtain values similar to probability we cannot use the `decision_function` method provided by scikit-learn for the SVM classifier, because this method computes

the distance from the boundary, but it is not the same as computing the probability that a given datapoint belongs to a particular class. To do that, we need to use `predict_proba`. This method computes the probability that a given datapoint belongs to a particular class using Platt scaling [100]. Platt scaling computes the probabilities using the following method:

$$P(class/input) = \frac{1}{1 + exp(A * f(input) + B)} \tag{5.3}$$

where $P(class/input)$ is the probability that "input" belongs to "class" and $f(input)$ is the signed distance of the input datapoint from the boundary, which is basically the output of `decision_function`.

To apply Platt scaling to SVM we need to train the SVM as usual and then optimize the parameters A and B. To optimize A and B we need to train a probability model on top of our SVM. Also, to avoid overfitting, Platt scaling uses n-fold cross validation.

So, this is a lot more expensive than training a non-probabilistic SVM.

What we obtain by applying the `predict_proba` method on the SVM classifier is a tuple of two values, each of which representing the confidence score assigned by the classifier to class 0 and class 1 respectively.

To enable the use of Platt scaling, we simply set the parameter `probability = True` in the scikit-learn SVM classifier. Therefore, in order to obtain proability values from our SVM classifiers we had to re-train the best models allowing the use of Platt scaling in the training phase.

We have noticed that classifiers provide the same result through the `predict` method whether trained with Platt scaling enabled or not. Therefore, all SVM model considerations reported in *Section 5.6* still hold.

An important thing to point out is that the value predicted by the classifier through the calculation of probabilities may not match the value associated by the classifier through the `predict` method. As reported in the scikit-learn documentation "the probability estimates may be inconsistent with the scores, in the sense that the argmax of the scores may not be the argmax of the probabilities (e.g., in binary classification, a sample may be labeled by predict as belonging to a class that has probability $\leq 0.5$ according to `predict_proba`). Platt's method is also known to have theoretical issues". Apart from Platt scaling, there exist also other strategies for obtaining probability values from an SVM classifier, but we will mention them in *Section 7*.

After this introduction on the method used to obtain probability values from SVM classifiers, we proceed with the description of the aggregation strategies that have been developed. To test the effect of using Platt scaling on the results we have created two different strategies:

- **Average 1**: in the first case, we computed the average prediction obtained through the convolutional models and we added it to the SVM predictions obtained through the `predict` method. Then, we computed the average value of this new list of predictions and we associated a class label to the average probability using a threshold value.
- **Average 2**: in the second case we computed the average prediction considering all the outputs of the convolutional models and all the outputs of the SVM models, obtained through the `predict_proba` method. Also in this case, we associated a class label to the average probability using a threshold value.

We experimented the use of two thresholds: 0.5 and 0.7. These threshold values were chosen arbitrarily. It would be interesting to use statistical methods to calculate the ideal threshold value, but we will talk about it in *Section 7*.

### 5.7.2 Validation

In this section we will describe the ensemble evaluation process starting from the datasets chosen to carry out the validation.

#### Datasets

In order to carry out the evaluation of the different aggregation strategies for the ensemble, two different datasets have been used: RAVDESS and CREMA.
RAVDESS and CREMA are the only two datasets in which we can find different speakers of different genders.
In particular, for each dataset we took the test set created in the *Data Preparation* phase. Between the two, we consider the evaluation performed on RAVDESS to be more reliable since, as previously described, the process of building the dataset was extremely precise and the audio files have a much higher quality than the CREMA audio files.
While the RAVDESS test set is fairly class-balanced, the CREMA test set is not. This is not important for the purposes of the evaluation since the metric used was in any case the F1-score on the positive class (for positive class we intend class 1).

Obviously, the validation performed on a dataset containing noise-free audio files allows us to obtain very optimistic evaluation with respect to the use of the model in the real use case. It would therefore be advisable to repeat all the tests using a dataset constructed in the reference context or augmenting the current available audio files through noise recorded from inside a train. We will return to these considerations in the following chapters.

#### Results

In this section we show the results of the ensemble validation. In particular, we tested the strategies described in (5.7.1) and (5.7.1) on the datasets described in 5.7.2.
As previously described, the strategies are compared on the basis of the F1-score obtained on the positive class and the objectives of this analysis are:

1. understand if there is a strategy which showed better results, compared to the others

2. have proof of the fact that the ensemble's generalization capacity is not sufficient to manage CREMA's data (known to be of a distinctly different quality from that of RAVDESS)

3. understand what are the models that we could exclude from the ensemble, looking at the models that have obtained a large amount of false negatives and few false positives

4. understand if the candidate models for exclusion from the ensemble are the same ones that we would eliminate by looking at both the results obtained on RAVDESS and the results obtained on CREMA

Table 5.9 shows the results that the ensemble obtained on the test set of RAVDESS dataset.

| **Strategy** | **Threshold** | **F1-score (positive class)** |
|---|---|---|
| **Voting** | 0.5 | 0.85 |
| **Voting** | 0.7 | 0.74 |
| **Average 1** | 0.5 | 0.57 |
| **Average 1** | 0.7 | 0.35 |
| **Average 2** | 0.5 | 0.90 |
| **Average 2** | 0.7 | 0.74 |

Table 5.9: Results of the validation of the ensemble on RAVDESS dataset

As we can see from Table 5.9, the strategy that achieved the best result is the one we called **Average 2**. Moreover, a trend that can be observed from the table is that for each strategy the worst results were obtained through the use of the threshold value of 0.7; this let us think that good threshold values are those which are less than 0.7.
The trend observed on the threshold values makes sense since the single models of the ensemble overfit a little bit, and so they have little confidence when predicting. This trend can also be useful to continue with future studies on the choice of the most appropriate threshold value, reducing the range of testable values.
With **Average 2** strategy, the performances in terms of F1-score (positive class) are very good and we can add that, in this experiment, the F1-score for the negative class was also very high (0.89). Since the RAVDESS test set quite balanced (56 class 0 files and 64 class 1 files) we also looked at the accuracy which in this case is 0.89.
We can therefore conclude that the ensemble has achieved excellent results on RAVDESS.

The results obtained are comparable to those of the state-of-the-art.

Figure 5.9 shows the models that contributed to more than 50% of all the False Positives and False Negatives, obtained with each of the strategies.
Histograms in Figure 5.9 and Figure 5.10 were constructed through the analysis of graphs like the one in Figure 5.8:

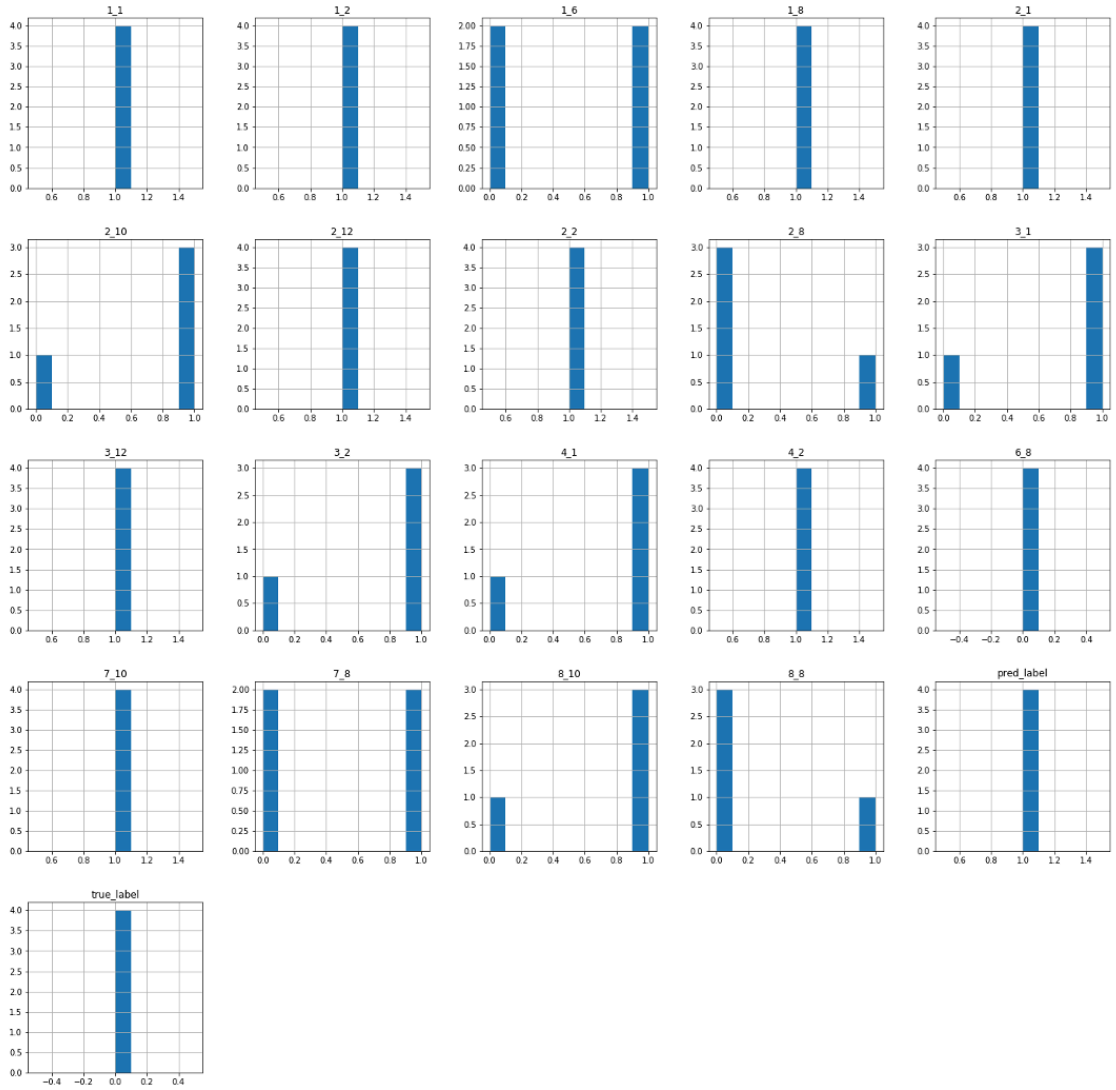


Figure 5.8: Distribution of False Positives predictions per model. Ensemble Configuration: Voting Strategy - Threhsold 0.5 - RAVDESS. On the y-axis there is the count of False Positives that have been retrieved by a single model through the use of the Ensemble configuration just mentioned. On the x-axis there are the predictions of a model. The name of the model is reported in the top-part of each histogram.

We showed only one figure (Fig. 5.8) as an example.

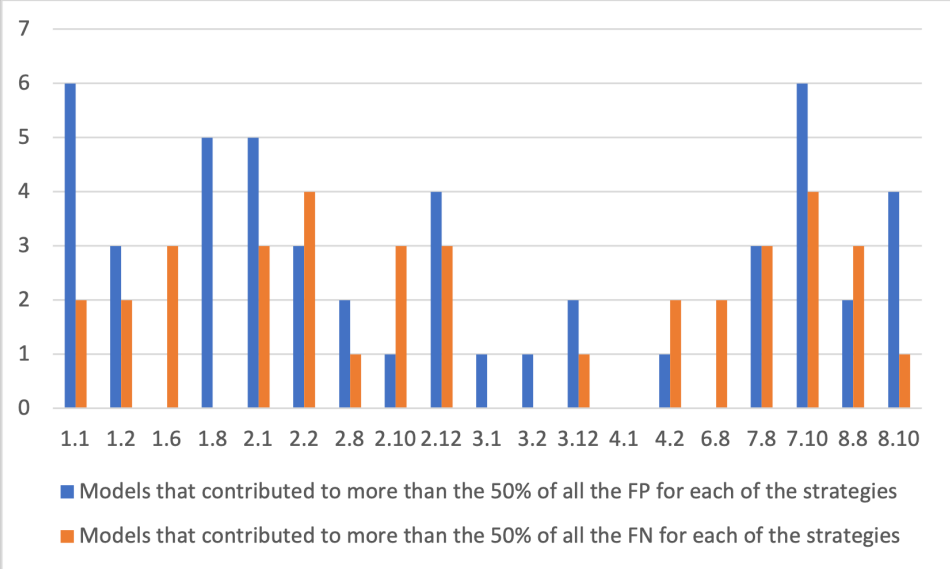


Figure 5.9: Models which contributed to more than the 50% of the total amount of False Positives and False Negatives for each of the strategies - RAVDESS. On the y-axis there is the count of the strategies; 6 is the maximum since we used 6 different aggregation strategies. On the x-axis there are the models.

As can be noted from Figure 5.9, the models that contributed to the generation of the greatest amount of false negatives are: 1.6, 2.2, 6.8, 7.10. In particular, models 1.6 and 6.8, among all the misclassified files, contributed to generate only false negative results (in 3 and 2 strategies out of 6 respectively). This let us think that these 2 models could be excluded from the ensemble, since the quantity that we want to minimize is the amount of false negatives. As regards models 2.2 and 7.10, we cannot exclude them since these models have also contributed to the generation of a fair amount of false positives; therefore if our criterion is to minimize false negatives and prefer false positives to false negatives, we cannot proceed with the elimination of these models.
We now focus on the results obtained on CREMA dataset.

Table 5.10 shows the results that the ensemble obtained on the test set of CREMA dataset.

| Strategy | Threshold | F1-score (positive class) |
|---|---|---|
| **Voting** | 0.5 | 0.61 |
| **Voting** | 0.7 | 0.44 |
| **Average 1** | 0.5 | 0.87 |
| **Average 1** | 0.7 | 0.80 |
| **Average 2** | 0.5 | 0.25 |
| **Average 2** | 0.7 | 0.34 |

Table 5.10: Results of the validation of the ensemble on CREMA dataset

As we can see in Table 5.10 the strategy which obtained best results is **Average 1**. Differently from what we observed with RAVDESS, we cannot conclude a lot about the trend of the threshold values. It would have been interesting if we had been able to observe that the winning strategy was the same on both datasets and with the same trend for the threshold values. Unfortunately this is not the case and, moreover, since CREMA is an unreliable dataset, we cannot conclude much from these results.
Figure 5.10 shows the models that contributed to more than 50% of all the False Positives and False Negatives obtained with each of the strategies.

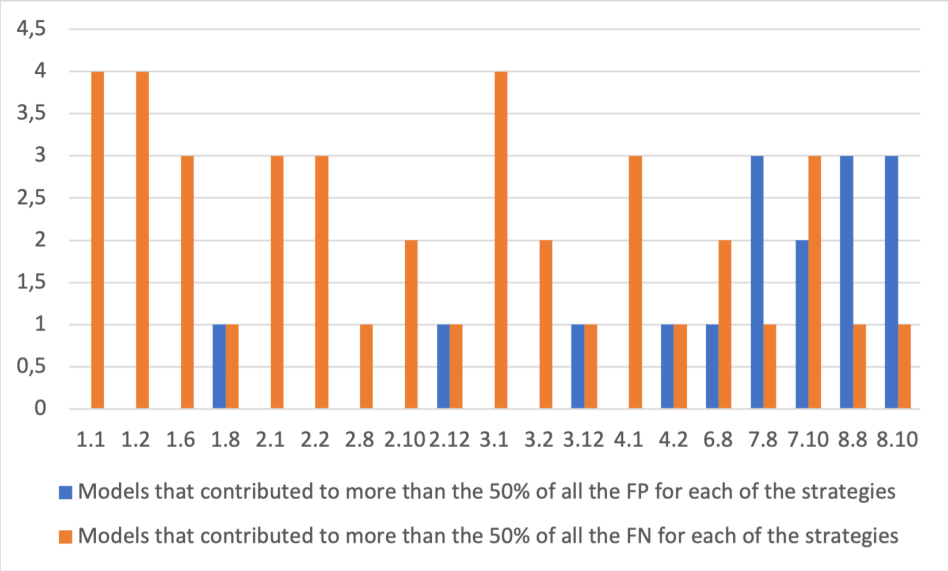


Figure 5.10: Models which contributed to more than the 50% of the total amount of False Positives and False Negatives for each of the strategies - CREMA. On the y-axis there is the count of the strategies; 6 is the maximum since we used 6 different aggregation strategies. On the x-axis there are the models.

As can be noted from the figure above, the models that contributed to the generation of the greatest amount of false negatives are: 1.1, 1.2, 1.6, 2.1, 2.2, 3.1, 4.1.
From the observation of these results, we could think of excluding all these models from the ensemble, for the same reasons mentioned during the analysis of the results obtained on the RAVDESS dataset. In reality, considering the unreliability of CREMA, it would be better to evaluate the exclusion only of the models that have also been candidates for RAVDESS; in particular, the candidate models for the exclusion, common to both datasets, are: 1.6 and 2.2.

We can conclude that the ensemble has achieved very satisfactory results on the RAVDESS dataset, which is the most reliable and widely used dataset for Speech Emotion Recognition research. Certainly, the research carried out in this thesis still provides ample room for improvement and open issues, especially regarding the application of such architectures in real scenario. We will discuss all this more in depth in *Chapter 6*.

# Chapter 6

# Discussion

In this chapter we will discuss the challenges encountered while building the application and the possible margins of improvement.

## 6.1 Integration

All the modules explained before have been realized from scratch or taken from packages available off the shelf; this has required a considerable effort of integration and engineering. The result is a stand-alone package described in *Appendix A.5*.

#### Limitations

The integration with the microphone that will be used in the real scenario is totally missing. It has been outlined what needs to be done to process incoming audio in streaming mode and which communication protocol need to be used, but both functionalities still need to be developed. Therefore, the application is not yet ready to be used in the real case scenario.

Moreover, the biggest problem of the system is the total execution time. As stated in *Appendix A.5*, the SER system lasts around 10 seconds for each prediction and it would be not possible having this execution time when processing streaming audio.
This problem is mainly related to the fact that the models are instantiated sequentially and not in parallel.

### Future Developments

The next step for the development of this application will concern the passage from a sequential application to a multi-threaded one, to ensure that all the models make the prediction simultaneously and that the loading of the single models is done only once during the life cycle of the application.
This would require creating a thread for each model and only terminating the threads when the application is closed.
It was not possible to implement this functionality due to the computational resources available during the integration phase. However, based on our initial analysis, the infrastructure that will host the deployed system has the necessary computing power to allow the use of a multi-thread application.
Among the possible improvements to our model, we can identify several future developments that would change the structure of the architecture, by adding new features such as Keyword Spotting (KWS), Speech-To-Text (STT), Automatic Hate Detection and Environmental Noise Classification.
Let's see in what purposes it would be useful to develop these features:

- **KWS and Speech-To-Text**: Keyword Spotting models can be useful for emergency word detection such *help*, *aiuto*, *ayudame*. These models usually compute the spectogram of fragments of audio and classify them between several labels, such as a *specific keyword* or *unknown*. Most common approaches to perform KWS rely on Convolutional Neural Networks. There have been also promising advances in this field by using encoder-decoder architectures such as the Transformer [101], which has led to significant improvements in several NLP tasks including Machine Translation and Language Modeling. Some of these works focus on adapting the Transformer architecture to the KWS problem [102]. It would also be interesting to use a Speech To Text model in parallel to the KWS model to transcribe the speech fragments and analyze if, in these fragments, there are words that match one of the emergency keywords. Moreover, it would be possible to compare the results obtained from both approaches on the same speech fragment to evaluate the effective presence of emergency keywords in the incoming audio. The most prominent models for offline Speech-To-Text to date are Vosk API (Python) and Deep Speech. Although Deep Speech is more famous than Vosk, we have had

the opportunity to experience that the Vosk model (vosk-model-small-en-us-0.15) works much better than Deep Speech (model 0.9.3) [103] in terms of execution time, accuracy and integration with edge systems. The Vosk model is so light (40M) that it can run even on a Raspberry Pi.

- **Automatic Hate Speech Detection**: for the identification of disruptive situations, it would be interesting to integrate an Automatic Hate Speech Detection (AHSD) system, on the idea of those presented in [104]. All the solutions presented in [104] refer to textual content analysis. Therefore we could think of devising a pipeline consisting of a Speech-To-Text system followed by a model for AHSD to analyze the transcribed sentence.

- **Environmental Noise Classification**: all the solutions developed within this thesis project, and presented so far among future developments, refer to speech analysis. Instead, it would be interesting to build a system to also classify the noise audio segments, selected by the VAD system. In particular, we could think of considering disruptive situations in the event that the noise is classified, for example, as a ”traffic accident” or ”gunshot”. There are several research works in this field, and the main datasets used for this purpose are [105] and [106]. Both of them are public and free to use.

The proposed improvements could be integrated in an extended architecture such as the one shown in Figure 6.2 (Figure 6.1 shows the current architecture for comparison).

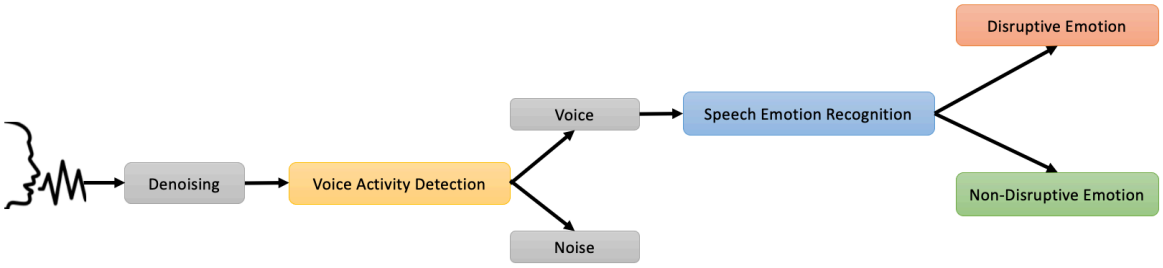


Figure 6.1: Proposed Architecture

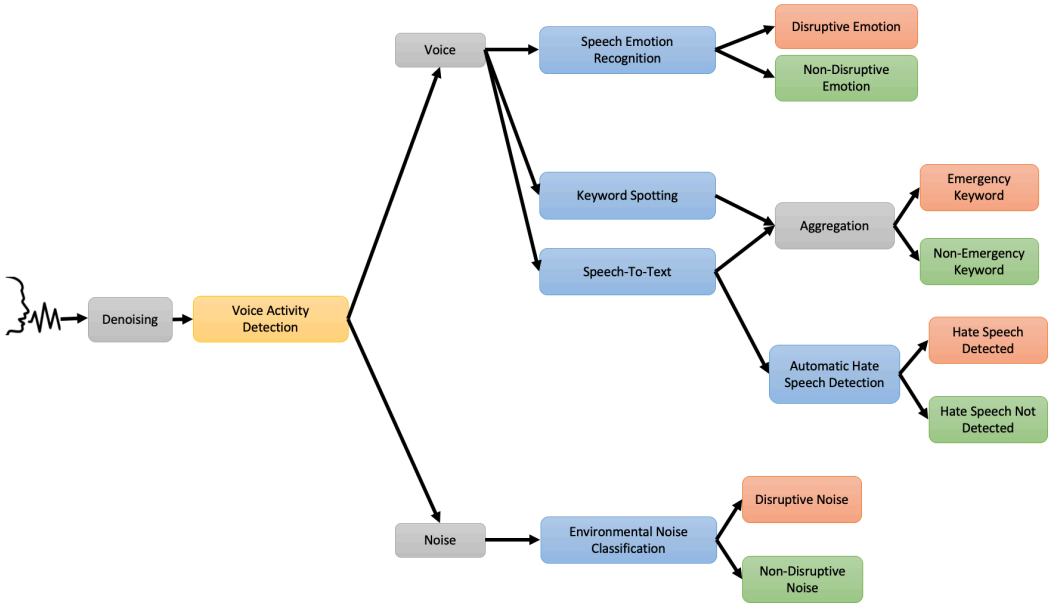


Figure 6.2: Extension of the Proposed Architecture

## 6.2 Speech Emotion Recognition

Moving now the focus from the final application to the individual components, we start with the discussion of the Speech Emotion Recognition system. As well as for the final application, here too it is possible to list the difficulties encountered, the critical issues and the possible margins for improvement.

### Limitations

As mentioned at the end of *Chapter 5*, the ensemble has achieved very satisfactory results on the RAVDESS dataset, but certainly the research carried out in this thesis still provides ample room for improvement and open issues, especially regarding the application of such architectures in the real scenario.
The main challenges encountered during the construction of the SER system are:

- **Lack of reliable and reproducible baselines**: as previously mentioned in *Section 2.3.2*, most of the research works present nowadays do not contain a complete description of the work done, in order to be completely reproducible. In most cases there is a lack of information on how the data was handled or detailed information on the characteristics of the classifier or training details. Even the lack of just one of the information just mentioned makes it impossible to faithfully reproduce the model described in the research work, thus preventing it from possible improvements. Therefore, in order to build baselines for this thesis project we took inspiration from different works.

- **Lack of real data**: in order to be able to conduct a non-optimistic assessment of the architectures it would be ideal to have a dataset of sounds recorded within the reference context (i.e. trains on which the architecture will be installed). Having those data we might either repeat the training of models or review the data augmentation process; as for the last thing, it would be interesting to use the ambient noise present in real recordings instead of the audio files used for this thesis project. This could help us in setting the correct SNR level while augmenting the clean datasets with the noisy sounds.

- **Choice of features**: many works use completely different features, therefore to date it is not clear if there is a favorite feature among those used by the various authors. It would seem that the most used features are MFCC and RMS, but both have different criticalities [17]. Therefore it would be certainly interesting repeating the experiments with features different from MFCCs and RMS, to see if using or combining other features the performance of the models improve.

- **Construction of a binary classifier for SER**: as stated in *Section 5.3*, in order to encode labels on a binary basis we need to make a strong assumption and this assumption may introduce noise in the classification task. Almost all the works in the literature, as mentioned in *Section 2.3.2*, use classifiers to solve multi-class classification problems; all those works are based on a categorical division of emotions that resides on the study of an expert team of scholars. To binary codify emotions we did not have a team of experts at our disposal, therefore it would be better to ask for an external opinion to validate the choices made while encoding the emotion labels.

**Future Developments**

The possible margins for improvement of the SER architecture are:

- **Change of data splitting for CREMA dataset**: as mentioned in *Section 5.3* we made a small mistake when splitting CREMA dataset. Since we have decided to only keep files characterized by a *high level* of emotional intensity, we have automatically excluded the ”neutral” class, since, by its nature, it does not exist at different levels of intensity. It would therefore be advisable to repeat the division of the dataset by including the neutral class. In this way we would obtain a more class-balanced dataset.

- **Build Noisy Test Sets**: in order to test the generalization capabilities of the model in a noisy environment, it would be interesting to apply the data augmentation process described in the *Section 5.3* also on the test sets of the individual datasets.

- **Find reliable datasets in languages other than North American English**: to study the generalization capabilities of the model and/or build a cross-linguistic model, as in [51], it would be useful to obtain / build reliable datasets also in other languages. Having a cross-linguistic model in the context of public transports could be really important. We have seen that EMO-DB can be used as far as the German language is concerned, but it is a dataset made up of very few samples. CREMA could be useful because the actors, although they all are english speakers, are of different ethnicities and consequently have different accents, but we have found that it is a completely unreliable dataset and of not high quality.

- **Try the use of delta and delta-delta MFCCs**: although not widely used in the literature, it would be interesting to see if the addition of information about the first and second derivative of the MFCC coefficients brings improvements in terms of performance.

- **Change of the Convolutional Model**: as mentioned in the previous chapters, we focused on the study of different models as the dataset and the features used changed, but we never changed the parameters of the convolutional model layers after an initial assessment. It would be interesting to conduct an in-depth analysis of the architecture by trying for example to change the kernel size of the convolutional layers or the number of neurons of the fully connected layers.

- **Try other Neural Architectures**: try other prominent SER architectures such as LSTM and RNN.

- **Cross-Validation**: in order to conduct a detailed analysis of both the individual models and the ensemble, and study their generalization capabilities, it would be advisable to conduct a cross-validation.

- **Threshold selection for the Ensemble**: it would be advisable to repeat the experiments performed on the aggregation strategies of the ensemble by selecting the threshold values more accurately.

- **Build new aggregation strategies**: we have tried only some of the possible aggregation strategies, while it would be interesting to broaden the analysis by

also trying, for example, some *weighted* strategies such as *weighted voting* or *voting using weighted sum probabilities*.

- **Change the strategy to obtain confidence scores from SVM**: as we have seen in *Section 5.7* one of the main problems in using SVM within the ensemble is that it does not allow us, by its nature, to directly obtain values that can be interpreted in terms of probability. As explained in that section, we have solved the problem through the use of Platt Scaling. However, we have discovered that there exist several alternative ways to deal with this problem (and so obtain confidence scores), such as:

  - **Normalisation**: to do that we can take the maximum value retrieved by the `decision_function` on the training set and use that value to normalise the values retrieved with the use of the `decision_function` on test data.
  - **RVM**: RVM is a machine learning technique whose main principles are the same of SVM, but with the difference that RVM provides probabilistic classification, since it is based on a Bayesian Probabilistc framework [107]. There exist many Python implementations of this algorithm, such as [108–110]. It would therefore be interesting to conduct a study of this algorithm and try to perform all the experiments previously conducted with SVM, with RVM. In this way it would be possible to judge which of the two algorithms is more suitable for our task and if there are differences between the different implementations of the algorithm. Furthermore, it would be possible to draw up a comparison between the different methods for obtaining confidence scores from SVM (and its variations) classifiers.

## 6.3 Voice Activity Detection

### Limitations

As said several times, the construction of the VAD system was not the main object of this thesis project.

### Future Developments

Although the results obtained with our system are satisfactory, it would be interesting to explore other approaches and build an ad-hoc model for our use case. Moreover, if real data would be obtained, it would undoubtedly be ideal to test the functioning of the VAD module on such data.
Furthermore, it would be advisable to extensively evaluate the effectiveness of the denoising module inserted in the architecture (i.e. Wiener filter) and look for alternative, and possibly more reliable, approaches.

# Chapter 7

# Conclusion

In this thesis we proposed an architecture to perform disruptive situations detection in public transports through Speech Emotion Recognition. The result is a stand-alone application consisting of three modules: denoising, Voice Activity Detection and Speech Emotion Recognition. Although we took care of the design of the whole system, the core part of this research is the construction of the Speech Emotion Recognition architecture. The work described in this dissertation is original and unique in the sense that, at the time of writing, to the best of our knowledge there is no other working architecture that supports all these modules with comparable performance.

Even though the system is not yet ready to be introduced in the real scenario (i.e. public transports), we achieved satisfying results. In particular, through the Speech Emotion Recognition ensemble model we achieved 89% both in accuracy and in F1 score on the positive class (*disruptive* emotions) on RAVDESS dataset. Unlike most of the systems in literature, our model is distinguished by the independence from the speaker and the independence from the gender of the speaker. This contributes to the generality of the model, which is superior to that of state-of-the-art baselines.

As stated in *Chapter 6*, the system still has many limitations and much room for improvement. Among the main limitations, there are the execution time of the application and the challenges faced while building the classifiers for SER, such as lack of real data and lack of reliable and sufficiently large dataset. Further to what has just been said, the lack of reliable datasets in languages other than English, prevents from having a system with high generalization capacity in real environments.

Rooms for improvements can be can be identified at different levels.

At integration level, it could be advisable developing a multi-thread model, instead of a sequential one, for the ensemble of the Speech Emotion Recognition system. Also as regards system integration, several possible features to extend the architecture have been identified, such as a KWS model to detect emergency keywords or a AHSD system to detect hateful contents in the speech.
At level of SER, although the research has led to satisfactory results, there are many aspects to be explored in depth such as the construction of realistic and reliable datasets and the identification of features capable of taking into account personalized differences, such as language or age.

The challenges that have been addressed in this thesis are not only referable to this application context, but can be found in several application and technological areas.
Examples of other contexts of application of SER are systems to assist in the early diagnosis and treatment of patients or SER-integrated surveillance systems for identifying risky situations in calls for help to law enforcement. Among systems for early diagnosis, we find depression speech recognition. Because the abnormal speech features of patients with depression are related to their mental state to some extent, it is valuable to use speech acoustic features as objective indicators for the diagnosis of depression [111]. Another interesting application context is that of systems that adapt their operation or interface based on the emotional state of the user; think for example of integrated driving systems that can activate autonomous driving in case of detection of abnormal moods of the driver. And, last but not least, is the contribution that SER systems could make in preventing cases of violence in public facilities (such as hospitals, nursing homes for the elderly) and domestic violence.
All the applications just listed share a lot of problems, such as the fact that the state-of-art has to deal with datasets that are small, acted (i.e. not realistic), of low audio quality, incorrectly divided, clean (i.e. recorded in the absence of noise and therefore not indicative of the audio signal that is in the real world), biased with respect to a very specific population type or language.
This confirms that research in this sector is still in its infancy, and it is important to invest time and resources in order to introduce services with a strong social impact, such as those mentioned above.

However, this work also demonstrated that an ensemble architecture of CNN and SVM classifiers, combined with datasets built under certain robustness constraints, is able to yield very promising results. This gives us ground to believe that, following a more extensive experimentation and re-engineering, the adoption in real-world settings of SER-based solutions, possibly based on the technologies we developed and described, may soon become a reality.

# Bibliography


[1] *5GMed*. Accessed on 06.11.2021. Feb. 2021. url: https://i2cat.net/projects/5gmed/.

[2] *5GMED*. Accessed on 06.11.2021. url: https://5g-ppp.eu/5gmed/.

[3] Changzeng Fu et al. "An End-to-end Multitask Learning Model to Improve Speech Emotion Recognition". In: *2020 28th European Signal Processing Conference (EUSIPCO)*. 2021, pp. 1–5. doi: 10.23919/Eusipco47968.2020.9287484.

[4] M. S. Likitha et al. "Speech based human emotion recognition using MFCC". In: *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. 2017, pp. 2257–2260. doi: 10.1109/WiSPNET.2017.8300161.

[5] V. B. Sowmya and A. Rajeswari. "Speech Emotion Recognition for Tamil Language Speakers". In: 2019.

[6] Jon Guðnason and Mike Brookes. "Voice source cepstrum coefficients for speaker identification". In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing* (2008), pp. 4821–4824.

[7] Muljono et al. "Speech Emotion Recognition of Indonesian Movie Audio Tracks based on MFCC and SVM". In: *2019 International Conference on contemporary Computing and Informatics (IC3I)*. 2019, pp. 22–25. doi: 10.1109/IC3I46837.2019.9055509.

[8] K. Rao and Manjunath k e. *Speech Recognition Using Articulatory and Excitation Source Features*. Jan. 2017. isbn: 978-3-319-49219-3. doi: 10.1007/978-3-319-49220-9.

[9] Daniel Jurafsky and James H. Martin. "Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 3rd Edition". In: (2020), pp. 548–573.

[10] Joseph Picone. "Signal modeling techniques in speech recognition". In: 1993.

[11] Muhammad Zafar Iqbal. *MFCC and machine learning based speech emotion recognition over Tess and IEMOCAP datasets*. url: `https://fui.edu.pk/fjs/index.php/fujeas/article/view/321`.

[12] Thomas Fang Zheng, Guoliang Zhang, and Zhanjiang Song. "Comparison of different implementations of MFCC". In: *Journal of Computer Science and Technology* 16 (2008), pp. 582–589.

[13] Todor Dimitrov Ganchev, Nikos Fakotakis, and George Kokkinakis. "Comparative Evaluation of Various MFCC Implementations on the Speaker Verification Task". In: 2007.

[14] *librosa.feature.mfcc*. url: `https://librosa.org/doc/main/generated/librosa.feature.mfcc.html`.

[15] Mehmet Bilal Er. "A Novel Approach for Classification of Speech Emotions Based on Deep and Acoustic Features". In: *IEEE Access* 8 (2020), pp. 221640–221653. doi: `10.1109/ACCESS.2020.3043201`.

[16] *librosa.feature.rms*. url: `https://librosa.org/doc/main/generated/librosa.feature.rms.html`.

[17] Ossama Abdel-Hamid et al. "Convolutional Neural Networks for Speech Recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.10 (2014), pp. 1533–1545. doi: `10.1109/TASLP.2014.2339736`.

[18] Ilias Papastratis. *Speech Recognition: a review of the different deep learning approaches*. Accessed on 02.11.2021. July 2021. url: `https://theaisummer.com/speech-recognition/`.

[19] Dr Shankar et al. "Optimal feature-based multi-kernel SVM approach for thyroid disease classification". In: *The Journal of Supercomputing* 76 (Feb. 2020), pp. 1–16. doi: `10.1007/s11227-018-2469-4`.

[20] R. Vijayarajeswari et al. “Classification of mammogram for early detection of breast cancer using SVM classifier and Hough transform”. In: *Measurement* (2019).

[21] Maricel P. Naviamos and Jasmin D. Niguidula. “A Study on Determining Household Poverty Status: SVM Based Classification Model”. In: ICSIM ’20. Sydney, NSW, Australia: Association for Computing Machinery, 2020. isbn: 9781450376907. doi: 10.1145/3378936.3378969. url: https://doi.org/10.1145/3378936.3378969.

[22] Anagha Sonawane, M. U. Inamdar, and Kishor B. Bhangale. “Sound based human emotion recognition using MFCC amp; multiple SVM”. In: *2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC)*. 2017, pp. 1–4. doi: 10.1109/ICOMICON.2017.8279046.

[23] Nivedita Patel, Shireen Patel, and Sapan Mankad. “Impact of autoencoder based compact representation on emotion detection from audio”. In: *Journal of Ambient Intelligence and Humanized Computing* (Mar. 2021), pp. 1–19. doi: 10.1007/s12652-021-02979-3.

[24] Hadhami Aouani and Yassine Benayed. “Deep Support Vector Machines for Speech Emotion Recognition”. In: Jan. 2021, pp. 406–415. isbn: 978-3-030-49341-7. doi: 10.1007/978-3-030-49342-4_39.

[25] Linhui Sun, Sheng Fu, and Fu Wang. “Decision Tree SVM Model with Fisher Feature Selection for Speech Emotion Recognition”. In: *EURASIP J. Audio Speech Music Process.* 2019.1 (Dec. 2019). issn: 1687-4714. doi: 10.1186/s13636-018-0145-5. url: https://doi.org/10.1186/s13636-018-0145-5.

[26] Aravind Ganapathiraju, Jonathan Hamaker, and Joseph Picone. “Applications of support vector machines to speech recognition”. In: *IEEE Transactions on Signal Processing* 52 (2004), pp. 2348–2355.

[27] Rubén Solera-Ureña et al. “SVMs for Automatic Speech Recognition: A Survey”. In: Jan. 2007, pp. 190–216. isbn: 978-3-540-71503-0. doi: 10.1007/978-3-540-71505-4_11.

[28] Thad Hughes and Keir Mierle. "Recurrent neural networks for voice activity detection". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 7378–7382. doi: 10.1109/ICASSP.2013.6639096.

[29] Hong Su et al. "Convolutional neural network for robust pitch determination". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 579–583. doi: 10.1109/ICASSP.2016.7471741.

[30] Houman Ghaemmaghami et al. "Noise robust voice activity detection using features extracted from the time-domain autocorrelation function". In: *INTERSPEECH*. 2010.

[31] Ekapol Chuangsuwanich and James R. Glass. "Robust Voice Activity Detector for Real World Applications Using Harmonicity and Modulation Frequency". In: *INTERSPEECH*. 2011.

[32] Matthew McEachern. *Neural Voice Activity Detection and its Practical Use)*. 2018.

[33] L. Bai, Zhen Zhang, and Jun Hu. "Voice activity detection based on deep neural networks and Viterbi". In: 2017.

[34] Jongseo Sohn, Nam Soo Kim, and Wonyong Sung. "A statistical model-based voice activity detection". In: *IEEE Signal Processing Letters* 6 (1999), pp. 1–3.

[35] Thad Hughes and Keir Mierle. "Recurrent neural networks for voice activity detection". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), pp. 7378–7382.

[36] Chang et al. "Temporal Modeling Using Dilated Convolution and Gating for Voice-Activity-Detection". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 5549–5553. doi: 10.1109/ICASSP.2018.8461921.

[37] Fei Jia, Somshubra Majumdar, and Boris Ginsburg. *MarbleNet: Deep 1D Time-Channel Separable Convolutional Neural Network for Voice Activity Detection*. 2021. arXiv: 2010.13886 [eess.AS].

[38] Hervé Bredin et al. *pyannote.audio: neural building blocks for speaker diarization*. 2019. arXiv: 1911.01255 [eess.AS]. url: https://github.com/pyannote/pyannote-audio.

[39] David Doukhan et al. “An Open-Source Speaker Gender Detection Framework for Monitoring Gender Equality”. In: *Acoustics Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE. 2018.

[40] David Doukhan et al. “INA’S MIREX 2018 MUSIC AND SPEECH DETECTION SYSTEM”. In: *Music Information Retrieval Evaluation eXchange (MIREX 2018)*. 2018.

[41] Google. *WebrtcVAD*. 2011. url: `https://webrtc.org//`.

[42] Mustaqeem and Soonil Kwon. “1D-CNN: Speech Emotion Recognition System Using a Stacked Network with Dilated CNN Features”. In: *Cmc-computers Materials & Continua* 67 (2021), pp. 4039–4059.

[43] Kannan Venkataramanan and Haresh Rengaraj Rajamohan. *Emotion Recognition from Speech*. 2019. arXiv: `1912.10458 [cs.SD]`.

[44] Andrew Huang and Puwei Bao. “Human Vocal Sentiment Analysis”. In: *ArXiv* abs/1905.08632 (2019).

[45] Marco Giuseppe de Pinto et al. “Emotions Understanding Model from Spoken Language using Deep Neural Networks and Mel-Frequency Cepstral Coefficients”. In: May 2020, pp. 1–5. doi: `10.1109/EAIS48028.2020.9122698`.

[46] Hadhami Aouani and Yassine Ben Ayed. “Speech Emotion Recognition with deep learning”. In: *KES*. 2020.

[47] Soham Chattopadhyay, Arijit Dey, and Hritam Basak. “Optimizing Speech Emotion Recognition using Manta-Ray Based Feature Selection”. In: *ArXiv* abs/2009.08909 (2020).

[48] Youddha Beer Singh and Shivani Goel. “1D CNN based approach for speech emotion recognition using MFCC features”. In: *Artificial Intelligence and Speech Technology* (2021).

[49] Mayank Chourasia et al. “Emotion Recognition from Speech Signal Using Deep Learning”. In: *Intelligent Data Communication Technologies and Internet of Things* (2021).

[50] Xuemei An and Zhou Ruan. "Speech Emotion Recognition algorithm based on deep learning algorithm fusion of temporal and spatial features". In: *Journal of Physics: Conference Series* 1861 (2021).

[51] Changzeng Fu et al. "An End-to-end Multitask Learning Model to Improve Speech Emotion Recognition". In: *2020 28th European Signal Processing Conference (EUSIPCO)* (2021), pp. 1–5.

[52] U.Raghu Vamsi et al. *Speech Emotion Recognition(SER) using Multilayer Perceptron and Deep learning techniques*. 2021.

[53] *Wiener filter*. Accessed on 02.11.2021. Mar. 2021. url: `https://en.wikipedia.org/wiki/Wiener_filter`.

[54] *scipy.signal.wiener¶*. Accessed on 02.11.2021. url: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.wiener.html`.

[55] *Voice activity detection*. Oct. 2021. url: `https://en.wikipedia.org/wiki/Voice_activity_detection`.

[56] Wiseman. *Wiseman/Py-webrtcvad: Python interface to the WebRTC voice activity detector*. url: `https://github.com/wiseman/py-webrtcvad`.

[57] Silas Rudolf. *A comparative analysis of the speech detection pipeline*. 2020. url: `https://www.zhaw.ch/storage/engineering/institute-zentren/cai/MSE_VT1_20_SpeechDetection_Silas.pdf`.

[58] Ina-Foss. *ina-foss/inaSpeechSegmenter: CNN-based audio segmentation toolkit. Allows to detect speech, music and speaker gender. Has been designed for large scale gender equality studies based on speech time per gender*. Accessed on 02.11.2021. url: `https://github.com/ina-foss/inaSpeechSegmenter`.

[59] Felix Burkhardt et al. "A database of German emotional speech". In: vol. 5. Sept. 2005, pp. 1517–1520. doi: `10.21437/Interspeech.2005-446`.

[60] Piyush Agnihotri. *EmoDB Dataset*. Sept. 2020. url: `https://www.kaggle.com/piyushagni5/berlin-database-of-emotional-speech-emodb`.

[61] *AudioSet*. url: `https://research.google.com/audioset/`.

[62] Moataz Ayadi, Mohamed S. Kamel, and Fakhri Karray. "Survey on speech emotion recognition: Features, classification schemes, and databases". In: *Pattern Recognition* 44 (Mar. 2011), pp. 572–587. doi: 10.1016/j.patcog.2010.09.020.

[63] Monorama Swain, Aurobinda Routray, and Prithviraj Kabisatpathy. "Databases, features and classifiers for speech emotion recognition: a review". In: *International Journal of Speech Technology* 21 (2018), pp. 93–120.

[64] Lisa Graziani. "Constrained Affective Computing". In: (2021).

[65] Steven R. Livingstone and Frank A. Russo. *The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)*. Version 1.0.0. Funding Information Natural Sciences and Engineering Research Council of Canada: 2012-341583 Hear the world research chair in music and emotional speech from Phonak. Apr. 2018. doi: 10.5281/zenodo.1188976. url: https://doi.org/10.5281/zenodo.1188976.

[66] Steven R. Livingstone. *RAVDESS Emotional speech audio*. Jan. 2019. url: https://www.kaggle.com/uwrfkaggler/ravdess-emotional-speech-audio.

[67] M. Kathleen Pichora-Fuller and Kate Dupuis. *Toronto emotional speech set (TESS)*. Version DRAFT VERSION. 2020. doi: 10.5683/SP2/E8H2MF. url: https://doi.org/10.5683/SP2/E8H2MF.

[68] R. Matin. *Developing a speech emotion recognition solution using ensemble learning for children with autism spectrum disorder to help identify human emotions (Unpublished thesis)*. 2020.

[69] *Toronto emotional speech set (TESS) Collection*. Accessed on 21.10.2021. url: https://tspace.library.utoronto.ca/handle/1807/24487.

[70] Eu Jin Lok. *Toronto emotional speech set (TESS)*. Accessed on 21.10.2021. Oct. 2019. url: https://www.kaggle.com/ejlok1/toronto-emotional-speech-set-tess.

[71] Accessed on 21.10.2021. url: http://kahlan.eps.surrey.ac.uk/savee/.

[72] John S. Garofolo et al. *TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1*. Philadelphia, 1993.

[73] P. Ekman. *Universals and cultural differences in facial expressions of emotion*. 207-283. 1971.

[74] Zhihong Zeng et al. *A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions*. 2009. doi: 10.1109/TPAMI.2008.52.

[75] Tarun Sunkaraneni. *SAVEE database*. Accessed on 21.10.2021. July 2019. url: https://www.kaggle.com/barelydedicated/savee-database.

[76] Xin Chang and Władysław Skarbek. *Multi-modal Residual Perceptron Network for Audio-Video Emotion Recognition*. 2021. arXiv: 2107.10742 [eess.SP].

[77] CheyneyComputerScience. *CheyneyComputerScience/Crema-D: Crowd sourced emotional multimodal actors dataset (crema-D)*. Accessed on 21.10.2021. url: https://github.com/CheyneyComputerScience/CREMA-D.

[78] Eu Jin Lok. *Crema-D*. Accessed on 21.10.2021. Oct. 2019. url: https://www.kaggle.com/ejlok1/cremad.

[79] *Soundfile*. url: https://pysoundfile.readthedocs.io/en/latest/.

[80] Slhck. *ffmpeg-normalize: Audio normalization for python/ffmpeg*. url: https://github.com/slhck/ffmpeg-normalize.

[81] Babak Joze Abbaschian, Daniel Sierra-Sosa, and Adel Said Elmaghraby. "Deep Learning Techniques for Speech Emotion Recognition, from Databases to Models". In: *Sensors (Basel, Switzerland)* 21 (2021).

[82] Xu Dong An and Zhou Ruan. "Speech Emotion Recognition algorithm based on deep learning algorithm fusion of temporal and spatial features". In: 1861.1 (Mar. 2021), p. 012064. doi: 10.1088/1742-6596/1861/1/012064. url: https://doi.org/10.1088/1742-6596/1861/1/012064.

[83] Marco Giuseppe de Pinto et al. "Emotions Understanding Model from Spoken Language using Deep Neural Networks and Mel-Frequency Cepstral Coefficients". In: *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*. 2020, pp. 1–5. doi: 10.1109/EAIS48028.2020.9122698.

[84] Sarala Padi, Dinesh Manocha, and Ram D. Sriram. "Multi-Window Data Augmentation Approach for Speech Emotion Recognition". In: *CoRR* abs/2010.09895 (2020). arXiv: 2010.09895. url: https://arxiv.org/abs/2010.09895.

[85] *YouTube*. url: https://www.youtube.com/.

[86] Mike Koenig. *SoundBible-Free Sound Clips*. url: https://soundbible.com/.

[87] Jiaaro. *Jiaaro/pydub: Manipulate audio with a simple and Easy High Level Interface*. url: https://github.com/jiaaro/pydub.

[88] *librosa.load*. url: https://librosa.org/doc/main/generated/librosa.load.html.

[89] *librosa.feature.melspectrogram*. url: https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html.

[90] *Librosa.core.stft*. url: https://librosa.org/doc/0.7.2/generated/librosa.core.stft.html.

[91] *librosa.filters.mel*. url: https://librosa.org/doc/main/generated/librosa.filters.mel.html.

[92] Maged M.M. Fahmy. *Palmprint recognition based on Mel frequency cepstral coefficients feature extraction*. Nov. 2010. url: https://www.sciencedirect.com/science/article/pii/S2090447910000067.

[93] *Sklearn.preprocessing.StandardScaler*. url: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[94] *Sklearn.svm.SVC*. url: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[95] Y. Bengio and Yann Lecun. "Convolutional Networks for Images, Speech, and Time-Series". In: (Nov. 1997).

[96] *sklearn.model*$_s$*election.RandomizedSearchCV*. url: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

[97] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. 1st. Chapman Hall/CRC, 2012. isbn: 1439830037.

[98] Jason Brownlee. *How to Combine Predictions for Ensemble Learning*. Apr. 2021. url: https://machinelearningmastery.com/combine-predictions-for-ensemble-learning/.

[99] Lior Rokach. "Pattern Classification Using Ensemble Methods". In: *Series in Machine Perception and Artificial Intelligence*. 2009.

[100] John Platt. "Probabilistic Outputs for Support vector Machines and Comparisons to Regularized Likelihood Methods". In: 1999.

[101] Ashish Vaswani et al. "Attention is All you Need". In: *ArXiv* abs/1706.03762 (2017).

[102] Axel Berg, Mark O'Connor, and Miguel Tairum Cruz. "Keyword Transformer: A Self-Attention Model for Keyword Spotting". In: *ArXiv* abs/2104.00769 (2021).

[103] *Deepspeech's documentation*. Accessed on 18.11.2021. url: `https://deepspeech.readthedocs.io/en/r0.9/`.

[104] Mohiyaddeen and Dr. Shifaulla Siddiqui. "Automatic Hate Speech Detection: A Literature Review". In: *International Journal of Engineering and Management Research* 11 (2021), pp. 116–121.

[105] Karol J. Piczak. "ESC: Dataset for Environmental Sound Classification". In: *Proceedings of the 23rd ACM international conference on Multimedia* (2015).

[106] Javier Naranjo-Alcazar et al. "An Open-set Recognition and Few-Shot Learning Dataset for Audio Event Classification in Domestic Environments". In: *ArXiv* abs/2002.11561 (2020).

[107] Yi Shi et al. "A comparative study of relevant vector machine and support vector machine in uncertainty analysis". In: July 2013, pp. 469–472. isbn: 978-1-4799-1014-4. doi: `10.1109/QR2MSE.2013.6625625`.

[108] $sklearn_rvm.em_rvm.EMRVC$¶. url: `https://sklearn-rvm.readthedocs.io/en/latest/generated/sklearn_rvm.em_rvm.EMRVC.html#sklearn_rvm.em_rvm.EMRVC`.

[109] *sklearn-rvm*. url: `https://pypi.org/project/sklearn-rvm/`.

[110] JamesRitchie. *JamesRitchie/scikit-rvm: Relevance Vector Machine implementation using the scikit-learn API*. url: `https://github.com/JamesRitchie/scikit-rvm`.

[111] Hongbo Wang et al. "Depression Speech Recognition With a Three-Dimensional Convolutional Network". In: *Frontiers in Human Neuroscience* 15 (2021).

[112] Tal Baram. *Classifying emotions using audio recordings and python*. Mar. 2021. url: `https://towardsdatascience.com/classifying-emotions-using-audio-recordings-and-python-434e748a95eb`.

[113] Youddha Beer Singh and Shivani Goel. "1D CNN based approach for speech emotion recognition using MFCC features". In: *Artificial Intelligence and Speech Technology* (2021).

# Appendix A

## A.1 Volume Normalisation Experiments

To evaluate the effect of audio normalization, several experiments were performed on the convolutional model. In particular, the convolutional model used in these experiments is the same as shown in Table 5.7 in section 5.4. A RandomizedSearchCV with k-folds = 3 was performed to find the best hyperparameters for each experiment.
The hyperparameters analyzed are the following: learning rate for Adam optimizer, Kernel Initializer, batch size. The number of epochs has been set at 50. The results of the experiments are shown in Table A.1.

| Dataset | Number of MFCCs | Kernel Initializer | Learning Rate (Adam Optimizer) | Batch Size | Accuracy | F1 score (positive class - 1) | F1 score (negative class - 0) |
|---|---|---|---|---|---|---|---|
| RAVDESS | 12 | He Normal | 0.001 | 8 | 0.83 | 0.84 | 0.83 |
| RAVDESS normalized | 12 | He Normal | 0.001 | 8 | 0.81 | 0.81 | 0.81 |
| CREMA | 12 | Lecun Uniform | 0.001 | 8 | 0.83 | 0.90 | 0.58 |
| CREMA normalized | 12 | Uniform | 0.001 | 16 | 0.78 | 0.87 | 0.43 |
| TESS | 12 | Lecun Uniform | 0.001 | 8 | 1.00 | 1.00 | 1.00 |
| TESS normalized | 12 | Uniform | 5e-05 | 4 | 0.53 | 0.69 | 0.08 |

Table A.1: Volume Normalisation Experiments

The trend that can be observed is that, in any case, by applying the normalization of the volume there is a deterioration in the performance of the model. This deterioration is especially noticeable when using the TESS dataset.

## A.2 Initial Experiments on the Dataset

Our baseline model is inspired by [112] and [113].

At the beginning we were using an audio length, in samples, equal to 120378. This values correspond to the mean length of RAVDESS audio files. Moreover, we were using 40 MFCCs component.

By cutting and padding all audio files to that length, we obtained input feature vectors of shape (`batch_size, 236,40`).

Our baseline architecture is shown in Figure A.1:

```
model = Sequential()
model.add(layers.Conv1D(256, 5,padding='same',
                 input_shape=(236,40)))
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling1D(pool_size=(8)))
model.add(layers.Dropout(0.2))
model.add(layers.Conv1D(128, 5,padding='same'))
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling1D(pool_size=(4)))
model.add(layers.Dropout(0.1))
model.add(layers.Flatten())
model.add(layers.Dense(64))
model.add(layers.Dense(1))
model.add(layers.Activation('sigmoid'))
```

Figure A.1: Baseline Architecture

Moreover, at the beginning we were using only RAVDESS, SAVEE and TESS datasets and we performed random splitting.

We performed few experiments to study the effect of scaling data through standard scaling. We obtained the results showed in Table A.2:

| | **F1 score Positive Class (0)** | **F1 score Negative Class (1)** | **Accuracy** |
|---|---|---|---|
| **Baseline - No Data Scaling** | 0.93 | 0.94 | 0.93 |
| **Baseline - Data Scaling** | 0.96 | 0.97 | 0.97 |

Table A.2: Results obtained by the baseline model with and without performing Data Standardization on Input Data. In this case Input Data is constituted by 410 samples of class 0 and 486 samples of class 1.

From the high performances of these experiments we understood that the division of the datasets into training, validation and test set was not correct and we proceeded, from here, to the division that we used in this thesis.

## A.3 Experimental Results for the Ensemble Model

For each experiment we highlighted in green the models that have been selected for the ensemble.

For the experiments on SVM models we report the F1 score of only the best models.

### A.3.1 Results of Experiment 1

Table A.3 shows the results obtained by the CNNs models of Experiment 1.

| Experiment ID | Classifier | Accuracy | F1 score (positive class - 1) | F1 score (negative class - 0) |
|---|---|---|---|---|
| 1.1 | CNN | 0.88 | 0.88 | 0.87 |
| 1.2 | CNN | 0.85 | 0.85 | 0.85 |
| 1.3 | CNN | 0.48 | 0.65 | 0.03 |
| 1.4 | CNN | 0.57 | 0.73 | 0.00 |
| 1.5 | CNN | 0.57 | 0.41 | 0.67 |
| 1.6 | CNN | 0.85 | 0.85 | 0.85 |
| 1.7 | CNN | 0.82 | 0.88 | 0.56 |
| 1.8 | CNN | 0.87 | 0.92 | 0.67 |
| 1.9 | CNN | 0.60 | 0.70 | 0.40 |
| 1.10 | CNN | 0.55 | 0.52 | 0.57 |
| 1.11 | CNN | 0.57 | 0.63 | 0.48 |
| 1.12 | CNN | 0.68 | 0.70 | 0.66 |

Table A.3: Results obtained by the CNNs models of Experiment 1. Performances have been evaluated in terms of Accuracy and F1 score. In green, the models that we selected to build the ensemble.

### A.3.2 Results of Experiment 2

Table A.4 shows the results obtained by the CNNs models of Experiment 2.

| Experiment ID | Classifier | Accuracy | F1 score (positive class - 1) | F1 score (negative class - 0) |
|---|---|---|---|---|
| 2.1 | CNN | 0.82 | 0.81 | 0.82 |
| 2.2 | CNN | 0.80 | 0.79 | 0.81 |
| 2.3 | CNN | 0.56 | 0.72 | 0.01 |
| 2.4 | CNN | 0.57 | 0.73 | 0.00 |
| 2.5 | CNN | 0.53 | 0.20 | 0.66 |
| 2.6 | CNN | 0.53 | 0.15 | 0.67 |
| 2.7 | CNN | 0.83 | 0.90 | 0.58 |
| 2.8 | CNN | 0.87 | 0.92 | 0.64 |
| 2.9 | CNN | 0.57 | 0.60 | 0.54 |
| 2.10 | CNN | 0.85 | 0.85 | 0.86 |
| 2.11 | CNN | 0.57 | 0.63 | 0.48 |
| 2.12 | CNN | 0.73 | 0.75 | 0.70 |

Table A.4: Results obtained by the CNNs models of Experiment 2. Performances have been evaluated in terms of Accuracy and F1 score. In green, the models that we selected to build the ensemble.

### A.3.3 Results of Experiment 3

Table A.5 shows the Results obtained by the CNNs models of Experiment 3.

| Experiment ID | Classifier | Accuracy | F1 score (positive class - 1) | F1 score (negative class - 0) |
|---|---|---|---|---|
| 3.1 | CNN | 0.89 | 0.90 | 0.89 |
| 3.2 | CNN | 0.90 | 0.91 | 0.89 |
| 3.3 | CNN | 0.68 | 0.77 | 0.49 |
| 3.4 | CNN | 0.67 | 0.76 | 0.47 |
| 3.5 | CNN | 0.73 | 0.77 | 0.69 |
| 3.6 | CNN | 0.49 | 0.66 | 0.00 |
| 3.7 | CNN | 0.85 | 0.91 | 0.53 |
| 3.8 | CNN | 0.82 | 0.89 | 0.42 |
| 3.9 | CNN | 0.65 | 0.66 | 0.65 |
| 3.10 | CNN | 0.71 | 0.74 | 0.69 |
| 3.11 | CNN | 0.70 | 0.74 | 0.64 |
| 3.12 | CNN | 0.70 | 0.74 | 0.64 |

Table A.5: Results obtained by the CNNs models of Experiment 3. Performances have been evaluated in terms of Accuracy and F1 score. In green, the models that we selected to build the ensemble.

### A.3.4 Results of Experiment 4

Table A.6 shows the Results obtained by the CNNs models of Experiment 3. The models highlighted in orange are models that have performed significantly better on the test set than on the validation set, by an amount of 20% in terms of accuracy. This led us to think that these results are attributable to chance and therefore not reliable.

| Experiment ID | Classifier | Accuracy | F1 score (positive class - 1) | F1 score (negative class - 0) |
|---|---|---|---|---|
| 4.1 | CNN | 0.89 | 0.90 | 0.89 |
| 4.2 | CNN | 0.85 | 0.85 | 0.85 |
| 4.3 | CNN | 0.66 | 0.73 | 0.51 |
| 4.4 | CNN | 0.66 | 0.55 | 0.48 |
| 4.5 | CNN | 0.58 | 0.49 | 0.65 |
| 4.6 | CNN | 0.53 | 0.68 | 0.10 |
| 4.7 | CNN | 0.90 | 0.94 | 0.73 |
| 4.8 | CNN | 0.82 | 0.90 | 0.15 |
| 4.9 | CNN | 0.95 | 0.95 | 0.94 |
| 4.10 | CNN | 0.85 | 0.87 | 0.84 |
| 4.11 | CNN | 0.88 | 0.88 | 0.89 |
| 4.12 | CNN | 0.67 | 0.72 | 0.59 |

Table A.6: Results obtained by the CNNs models of Experiment 4. Performances have been evaluated in terms of Accuracy and F1 score. In green, the models that we selected to build the ensemble.

### A.3.5 Results of Experiment 5

Table A.5 shows the Results obtained by the CNNs models of Experiment 5.

| Experiment ID | Classifier | Accuracy | F1 score (positive class - 1) | F1 score (negative class - 0) |
|---|---|---|---|---|
| 5.1 | SVM | 0.63 | - | - |
| 5.2 | SVM | 0.63 | - | - |
| 5.3 | SVM | 0.58 | - | - |
| 5.4 | SVM | 0.57 | - | - |
| 5.5 | SVM | 0.52 | - | - |
| 5.6 | SVM | 0.55 | - | - |
| 5.7 | SVM | 0.60 | - | - |
| 5.8 | SVM | 0.68 | - | - |
| 5.9 | SVM | 0.56 | - | - |
| 5.10 | SVM | 0.57 | - | - |
| 5.11 | SVM | 0.54 | - | - |
| 5.12 | SVM | 0.54 | - | - |

Table A.7: Results obtained by the SVMs models of Experiment 5. Performances have been evaluated in terms of Accuracy and F1 score. In green, the models that we selected to build the ensemble.

### A.3.6 Results of Experiment 6

Table A.8 shows the Results obtained by the SVMs models of Experiment 3.

| Experiment ID | Classifier | Accuracy | F1 score (positive class - 1) | F1 score (negative class - 0) |
|---|---|---|---|---|
| 6.1 | SVM | 0.63 | - | - |
| 6.2 | SVM | 0.61 | - | - |
| 6.3 | SVM | 0.53 | - | - |
| 6.4 | SVM | 0.57 | - | - |
| 6.5 | SVM | 0.52 | - | - |
| 6.6 | SVM | 0.55 | - | - |
| 6.7 | SVM | 0.70 | - | - |
| 6.8 | SVM | 0.83 | 0.90 | 0.58 |
| 6.9 | SVM | 0.55 | - | - |
| 6.10 | SVM | 0.55 | - | - |
| 6.11 | SVM | 0.52 | - | - |
| 6.12 | SVM | 0.58 | - | - |

Table A.8: Results obtained by the SVMs models of Experiment 6. Performances have been evaluated in terms of Accuracy and F1 score. In green, the models that we selected to build the ensemble.

### A.3.7 Results of Experiment 7

Table A.9 shows the Results obtained by the CNNs models of Experiment 3.

| Experiment ID | Classifier | Accuracy | F1 score (positive class - 1) | F1 score (negative class - 0) |
|---|---|---|---|---|
| 7.1 | SVM | 0.73 | - | - |
| 7.2 | SVM | 0.68 | - | - |
| 7.3 | SVM | 0.57 | - | - |
| 7.4 | SVM | 0.57 | - | - |
| 7.5 | SVM | 0.51 | - | - |
| 7.6 | SVM | 0.50 | - | - |
| 7.7 | SVM | 0.47 | - | - |
| 7.8 | SVM | 0.82 | 0.89 | 0.42 |
| 7.9 | SVM | 0.67 | - | - |
| 7.10 | SVM | 0.85 | 0.91 | 0.53 |
| 7.11 | SVM | 0.72 | - | - |
| 7.12 | SVM | 0.58 | - | - |

Table A.9: Results obtained by the SVMs models of Experiment 7. Performances have been evaluated in terms of Accuracy and F1 score. In green, the models that we selected to build the ensemble.

### A.3.8 Results of Experiment 8

Table A.10 shows the Results obtained by the SVMs models of Experiment 3.

| Experiment ID | Classifier | Accuracy | F1 score (positive class - 1) | F1 score (negative class - 0) |
|---|---|---|---|---|
| 8.1 | SVM | 0.65 | - | - |
| 8.2 | SVM | 0.70 | - | - |
| 8.3 | SVM | 0.57 | - | - |
| 8.4 | SVM | 0.57 | - | - |
| 8.5 | SVM | 0.52 | - | - |
| 8.6 | SVM | 0.50 | - | - |
| 8.7 | SVM | 0.46 | - | - |
| 8.8 | SVM | 0.85 | 0.91 | 0.53 |
| 8.9 | SVM | 0.65 | - | - |
| 8.10 | SVM | 0.88 | 0.88 | 0.87 |
| 8.11 | SVM | 0.66 | - | - |
| 8.12 | SVM | 0.56 | - | - |

Table A.10: Results obtained by the SVMs models of Experiment 8. Performances have been evaluated in terms of Accuracy and F1 score. In green, the models that we selected to build the ensemble.

### A.3.9 Hyperparameters of Best Models

The performance of machine learning and deep learning models is greatly influenced by the setting of initial values, such as neural network weights. To ensure reproducibility, we set `random_seed = 7`. Tables A.11 and A.12 show the hyperparameters selected through Randomized Search . In particular, the models reported in Tables A.11 and A.12 are those that showed the best perfomance among all the CNNs and SVMs models respectively.

**CNN**

| Experiment ID | Initializer | Learning Rate (Adam Optimizer) | Batch Size |
|---|---|---|---|
| 1.1 | glorot uniform | 0.001 | 8 |
| 1.2 | glorot normal | 0.0001 | 4 |
| 1.6 | uniform | 0.001 | 8 |
| 1.8 | glorot normal | 0.001 | 8 |
| 2.1 | glorot normal | 0.001 | 8 |
| 2.2 | uniform | 0.0001 | 4 |
| 2.8 | uniform | 0.001 | 8 |
| 2.10 | uniform | 0.0001 | 4 |
| 2.12 | uniform | 0.0001 | 4 |
| 3.1 | glorot normal | 0.001 | 8 |
| 3.2 | uniform | 0.001 | 8 |
| 3.12 | uniform | 0.0001 | 4 |
| 4.1 | glorot normal | 0.001 | 8 |
| 4.2 | uniform | 0.001 | 8 |

Table A.11: Hyperparameters selected through RandomizedSearchCV. The models reported in this table are those that showed the best perfomance among all the CNNs models of our experiments

**SVM**

| Experiment ID | Kernel | C |
|---|---|---|
| 6.8 | rbf | 10 |
| 7.8 | rbf | 10 |
| 7.10 | rbf | 10 |
| 8.8 | rbf | 10 |
| 8.10 | rbf | 10 |

Table A.12: Hyperparameters selected through RandomizedSearchCV. The models reported in this table are those that showed the best perfomance among all the SVMs models of our experiments

### A.3.10 Learning Curves for Best CNN models

Figures A.2 and A.3 show the learning curves of the CNN models that led to best perfomances among all our experiments.

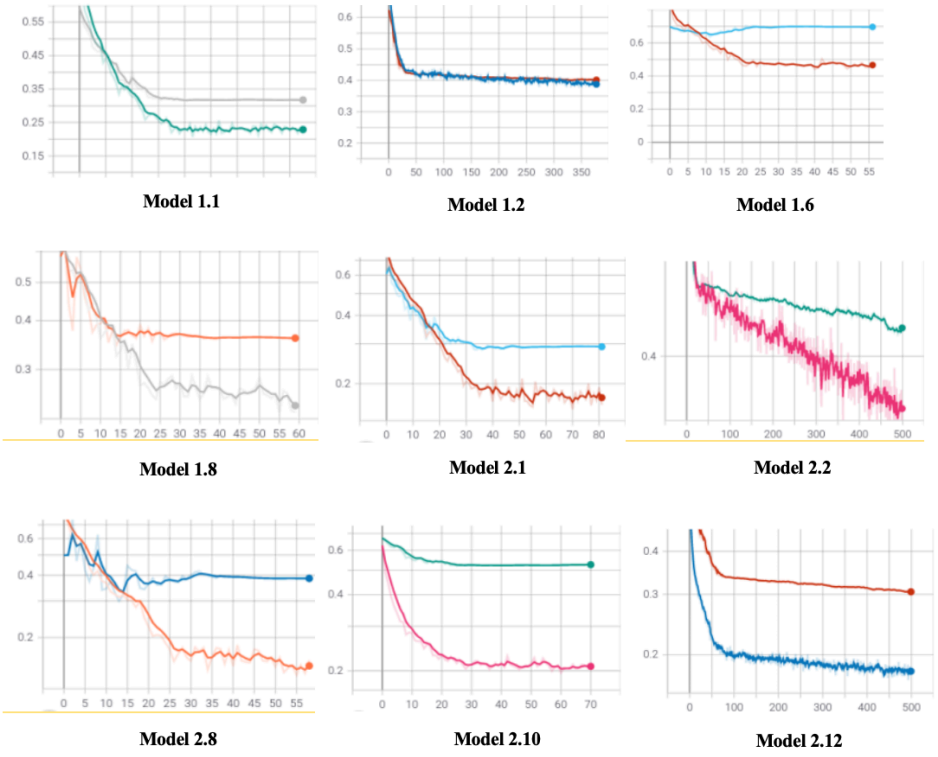


Figure A.2: Learning Curves of CNN models that led to best performances among all our experiments - 1

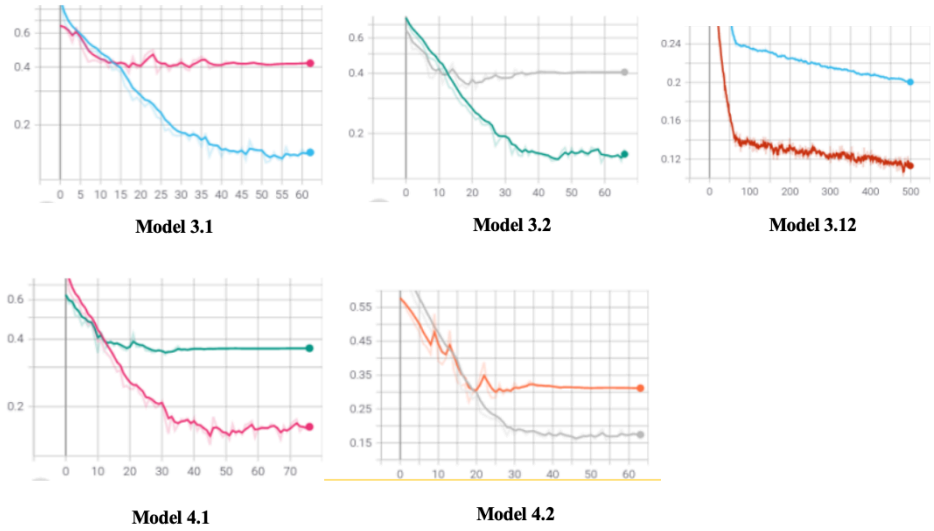


Figure A.3: Learning Curves of CNN models that led to best performances among all our experiments - 2

## A.4 Experimental Setup - Hardware

All the experiments (i.e. training and validation of the single models) have been carried out on a laptop DELL-Latitude 5420. The device specifications are listed in Table A.13.

| | |
|---|---|
| **Processor** | 11th Gen Intel(R) Core(TM) i7 3.00 GHz |
| **RAM** | 16.0 GB |
| **System Type** | 64-bit operating system, x64-based processor |
| **OS** | Linux Ubuntu |

Table A.13: Hardware specifications - 1

The construction of the ensemble, its validation and the integration of the different systems was performed on a MacBook Air laptop (Retina, 13-inch, 2019). The device specifications are listed in Table A.14.

| | |
|---|---|
| **Processor** | 1,6 GHz Intel Core i5 dual-core |
| **RAM** | 8.0 GB |
| **System Type** | 64-bit operating system, x64-based processor |
| **OS** | macOS Monterey |

Table A.14: Hardware specifications - 2

The final architecture, containing all the AI modules that have been developed to perform the "Disruptive Situations Detection" function, must be hosted on hardware with the following technical specifications:

- Intel CPU with $\geq$ 8 cores.
- Memory $\geq$ 32 GB RAM
- Storage $\geq$ 256 GB SSD
- OS: Linux distribution, Ubuntu is preferred
- Recent NVIDIA GPU with $\geq$ 11GB, such as GeForce RTX 2080 Ti or Titan V.

As for the cameras and microphones that will provide real-time audio from the train to the AI module, a closed-circuit television (CCTV) camera with a microphone will be used to detect disruptive or tense situations inside the train. The model selected is the Hikvision DS- 2DE2A404IW-DE3, which is a compact 4 Megapixel camera with pan, tilt and zoom (PTZ) capabilities that will allow the operator in the Cloud Control Centre (CCTV Management Module) to manipulate it remotely.

## A.5 Integration: a stand-alone Python application

After conducting the construction and validation of the individual components (denoising module, VAD system, SER system), it was necessary to integrate them in order to build an application that can be run on the Ground AI module of the Edge Server in Figure 1.1 (please refer to *Chapter 1* for further details).

The whole application is self-contained within a folder called `ambient-intelligence`. The application can be run from inside this folder through a UNIX terminal using the following command: `python3 main.py`.
The arguments that need to be specified when running the script are:

- `-m`: to specify the method of execution. Possible options are:
  - `mic`: for local real-time execution. The audio is captured from the audio interfaces of the laptop through the `SpeechRecognition` library for Python.
  - `file`: for offline execution. The audio is provided through a *.wav* audio file.
  - `real-mic`: for streaming execution in the real environment. The audio is provided by the microphone hosted on the CCTV camera described in *Section A.4*. While the first two modes have been implemented, the last one has yet to be developed.
- `-f`: to specify the audio file when `- m file` execution is enabled. After typing the `-f` option, the full path to an audio file should be provided.
- `-p`: to specify the aggregation strategy to be used when using the ensemble. Possible options are:
  - `voting`
  - `avg_1`
  - `avg_2`

After starting the application through the just described command, the Segmenter object of Ina Speech Segmenter API is instantiated. The instatiation of the Segmenter lasts

around 1 second.
At this point, three different possible scenarios may arise:

- The modality of execution is set to `mic`: the audio is captured from local audio interfaces and converted to a temporary *.wav* file at 16 KHz sample-rate. The audio file is then ready to be processed.
- The modality of execution is set to `file`: the file path of the audio file is directly processed.
- The modality of execution is set to `real-mic`: the audio is received from the CCTV camera through a Real Time Streaming Protocol (RTSP) and divided into parts lasting 5 seconds each using a windowing mechanism. Each subsequent part of the input audio is written on a temporary *.wav* file and then is ready to be analyzed by the VAD system. The process just described has been devised but not yet developed.

After receiving the audio from the input interface, it needs to be prepared to be used as input for the VAD system. The steps of data preparation are the following:

- Sampling and quantization
- The samples are resampled to 16 KHz sample rate
- To be sure that the audio length respect the length chosen in 5.3 (5 seconds) it needs to cut or padded
- A Wiener filter is then applied to the input samples
- After denoising the input audio, it is analyzed through the VAD system

If VAD system recognizes the presence of speech in the input, the SER process is started; otherwise the system stops. In the real scenario the system should not stop, but it has to start processing the next fragment of input audio.
Once and if the SER process is started, what happens is the following:

- Each model is instantiated one after the other, i.e. sequentially
- For each model, the corresponding scaler is loaded in order to perform data standardization

- After loading the scaler, feature extraction phase is started. A specific feature extraction is performed for each of models, depending on the features used to train it. Feature extraction includes the following steps:
  - MFCCs computation
  - Elimination of first MFCC component
  - Energy computation (if required by the model)
  - Data standardization
  - Reshaping of the feature vector in order to have a feature vector of shape (`1, 157, n_mfccs`) for the convolutional models or a feature vector of shape (`1, n_mfccs`) for the SVM models
- Once the feature extraction ends, each of the model classifies the input feature vector
- The predictions of the classifiers are aggregated through one of the aggregation strategies and the final prediction (*disruptive* or *non-disruptive*) is given as output

The whole SER process lasts around 10 seconds.

# Acknowledgements

I would like to thank Professor Torroni for having accompanied me from the beginning of this short and intense journey in the world of Artificial Intelligence. I would like to thank him not only for the supervision work done during the writing of this thesis, but also for all the availability and support provided in this path, indispensable to face these difficult years. Finally, I thank the professor for passing on to me the passion for a subject, which I hope will be my daily bread.

I would like to thank my co-supervisors, Andrea and Federico, for the helpfulness and competence they showed whenever I needed a comparison or a cue.

I thank my i2CAT internship supervisors, Josep Escrig Escrig and Ivan Huerta Casado, for following me closely during the wonderful months I spent in Barcelona. Thank you for sharing your knowledge with me and for allowing me to acquire important skills that are the foundational blocks of my entire future path.

Finally, thank you to everyone who has walked alongside me during these very important years of my life. I am infinitely grateful.