POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Master Degree in Music and Acoustic Engineering

Audio Splicing Detection and Localization Based on Recording Device Cues

by:
Daniele Ugo Leonzio

matr.:
940102

Supervisor:
Prof. Paolo Bestagini

Co-supervisor:
M.Sc. Luca Cuccovillo

Academic Year
2020-2021

Abstract

In recent years, we have witnessed an increasing spread of technology. Artificial intelligence and machine learning are now part of our daily lives. The availability of these sophisticated techniques, even on the consumer market, has made it possible for anyone to create multimedia content at a professional level. This has also created a new kind of problem to deal with: it has become very easy to create very realistic fake content that can be used to convey targeted messages by exploiting the notoriety of certain people. For this reason, the possibility of verifying the reliability of a multimedia object is becoming of paramount importance, especially if these files are used as evidence in trials. The problem we have addressed in this thesis goes in this direction. Our goal is to determine whether an audio track under analysis has been manipulated through splicing. Moreover, if a recording is detected as spliced, we identify where it has been modified. The method we propose is based on a Convolutional Neural Network (CNN) to extract certain features from the audio recording. After extracting the features, we determine through a clustering algorithm if there has been a manipulation. Finally, we identify the point where the modification has been introduced with a distance-based technique. The results achieved are very satisfactory as we are able to reach 98% accuracy for the identification phase and a very small error for the localisation task on a dataset we built on purpose to study this problem.

Sommario

Negli ultimi anni, abbiamo assistito a una crescente diffusione della tecnologia. L'intelligenza artificiale e l'apprendimento automatico fanno ormai parte della nostra vita quotidiana. La disponibilità di queste tecniche sofisticate, anche sul mercato consumer, ha reso possibile a chiunque creare contenuti multimediali a livello professionale. Questo ha anche creato un nuovo tipo di problema da affrontare: è diventato molto facile creare contenuti falsi molto realistici che possono essere utilizzati per trasmettere messaggi mirati sfruttando la notorietà di alcune persone. Per questo motivo, la possibilità di verificare l'affidabilità di un oggetto multimediale sta diventando di fondamentale importanza, soprattutto se questi file vengono utilizzati come prove nei processi. Il problema che abbiamo affrontato in questa tesi va in questa direzione. Il nostro obiettivo è quello di determinare se una traccia audio in analisi è stata manipolata attraverso lo splicing. Inoltre, se una registrazione viene rilevata come manipolata, identifichiamo dove è stata modificata. Il metodo che proponiamo si basa su una rete neurale convoluzionale (CNN) per estrarre alcune caratteristiche dalla registrazione audio. Dopo aver estratto le caratteristiche, determiniamo attraverso un algoritmo di clustering se c'è stata una manipolazione. Infine, identifichiamo il punto in cui la modifica è stata introdotta con una tecnica basata sulla distanza. I risultati ottenuti sono molto soddisfacenti in quanto siamo in grado di raggiungere il 98% di accuratezza per la fase di identificazione e un errore molto piccolo per la localizzazione su un set di dati che abbiamo costruito appositamente per studiare questo problema.

Ringraziamenti

Questa tesi rappresenta la conclusione del mio percorso al Politecnico di Milano.

Vorrei iniziare ringraziando il Prof. Paolo Bestagini e il M.Sc. Luca Cuccovillo per avermi guidato in questo progetto di ricerca. Grazie per tutti i consigli utili dati in questi mesi di lavoro insieme, avete reso sicuramente tutto più semplice.

Ringrazio poi i miei genitori per avermi sempre dato la possibilità di seguire le mie passioni, sostenendomi e aiutandomi in ogni occasione.

Ringrazio mia sorella Eugenia per aver sempre creduto in me. Magari a volte non sono capace di dimostrarlo, ma sai che ti voglio bene.

Vorrei poi ringraziare Gabriella per essermi stata, nonostante le difficoltà che sono passate, sempre vicino.

Ringrazio Aldo, per avermi fatto scoprire diverso ai miei occhi. Mi ha dato una sicurezza che era ancora inespressa, ti sarò per sempre grato.

Ringrazio anche Ema, Franci e Andrea. Amici sinceri che hanno reso sicuramente l'ultimo periodo delle mia vita più divertente.

Ringrazio Simone, mio compagno di avventure per tutta la magistrale. Grazie per tutto l'aiuto che mi hai dato nei mille progetti da fare.

Ringrazio infine tutti coloro che per un motivo od un altro hanno incrociato la mia vita in questo percorso.

Daniele

Contents

List of Figures

List of Tables

Glossary

ANN Artificial Neural Network. 16, 18, 21

AUC Area Under the Curve. 41, 42, 52, 55

BFCC Bark Frequency Cepstrum coefficient. 7

CNN Convolutional Neural Network. vi–viii, 2, 3, 21, 22, 26–31, 34, 38, 42, 44–47, 51, 52, 63, 64

DFT Discrete Fourier transform. vi, 11, 12

DL Deep Learning. 16

ENF Electric Network Frequency. 2, 8

FN False Negative. 39

FP False Positive. 39, 40

FPR False Positive Rate. 41, 42

GMM Gaussian Mixture Model. 6, 7

GSV Gaussian Support Vector. 6, 7

LFCC Linear Frequency Cepstrum coefficient. 7

LPCC Linear Prediction Cepstrum coefficient. 7

MFCC Mel Frequency Cepstrum coefficient. 2, 6, 7

ML Machine Learning. 2, 16, 22

MLP Multilayer Perceptron. 20–22

RBF Radial Basis Function. 9

ROC Receiver Operating Characteristic. 41, 52, 55, 64

STFT Short-Time Fourier Transform. vi–viii, 2, 12–15, 23, 27–29, 42, 45–47, 52, 56

SVM Support Vector Machine. 2, 6, 9

TN True Negative. 39, 40

TP True Positive. 39, 40

TPR True Positive Rate. 42

1 Introduction

Rapid developments in technology and the increased widespread availability of more advanced techniques have made the creation and distribution of multimedia content more accessible to everyone. It has also become much easier to create fake content or modify existing content without anyone being able to tell the difference between originals and copies anymore. This is especially true in the field of audio files where the latest technologies developed can synthesize very realistic voices or put together different tracks as if they were a single piece. The latter is the problem we have chosen to address, that is, trying to determine if an audio track has been manipulated and to identify the point at which it has been modified. This is an important problem to solve because, for example, if audio recordings are presented as evidence in a court of law, it becomes crucial to determine if they present traces of manipulation or not.

For instance, several methods that investigate the task of microphone identification have been proposed in the literature. There are methods based on some audio features such as [3], that use descriptors extracted from the audio recording and then cluster them with K-Means. Alternatively, in [4] the authors use Mel Frequency Cepstrum coefficients (MFCCs) and a Support Vector Machine (SVM) to identify different microphones. More advanced methods are based on CNNs and extract some features directly from the audio recording. An example in this direction is [1] in which the authors adopt a CNN to extract the microphone characteristics from the STFT of the signal.

The problem we face in this thesis is that of detecting whether an audio recording is pristine, or it is a composition of multiple recordings from different devices. If the track is a splicing composition, we are also interested in detecting the splicing point location in time. While microphone identification is a topic full of studies and different methodologies, the task of determining the location of a editing point by exploiting microphone characteristics has not been thoroughly investigated. There are some studies that deal with the localization of editing points by exploiting different clues, such as [5] that use the Electric Network Frequency (ENF) to detect the discontinuity. However, very few methods use microphone traces as source to localize the splicing point. To our knowledge, the only work which investigated the usage of microphone clues to detect editing traces is [6], which however present some issues that can be improved. In particular, it started from the assumption of knowing the splicing point. So our study wants to focus the attention on a problem not deeply investigated until now and on the same gives a solution that improves the previous studies.

To develop our work we rely on some basic tools of signal processing and some Machine Learning (ML) techniques. In particular, we use the STFT and Mel spectrogram to transform the signal into time-frequency domain and a CNN to extract audio features. We adopt also the K-means as clustering technique.

To solve the problem described above we propose a method based on a CNN and a clustering technique. We start from the audio recording and we transform it with a STFT in order to have a 2-D signal. After the time-frequency transformation we exploit a CNN based on the work

[1] as a features extractor. We decided to use this type of network because it has good performance for the microphone identification task [1]. After the feature extraction phase, we make use of K-Means algorithm to understand if the analyzed track has been tampered or not. In fact the output of the K-Means are k clusters, for each cluster we compute the centroid and then we compute the distances among the centroids. If this distance is higher than a predefined threshold we can say that the track has undergone tampering.

Furthermore, we localize the splicing point in time. To do this we rely on the euclidean distance. From the feature array extracted from the CNN, we compute the euclidean distances between consecutive samples. We find the maximum of this distance array and then the sample position of the found maximum is our splicing point.

Our algorithm reached remarkable results for the task we were able to reach a balanced accuracy of 98%.

In the localization pipeline we measured the error of localization as the difference in seconds between the predicted splicing point and the true one. The final result show that we were able to detect the correct point position and in the worst case we did and an average error of $0.5s$.

This thesis is organized as follows:

In Chapter 2 there is a summary of the-state-of-the-art regarding the microphone identification and localization. In Chapter 3 we introduce the theoretical background needed to understand the rest of the work. In Chapter 4 we describe the problem we want to solve by means of a rigorous mathematical formulation and describe in details our proposed method. In Chapter 5 we introduce the dataset used in the study, we define the metrics adopted to evaluate our results, and we illustrate all the experiments and test done to evaluate both the identification and the localization algorithms. Chapter 6 contains our conclusions and some future improvements.

2
State of the Art

In this chapter we report an overview of the state-of-the-art methods related to the topic of this thesis. We start introducing methods allowing to solve the microphone identification problem, i.e., recognizing which microphone was used to acquire an audio recording under analysis. We then report a few examples of methods using microphone traces to expose audio forgeries. Finally, we conclude the chapter with a few conclusive remarks.

2.1 Microphone Identification

Due to the continuous development of technology and its application on consumer devices, we are all now able to produce audio and video items at an almost professional level. It is also becoming increasingly easy to produce fake multimedia contents that are indistinguishable from the authentic ones. This is the case of politician speech for example, in which we can extract just words from different speeches and create a new file convey a message we want with the politician voice. These malicious files, however, may end up being used as evidence in a trial, so it becomes essential to be able to recognize their authenticity.

Audio forensics is a branch of the wide field of forensic science. Audio forensics, as said in [7], refers to the acquisition, analysis, and interpretation of audio recordings as part of an official investigation. Typical investigations involve these three primary fields:

- Authenticity
- Enhancement
- Interpretation

As explained in [8], the authenticity and integrity of an audio source can be divided in two main categories: container based method and content based method. The file structure and file metadata are part of container based method, whereas the actual bytes of the file are content based. The container analysis consists of HASH calculation, MAC and file format analysis. The content analysis is the core of audio forensic examination and relies on exploring the audio recording to find traces of tampering. Most of existing audio forensic methods use the actual content for authentication and integrity.

In our work we focus on audio source identification from a recorded audio signal, that can be used to verify the authentication and integrity of the file. Figure 2.1 illustrate what is the goal of this investigation. The initial works on audio source identification used microphones as source devices. But due to the increasing amount of mobile phone users, most recordings are required nowadays using cellphones or–even more often–smartphones. Hence, also the source identification domain evolved, and the most recent studies focus on associating an audio recording to the mobile phone used to record it [9].

In literature there are several methods to solve the problem of audio source identification. One of the first method has been proposed by Kraetzer et al. in [3]. In this work the authors suggest a set of audio steganalysis-based features to cluster, with K-means, or to predict, with Naive Bayes classifiers, both the microphone and the environment.

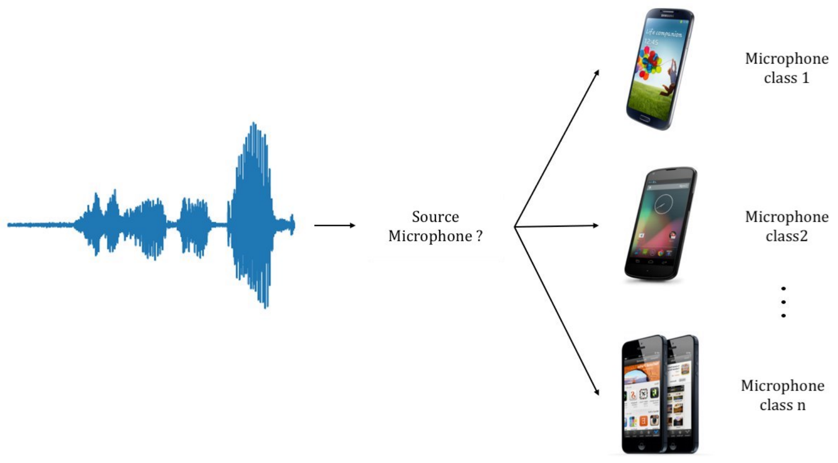


Figure 2.1: This figure shows the goal of the identification problem.

Buchholz et al. [10] instead, proposed to use features extracted from the Fourier coefficients from the near-silence segments to solve the microphone classification task used. Both these works were improved by Kraetzer et al. in [11], where was demonstrated that with the combination of statistical features and unweighted information fusion it is possible to improve the classification accuracy. Another method was proposed by Garcia-Romero and Wilson in [12]. In this paper the classification was performed using Gaussian Support Vectors (GSVs), derived from the means of a trained Gaussian Mixture Model (GMM). A new study by Jiang and Leung [13] added a kernel-based transformation from the original GSV feature vector to another projected space, resulting in a better performance for the microphone recognition task. In [14] Panagakis and Kotropoulos proposed random spectral features and a classifiers based on the sparse representation. The same authors improved this study in [15] with labeled spectral features and SVM as classifier.

Given that the most commonly used devices nowadays are mobile phones, and given that our study is also based on microphones from mobile devices, we now discuss the state of the art on cell phone identification. The first study focusing on mobile phones is [4], in which Hanilci et al. used MFCC and a SVM to classify 14 different cell-phones. In another study [16] they did a comparison among different acoustic features for mobile phone classification. The study concluded that in general, MFCC performs better compared to other cepstral based features such as Linear Frequency Cepstrum coefficients (LFCCs), Bark Frequency Cepstrum coefficients (BFCCs) and Linear Prediction Cepstrum coefficients (LPCCs). Nevertheless, with mean and variance normalization, LPCCs, provided the best classification results. Moreover, they observed that the addition of corresponding delta features (derivatives of order one and two) to the original cepstral features resulted in better performances.

Non-speech regions of the recorded audio were used in [17] for the cell phone recognition task. Pandey et al. [9] have used the estimate of power spectral density of the speech-free regions of the audio recording for source cell-phone classification tasks. Noisy part of the speech was utilized in [18] to extract MFCC feature vectors. In [19], intrinsic traces of cell-phone left on the recorded audio were captured by first extracting the MFCC feature vector at the frame level, then training a GMM, and finally GSVs were taken as a template for each of the devices. Maximum classification accuracy of 97.6% has been achieved on 21 cell-phones of seven different brands. GSVs have also been used for cell-phone verification [20].

Li et al. in [21] proposed an unsupervised method for cell-phone clustering. Deep auto-encoder networks were used to extract intrinsic signatures of a recording device. Spectral clustering was used to form a single cluster for the audio recordings of the same cell-phone. Concatenation of MFCC and inverted MFCC (IMFCC) feature vectors was used in [22] to depict device specific traces. Additionally, Luo et al. [23] proved that the frequency response curve computed from the recorded audio could represent a robust device signature. Feature vector named as BED (Band Energy Difference) has been derived to capture device-specific signatures. Qin et al. in [24] have explored the problem of cell-phone classification in the presence of five different types of noises. Cell-phone identification in presence of AWGN noise has been faced in [1], and [25]. However, these two systems use synthetic tones as compared to the real-world recorded audio signals. The method in [25] requires the suspected cell-phone to be available during the identification or authentication phase.

2.2 Localization

Another problem of audio forensics is the localization of a forgery. With localization we mean to identify, within a manipulated track, the point where the original file has been altered.

To our knowledge, localization has not been still deeply explored in the literature. Furthermore, the existing works on tampering localization are not based on microphone analysis.

An example is the study [5] in which the authors proposed and audio forensic tool for access audio authenticity. Their tool is based on the ENF, that is a signal embedded in audio files when the recording is taken with the device connected to an electrical outlet or when certain microphone are inside a magnetic field. They analyzed discontinuity in the phase of the power grid signal and then localized the splicing point at the discontinuities.

Another approach was used by Grigoras and Smith in [26], based on quantization level analysis to detect the tampering in the audio signal. The authors underlined that PCM files with low bit-depth, whenever processed by classic editing softwares, exhibits 16-bit of quantization instead than the original 8 or 12 only in correspondence of the tampering borders, or whenever external 16-bit content was spliced in the file.

In [27] Gartner et al. the tool chosen to detect manipulation inside the audio signal was the analysis of discontinuities in the framing grid. Also in this case when the discontinuity is found a tampering is detected and localized at the corresponding time instant.

Another different approach by Alan J. Cooper can be found in [28]. In this work the author proposed a method based on the analysis of butt-spliced edits. Forgeries are often created using simple editing technique as butt-splicing. This can leave traces as discontinuity in the audio waveform. The method is time domain based, uses an high pass filter on the audio data and models the discontinuity at higher frequencies. The method than adopts a template to discover potential edits in the filtered signal. A different process was used by Capoferri et al. [29]. They explored in this paper the reverberation cues as a feature to detect and localize the point of splicing in an audio recording. In fact distinct recordings may be recorded in different environments that are typically characterized by different reverberation cues. This method can be explained in three steps. They firstly apply a time-frequency transform on the time domain signal, then they estimate the reverberation time in the free decay region and finally they analyzed the estimated reverberation times to determine the splicing location.

The authors got successfully results but the approach present some intrinsic problem that affected the localization results. The main problem is that the reverberation time can be estimated only in the free decay region, so they were not able to localize the correct time instant of the splicing. In our work we based the localization on a different method, we compute the euclidean distance between consecutive feature frames and lead us to get a better precision in localization.

Now we introduce the work in the literature that deals with localization based on the microphone analysis. A study in this direction can be found in [6]. In this work the authors proposed a method of audio tampering based on the microphone classification but without providing any algorithm for localization. The authors assumed that the location of the splicing borders was known beforehand, to simulate a setup similar to the one often happening in court cases. This setup is of course not applicable if the splicing location is unknown, which is often the case and that we are going to address in the following pages. The microphone classification algorithm is based on the work [30], but in this case the channel models the frequency response of the microphone instead of the environment. After that they built the tampering algorithm that rely on a SVM with a Radial Basis Function (RBF) kernel. They tested the method on different encoding types and different bitrates reaching an overall accuracy of 95%.

Since the methods that explored this problem are few, we were encouraged to investigate this task.

2.3 Conclusive Remarks

In this chapter we showed the state of the art regarding microphone identification and the problem of forgery localization. We have seen that there are several methods for the microphone identification and classification, whereas the localization problem has been less explored in the literature despite its interesting aspects. In the next chapter we will introduce the theoretical background needed to understand the following sections.

3 Theoretical Background

In this chapter we describe the basic theoretical background and the tools we used in this work. This will give to the reader the theory knowledge to understand completely what it is explained in the next sections. In particular in this chapter we focus on the time-frequency representation of audio signals, neural networks and clustering.

3.1 Time-Frequency Representation

An audio signal is the variation of a specific variable, usually pressure, over time. An analog time domain signal can be transformed in its digital form with a process called sampling. The sampled signal is the discrete version of the analog one. We can pass from the time domain to the frequency domain using the Fourier transform. The Fourier transform gives us the variation of the signal with respect to the frequency domain. In case of digital signal, i.e. discrete signals, we deal with the DFT. So given a digital signal $x[n]$ in the time domain, with $n \in 0, ..., N-1$, its

DFT is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} = \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right)\right], \quad (3.1)$$

with $k \in 0, ..., N-1$.

In Figure 3.1 and Figure 3.2 we have an example of digital signal with its DFT.

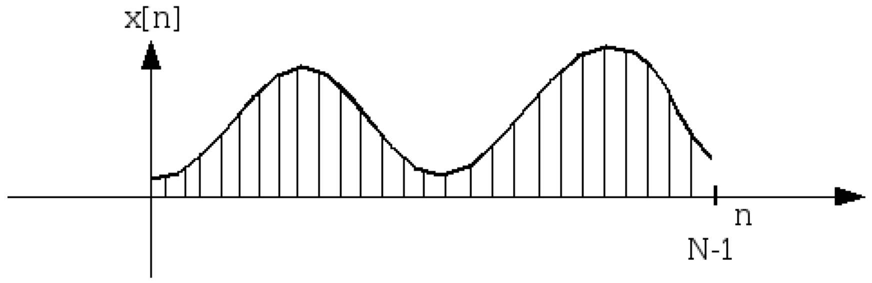


Figure 3.1: Example of digital signal

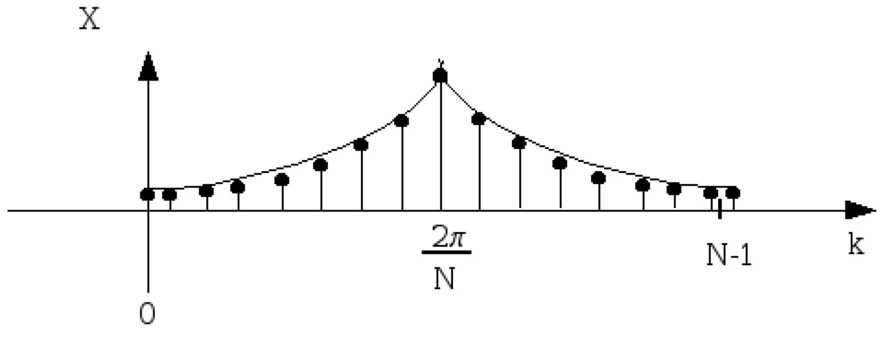


Figure 3.2: Example of DFT of the signal in 3.1

Time-Frequency transforms are representations of a signal in which we can obverse both temporal and frequency evolution. There are different type of time-frequency transforms, depending on how the frequency scale is selected.

3.1.1 STFT

The Short-Time Fourier Transform (STFT) is a time-frequency representation obtained directly from the discrete Fourier transform of the signal.

The STFT adopts the linear frequency scale. Given a digital signal $x[n]$ in the time domain, its STFT is defined as:

$$STFT\{x[n]\}(m,\omega) = X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n] \cdot w[n-m] \cdot e^{-j\omega n}, \quad (3.2)$$

where $w[n]$ represents the chosen window, m the window bin and ω the frequency bin. Several windows have been proposed in the literature, for instance the hann (eq. (3.3a)), sine (eq. (3.3b)), kbd (eq. (3.3c)) ones:

$$w(n) = 0.5 \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right) \quad (3.3a)$$

$$w(n) = \sin\left(\frac{\pi n}{N-1}\right) \quad (3.3b)$$

$$w(n) = \frac{I_0\left(\pi\alpha\sqrt{1-(\frac{2n}{N-1}-1)^2}\right)}{I_0(\pi\alpha)}, \quad (3.3c)$$

the type of windows determines the spectral leakage, which may also influences the analysis quality and resolution. Since the STFT is a complex quantity we can represent it with its magnitude and phase. In Figure 3.3 an example of an STFT magnitude is reported. In this image we have a log spectrogram, this means that the vertical axis has been scaled with a logarithmic function.

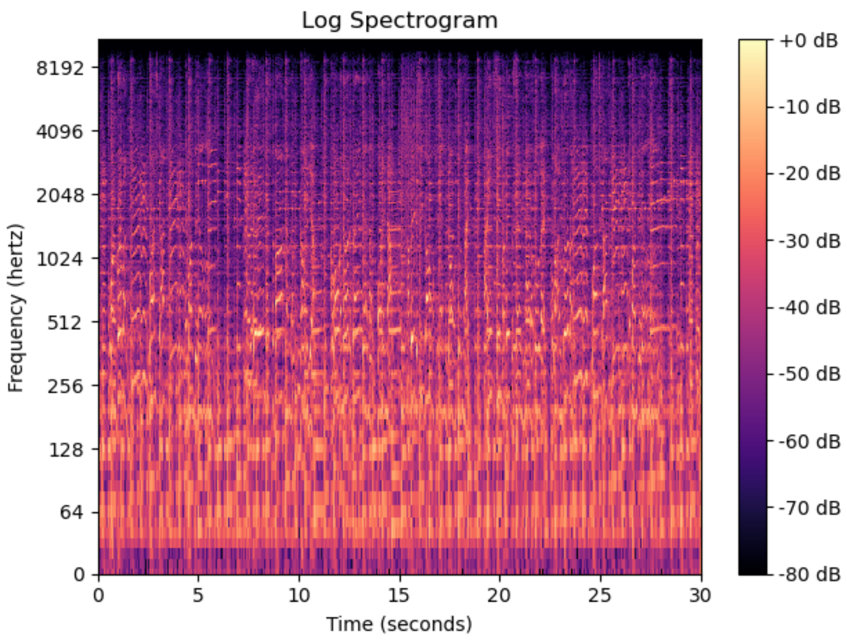


Figure 3.3: An example of log spectrogram.

3.1.2 Mel Spectrogram

The Mel spectrogram is obtained starting from the magnitude of the STFT, but the frequency axis is converted from the liner to the mel scale. The mel scale is a perceptual scale in which pitches are judged to be equal in distance from one another. The following formula describes the relationship between Hertz and mels:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.4}$$

where f is the frequency in Hertz and m the frequency in mels. In Figure 3.4 there is curve that represents the equation 3.4.

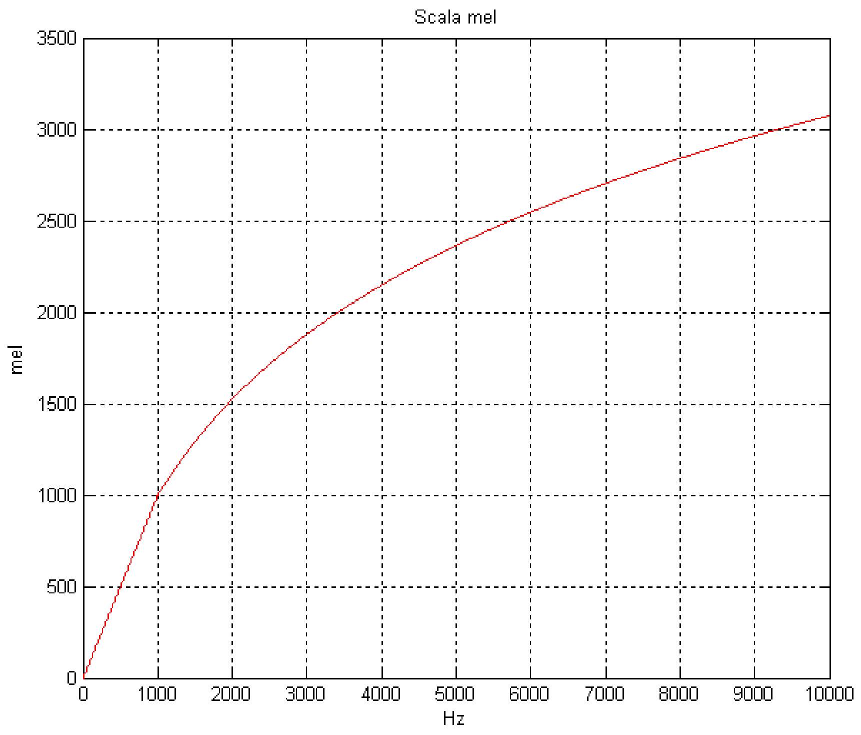


Figure 3.4: Mel frequency conversion.

In Figure 3.5 an example of Mel spectrogram, i.e., of the projection of STFT magnitudes into a space in which the frequency bins are uniformly spaced in Mel domain is reported. On the x axis there is the temporal evolution whereas on the y axis the frequency one. The color of each pixel denotes the magnitude associated to the mel bin at the corresponding time instant. The frequency axis in based on the mel scale and also in this case the axis is scaled using a logarithmic function.

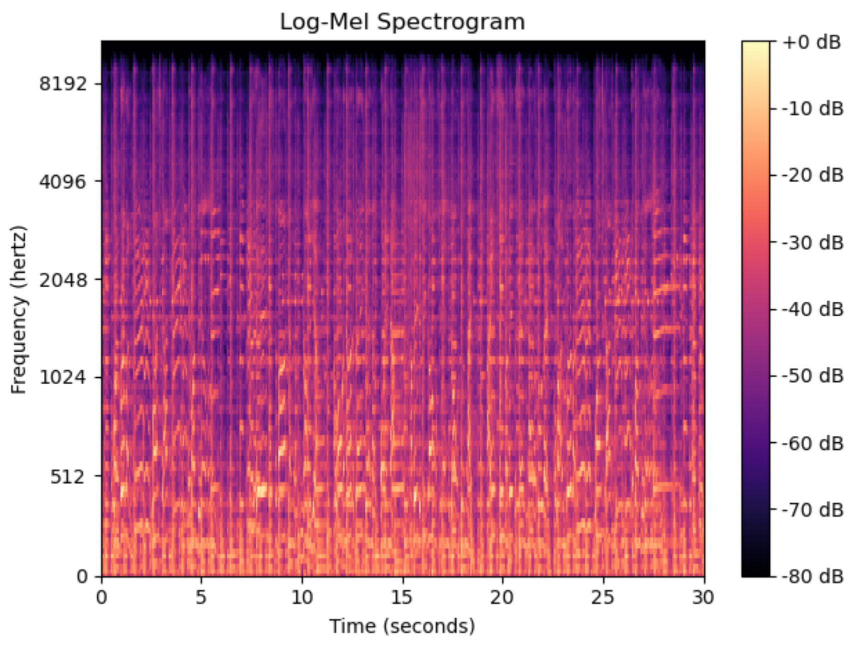


Figure 3.5: An example of log Mel spectrogram.

3.2 Machine Learning

In this section, we will give a brief overview to the core ideas and concepts behind Machine Learning (ML) and Deep Learning (DL).

3.2.1 Overview

Machine Learning (ML) is a field of study which allow computers to learn from data or experience in order to make predictions that are based on this knowledge. These programs or algorithms are designed in a way that they learn and improve over time when are exposed to new data. Deep Learning (DL) instead, is a subcategory of machine learning where the used algorithms attempt to find a mathematical representation of information processing that emulates the structure and functioning of the brain. For this reason, they are called Artificial Neural Network (ANN).

3.2.2 Taxonomy

ML problems can be broadly divided in three classical categories:

- Supervised Learning;
- Unsupervised Learning;
- Reinforcement Learning.

In addition to these three main categories, there are two main subcategories of supervised learning: *Semi-Supervised Learning*, which can be considered a mix between supervised and unsupervised learning, and *Self-Supervised Learning* that can be regarded as autonomous learning in that the model does not necessarily require sample data classified in advance by humans.

3.2.2.1 Supervised Learning

Supervised learning is focused on predicting a target value given input observations. In machine learning, input data to a model are normally called "features". The target values instead are often called "labels". The latter are the elements on which the supervised models are trained to make a prediction. Supervised learning problems can be categorized into two major subcategories: *regression* analysis and *classification*.

In regression analysis, the labels are continuous variables. In classification, the labels are so-called class labels, which can be understood as discrete class-membership or group-membership indicators.

3.2.2.2 Unsupervised Learning

While supervised learning is based on labeled data, unsupervised learning aims to model the hidden structure of the input features without label information. The main tasks addressed in unsupervised learning are:

- Feature Learning;
- Dimensionality Reduction;
- Clustering.

Feature learning techniques replace manual feature engineering and allow a machine to automatically discover the representations needed to perform a specific task.

Instead, the *dimensionality reduction* methods, aims to a data transformation from a high-dimensional space into a low-dimensional one so that the low-dimensional representation still preserves some meaningful properties of the original data. This is crucial since working in high-dimensional spaces can be undesirable: raw data are often sparse as a consequence of the curse of dimensionality, and analyzing the data in this sparse domain is usually computationally intractable.

Lastly, *clustering methods* can be seen as a task similar to classification but without labeling information in the training dataset. Without this information, the main goal is to group data by similarity and define distinct groups based on similarity thresholds. These algorithms can be divided into three major groups: prototype-based, density-based, and hierarchical clustering. While in the first type, a fixed number of cluster centers is defined, in a density-based clustering this number is not fixed but assigned by identifying regions of a high density of data. Finally in hierarchical clustering a distance metric is used to group examples in a tree-like fashion, in such a way that examples at the root are more related to each other. The depth of the tree defines the number of clusters to be used.

3.2.2.3 Reinforcement Learning

Reinforcement learning is tightly connected to the development of reward systems to model complex decision processes and learning a series of actions that lead to a particular outcome. In a reinforcement learning algorithm, a so-called agent learns how to act by interacting with its environment. The agent receives rewards for performing correctly and penalties for performing incorrectly. The agent learns without human intervention by maximizing its reward and minimizing its penalty.

3.2.3 Neural Networks

3.2.3.1 Multilayer Perceptron

The basic building block of an ANN is the perceptron, which is the equivalent of a brain neuron. The perceptron follows the “feed-forward” model, meaning that inputs are sent into the neuron, are then processed for producing an output result, which eventually becomes the input to a new layer of perceptrons. Figure 3.6, represents a generic neuron j. Given a series of input s_i^{in}, the output of a neuron is computed as:

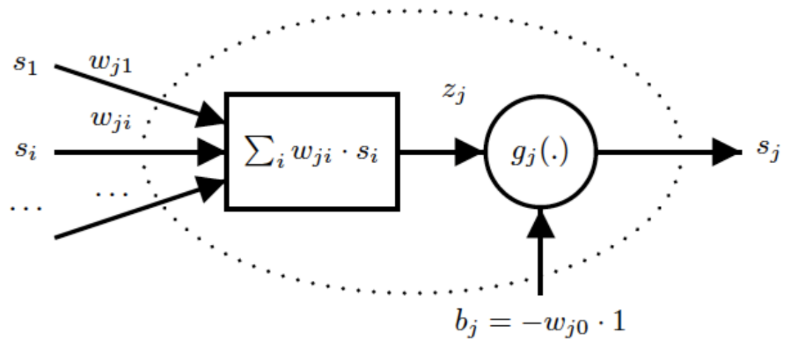


Figure 3.6: An example of perceptron. In this example we have a perceptron j with N inputs, each one with its proper weight. The input are multiplied for the weights and then summed up together. Then the bias is added to the result and this sum pass through the activation function g_j.

$$s_j^{\text{out}} = g_j\left(z_j + b_j\right) = g_j\left(\sum_{i=0}^{N} w_{ij} s_i^{\text{in}} + b_j\right). \tag{3.5}$$

Where ω_{ji} are the weights of each input. The activation procedure is divided in:

- the activation value z_j
- the activation function g_j
- the bias b_j

There exist different activation functions that can be used and they are chosen in relation to task we want to resolve. Initially the weights and the bias are assigned randomly, but to work well the net needs to learn the proper values. In order to learn these values we need a measure of error, $L(\hat{y}, y)$ where $\hat{y}$ is the target value and y the ground truth value. Neural network compute the gradient of this error with respect to the model weights, i.e., they compute a term

$$\frac{\partial}{\partial W^{(l)}} L, \tag{3.6}$$

for every (l)-th layer of the network. In order to minimize the error of the network, the aforementioned gradient is used to move the weights in the opposite direction, e.g. by applying:

$$W^{(l)}(t+1) = W^{(l)}(t) - \alpha \frac{\partial}{\partial W^{(l)}} L, \tag{3.7}$$

an update rule which is is often called either “backpropagation,” since the error influence is propagated backward through the network layers, or “gradient descent”, since the weights are moving in the opposite direction to the gradient. In Figure 3.7 there is the plot of a loss function with respect the model weights. The arrows indicate the iterative process done for the optimization.

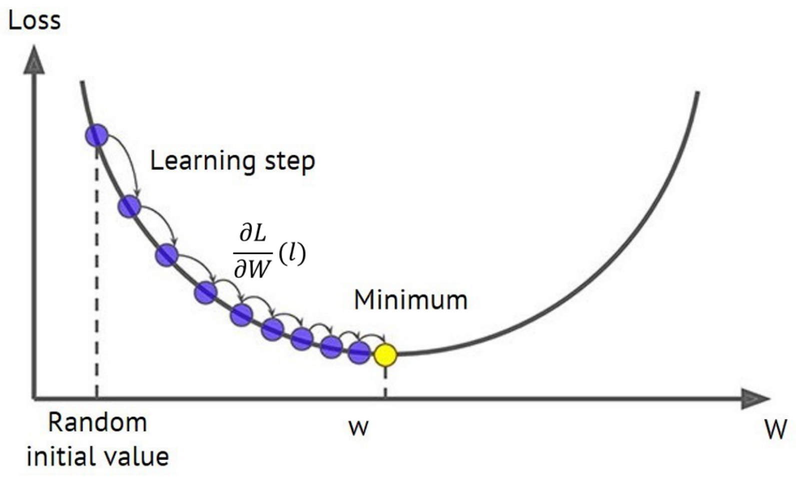


Figure 3.7: Gradient descent method.

The Multilayer Perceptron (MLP) is a artificial neural network where multiple perceptron are linked together. Figure 3.8 is an example of MLP network. Each neuron of an MLP works as explained before, but in this case the “feed-foward” and “backward propagation” involve the entire network. In this case, the first layer is called input layer. The last layer is called output layer. All middle layers of stacked neurons are known as hidden layers. In a MLP, one can decide how many layers and how many neurons per layer should be used, thus defining different architectures. Architectures with different depths may be more appropriate to certain problems, also depending on the available amount of data. As we will see in the next section, CNNs increase even more the amount of the available degrees of freedom in designing the architecture.

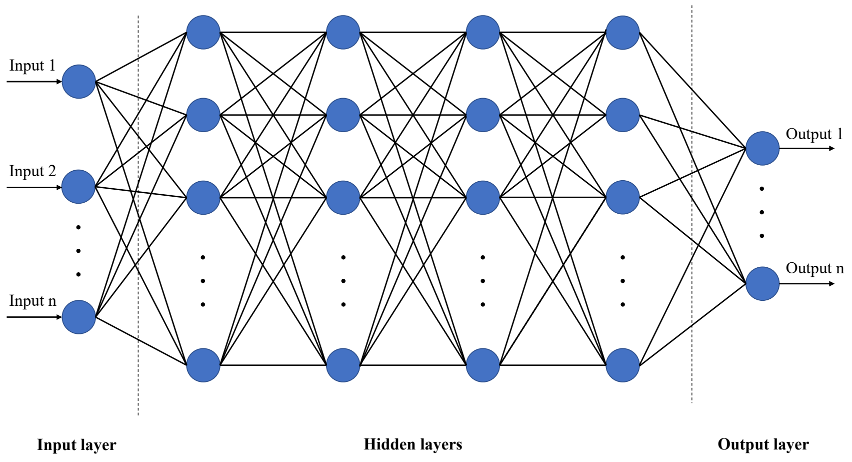


Figure 3.8: An example of Multilayer Perceptron Network. In this example we have n inputs that are processed through a series of four hidden layers. The output layer of the network contains n neurons.

3.2.3.2 Convolutional Neural Netwoks

CNNs a type of ANN inspired by the human biology of the visual cortex. Each neuron of the CNN operates in a restricted region, the receptive region. As in the case of MLP, a CNN is composed of an input layer, a variable number of hidden layers and an output layer. The hidden layer is usually composed by a convolutional layer, an activation function and a pooling layer. The result of the convolutional operation is called feature map. In case of a 2-D input signal the mathematical operation can be expressed using the following equation:

$$y(t,f) = (x * w)[t,f] = \sum_{m}\sum_{n} x[m,n] \cdot w[t-m, f-n] \tag{3.8}$$

where x is the input signal, w is the kernel or filter matrix and $*$ is the operation of convolution. The feature map obtained is then the input of the activation function layer, usually a Rectified Linear Unit (ReLU). The pooling layer instead is used to reduce the dimension of the feature map to decrease its complexity. Also in this case we need a minimize a cost function to learn the proper weights and biases. The output layer normally is a fully-connected MLP. In the image 3.9 there is an example of CNN used for image classification. In this example the input is image that we want to classify. The network is composed by a convolutional layer with ReLU as activation function followed by a pooling layer. After that we have another similar block formed by convolutional layer and ReLU followed by a pooling layer. Then we have a flatten layer and after it there is a MLP fully connected layer. The last layer is a softmax layer.

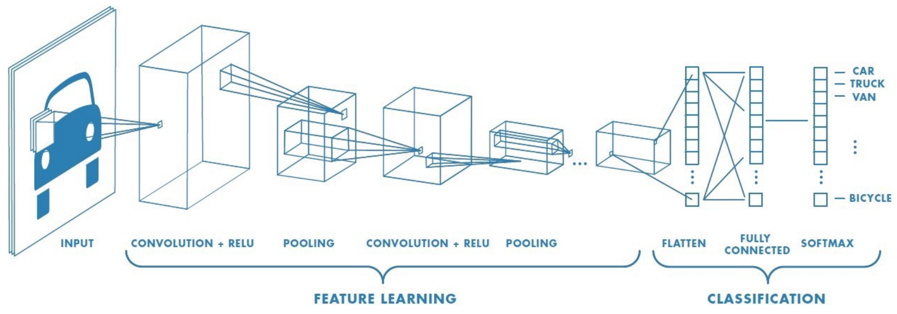


Figure 3.9: An example of CNN. In this example we have a CNN used for image classification. After the the input layer we have the first convolutional block formed by a convolutional layer with ReLU as activation function and a pooling layer. The second block is made up as the first one. After the last convolutional layer there is a Flatten layer, then a MLP fully connected layer and the output is a softmax function.

3.2.4 Clustering and K-Means

Clustering is the task of grouping a set of objects in such a way that objects in the same group, called cluster, are more similar to each other that to those in other groups. Clustering algorithms are a part of ML, specifically they are part of unsupervised learning. Unsupervised learning means to deal with unlabeled data.

A good clustering method will produce clusters with an high intra-class similarity and a low inter-class similarity. The similarity between objects is measured using a proper distance, e.g., the Euclidean distance, or the cosine distance, or any metric is proper for the specific feature space.

The K-Means is an iterative algorithm to cluster n objects into k clusters. The aim of this algorithm is to find the local maxima in each iteration. Given a predefined amount of clusters, denoted by k, K-means is carried out in three steps after the initialization:

1. Initialization: set seed points randomly.
2. Assign each object to the cluster of the nearest seed point measured with a specific distance measure.
3. Compute new seed points as the centroids of the clusters of the current partition.
4. Go back to step 2 until there are not new assignment, so objects in each cluster remain the same.

In Figure 3.10 there is a visual example of how this algorithm works. The three clusters are highlighted with different colors and for each cluster we have the centroid represented by the cross.

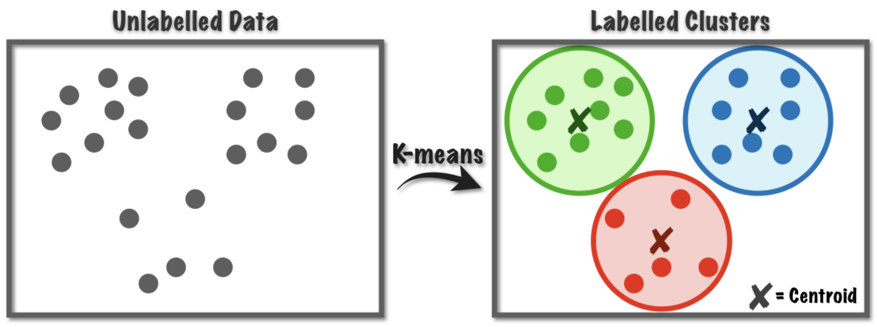


Figure 3.10: K-Means algorithm.

3.3 Conclusive Remarks

In this chapter we explained the theory of the techniques that we will use in the our work. In particular we explained:

- Two time-frequency transform: STFT and Mel spectrogram.
- A quick overview of the working principles of Neural Networks.
- How clustering, specifically K-Means, works.

Given the theory knowledge we can now introduce the problem we want to solve and the solution we adopted to solve it.

4 Splicing Detection and Localization

In this chapter we present the proposed method we developed to solve the problem of splicing detection and localization based on traces left by the recording device. At first there is a brief explanation of the problem we face. Then, the method is divided in blocks and each block is formally discussed in details.

4.1 Problem formulation

In this work we focus on the problem of detecting if an audio recording has been modified. It is very important to study deeply this problem because nowadays we can easily create fake audio recordings and change completely the original meaning of the speech tampered with. This can lead to serious problem if we think, e.g., to a politician speech.

If we want to modify the meaning of an audio recording often we add excerpts from different recordings to the original one. It is possible that this added tracks were recorded using different mobile phones, each one with its own microphone. Since each microphone leaves traces in the recording, it is possible to explore these traces to find if the audio was manipulated or not. Moreover with this presented method we were able not only to detect the presence of a tampering, but also to localize the point where the change of device was introduced.

Formally, let us take into account two speech audio recordings $x_1(n), n = 0, 1, ..., N_1-1$ and $x_2(n), n = 0, 1, ..., N_2-1$. A spliced recording $x_{tampering}(n)$ can be built by concatenating $x_1(n)$ and $x_2(n)$, i.e., by applying:

$$x_{tampered}(n) = [x_1(n), x_2(n)] \tag{4.1}$$

. At this point, we would like to highlight that if the two files $x_1(n)$ and $x_2(n)$ belong to two different microphone classes, then the microphone characteristics of $x_{tampered}(n)$should change at the $(N_1 - 1)$-th sample. In Figure 4.1 there are the two signals x_1 and x_2 respectively. The Figure 4.2 shows the concatenation of the two signals. Moreover we highlighted the splicing point too.

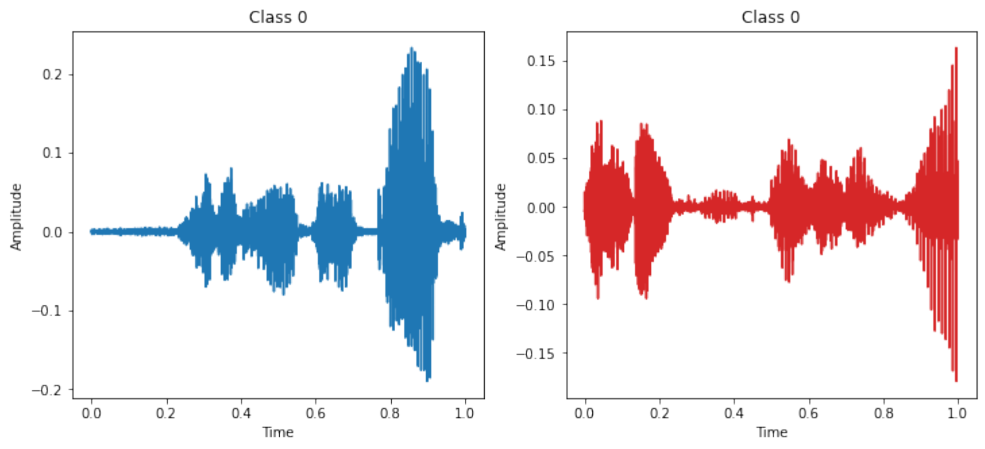


Figure 4.1: In this figure there are two examples of generic audio recordings. Each recording has class 0 and this means that they are pristine recordings.

The objective of our study is to detect if an audio recording is a composition of two different recordings or not. Formally, given a *generic* input audio recording $x(n)$, solving the detection problem is equivalent to associating a label $c \in [0, 1]$, where 0 means that the recording is pristine, whereas 1 means that the recording results from a splicing operation.

If we detect a tampering in the audio recording we also try to estimate the splicing point, i.e, an approximation of the sample index where there

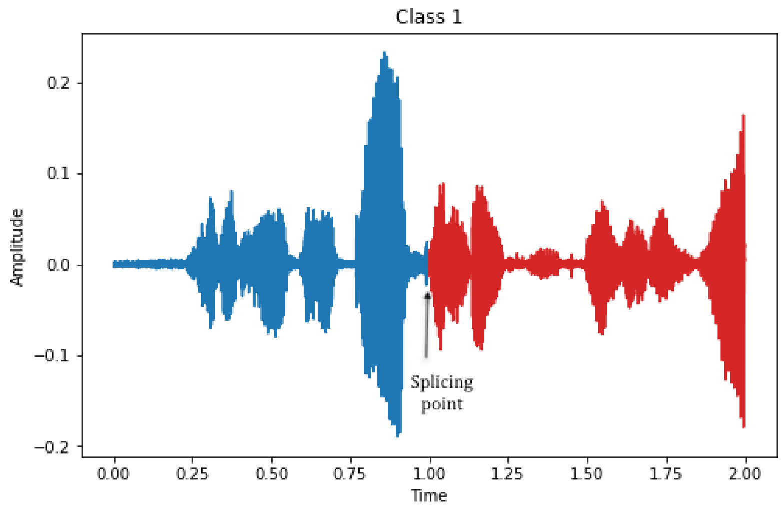


Figure 4.2: In this figure we show the concatenation between the two signals x_1 and x_2. The final result has class 1, so it means that was manipulated. With the black arrow we point out the splicing point.

the concatenation took place. Formally, this means estimating the sample position $\hat{n}$, which in principle should be coincident to N_1 in the reported example.

We will now see a general description of the method we adopted to reach this goal.

4.2 Proposed Method

In this section we discuss the general idea of the proposed method, describe the corresponding processing pipeline, and define in details every block which are composing it.

The coarse idea behind our pipeline is to use a microphone classification algorithm and try to explore it to build an identification and localization algorithm. After an initial pre-processing step, we use a modified version of a CNN architecture originally devised for microphone identification [1] to extract some features vectors, that supposedly contain the intrinsic characteristics of the input microphone, are used as input to a clustering technique, to decide if the analyzed recording is made of two different files or not. If we detect a tampering in the recording we apply a distance procedure on the features vector samples to localize the splicing cut point, an overview of the pipeline is also depicted in Figure 4.3

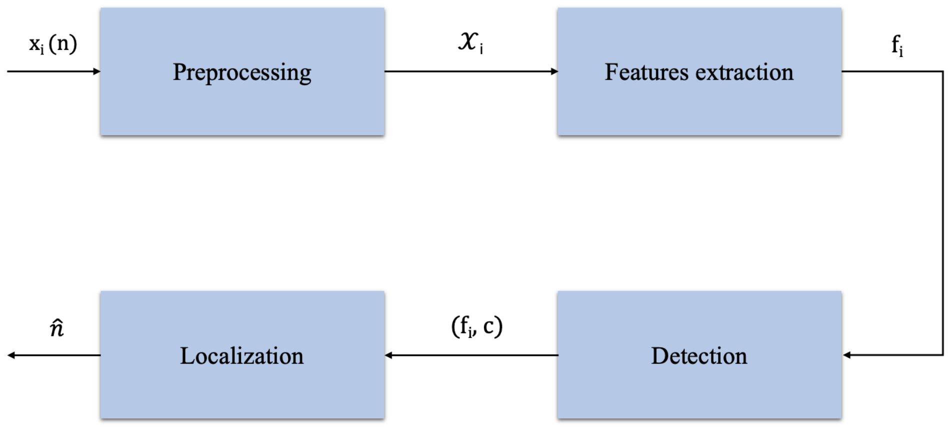


Figure 4.3: Proposed method pipeline.

4.2.1 Preprocessing

The first block of our pipeline is the preprocessing one. In this section we explain all the operations we apply to the signal before the CNN and the features extraction process.

Given that the audio recordings may have different lengths, as first operation we set a fixed maximum length of T seconds. Then, from each recording we extract N tracks, each one with a length of T/N seconds [1]. To each extracted track we apply the STFT obtaining a log magnitude spectrum. This spectrum is the starting point of the feature extraction algorithm.

Formally consider we have an audio recording as $x(n)$. The i-th track or excerpt is defined as $x_i(n)$, and $i \in 0, ..., N-1$. The STFT of the i-th track is X_i, where we omit the time and frequency indexes for the sake of notational compactness. The output of the pre-processing step is a matrix $\mathcal{X}_i$, which contains the log magnitude of the STFT:

$$\mathcal{X}_i = \log\left(\|X_i\|\right), \tag{4.2}$$

[1]If T is not an exact multiple of N the last sample should not be analyzed. However this procedure was chosen by us to increase the number of input samples but it does not influence the network.

with $i \in [1, N]$ once again denoting the track index. In order to visualize the input and the output of this stage, Figure 4.4 reports an example of a input recording, whereas Figure 4.5 shows the output of a processed track. This is the STFT of the audio file showed in Figure 4.4.

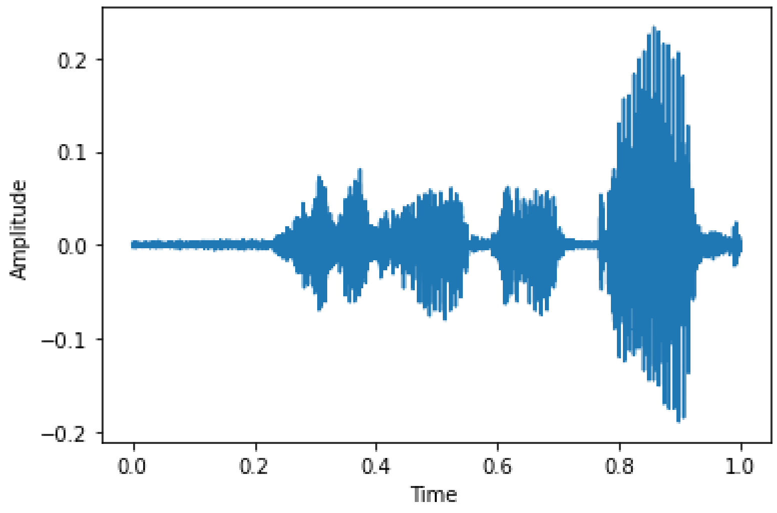


Figure 4.4: Input audio signal cut to T/N. This is the signal we used as input for our algorithm. The image represent the audio waveform of the signal. The x axis represents the time in seconds, whereas on the y axis there is the amplitude of the signal.

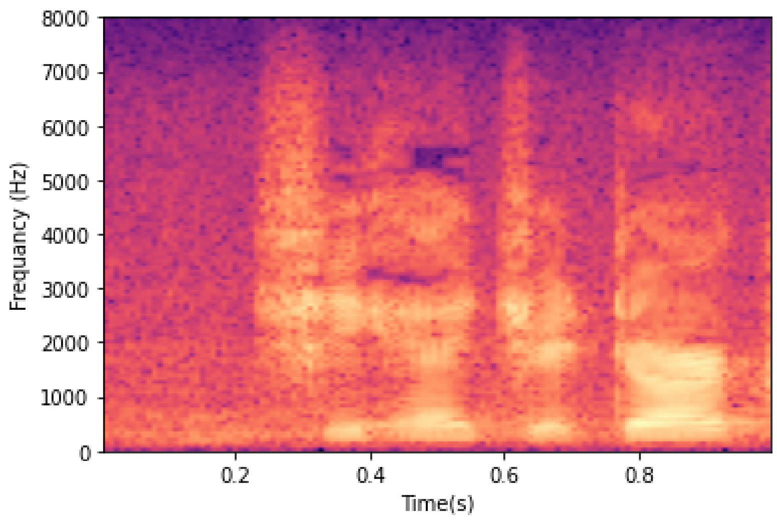


Figure 4.5: STFT of the audio signal in Figure 4.4. This is the input of our CNN from which we extract a corresponding feature vector.

4.2.2 Features Extraction

Before explaining how the features are extracted, we introduce the CNN we use to obtain these features. This CNN is based on the work presented in [1], in which the authors illustrate a possible approach to solve a microphone classification problem. The results they achieved were very successful, so we use it as a starting point for our algorithm. This type of network is trained to solve a microphone classification task, but we use the features extracted at specific point of the net. Figure 4.6 shows the whole scheme of the CNN. The first element of the network is an input layer used to adjust the size of the STFT, it add a channel to the input signal in order to have an size input compatible with the CNN. After that there are two first convolutional blocks, each one composed by this elements:

- a convolutional layer
- ReLU activation function
- MaxPooling layer

The convolutional layer takes care of extracting high-level information from the input feature map. The ReLU activation function, which is common to associate with convolutional layer, introduce the non-linearity needed by the network. The MaxPooling layer is necessary to reduce the dimension of the convolution output. The network ends with a convolutional layer and a dense layer with softmax as activation function. The softmax function is defined as:

$$\sigma(z)_i = \frac{\exp z_i}{\sum_{j=1}^{k} \exp z_j}, \tag{4.3}$$

with $i \in 1, ..., K$ and $z = (z_1, ..., z_k) \in \mathbb{R}^K$. In Figure 4.7 there are summarized the CNN layers with the related parameters, which we adapted compared to the original one in [1].

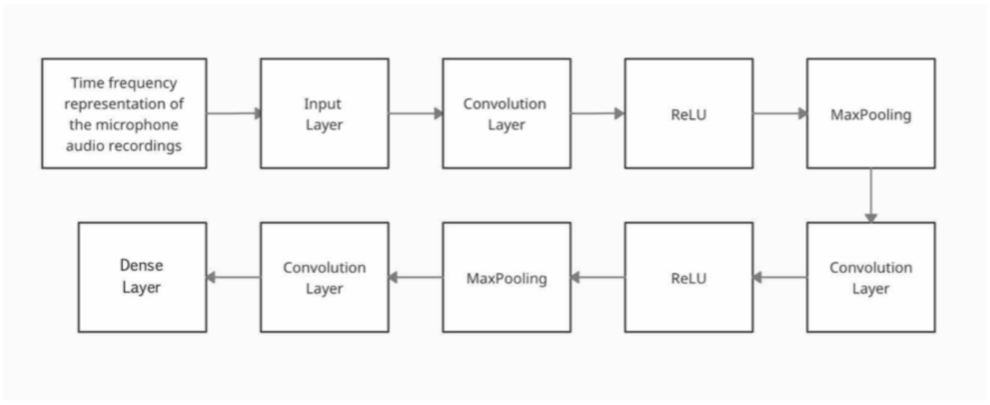
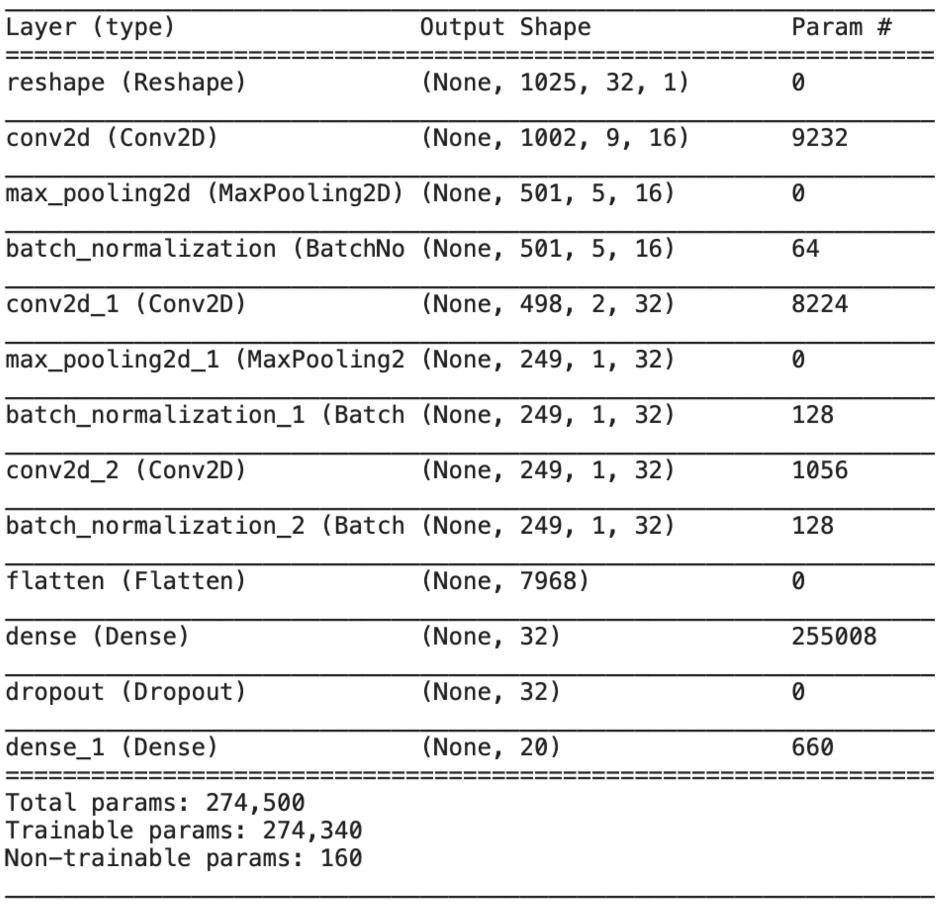


Figure 4.6: CNN used for feature extraction. In this image there are all the blocks we used to build our CNN.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 1025, 32, 1)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 1002, 9, 16)       9232
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 501, 5, 16)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 501, 5, 16)        64
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 498, 2, 32)        8224
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 249, 1, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 249, 1, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 249, 1, 32)        1056
_________________________________________________________________
batch_normalization_2 (Batch (None, 249, 1, 32)        128
_________________________________________________________________
flatten (Flatten)            (None, 7968)              0
_________________________________________________________________
dense (Dense)                (None, 32)                255008
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 20)                660
=================================================================
Total params: 274,500
Trainable params: 274,340
Non-trainable params: 160
_________________________________________________________________
```

Figure 4.7: CNN topology and parameters. In this figure there is the list of the layer of the CNN. The output we used to extract the feature is the first dense layer.

To train the CNN we use sparse categorical cross-entropy as loss measure, i.e., we effectively train the network for a microphone classification problem. After this stage, the output layer we chose to extract the features is, however, the first dense layer. The last layer of the network can be considered a one-hot encoding of the microphone label, which would not fit our task. By using as feature vector the output of the first dense layer, we instead reached a sufficient separation between the various classes and also an higher number of feature per sample, with respect the only label given as output by the last dense layer. Formally describing what we introduce before we can say that when the CNN is fed with the input $\mathcal{X}_i$ as in eq. (4.2), it returns as output the feature vector f_i with dimensionality $(1, m)$. So for each spectrogram X_i we extract a feature vector f_i having a fixed length F. We selected F as being equal to the amount of window in X_i

4.2.3 Detection

The next step is to detect if the audio recording has been tampered with by means of splicing. To do this we applied a clustering technique to the features vectors $F = f_i$ extracted by the CNN from each audio track of the input audio recording under analysis. The idea is that, a pristine recording is composed by tracks that all share very similar device information, hence feature vectors. Conversely, if an audio recording is a composition from multiple devices, at some point the feature vectors extracted in time should change.

In our method we proposed to use the K-Means algorithm. The detection pipeline can be summarize in the following steps:

- Apply the K-means clustering algorithm
- Compute the centroids of the corresponding clusters
- Compute the distance between cluster centroids
- Label the input recording as being tampered or not

In the first step we apply a K-Means algorithm on the feature vectors. From the K-Means we obtained as many clusters as we set for the algorithm. For each cluster we compute the centroid, i.e, the center of the cluster. Then we compute a distance between the found centroids, and in our study we decided to use the Euclidean distance as distance measure.

K-means assigns each feature vector f_i to a cluster. Let us consider $\mathcal{F}_k$ the set of all features belonging the cluster number k. The k-th cluster centroid can be computed as

$$\mu_k = \frac{1}{|\mathcal{F}_k|} \sum_{i:f_i \in \mathcal{F}_k} f_i, \tag{4.4}$$

where $|\cdot|$ indicates the cardinality of a set, and the sum is performed element-wise on each element of the vectors f_i. The distance between two centroids is defined as

$$d_{jk} = \|\mu_j - \mu_k\|_2, \tag{4.5}$$

where $\|\cdot\|_2$ is the $l2$ norm. In our case we set K=2 and the distance is defined as

$$d = \|\mu_1 - \mu_2\|_2 \tag{4.6}$$

The final decision is taken based on a threshold mechanism. We set a threshold and if the distance computed between the two centroids is higher than the threshold, we state that the recording has been modified.

Formally, we assign the tampering class label as

$$c = \begin{cases} 0 \text{ if } d \leq \gamma, \\ 1 \text{ if } d > \gamma. \end{cases} \tag{4.7}$$

The threshold γ has been set following a procedure explained in the next chapter.

In the following, we report the outcome of this block when applied to two examples: the case of a pristine recording, and the case of a manipulated recording obtained by splicing two different source files. For all the examples we plotted the results obtained with the K-Means, by projecting the result in a two dimensional space. The results are plotted in the cluster space, that is represented by two features. The features are directly derived from the K-Means algorithm.

In Figure 4.8 we have a block of samples belonging all to the same class. Instead in Figure 4.9 we have an example of two classes. As we can see clearly from this images in the first case we do not have a definite separation between the two clusters because all the samples are taken from the same microphone class. When we analyze a recording with two classes is very simple to identify the two clusters and the distance between the centroids is much bigger than the first example.

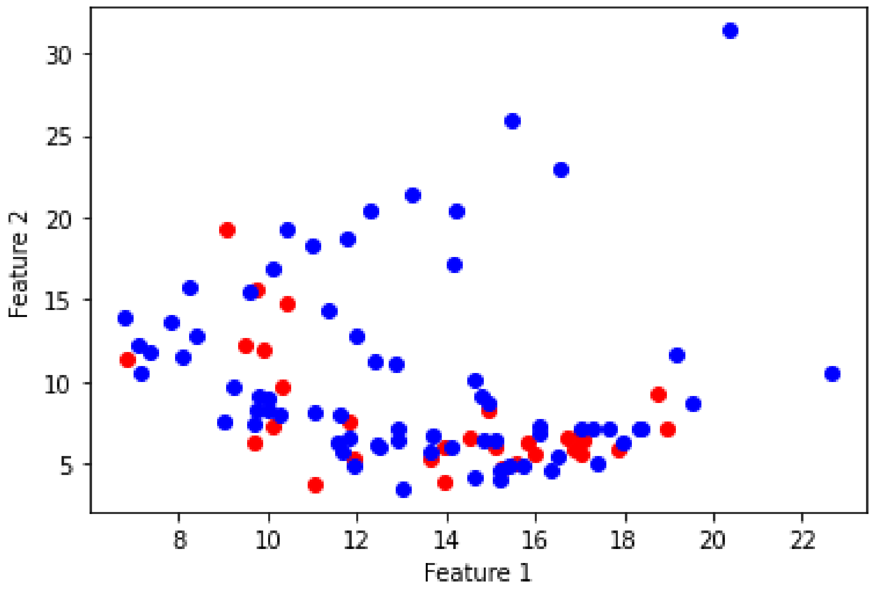


Figure 4.8: An example of clustering with same class samples. The samples are plotted in the cluster space, in fact on the axis the are the two features that describe this space. Since the samples are all from the same class we cannot identify to distinct clusters.

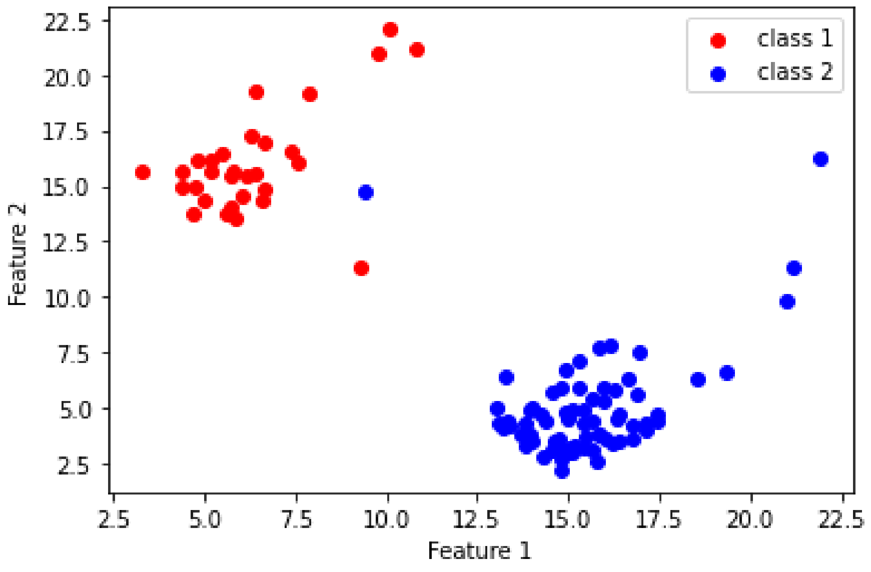


Figure 4.9: An example of clustering with two classes. In this case the classes are different and the clusters are easy to identify by sight.

4.2.4 Localization

If the analyzed signal is labeled as fake, i.e., formed by two different recordings as in eq. (4.1), the localization algorithm is applied to find the samples in which there is the change of device. Now we illustrate the proposed localization algorithm.

The starting point are the feature vectors computed by the CNN. From these vectors we compute the Euclidean distance between consecutive elements and save the distance results in an array. After we take the maximum of the distance array and the sample associated to it. The sample corresponding to the element with the highest distance will be our splicing cut point.

Formally, let us consider that we have an input recording $x_{tampered}$ defined as in eq. (4.1), i.e., by concatenating two different sources of length N_1 and N_2 and we want to find the sample index N_1.We can build the distance array y_{distance} by computing the Euclidean distance among consecutive features extracted from windows or tracks of the signals x_{tampered} under analysis. Formally, we compute

$$y_{\text{distance}}(i) = \|f_i - f_{i+1}\|_2, \; i \in \left[0, \frac{N_1 + N_2}{T} - 1\right], \tag{4.8}$$

with those being the lengths, and T being the length in sample of each analysis window, as discussed in Section 4.2.1.The $\|\cdot\|_2$ computes the $l2$ norm. The splicing point is identified in the position in which $y_{\text{distance}}(i)$ shows its maximum. The first and last 10 elements of the distance vector $y_{distance}(i)$ were discarded as explained in [29]. Formally,

$$\hat{n} = \arg\max_i \left(y_{\text{distance}}(i)\right) \cdot T. \tag{4.9}$$

The procedure can also be described with the help of some figures. The result after computing the distance vector $y_{distance}(i)$is shown in the Figure 4.10. From this vector we need to localize the maximum and its position. In the Figure 4.11 we highlighted with a red point the maximum of the sequence and its sample position will be our splicing point.

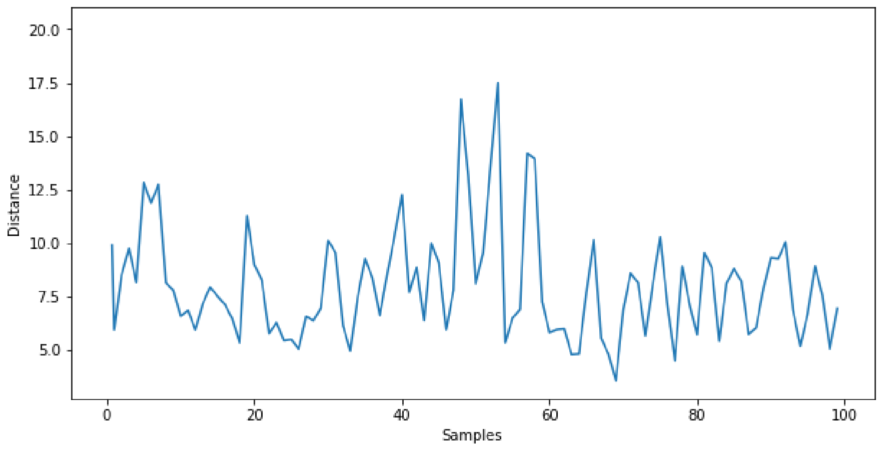


Figure 4.10: Distances between consecutive samples of the feature vector. On the x axis there is the sample number, on the y axis the distance value measured with the Euclidean distance.

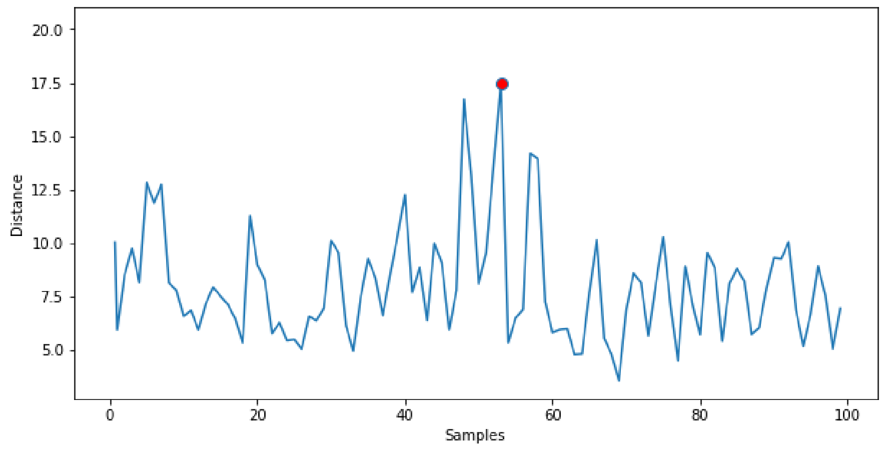


Figure 4.11: Distances between consecutive samples of the feature vector with highlighted the predicted splicing cut point. This point is obtained computing the maximum of the distance values.

4.3 Conclusive Remarks

In this chapter we introduced the formulation of the problem. Then we explained in details how we solve it with our proposed method. In particular we explained:

- The preprocessing phase
- The feature extraction
- The detection process
- The localization algorithm

In the next chapter we will discuss the parameters details and the results we obtained.

5
Results

In this chapter we introduce the dataset used in our study and the metrics adopted to measure the results. After that, we discuss the experiments we performed to tune the parameters of our method. At the end, we present the final results obtained for splicing detection and localization.

5.1 Dataset

In this section we describe the dataset adopted for the study and how we divided it for the training and test of the network.

The dataset we decided to use is the MOBIPHONE dataset [19]. MOBIPHONE is a collection of audio files, recorded with 21 cell phones model from seven different brands. For each phone there are 24 speakers from TIMIT [31], 12 male and 12 female speakers. For each speaker we have 10 sentences, so at the end the dataset is formed by 4800 utterances. The TIMIT is a corpus designed to provide speech data for acoustic-phonetic studies and for the development and evaluation of automatic

Table 5.1: List of MOBIPHONE devices

Class Name	Brand and Model
0	Apple iPhone 5
1	HTC desire c
2	HTC Sensation xe
3	LG GS290
4	LG L3
5	LG Optimus L5
6	LG Optimus L9
7	Nokia 5530
8	Nokia C5
9	Nokia N70
10	Samsung e1230
11	Samsung E2121B
12	Samsung E2600
13	Samsung Galaxy GT-I9100 s2
14	Samsung Galaxy Nexus S
15	Samsung GT-I8190 mini
16	Samsung GT-N7100
17	Sony ericsson c501i
18	Sony ericsson c902
19	Vodafone joy 845

speech recognition systems. We followed the same approach explained in [32] and [33] in which they excluded the device “Samsung s5830i" due to the small duration of its recording. In Table 5.1, we report the complete list of brands and cell phone models used in this study. There is also the class label tag associated to each mobile phone class. We used also a training/test split to train and then validate our CNN. In our work we use 67% for training and 33% for test. We split the dataset in this way: for each class we took the first 16 speakers for the training and the remaining 8 for the test class. In this way we a sufficient number of samples for the training and also all the class were represented in the test phase. During the training phase we divided the training set into training and and validation set with a proportion 67% to 33% respectively. The validation set is necessary to get an evaluation of the system and tune the model hyper-parameters.

5.2 Metrics

In the following we introduce the metrics adopted in this work to evaluate our results.

5.2.1 Balanced Accuracy

Before defining the balanced accuracy we need to defined the following quantities:

- True Positive (TP) = Recordings from different devices detected as such.
- True Negative (TN) = Recordings from the same devices detected as such.
- False Positive (FP) = Recordings from the same devices detected as being from different ones.
- False Negative (FN) = Recordings from different devices detected as being from the same device.

The Figure 5.1 shows graphically what we described before.

Balanced accuracy is a metric that it is used when evaluating how good a supervised classifier is. It is especially useful when the classes are not balanced, i.e. one of the two classes appears a lot more often than the other. This happens often in many settings such as anomaly detection. Balanced accuracy is based on two more simple metrics: the sensitivity, also known as true positive rate or recall, and the specificity, also known as true negative rate. Sensitivity is defined as:

$$Sensitivity = \frac{TP}{TP + FP} \tag{5.1}$$

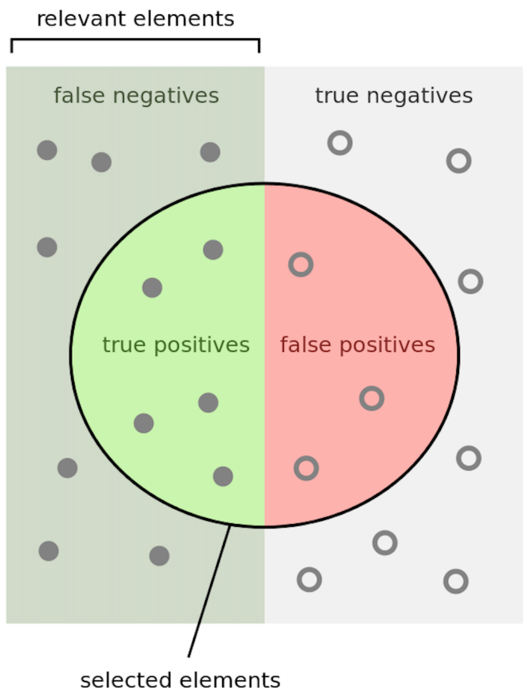


Figure 5.1: Definition of TP, TN, FP, FN. Positive samples are denoted by full circles, while negative samples are denoted by empty circles. In the picture we highlighted true positive with a green background, and false positives with a red one.

and Specificity is:

$$Specificity = \frac{TN}{TN + FP} \tag{5.2}$$

The balanced accuracy is simply the arithmetic mean of the two, so

$$Balanced\ accuracy = \frac{Sensitivity + Specificity}{2}. \tag{5.3}$$

5.2.2 Precision

Precision indicates the portion of relevant instances among the retrieved instances. It is computed as :

$$Precision = \frac{TP}{TP + FP} \tag{5.4}$$

5.2.3 F1-score

The F-score measure the test accuracy. It is derived from the precision and the recall. In particular the F1-score is the harmonic mean of precision and recall. Formally it is expressed as:

$$F_1 = \frac{2}{Recall^{-1} + Precision^{-1}} \tag{5.5}$$

5.2.4 ROC

The Receiver Operating Characteristic (ROC) curve is a graphical plot that shows the ability of a binary classifier as its threshold is varied. This curve is created by plotting the Sensitivity against the False Positive Rate (FPR). The FPR is obtained as $1 - Specificity$ and represents the fraction of samples of the negative class detected as positives.

The best possible prediction method would give a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity and 100% specificity. The (0,1) point is also called a perfect classification. A random guess would give a point along a diagonal line from the left bottom to the top right corners. The diagonal splits the ROC space in two parts. Points above the diagonal represent good classification results, points below the line represent bad results. The quantity Area Under the Curve (AUC) is the area under the ROC and represent the accuracy of the binary classifier. In Figure 5.2 there is an example of ROC.

5.2.5 Confusion Matrix

The Confusion Matrix is a specific table that allows to visualize the performance of a multi-class classifier. Each column of the matrix represents the predicted class instances. whereas each row represents the ground truth instances. The name derives from the fact that this table makes it easy to understand whether the system is confusing two classes.

In Figure 5.3 there is an example of a confusion matrix. In this case the values of the table have been normalized, so each value represents a percentage of correct predictions with respect to the total amount of samples of that class.

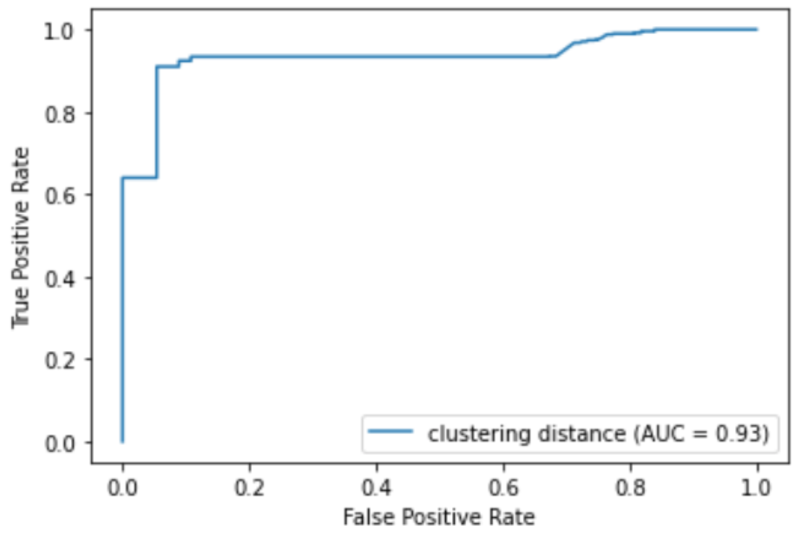


Figure 5.2: An example of ROC curve. On the x axis there is the FPR and on the y axis there is the True Positive Rate (TPR). The AUC value is specified in the right bottom corner of the image.

5.3 Experimental Setup

In this section we explain the parameters we chose in our pipeline. First of all, as explained in Section 4.2.1, the audio recordings have all different length so we decided to set $T = 20$s. Then the N parameter was set to 20, so at the end we ended up with 20 tracks for speaker each one with a duration of 1s.

The sampling frequency was equal for all the classes, so we decide to use the default one that is $16KHz$.

Then we set the parameters of the STFT. The STFT was computed with the default parameters, so with 2048 N_{FFT} points, hanning window as window function and no overlap between consecutive windows. After this stage we got as result a 2-D array, with size of $(1025, 32)$.

After that there is the CNN. The parameters of the network are summarized in the Figure 4.7. For the first block the convolutional layer has a kernel size of $(24, 24)$ and the padding layer has a stride equal to $(2, 2)$ and the padding option is set to “same”. For the second convolution block we set a kernel size of $(4, 4)$ for the convolutional layer whereas the pooling layer is equal to the previous one. The last convolutional layer has

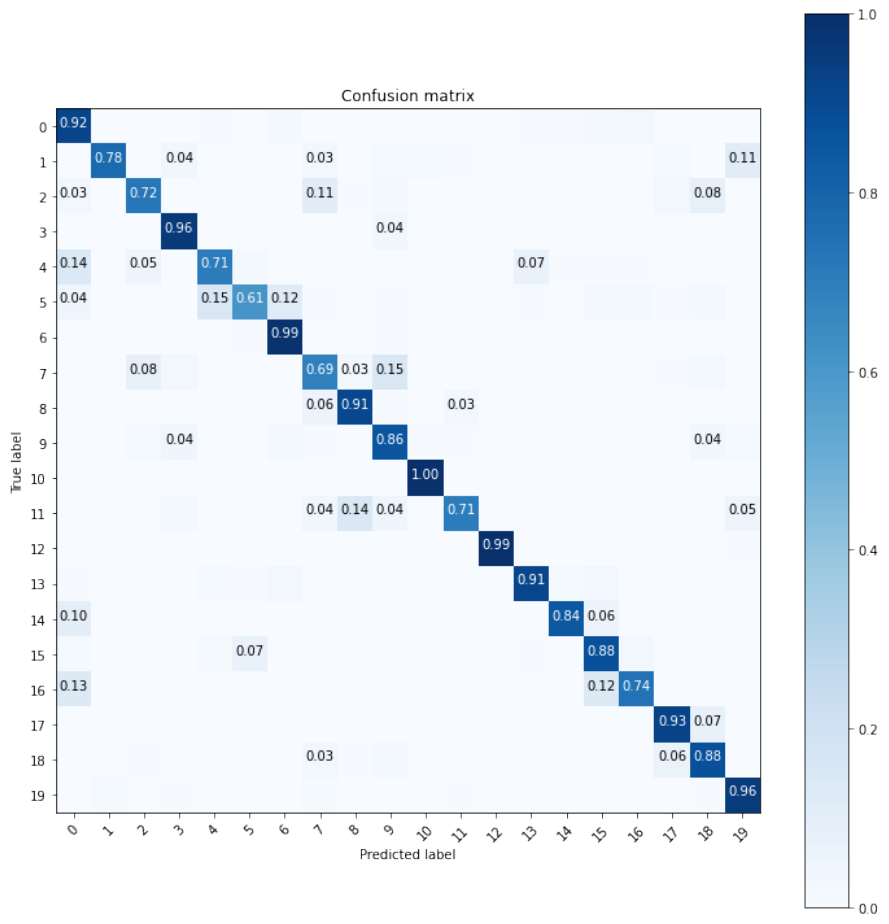


Figure 5.3: An example of Confusion Matrix with multiple class. Each cell represent the percentage of correct prediction with respect to the the total sample number of that class. In this example the values under a certain threshold have been discarded to have a better readability of the matrix.

a kernel of $(1, 1)$. In the first Dense layer we add also a kernel regularizer, in particular an L2 regularizer with a penalty of 0.01. The Dropout is equal to 0.3. The training was done on the training set, composed as explained before by 16 speakers per class. The optimizer used is the RMSPprop with a learning rate set to 0.0001 [34]. The number of epochs was set to 100. The loss function to minimize is the sparse categorical cross-entropy function. As explained in Section 4.2.2 the network has been trained to solve a microphone classification task.

We used also three callbacks function that are:

- Model Checkpoint
- Reduce Learning Rate on Plateau

- Early Stopping

The “Model Checkpoint” callback is used to save the model when the results get better. The quantity we monitor was the validation accuracy.

The “Reduce Learning Rate on Plateau” callback is useful to reduce the value of the learning rate when there are not progress in the learning. Also in this case the monitored quantity was the validation accuracy. The parameters used are the factor equals to 0.1 and the patience set to 2.

The “Early Stopping” callback is a callback function which allows to stop the training when there are not improvements in the monitored quantity. The monitored quantity was the validation loss with a patience of 15. The patience is the number of epochs with no improvement after which training will be stopped.

Figure 5.4 summarizes the training process. It represents how the accuracy and loss, both for training and validation, change along the number of epochs.

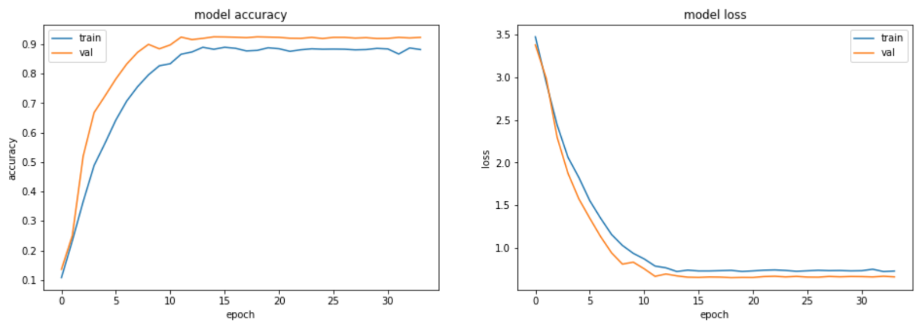


Figure 5.4: Training curves: Accuracy and loss of the training stage. The curves show the training process both for train set and for validation set. The horizontal axis corresponds to the epoch number, while the vertical corresponds to the the accuracy or the loss value, as appropriate.

5.4 Experiments

In this section we explain how we selected the CNN to use and how we computed the threshold γ we adopted in our algorithm.

5.4.1 Experiment 1: The CNN for feature extraction

In order to choose the best microphone classification networks, we did some experiment on different architectures and combinations. We tried two networks, the first one taken by Baldini et al. [1] and the second one by Zeighidour et al. [2]. The first one is the network we decided, with hyper-parameters set as discussed in the previous section. The second CNN is slightly different from the first one because it has more layer and a higher number of hyper-parameters, which we set as explained in [2]. The results obtained by each network and respective hyper-parameter configuration are discussed in the following with the help of a confusion matrix and the final classification report. The results presented here have been computed on the test set.

The first result we show is the output of the CNN introduced in [1] and the log magnitude of the STFT as input. With this combination we obtained the results shown in Figure 5.5 and Table 5.2:

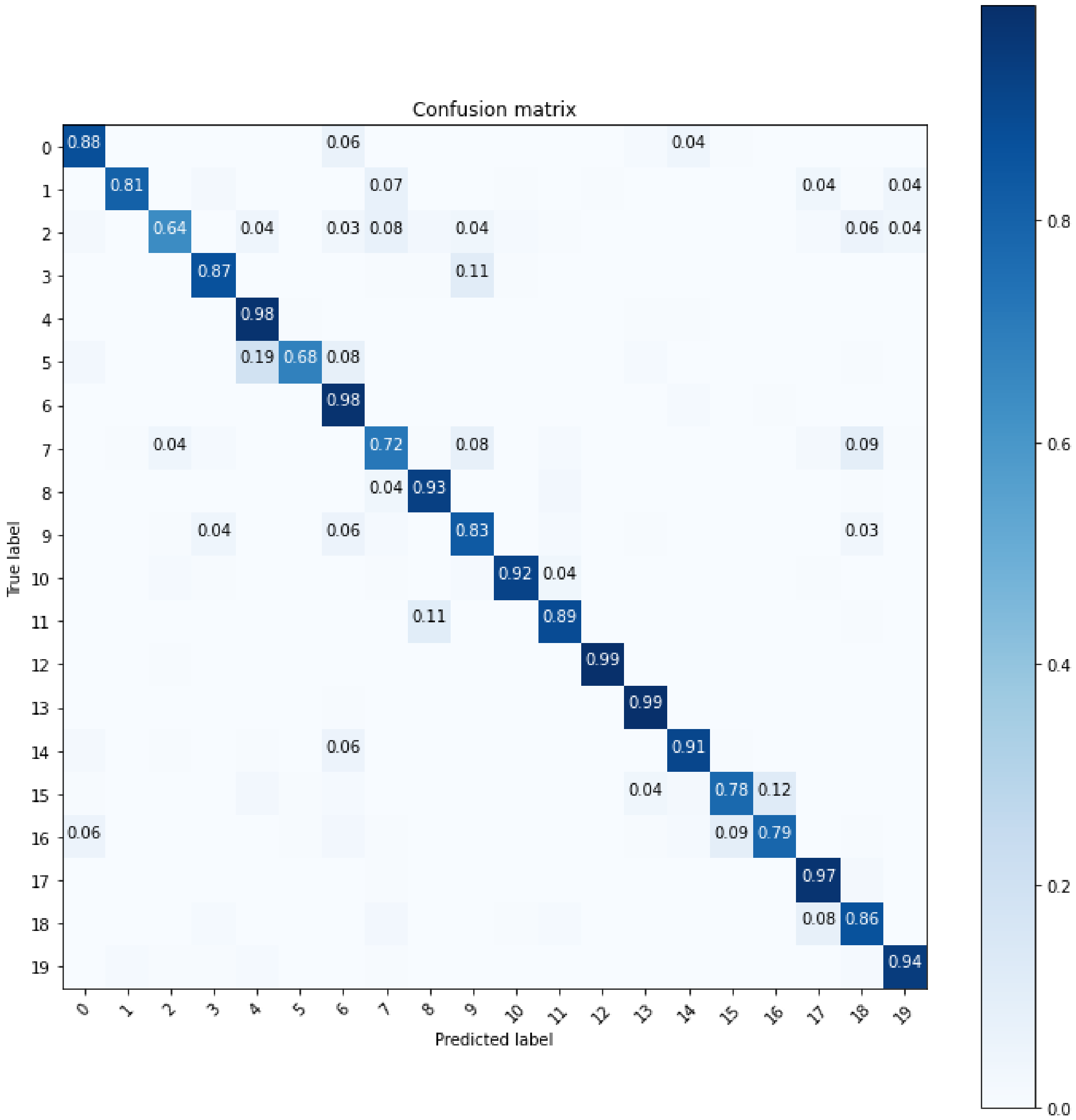


Figure 5.5: Confusion Matrix of CNN [1] with STFT

Table 5.2: Classification report of CNN [1] with STFT. The support column represent the number of true positive occurrences in that class. The accuracy results show both macro averaged and weighted average accuracy. The macro average mediates the unweighted mean per label, so it does not take into account a misbalance in the labels. The weighted average mediates the support-weighted mean per label.

Class Name	precision	recall	f1-score	support
0	0.88	0.86	0.87	160
1	0.96	0.85	0.90	160
2	0.93	0.62	0.75	160
3	0,95	0.88	0.91	160
4	0.90	0.96	0.93	160
5	0.93	0.70	0.80	160
6	0.82	0.98	0.89	160
7	0.79	0.69	0.74	160
8	0.91	0.90	0.91	160
9	0.70	0.91	0.79	160
10	0.97	0.89	0.93	160
11	0.84	0.91	0.87	160
12	0.98	0.99	0.98	160
13	0.93	0.97	0.95	160
14	0.89	0.91	0.90	160
15	0.84	0.87	0.85	160
16	0.77	0.74	0.76	160
17	0.89	0.94	0.91	160
18	0.81	0.88	0.94	160
19	0.87	0.99	0.92	160

accuracy			0.87	3200
macro avg	0.88	0.87	0.87	3200
weighted avg	0.88	0.87	0.87	3200

Then we tried the approach explained in [2], which consists on a deeper net and a log Melspectrogram as input[1]. The final results are showed in Figure 5.6 and Table 5.3:

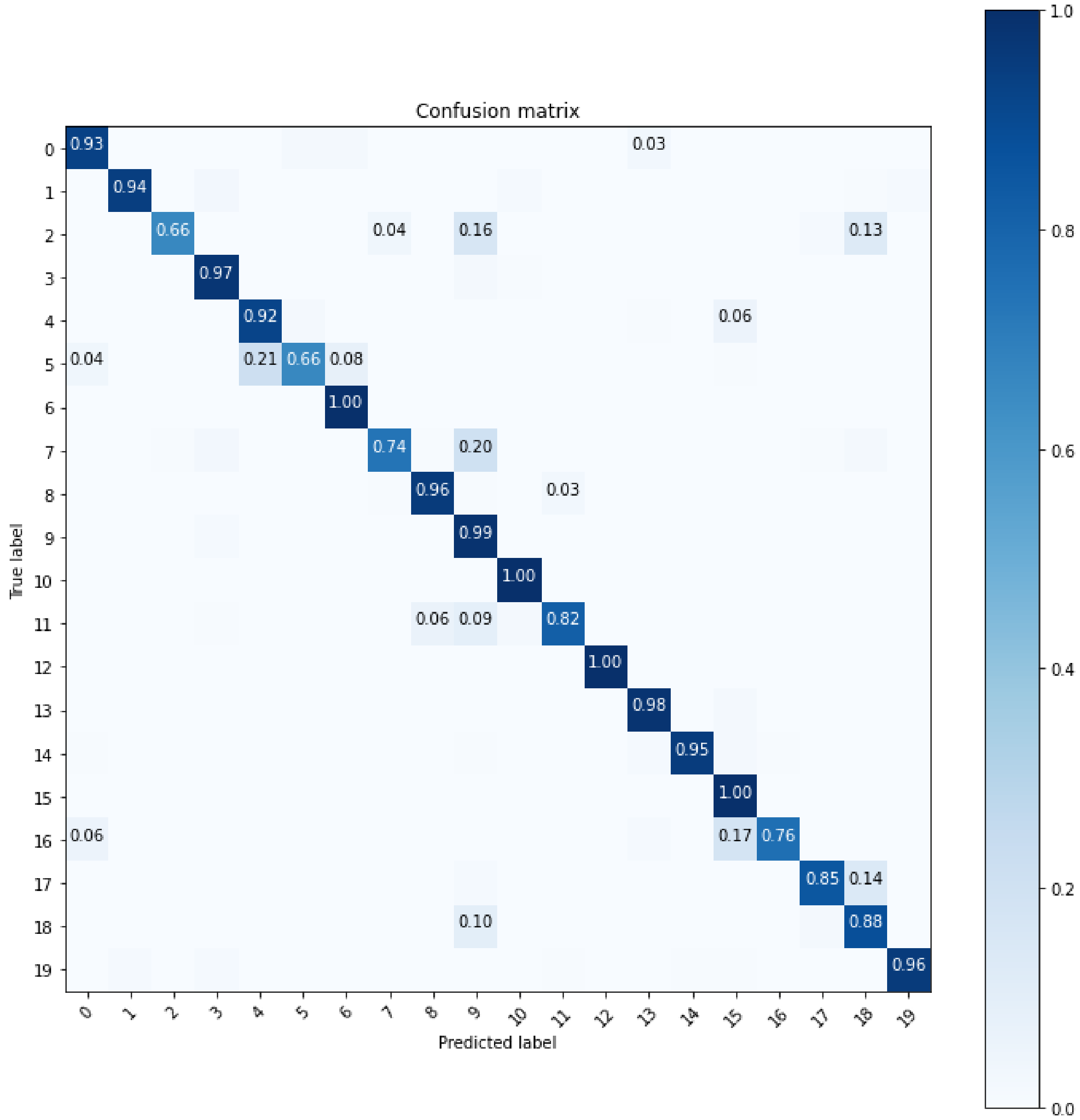


Figure 5.6: Confusion Matrix of CNN [2]

[1]See Section 3.1.2.

Table 5.3: Classification report of CNN [2]

	precision	recall	f1-score	support
0	0.91	0.96	0.93	160
1	0.97	0.93	0.95	160
2	0.98	0.68	0.80	160
3	0,99	0.95	0.97	160
4	0.81	0.92	0.86	160
5	0.95	0.70	0.81	160
6	0.95	1.00	0.97	160
7	0.87	0.77	0.81	160
8	0.95	0.95	0.95	160
9	0.68	1.00	0.81	160
10	0.98	1.00	0.99	160
11	0.95	0.89	0.92	160
12	1.00	1.00	1.00	160
13	0.96	0.96	0.96	160
14	0.99	0.94	0.96	160
15	0.81	1.00	0.90	160
16	0.92	0.76	0.83	160
17	0.96	0.88	0.92	160
18	0.73	0.89	0.81	160
19	0.96	0.91	0.94	160

accuracy			0.90	3200
macro avg	0.92	0.90	0.90	3200
weighted avg	0.92	0.90	0.90	3200

Lastly, we tried an hybrid approach, with the first network using a log melspectrogram as input. The results of this approach are summarize in Figure 5.7 and Table 5.4:

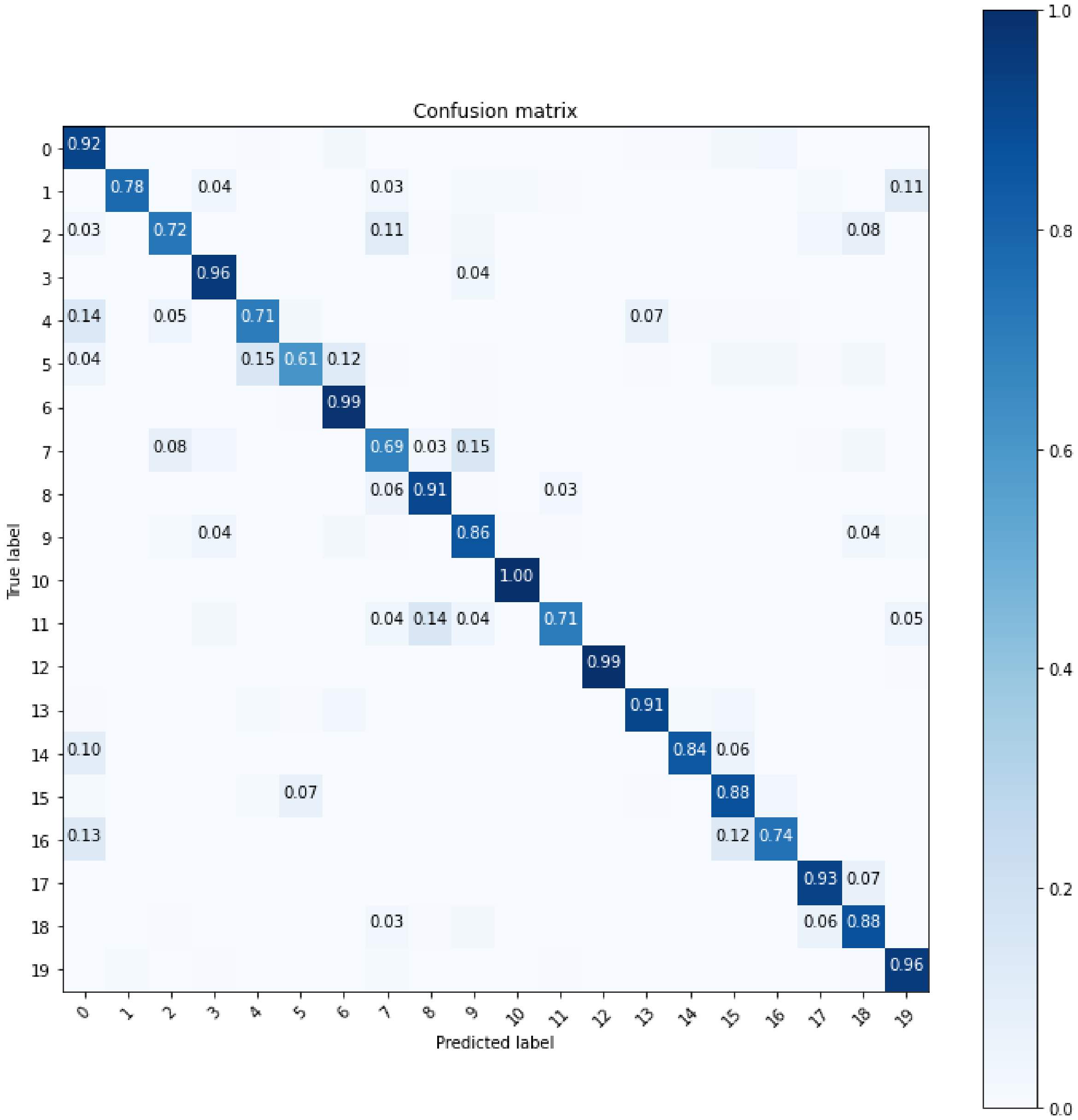


Figure 5.7: Confusion Matrix of CNN [1] with log Mel Spectrogram

Table 5.4: Classification report of CNN [1] with log Mel Spectrogram

	precision	recall	f1-score	support
0	0.84	0.89	0.86	160
1	0.99	0.80	0.89	160
2	0.88	0.77	0.82	160
3	0.91	0.98	0.95	160
4	0.80	0.79	0.80	160
5	0.79	0.62	0.70	160
6	0.82	0.96	0.88	160
7	0.80	0.76	0.78	160
8	0.88	0.95	0.92	160
9	0.71	0.90	0.79	160
10	0.99	0.97	0.98	160
11	0.95	0.81	0.88	160
12	1.00	0.99	1.00	160
13	0.86	0.95	0.90	160
14	0.99	0.85	0.91	160
15	0.81	0.89	0.85	160
16	0.92	0.83	0.87	160
17	0.91	0.85	0.88	160
18	0.76	0.86	0.80	160
19	0.90	0.97	0.93	160

accuracy			0.87	3200
macro avg	0.88	0.87	0.87	3200
weighted avg	0.88	0.87	0.87	3200

As we can observe from these results, all the combinations tested performed well on our dataset. So the choice of the network we adopted was taken according to the trade-off between results and computational costs. In fact the CNN of the paper [2] allowed us to reach an higher accuracy with respect the other two, but the computational cost associated with this net was very high and the gain in accuracy did not justify this cost.

So we decided to use the CNN as explained in [1], with log magnitude of the STFT as input: we preferred to follow the approach explained in the paper since the results were the same both for STFT and log Mel.

5.4.2 Experiment 2 : The Threshold for Splicing Detection

In order to detect the presence of splicing, we threshold the quantity d computed as in eq. (4.6) representing the distance between features from different devices. To do so, we computed the value d for multiple track pairs (belonging or not to the same device), and we evaluated the results by means of ROC curves and AUC.

To perform this evaluation, we had to work with pairs of classes, since the ROC measures a binary classifier. So we built all the possible pairs among the various classes. Then we built the distance array $y_{distance}(i)$, as explained in eq. (4.8).

From the ROC curve we extracted the threshold as the one that maximized the AUC. The value we obtained for the threshold γ used in eq. (4.7) is 12.2381.

After setting this threshold we computed the balanced accuracy for each pair, in order to verify that this approach was adequate for each class. From this analysis we extracted the Table 5.5, that summarize the value of Balanced Accuracy computed pairwise. This test had good results, in fact for every pair of classes we were able to get an accuracy higher than 70%. The minimum results are for the class 15, and in general the value is low for classes from the same brand. The maximum value we got is 97% of accuracy, which almost perfect accuracy.

Table 5.5: Balanced accuracy computed pairwise. Each cell represents the balanced accuracy with which the two classes are distinguished from each other by analyzing the distance between the features

CLASSES	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		0,9308	0,7852	0,9233	0,7834	0,9417	0,7772	0,9137	0,9291	0,7955	0,9705	0,9366	0,7152	0,9449	0,7709	0,8866	0,8027	0,9459	0,9677	0,9031
1	0,9308		0,7837	0,8245	0,8092	0,9280	0,8134	0,8721	0,9436	0,8092	0,7723	0,8571	0,9048	0,9372	0,9580	0,7208	0,7866	0,8862	0,9436	0,7150
2	0,7852	0,7837		0,9121	0,7791	0,9158	0,7895	0,7816	0,9229	0,8497	0,9489	0,9518	0,8265	0,9455	0,9664	0,7532	0,8591	0,9137	0,8014	0,8351
3	0,9233	0,8245	0,9121		0,8397	0,8793	0,8601	0,8775	0,9020	0,7146	0,7533	0,9377	0,7504	0,9378	0,9545	0,7523	0,8511	0,9479	0,9509	0,9070
4	0,7834	0,8092	0,7791	0,8397		0,9617	0,8096	0,8715	0,9378	0,8345	0,9419	0,9702	0,8144	0,9311	0,9728	0,7314	0,8173	0,9257	0,8877	0,9046
5	0,9417	0,9280	0,9158	0,8793	0,9617		0,8278	0,9413	0,8428	0,8818	0,9664	0,9040	0,7617	0,9362	0,8791	0,7235	0,8137	0,8506	0,9414	0,9508
6	0,7772	0,8134	0,7895	0,8601	0,8096	0,8278		0,9337	0,9410	0,8679	0,9621	0,9833	0,9470	0,9453	0,9097	0,7284	0,8317	0,9140	0,9327	0,9473
7	0,9137	0,8721	0,7816	0,8775	0,8715	0,9413	0,9337		0,8326	0,8665	0,8233	0,7594	0,8419	0,9132	0,9496	0,7506	0,7669	0,7129	0,7204	0,7076
8	0,9291	0,9436	0,9229	0,9020	0,9378	0,8428	0,9410	0,8326		0,7731	0,9138	0,7654	0,7679	0,9437	0,9150	0,7589	0,8405	0,9391	0,8495	0,9279
9	0,7955	0,8092	0,8497	0,7146	0,8345	0,8818	0,8679	0,8665	0,7731		0,9421	0,9215	0,7506	0,8961	0,9545	0,7256	0,7968	0,9414	0,8228	0,9171
10	0,9705	0,7723	0,9489	0,7533	0,9419	0,9664	0,9621	0,8233	0,9138	0,9421		0,8505	0,7344	0,9030	0,9513	0,7393	0,8139	0,9542	0,8905	0,8268
11	0,9366	0,8571	0,9518	0,9377	0,9702	0,9040	0,9833	0,7594	0,7654	0,9215	0,8505		0,7692	0,8902	0,9775	0,7205	0,8261	0,8114	0,8206	0,8761
12	0,7152	0,9048	0,8265	0,7504	0,8144	0,7617	0,9470	0,8419	0,7679	0,7506	0,7344	0,7692		0,7831	0,7734	0,7865	0,7144	0,8344	0,8734	0,8718
13	0,9449	0,9372	0,9455	0,9378	0,9311	0,9362	0,9453	0,9132	0,9437	0,8961	0,9030	0,8902	0,7831		0,9443	0,9284	0,8063	0,8990	0,9707	0,9145
14	0,7709	0,9580	0,9664	0,9545	0,9728	0,8791	0,9097	0,9496	0,9150	0,9545	0,9513	0,9775	0,7734	0,9443		0,9079	0,8252	0,9505	0,9570	0,9341
15	0,8866	0,7208	0,7532	0,7523	0,7314	0,7235	0,7284	0,7506	0,7589	0,7256	0,7393	0,7205	0,7865	0,9284	0,9079		0,7828	0,9232	0,9461	0,9022
16	0,8027	0,7866	0,8591	0,8511	0,8173	0,8137	0,8317	0,7669	0,8405	0,7968	0,8139	0,8261	0,7144	0,8063	0,8252	0,7828		0,9556	0,9411	0,9093
17	0,9459	0,8862	0,9137	0,9479	0,9257	0,8506	0,9140	0,7129	0,9391	0,9414	0,9542	0,8114	0,8344	0,8990	0,9505	0,9232	0,9556		0,8190	0,8699
18	0,9677	0,9436	0,8014	0,9509	0,8877	0,9414	0,9327	0,7204	0,8495	0,8228	0,8905	0,8206	0,8734	0,9707	0,9570	0,9461	0,9411	0,8190		0,8742
19	0,9031	0,7150	0,8351	0,9070	0,9046	0,9508	0,9473	0,7076	0,9279	0,9171	0,8268	0,8761	0,8718	0,9145	0,9341	0,9022	0,9093	0,8699	0,8742	

5.5 Detection and Localization

5.5.1 Clustering

After determining the optimal threshold and verifying that it was adequate for all the classes we tested the clustering algorithm.

As explained in Section 4.2.3, the clustering is based on a K-Means algorithm and was set to cluster the input in two groups. This because we built the testing array with two different class in each testing sample, but this can ideally be generalized to a higher class clustering. The test was done on 2000 class pairs, with the pairs chosen randomly. The outcome of the clustering algorithm is showed in Table 5.6 and Figure 5.8

Table 5.6: Result K-Means. The 0 class stands for same class samples, the 1 class is for different class samples. In the table there are the results for each metric and the final balanced accuracy.

	precision	recall	f1 score	
0	0.75	1.00	0.86	
1	1.00	0.67	0.80	
Balanced accuracy				0.834

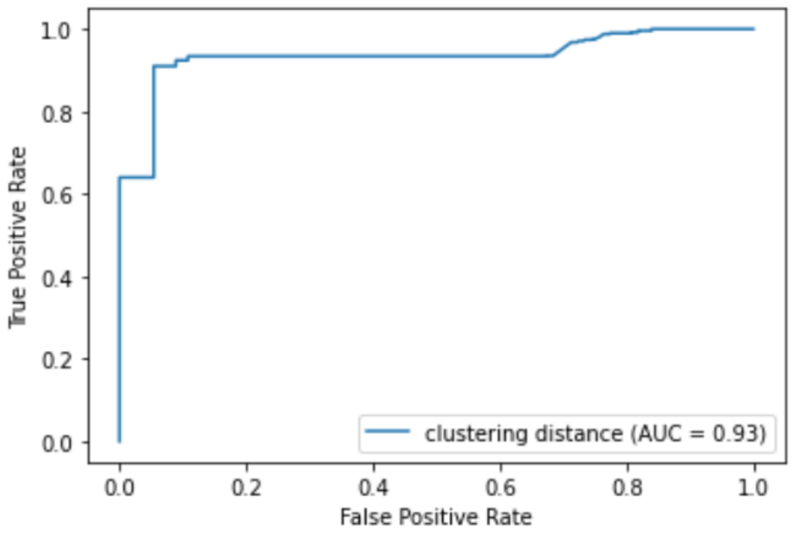


Figure 5.8: ROC of the clustering algorithm. This curve was obtained using the centroids distance and the groud truth labels.

Since with the pipeline explained in Section 5.4.2 we maximized the AUC of the ROC curve but not the accuracy, we tried also different value of thresholds for the clustering algorithm. We noticed that if we work around the selected threshold, but we lower its value about the 10% we can achieve better results in terms of accuracy. In fact in this way we were able to get the results showed in Table 5.7 and Figure 5.9.

Table 5.7: Result K-Means with modified threshold. The 0 class stands for same class samples, the 1 class is for different class samples. This is the best result achieved and was obtained by lowering the threshold of 10%.

	precision	recall	f1 score	
0	0.96	1.00	0.98	
1	1.00	0.96	0.98	
Balanced accuracy				0.9795

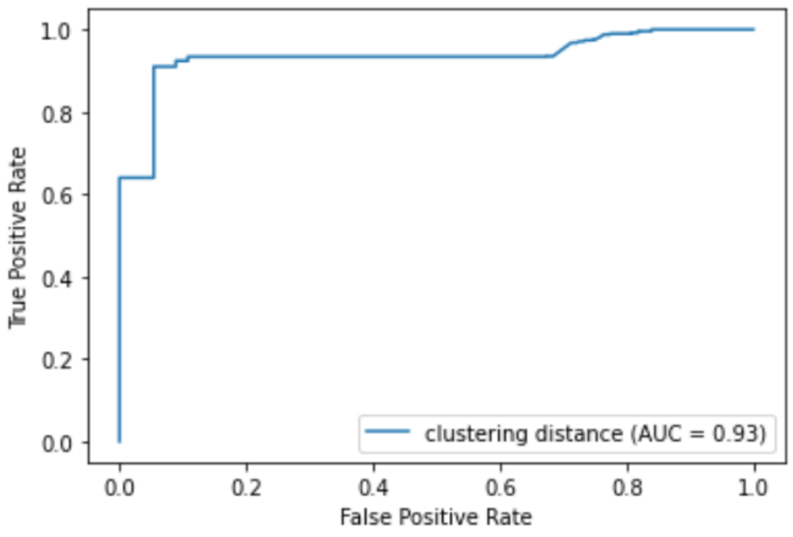


Figure 5.9: ROC of the clustering algorithm with modified threshold. This results is equal to the Figure 5.8 beacuse the cluster distances do not depend on the threshold value.

5.5.2 Localization

Once the tampering is detected we need to find the splicing point. To do this we used the localization algorithm explained in the Section 4.2.4.

To evaluate how well the localization works we set an error measure as the difference between the predicted point and the true splicing cut point. The error is defined as:

$$e = n - \hat{n}, \tag{5.6}$$

where n is the true splicing point and $\hat{n}$ is the predicted one. To visualize the performance we show the histogram of this error measure in Figure 5.10

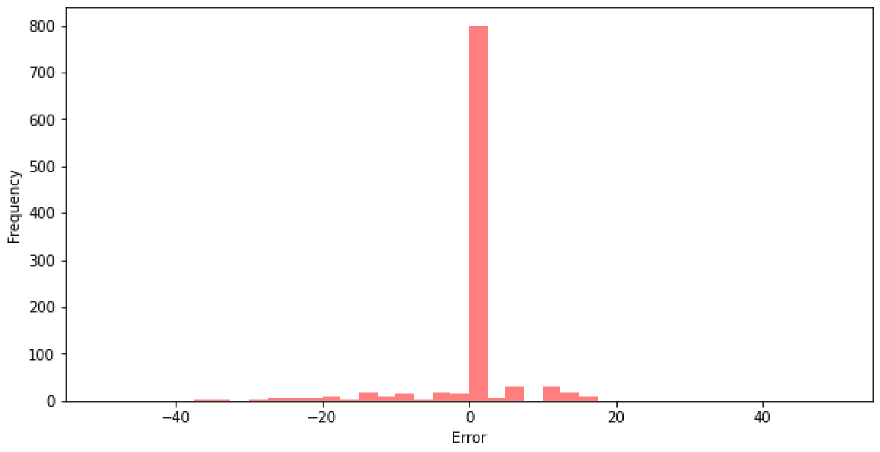


Figure 5.10: Histogram of the error measure. This plot represent the error frequency in a test of 1000 samples. The error was computed as the difference between the predicted splicing point and the true one.

The horizontal axis represents the localization error, divided in bins. The vertical axis represents the occurrence. Each bin represents the distance between the true and the predicted point. Since each point is a frame of the original STFT, this distance can be translate also in seconds. So we can measure how far in seconds we are from the correct prediction. Every bin is approximately 30ms. So as we can notice from the histogram the prediction is almost always correct, but in the worst case we have an error around 0.6s. We need to specify that to compute the distance between the consecutive samples we discarded the first and last 10 bins as suggested in [29].

5.5.3 Additional Results

In this section we present some additional results about clustering and localization based on some aspect we wanted to analyze. For example we investigated the differences between mobile phone classes of the same brand and classes of different brands. In Figure 5.11 we have the clustering results for classes 10 and 11. The two classes are from the same brand, Samsung, and in fact their clusters are not so well separated. We assumed they used the same microphone technology on the different devices and this results in a similarity between the two different classes. If we observe Figure 5.12 we can see two completely separated clusters in the cluster space. The two clusters are derived from classes 0 and 12, i.e., from two competitor brands Apple and Samsung. We can assume that the two brands use a totally different construction technology for the microphone and this give us the possibility to identify in a easy way the two clusters. The same reasoning can be applied to all classes of the same brand available in Table 5.1. In Figures 5.13 and 5.14 we can observe other comparisons done on the same brand and conclude are that maybe for technological reason the different classes present similarities reflected in the corresponding clusters.

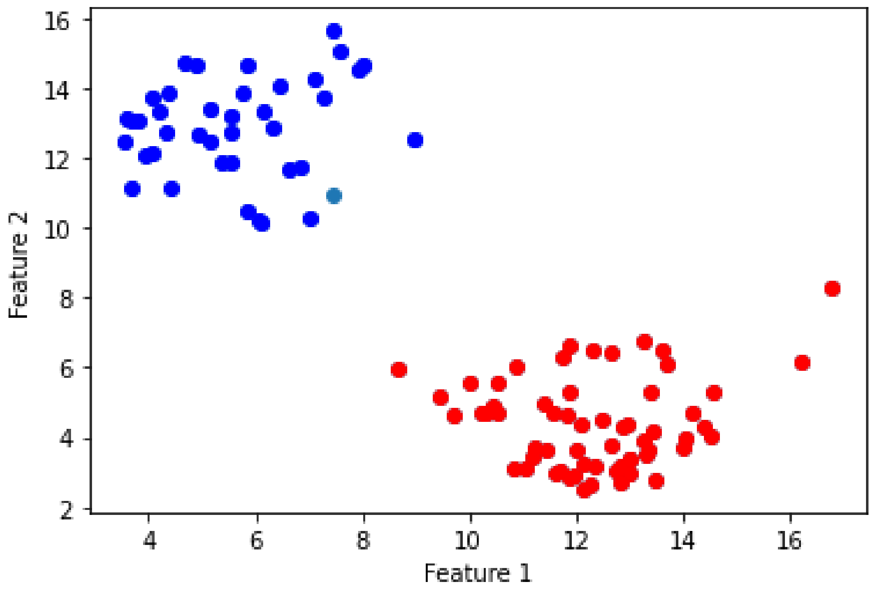


Figure 5.11: Cluster of classes 10 and 11 of the same brand "Samsung". Since the two classes are from the same brand the cluster are near in the cluster space.

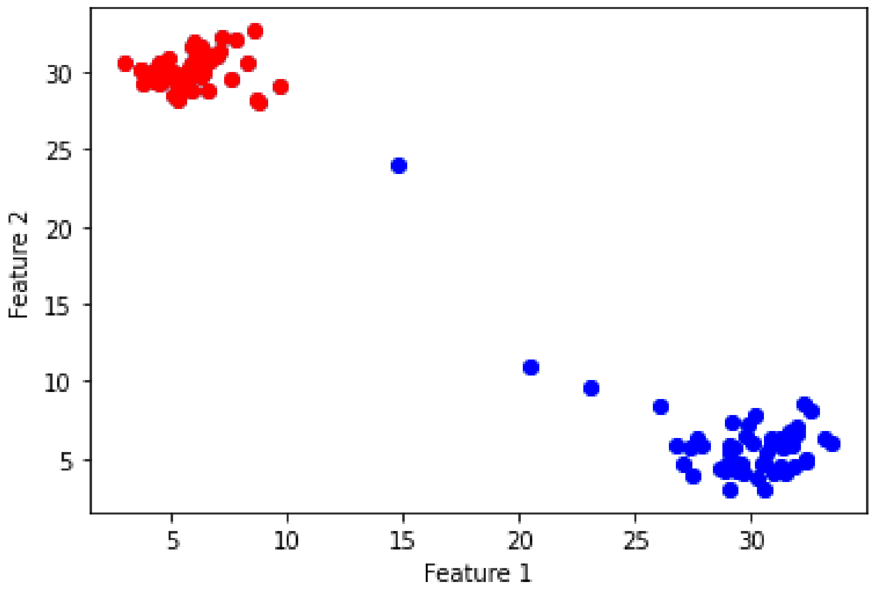


Figure 5.12: Cluster of classes 0 and 12. This is an example of two classes from different brands. In this case the clusters are clearly separated and recognizable.

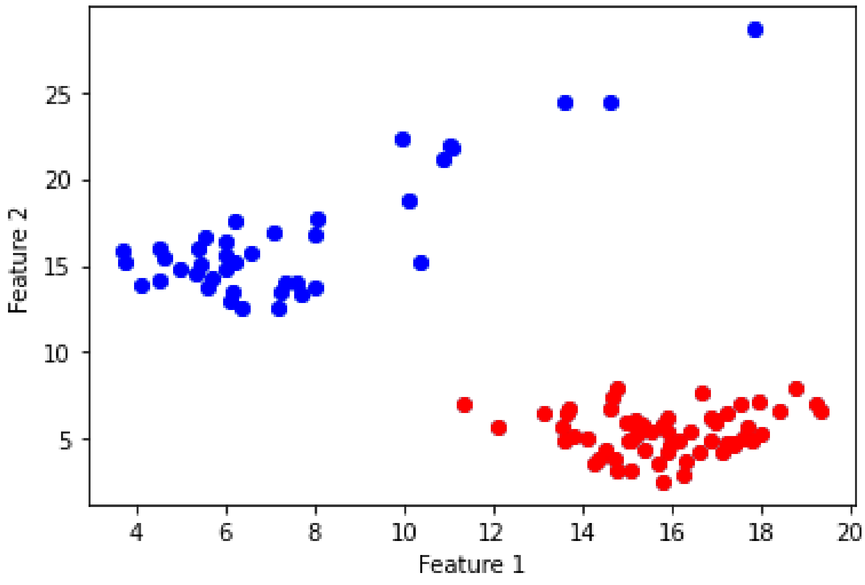


Figure 5.13: Cluster of classes 5 and 6 of the same brand "LG". This is another example of clusters from different class but same brand.

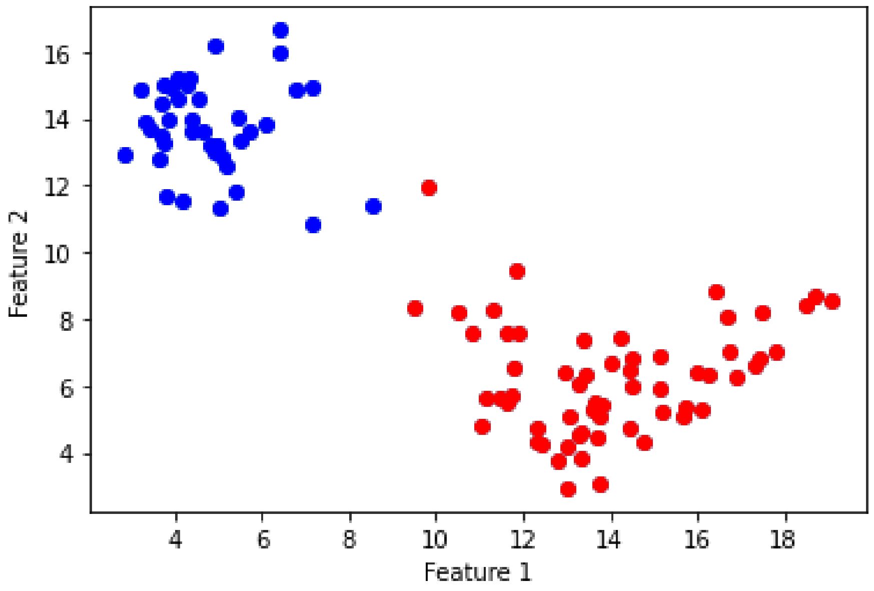


Figure 5.14: Cluster of classes 17 and 18 of the same brand "Sony Ericson". This is another example of clusters from different class but same brand.

This behavior can also be observed in the localization phase. Our algorithm is based on a distance measure and so, given that the distance is not so high in case of same brand classes, we have more difficulties in the localization of the splicing cut point. In fact the maximum error that we commit happens in this configuration. We present now some localization results both for different brand and for same brand classes. In Figure 5.15 we have the localization result applied on classes 0 and 12. The two classes, as also shown in the cluster Figure 5.12, are very far in the cluster space, this means that also the localization is simple and the splicing point is easily detectable. On the other hand in Figure 5.16 we have the localization algorithm applied on the class 17 and 18. The detected splicing point is on the analysis window 40 but the truth splicing point is on analysis window 60. So we have an error of 20 samples that is about 0.5s. Additionally the figure shows more fluctuations with respect to the Figure 5.16. We show also another example of two different classes, 9 and 12, from different brands in Figure 5.17. In this case we have a correct prediction, but it is useful to highlight that the splicing point does not need to be in the middle of the sequence.

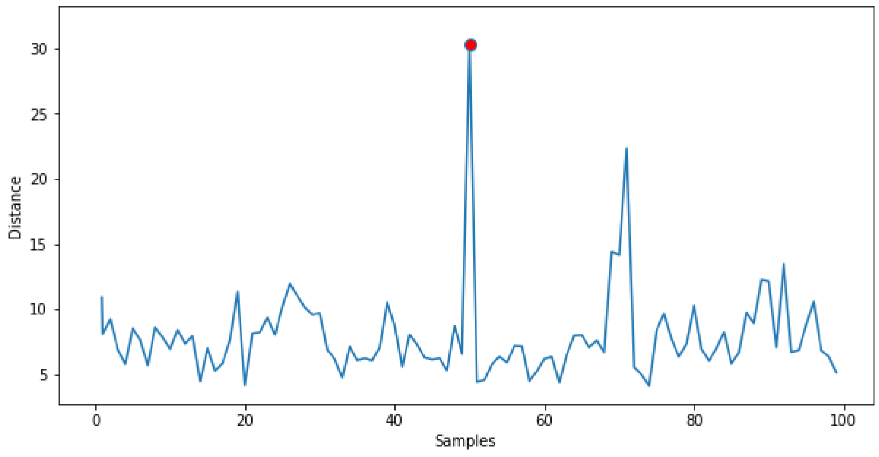


Figure 5.15: Localization on classes 0 and 12. Localization done on two classes very far in the cluster space, in fact the maximum, and so the splicing point, is easy to detect in this case.

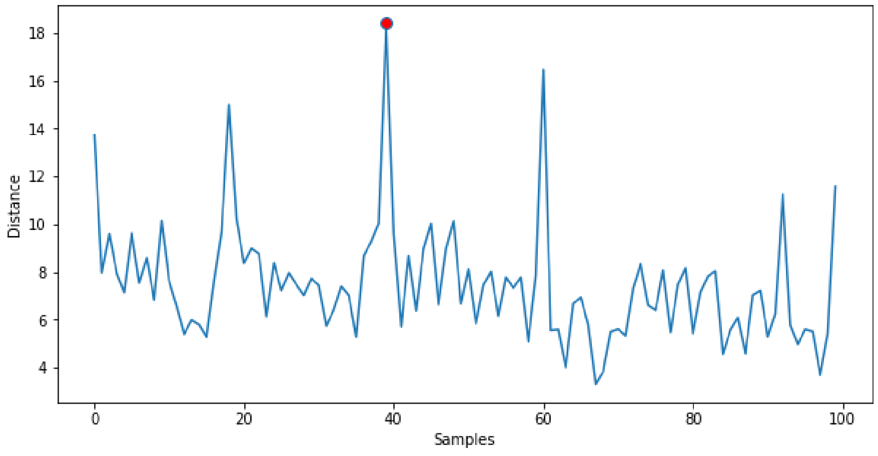


Figure 5.16: Localization on classes 17 and 18. Example of an error in localization. In this case the localization algorithm detected as splicing point the analysis window 40, but the true one is the 60. This is possible also because the two classes are from the same brand, so present similarity in the construction process.

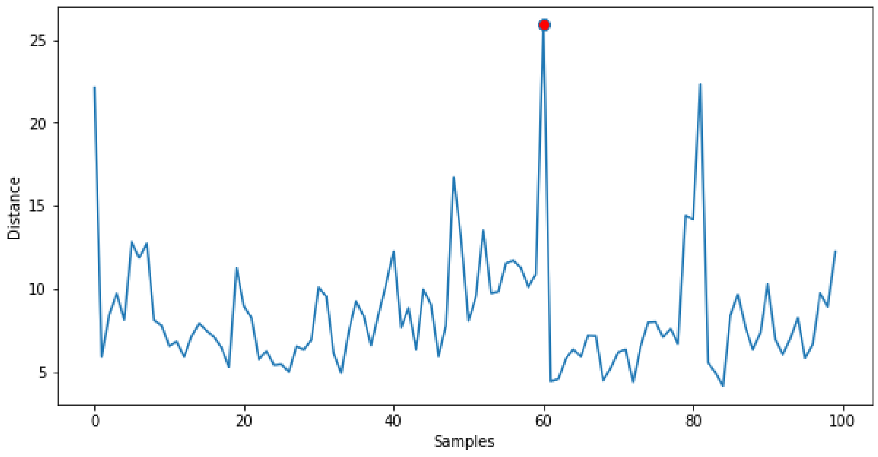


Figure 5.17: Localization on classes 9 and 12. Another example of localization. In this case the classes are from different brands, so it is easier to detect the true splicing point.

5.6 Conclusive Remarks

In this chapter we have evaluated the proposed methodology through simulations and experiments. We presented first the dataset and the metrics we used. Then we showed the process we followed to take our decision for the network and the threshold. At the end we illustrated the results we were able to get with our algorithm.

6
Conclusions and Future Works

This thesis proposes a new methodology for audio forensic analysis exploiting traces left on a recording by acquisition devices. Specifically, we have addressed two main tasks:

- Detecting if a an audio track comes from a single recording or it is a splicing generated concatenating recordings from different devices.
- Localizing the splicing cut point in order to separate the concatenated track from the original one.

The devised methodology is based on a CNN able to extract suitable features from the audio file, K-Means clustering algorithm to recognize the presence of traces from multiple devices, and a distance measure to localize the splicing point.

The main contributions of the proposed work are:

- We have proposed a splicing detection algorithm that performs very well thanks to the clustering stage. In fact, with the CNN we extract microphone intrinsic properties and with clustering we are able to identify the different classes.

- We are one of the first works that explore the problem of splicing localization. Within this context, we have been able to achieve an excellent result for the splicing point prediction, improving all the previous works in this field.

The dataset used in this work is the MOBIPHONE dataset, that we use both for training the CNN and to test the entire algorithm. We tested different CNNs and different types of input signals to select the best one to suit our goals. To choose the threshold value we evaluated the ROC characteristics taking into account all the possible class pairs.

The proposed approach has shown promising results both in detection and localization. In particular the pipeline described in our work is very fast and achieves high accuracy. The identification stage shows an accuracy of 98%, whereas the maximum error committed in localization is about $0.5s$. This results remark that the proposed method is a valid approach to resolve both the detection and localization task, and that we therefore improved the existing state of the art.

Even if we were able to achieve successfully results there are some aspects that need a deeper investigation. The first suggestion is about the dataset. In fact, we used a collection of mobile phone recordings, but the phones included in the MOBIPHONE dataset are a bit old. So it would be very interesting to see how this work performs also with the most modern mobile phones.

Another possible improvement is related to the identification pipeline. In our study we decided to use the CNN that gave us a fair balance between accuracy and computational cost. But if the computational time is not a problem it would be nice to compare also the features extracted from the CNN with the best accuracy on how they perform in the identification task. Moreover it is necessary to verify that the algorithm is valid for microphones other than those present during the CNN training, i.e. it can operate in total open set conditions.

Finally, also the localization needs a deeper research. We used a distance measure among consecutive samples but this is not the only method to retrieve the splicing point. Moreover we tested the algorithm with the euclidean distances, but it would be a great study also a comparison among the various distances and their performances.

Bibliography

[1] G. Baldini, I. Amerini, and C. Gentile, “Microphone identification using convolutional neural networks,” *IEEE Sensors Letters*, vol. 3, no. 7, pp. 1–4, 2019.

[2] N. Zeghidour, O. Teboul, F. D. C. Quitry, and M. Tagliasacchi, “Leaf: A learnable frontend for audio classification,” *ArXiv*, vol. abs/2101.08596, 2021.

[3] C. Krätzer, A. Oermann, J. Dittmann, and A. Lang, “Digital audio forensics: a first practical evaluation on microphone and environment classification,” in *MM&Sec*, 2007.

[4] C. Hanilçi, F. Ertas, T. Ertas, and Ö. Eskidere, “Recognition of brand and models of cell-phones from recorded speech signals,” *IEEE Transactions on Information Forensics and Security*, vol. 7, pp. 625–634, 2012.

[5] D. P. N. Rodríguez, J. A. Apolinário, and L. W. P. Biscainho, “Audio authenticity: Detecting enf discontinuity with high precision phase analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 534–543, 2010.

[6] L. Cuccovillo, S. Mann, M. Tagliasacchi, and P. Aichroth, “Audio tampering detection via microphone classification,” in *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 177–182, 2013.

[7] R. C. Maher, *Principles of Forensic Audio Analysis.* Springer, 1 ed., 2018.

[8] M. Zakariah, M. Khan, and H. Malik, “Digital multimedia audio forensics: past, present and future,” *Multimedia Tools and Applications*, vol. 77, pp. 1009–1040, 2016.

[9] V. Verma and N. Khanna, “Speaker-independent source cell-phone identification for re-compressed and noisy audio recordings,” *Multimedia Tools and Applications*, pp. 1–23, 2021.

[10] R. Buchholz, C. Krätzer, and J. Dittmann, “Microphone classification using fourier coefficients,” in *Information Hiding*, 2009.

[11] C. Krätzer, M. Schott, and J. Dittmann, “Unweighted fusion in microphone forensics using a decision tree and linear logistic regression models,” in *MM&Sec ’09*, 2009.

[12] D. Garcia-Romero and C. Espy-Wilson, “Automatic acquisition device identification from speech recordings,” *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1806–1809, 2010.

[13] Y. Jiang and F. H. F. Leung, “Source microphone recognition aided by a kernel-based projection method,” *IEEE Transactions on Information Forensics and Security*, vol. 14, pp. 2875–2886, 2019.

[14] Y. Panagakis and C. Kotropoulos, “Automatic telephone handset identification by sparse representation of random spectral features,” in *Proceedings of the on Multimedia and Security*, (New York, NY, USA), p. 9196, Association for Computing Machinery, 2012.

[15] Y. Panagakis and C. Kotropoulos, “Telephone handset identification by feature selection and sparse representations,” in *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 73–78, 2012.

[16] C. Hanilçi and F. Ertas, “Optimizing acoustic features for source cell-phone recognition using speech signals,” in *IH&MMSec ’13*, 2013.

[17] C. Hanilçi and T. Kinnunen, "Source cell-phone recognition from recorded speech using non-speech segments," *Digit. Signal Process.*, vol. 35, p. 7585, Dec. 2014.

[18] R. Aggarwal, S. Singh, A. K. Roul, and N. Khanna, "Cellphone identification using noise estimates from recorded audio," *2014 International Conference on Communication and Signal Processing*, pp. 1218–1222, 2014.

[19] C. Kotropoulos and S. Samaras, "Mobile phone identification using recorded speech signals," *2014 19th International Conference on Digital Signal Processing*, pp. 586–591, 2014.

[20] L. Zou, Q. He, and X. Feng, "Cell phone verification from speech recordings using sparse representation," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1787–1791, 2015.

[21] Y. Li, X. Zhang, X. Li, Y. Zhang, J. Yang, and Q. He, "Mobile phone clustering from speech recordings using deep representation and spectral clustering," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 965–977, 2018.

[22] V. Verma, P. Khaturia, and N. Khanna, "Cell-phone identification from recompressed audio recordings," *2018 Twenty Fourth National Conference on Communications (NCC)*, pp. 1–6, 2018.

[23] D. Luo, P. Korus, and J. Huang, "Band energy difference for source attribution in audio forensics," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 2179–2189, 2018.

[24] T. Qin, R. ding Wang, D. Yan, and L. Lin, "Source cell-phone identification in the presence of additive noise from cqt domain," *Inf.*, vol. 9, p. 205, 2018.

[25] G. Baldini and I. Amerini, "Smartphones identification through the built-in microphones with convolutional neural network," *IEEE Access*, vol. 7, pp. 158685–158696, 2019.

[26] C. Grigoras and J. M. Smith, “Quantization level analysis for forensic media authentication,” in *AES International Conference on Audio Forensics*, pp. 4–1, 2014.

[27] D. Gärtner, C. Dittmar, P. Aichroth, L. Cuccovillo, S. Mann, and G. Schuller, “Efficient cross-codec framing grid analysis for audio tampering detection,” in *AES 136th International Convention*, p. 9094, 2014.

[28] A. J. Cooper, “Detecting butt-spliced edits in forensic digital audio recordings,” in *AES International Conference on Audio Forensics*, pp. 1–1, 2010.

[29] D. Capoferri, C. Borrelli, P. Bestagini, F. Antonacci, A. Sarti, and S. Tubaro, “Speech audio splicing detection and localization exploiting reverberation cues,” in *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, 2020.

[30] N. D. Gaubitch, M. Brookes, P. A. Naylor, and D. Sharma, “Single-microphone blind channel identification in speech using spectrum classification,” in *2011 19th European Signal Processing Conference*, pp. 1748–1751, 2011.

[31] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue, “TIMIT Acoustic-Phonetic Continuous Speech Corpus,” 1993.

[32] V. Verma and N. Khanna, “Cnn-based system for speaker independent cell-phone identification from recorded audio,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[33] Z. Borsos, Y. Li, B. Gfeller, and M. Tagliasacchi, “Micaugment: One-shot microphone style transfer,” *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3400–3404, 2021.

[34] T. Tieleman and G. Hinton, “Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude.” COURSERA: Neural Networks for Machine Learning, 2012.