POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Master Degree in Music and Acoustic Engineering

Combining Automatic Speaker Verification and Prosody Analysis for Synthetic Speech Detection

by:
Luigi Attorresi

matr.:
945624

Supervisor:
Paolo Bestagini

Co-supervisor:
Davide Salvi
Clara Borrelli

Academic Year
2020-2021

POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Laurea Magistrale in Music and Acoustic Engineering

Rilevamento di Parlato Sintetico Combinando Caratteristiche Prosodiche e Identità del Parlatore

di:
Luigi Attorresi

matr.:
945624

Relatore:
Paolo Bestagini

Correlatore:
Davide Salvi
Clara Borrelli

Anno Accademico
2020-2021

Abstract

Recent developments in artificial intelligence have led to incredible innovations that are rapidly becoming part of our daily lives. This is the case of deepfakes, a new practice that allows generating hyper-realistic fake multimedia content. For example, it is possible to replace one person's face with that of another in a photo or video or imitate the voice of someone by making them say anything.

However, significant innovations often come with great threats. In fact, this technology can be very dangerous when exploited to steal someone's identity, discredit him, or spread false news. A video that portrays a public figure with a strong social impact in a speech that was never delivered would have tremendous consequences. The same would happen for pornographic material that depicts a world-renowned celebrity with a simple exchange of faces. Finally, a cloned voice could be suitable for phone fraud or forgery of judicial evidence. Unfortunately, this is not a dystopian description of the near future but rather actual facts.

Therefore, it is crucial to have the methodologies to neutralize such counterfeits. This thesis proposes a system capable of recognizing whether a given speech audio signal is synthetic or authentic. Our approach starts from the hypothesis that counterfeiting techniques are not yet able to recreate the most complex semantic aspects of the voice realistically. For this reason, we use a speech representation that takes into account two high-level features: vocal identity, related to timbre and intonation, and prosody, which we consider a very subtle but distinctive aspect of voice, related to behavioral characteristics and speech habits, such as style, accent or tone.

We evaluate our system from different perspectives through a series of ad-hoc experiments involving a large amount of real and deepfake audio tracks. In this way, we demonstrate the effectiveness and novelty of our method compared to existing ones. In addition, we analyze its ability to generalize in less controlled scenarios and robustness to additional manipulations, such as compression, typically applied to deepfakes to hide traces of counterfeiting. These analyses lead to excellent results and open up possible future scenarios and improvements to the model.

Sommario

I recenti sviluppi dell'intelligenza artificiale hanno portato a incredibili innovazioni che stanno rapidamente prendendo parte alla nostra vita quotidiana. È questo il caso dei deepfake, una nuova pratica che permette di generare dei contenuti multimediali iperrealistici, ma falsi. È possibile, ad esempio, sostituire il viso di una persona con quello di un'altra in una foto o in un video, oppure imitare la voce di qualcuno facendogli dire ciò che si vuole.

Tuttavia, molto spesso a importanti innovazioni corrispondono anche grandi minacce. Infatti, tale tecnologia può essere molto pericolosa quando la si sfrutta con lo scopo di rubare l'identità della vittima, screditarla o diffondere notizie false. Un video che ritrae un personaggio pubblico con un forte impatto sociale in un discorso mai pronunciato potrebbe avere conseguenze tremende. La stessa cosa accadrebbe per del materiale pornografico che, con un semplice scambio di volti, ritrae una celebrità di fama mondiale. Infine, una voce clonata potrebbe prestarsi a frodi telefoniche o contraffazioni di prove giudiziarie. Purtroppo, questa non è una descrizione distopica di un futuro prossimo, ma si tratta di fatti realmente accaduti.

È cruciale quindi avere a disposizione dei mezzi per contrastare tali contraffazioni. Questa tesi propone un sistema in grado di riconoscere se un dato audio contenga del parlato sintetico, oppure autentico. Il nostro approccio parte dall'ipotesi che le tecniche di contraffazione non siano ancora in grado di ricreare in modo realistico gli aspetti semantici più complessi della voce. Per questo motivo ci serviamo di una rappresentazione del parlato che tenga conto di due caratteristiche di alto livello: l'identità vocale, legata al timbro e all'intonazione, e la prosodia, che consideriamo un aspetto molto sottile, ma distintivo della voce, legato a caratteristiche comportamentali e abitudini nel parlare, come stile, accento o tono.

Valutiamo il nostro sistema sotto diversi aspetti mediante una serie di esperimenti ad-hoc che coinvolgono una grande quantità di audio reali e deepfake. In questo modo dimostriamo l'efficacia e l'innovazione del nostro metodo rispetto a quelli già esistenti. Inoltre, analizziamo la sua capacità di generalizzare in contesti meno controllati e la robustezza a manipolazioni aggiuntive, come ad esempio la compressione, tipicamente applicata ai deepfake per nascondere le tracce di contraffazione. I risultati di queste analisi sono eccellenti e aprono a possibili scenari futuri e miglioramenti del modello.

Ringraziamenti

Ringrazio innanzitutto Paolo, Davide e Clara per avermi guidato per tutto il lavoro di tesi, ho imparato tanto da voi. In particolare grazie Davide per essere stato sempre presente ed avermi spronato a dare il massimo. Grazie ai miei genitori, che da sempre mi supportano (e sopportano!).

Anche se lontani vi porto sempre con me, siete il mio punto di riferimento. Ringrazio i nonni, gli zii, i cugini e tutta la famiglia per laffetto che mi hanno sempre dimostrato. Ringrazio chi mi ha accompagnato in tutti

questi anni restando sempre al mio fianco, gli amici di una vita e quelli più recenti. Grazie a Chiara, Pico, Cate e Nenna per esserci sempre stati, ora che siete a Milano è tutto più bello. Marina, Alessio, Giulia, Arianna, Eleonora, Michele, Elisa, Andrea, Alice, Erika, Lucia e Marco, siete la mia casa, è sempre bello ritrovarsi dopo tanti anni come se nulla fosse cambiato. Grazie a Polifonia, una delle esperienze più belle mai vissute.

Grazie a chi ho conosciuto tramite questa associazione, Andrea, Marco, Nicole, Clara, i membri dei due direttivi e tutti i ragazzi con cui ho passato momenti indimenticabili. In particolar modo grazie a Bea, il mio braccio destro, una fonte di ispirazione e sostegno. Grazie ai Fotonici per

aver reso questi anni di magistrale indimenticabili. Grazie ai Lambrate Boys per i momenti passati insieme, vi voglio bene.

L.A.

Contents

List of Figures

List of Tables

Introduction

In recent years, Artificial Intelligence (AI) has made impressive progress, thanks to the advent of Machine Learning (ML) and Deep Learning (DL) techniques. We are witnessing a remarkable technological development that will not slow down and affect almost every field. AI-based algorithms are responsible for the most astonishing and futuristic discoveries: software capable of beating world champions in chess [19] or other creative-thinking games [20], computers able to recognize objects with greater accuracy than humans [21] and even self-driving cars [22]. Every day, we interact with voice assistants such as Siri, Alexa, Google Home, which can talk to us, understand our questions, and know how to answer by performing advanced research in a few instants. Automated translators are increasingly reliable, close to human accuracy and able to work in real-time [23]. New technologies are even being tested to translate brainwaves into messages and instructions that enable non-verbal communication [24].

However, while this technological revolution is leading to great innovations, it also has its downsides. AI, represents a tool as useful as harmful, depending on the intent with which it is used and in many cases it has paved the way for malicious purposes. One example that has been in the spotlight lately and is becoming increasingly popular is the deepfake (DF) technology. The term DF comes from the combination of "deep learning" and "fake". Indeed, it is a DL technique used to create realistic but fake images, videos and audio. For example, it allows to replace a person's face in a video, make someone's photo smile or more generally change facial expression, imitate the voice of a certain speaker by copying its main characteristics and much more. The recent developments in this technology have made such forgeries increasingly realistic and accessible. This enables producing manipulated media that are almost impossible to distinguish from the original ones. Moreover, there are many apps in circulation that allow anyone to generate a DF in a matter of seconds by just uploading a single image, audio or video. This has led to widespread popularity on social media as pure entertainment and even in the artistic field. However, together with this seemingly innocent use of this technology, other harmful uses have spread. They involve DFs to generate and promote identity-stealing multimedia content, passing it off as real in order to discredit the depicted victim, spread fake news, or even commit fraud and

appropriate their data. Many examples have already featured political or more generally public characters with a strong social impact [25]. In this regard, they can contribute to the proliferation of misinformation by creating situations that have never occurred in reality, portraying the subjects in contexts in which they have never been or that could appear compromising, to influence public opinion. Moreover, there are many cases in which face-swap has been applied in pornographic contexts, thus giving birth to new adult material in which the protagonists are often celebrities [26] or other victims. Generally, this technology can be used to discredit and damage people's image and is at everyone's reach [27]. In the same way, synthetically generated voices, corresponding to audio DFs, have proven equally dangerous. With the increasing quality and accessibility of the two main speech synthesis techniques, Text-to-Speech (TTS) and Voice Conversion (VC), it is possible to generate increasingly realistic voices with the same timbre and characteristics of target speakers. The first method generates speech that resembles a specific voice, starting from a textual input containing the words to be spoken. The second takes speech audio as input and transforms the voice characteristics into those of a target speaker, keeping the linguistic content unchanged. In this way it is possible to make the imitated people say anything so as to pretend to be them and access personal information. Real-life examples show how such voices can fool voice recognition systems and commit fraud [28] [29] or support voice phishing attacks.

As the development of DFs is moving very fast, it is necessary to work on countermeasures to defeat their misbehaving uses. With new systems for generating such counterfeit content, there is a need for new systems to recognize it, keeping pace with this evolving technology. As a result, the two branches of research involved in developing techniques for generating and recognizing DFs are constantly developing and challenging each other. One seeks to create increasingly realistic DFs capable of evading recognition systems, while the other constantly seek their weaknesses and traits relevant to identifying them. Several state-of-the-art methods have been proposed in relation to the DF detection task [30]. Some can search for artifacts introduced by the generator at pixel or sample level, which can be interpreted as fake fingerprints. This analysis usually concerns low-level features since such traces are not noticeable at high-level. For example, the authors of [31] perform detection of AI-generated fake images by looking for artifacts through high-frequency component analysis. On the other hand, the method proposed in [13] looks for them in synthesized speech through high-order spectral analysis. Other methods rely on more semantically meaningful features and exploit high-level inconsistencies to discriminate DFs, assuming their weakness in emulating the finest aspects of human nature. For example, [32] detects fake videos by modeling how people move as they speak, while [33] focuses on the detection of the lack of natural eye blinking. The authors of [34] show how synthetic voices lack natural emotional behavior and can be discriminated by feeding

a classifier with high-level features obtained from a Speaker Emotion Recognition (SER) system.

In this thesis we propose a method, called *ProsoSpeaker*, for audio DF detection, that, given an input speech signal, is able to determine if it is authentic or a fake. In particular, the types of fake audio that our system recognizes are those generated via TTS and VC techniques, which are the most advanced and common for DF generation. To do so, we consider two high-level features that describe the voice's semantic aspects. The first one refers to the spectro-temporal characteristics of the analyzed voice, i.e., timbre specific properties or pitch contour of the voice, extracted using a state-of-the-art model for Automatic Speaker Verification (ASV) tasks. The second describes prosodic aspects of the voice, like speech signal variations in rhythm, intonation and style. It is extracted by means of the prosody encoder presented in [17], originally designed to increase the expressiveness of TTS speech synthesis. We feed the concatenation of these two representations to a classifier which labels the corresponding audio as real or DF. The novelty of our approach lies in using these two features together to address the task of audio DF detection. In fact, there are methods that use either spectral [35] or prosody [36] features to implement ASV systems, but not some that jointly consider them for the task at hand. Indeed, we believe that combining two semantic representations as speaker-identity and prosody, generates a more informative one, able to model both the voice's physiological and behavioral characteristics. This constitutes a basis we can leverage to identify DF speech generated via different technologies that may be flawed in one semantic aspect or the other.

To validate our system from different perspectives, we performed five experiments that include a total of almost 800000 traces between authentic and counterfeits. We keep the same training set for all experiments, while varying the test set according to the aspect of the model to be evaluated. Moreover, the datasets involved in the various test phases are not necessarily the same ones considered during training. We guarantee uniformity of the experiments and a robust evaluation of the results on unseen generation algorithms and real-world data. The evaluation setup aims to analyze the chosen features, demonstrate the effectiveness of the method compared to the state-of-the-art and verify the generalization capabilities and robustness to post-processing manipulations. The results are promising since *ProsoSpeaker* proves to be better than the state-of-the-art methods considered, with a Area Under the Curve (AUC) of 98.85 in a controlled scenario, while 91.59 in the most complex case.

This thesis is organized as follows. In Chapter 1 we provide the reader with all the specific knowledge needed to understand the content of our work, i.e., we describe the features we use as input to the system and the Machine Learning (ML) and Deep Learning (DL) concepts involved. In Chapter 2 we describe the state-of-the-art related to our work, i.e., Automatic Speaker Verification (ASV), audio deepfake (DF) generation

techniques, prosody modeling and DF detection. Then, in Chapter 3 we formalize the problem that this thesis addresses and describe the system proposed to solve it, that we call *ProsoSpeaker*. Chapter 4 describes the experimental setup implemented to test our method, that is, the datasets and metrics chosen, the details for feature extraction and model training, the baselines used as reference, and the series of experiments performed. In Chapter 5 we show and analyze the obtained results interpreting them and validating our starting hypothesis. Finally, in Chapter 6 we summarize our work and propose possible future scenarios and improvements to the model.

1 Theoretical Background

In this chapter we will present the theoretical background needed to understand the following sections completely.

1.1 Audio Feature Extraction

Feature extraction is a crucial procedure needed to build an informative representation of the data. It consists of extracting meaningful characteristics that can be used to fully describe and distinguish the data. This section will present the main representations and features involved in our study.

1.1.1 Mel Spectrogram

A Spectrogram is a representation of an audio signal in both its time and frequency domains. In the Mel Spectrogram, shown in Figure 1.1, the frequency axis is converted into the Mel scale, which is an approximation of the human psychological sensation of heights of a pure sinusoid. This scale resembles the logarithmic behavior of the auditory filters in the cochlea — organ which plays a key role in the sense of hearing — and thus, is used to provide sound information similar to what a human would perceive [37]. There is not an exact formula for the Hz-to-Mel conversion, being the Mel scale a psychoacoustic concept, but there are many approximations. In our work we adopt the popular formula proposed by O'Shaughnessy [38]:

$$m = 2595 * \log_{10}(1 + f/700), \tag{1.1}$$

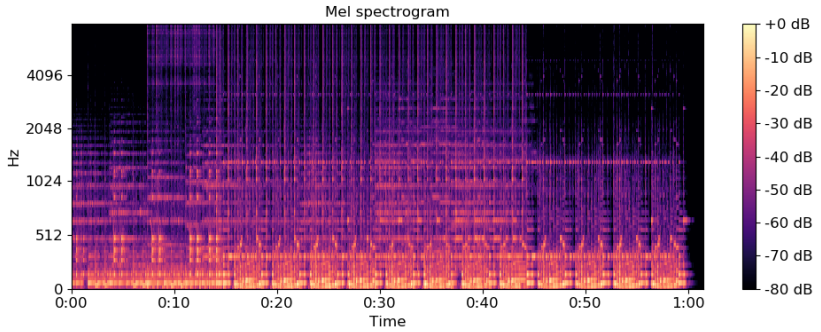


Figure 1.1: Example of a Mel Spectrogram [1].

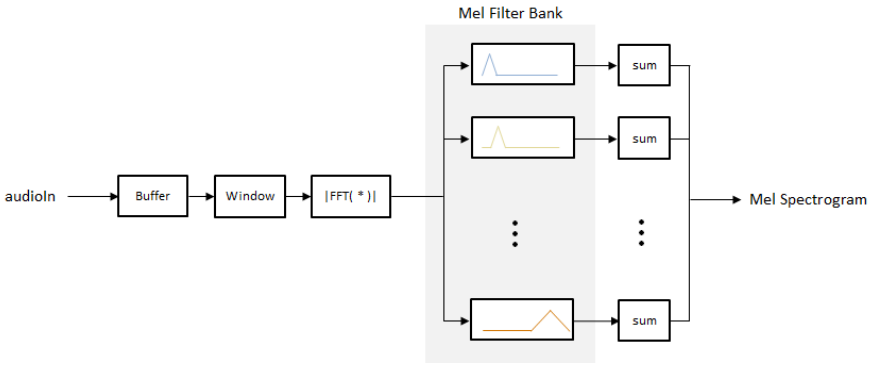


Figure 1.2: Block scheme showing the computation of a Mel Spectrogram [2].

where f is the considered frequency and m is the associated value on the Mel scale.

The general algorithm to compute a Mel Spectrogram is described in Figure 1.2. First, a time-frequency representation of the input signal is extracted through the Fast Fourier Transform (FFT), a fast algorithm for computing the Discrete Fourier Transform (DFT). It is performed by dividing the signal into shorter overlapping segments through the windowing process, as in

$$x_l[n] = w[n]x[n + lR] \qquad 0 \leq n \leq N - 1, \tag{1.2}$$

and then applying the DFT at frame-level (1.3),

$$X(l, k) = \sum_{n=0}^{N-1} x_l[n]e^{-j2\pi nk/N}. \tag{1.3}$$

Here, $x_l[n]$ is the windowed frame l extracted using a window $w[n]$ with a hop-size R, $X(l, k)$ is the value of the DFT of $x_l[n]$ at the frequency bin k and N is length of $x_l[n]$. Then, the result is multiplied by a Mel filter bank, computed from a series of overlapping triangular windows at a series of evenly spaced Mels. Finally, the Mel Spectrogram is obtained by summing all the filter bank outputs.

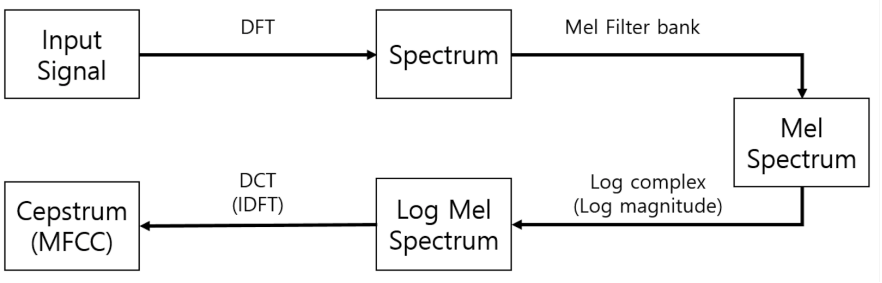


Figure 1.3: Block scheme showing the computation of MFCCs [3].

1.1.2 Mel Frequency Cepstral Coefficients

The Mel Frequency Cepstral Coefficients (MFCCs) are a compact representation of a signal spectrum, widely used in speech processing. The name comes from the fact that they express in the Mel scale the rate of change in spectral bands, often referred to as *cepstrum* [39].

Figure 1.3 shows the steps required to extract the MFCCs. The first two are intended to calculate the Mel Spectrogram of the signal, as described previously. Therefore, once the signal has been divided into N fragments of the same length, the DFT computes the spectrum of the signal from them and a Mel filter bank of K filters transforms the frequency into the Mel scale. Then, we compute the log filter bank energies as a logarithm of the Spectrogram and use the Discrete Cosine Transform (DCT) to decorrelate their values. The resulting $K \times N$ matrix contains the MFCCs values of the signal.

1.1.3 Prosodic Features

Prosody Definition

Prosody is a general term used to describe the confluence of many phenomena in speech, such as temporal variations in rhythm, intonation, stress and style. Such elements, known as *suprasegmentals*, convey information that is not explicitly specified by the text. The same sentence might denote a number of different intentions and nuances, depending on how it is pronounced. Such aspects are only capable through suprasegmentals and are closely influenced by the speaker's emotion, language, and speaking style. Moreover, different languages can have very different intonation rules [40] as well as other individuals can have their own speaking habits [41].

Many methods have been proposed to model prosody, such as [42, 43]. These approaches decouple the concept of prosody from that of speaker identity and model them separately. In this work we will intend the prosody based on the "subtractive definition" adopted in [42] that is: *prosody is the variation in speech signals that remains after accounting for variation due to phonetics, speaker identity, and channel effects (i.e. the recording environment).*

Features

Among those aspects related to the definition of prosody, two of the most used are the *fundamental frequency (f_0)* and *energy (E_0) contours* of the audio signal, which can be expressed as functions of time. Their behaviour provides lots of information about most phenomena ascribable as prosody. Among the most commonly used features are the *minimum* and *maximum* of these functions computed over short fragments and their first four statistical moments, namely *average*, *standard deviation*, *skewness* and *kurtosis*.

Among all the proposed methods for extracting such features, the majority can be grouped into two main categories:

1. **Windowed-segments Methods**: the audio signal is divided into short-term frames via fixed-length, overlapping windows and a feature vector is extracted for each one of them.

2. **Voiced-segments Methods**: the audio signal is divided into voiced segments, like words or syllables [44], and a feature vector is extracted for each one of them. These methods discard the pauses between each segment, assuming that their placement does not considerably influence prosody.

Depending on the length of the inputs, these methods can provide long feature vectors with variable sizes.

Another set of prosody features used occasionally is based on the duration and position of voiced and unvoiced segments [45]. Among them are:

- *Duration of voiced segments*: duration of speech voiced sounds.
- *Duration of unvoiced segments*: duration of speech voiceless sounds.
- *Duration of pauses*: duration of silences between words.
- *VUP ratios*: ratios between the voiced, unvoiced and pause segments.

This kind of prosodic features is normally extracted in a global fashion from the whole audio, providing fixed-length vectors regardless of the inputs lengths.

1.1.4 Feature Preprocessing

Once the features have been extracted, for most cases it is important to include a preprocessing phase in which they are rearranged, transformed and optimized for the problem at hand.

Feature Scaling

Feature scaling is needed whenever the feature supports do not coincide [46]. This technique standardizes each feature independently to have them all defined over the same range. In this way they will be comparable, otherwise, the used system will assign different weights to those with different domains. There are two commonly used scaling techniques:

1. **Min-Max Normalization**: re-scales a feature set in the range $[0, 1]$
$$\hat{x} = \frac{x - min(x)}{max(x) - min(x)}, \tag{1.4}$$
where $\hat{x}$ is the normalized feature and x is the original one.

2. **Standardization**: re-scales a feature value so that it has distribution with 0 mean value and unitary variance
$$\hat{x} = \frac{x - \overline{x}}{\sigma}, \tag{1.5}$$
where $\hat{x}$ is the standardized feature, x is the original one, $\overline{x} = avg(x)$ is the mean of the feature vector and σ is its standard deviation.

Feature Correlation

Feature correlation is considered a crucial step in the preprocessing phase. It is a way to understand the amount of shared information among the extracted features. Correlation is a statistical term, which, in common usage refers to how close two variables are to having a linear relationship with each other. Hence, it can be used as a measure of similarity. Suppose two features are uncorrelated or have a low correlation degree. In that case, it means that they can model and focus on independent characteristics of the input data. On the other hand, if they show a high degree of correlation, they share a significant amount of information and, therefore, are redundant. In this work we intend the linear correlation between two variables x and y in terms of Pearson's correlation, defined as the ratio of their covariance $cov(x, y)$ and the product of their standard deviations σ_x and σ_y

$$r_{xy} = \frac{cov(x, y)}{\sigma_x \sigma_y} \tag{1.6}$$

The resulting value r_{xy} ranges from -1 to 1 and the higher its module the greater the correlation, so it is maximum at the extreme of the range, and minimum at the center. A negative correlation means that if a feature increases, the other decreases, while they are directly proportional for a positive correlation. An easy way to visualize the correlation between all possible feature pairs is the *correlation matrix*. It is symmetrical since rows and columns contain the same values and each element of the matrix is the correlation coefficient between the features at its indices.

1.2 Machine Learning

ML [47] is a branch of Artificial Intelligence (AI) [48], which encompasses systems able to perform a task without having been explicitly pre-programmed to succeed it. ML models are presented with many examples relevant to a task so that they can find a statistical structure in the data that eventually allows them to come up with rules to automate it. That is why such systems are *trained* rather than programmed. The set of examples used during the training phase is called *training set* and it should be comprehensive and representative enough for the given problem. The one used to test the model performances is called *test set*. It should contain *unseen* samples, not provided during training, so that it tests the ability of the model to *generalize*, i.e., to perform well on never-seen-before data. In addition to the first two, a *validation set* can also be used to get an early estimate of the system's performance. Validation monitors weather the model has properly learnt by evaluating it at the end of each training epoch. This serves as a guide to eventually modify its parameters before testing the final version on the test set.

Overfitting is a common ML problem highlighted by a considerable difference in performances between training and test sets. This occurs when the model adapts too much to the training data and learns its structure "by heart", without being able to generalize over unseen data. Its cause could be the high complexity of the model with respect to the low quantity of available training data. The opposite situation, called *underfitting*, happens when the model, being too simple, is not able to catch the structure of the input data. Poor performances on the training set underline this case.

There are different approaches in training a ML system. The two most common are:

- **Supervised Learning**: it consists of learning a function that maps input data to known targets, also called *labels*, given a set of examples that have been previously annotated. Then, the inferred function is used to predict the output from any unseen instance of input. The model's goodness can be described according to various metrics and depends on how close the predictions on unseen test data are to the ground truth ones.

- **Unsupervised Learning**: in this training approach, data are supplied to the system without any indication of the desired result. The purpose of this second method of learning is to "trace" hidden patterns and models, i.e., to identify a logical structure in the input data, without previously labeling it. In this approach, the model's accuracy depends on how close the original distribution of the data is to the learned one.

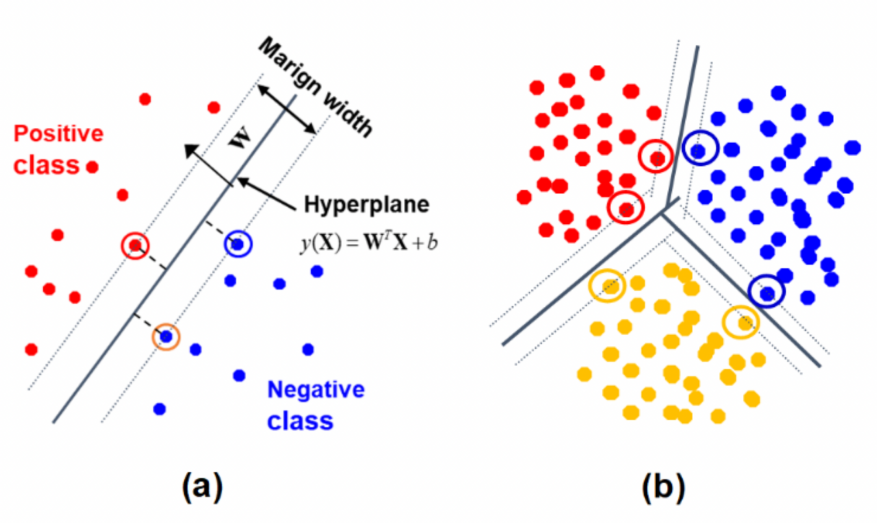


Figure 1.4: SVM binary (a) and SVM multiclass classification (b) [4].

1.2.1 Support Vector Machine

Support Vector Machine (SVM) [49] is a supervised ML method widely used for the task of *classification* [50]. Classification consists of assigning to each input sample a label selected from a given set. This predictive task is accomplished by learning a function $f(x_i) = y_i$ where $x_i \in x_1, x_2, ..., x_n$ is the input sample and $y_i \in y_1, y_2, ..., y_n$ is the predicted label from the given set.

SVM approaches this problem by learning good *decision boundaries* between the set of points belonging to different categories and then checking in which region the unseen samples fall. As shown in Figure 1.4, these boundaries can be geometrically represented as hyperplanes separating the training data into spaces corresponding to different categories. The method first maps the data into a new high-level representation, called feature space. Then, it computes hyperplanes by maximizing their distance from the nearest data points in each class. This step, defined margin maximization, allows the boundaries to generalize well to the new samples. The higher the margins, the better the separation between classes and the more robust the model.

If the input data are not linearly separable, SVM cannot partition the space with hyperplanes and, therefore, needs to perform a data transformation. It maps the input data into a higher-dimensional space where the classes are more readily separable and the classification problem becomes more straightforward. However, explicitly computing the coordinates of the points in the new space can be computationally intractable and for this reason it employs a shortcut called the *kernel trick*, named after the *kernel function*. A kernel function is an operation that maps any two points from the initial space to the distance between them in the target space, bypassing the explicit computation of the new representation.

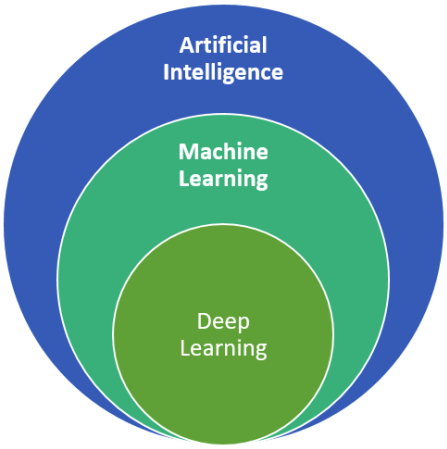


Figure 1.5: Relations between DL, ML and AI [5].

1.2.2 Grid Search

One of the most challenging parts in ML is tuning the *hyperparameters* — the set of external characteristics of a model whose values cannot be estimated from data — since only experience can help to optimize them. However, there are algorithms which try to find their optimum values via trial and error approaches. One of these is Grid Search [51], which receives as input the set of trials as different hyperparameters values to try, and trains and evaluates the model performances for each possible combination. The one that maximizes a chosen metric over the validation set is considered the best choice.

1.3 Deep Learning

DL [52] is a a specific sub-field of ML as shown in Figure 1.5. It is based on the idea of learning successive *layers* of increasing meaningful representations from data.

In DL the learning of high-level representations is mostly carried out by models called *Artificial Neural Networks (ANNs)*. An ANN is a computing system inspired by how biological neural networks work. It is composed of a series of computing units, called *neurons*, connected and organized in layers. The input flows through the first layer, whose output is the input of the second one and so on. By going deeper into the ANN, the representation becomes increasingly different from its original form and informative about the final result. The output of each layer depends on how it weights the inputs. Therefore, *learning* means finding a set of weights for all the layers, such that the network will correctly map the inputs to their associated targets. It is crucial in this sense to adopt a score as feedback to adjust the weights and push the learning in the right direction. This process is performed by *backpropagation*, which is a fundamental algorithm in DL. Once the input has passed through

the ANN, resulting in the first output (or prediction), backpropagation computes the gradient of a defined cost function, called *loss function*, and propagates back the error. This serves to update the weights of each layer and marks the end of an *epoch*. The goal is to minimize the loss function across the epochs to obtain the desired output. We can do so using different optimization methods, and the most known is the Stochastic Gradient Descent.

The reason why DL is usually more performing than ML is that it completely automates what used to be the most crucial step in a ML workflow: feature engineering. DL models do not require the input to be provided as a set of features since they can directly work on raw data. By building high-level representations, each layer can be considered a feature extraction step where features, i.e., the layer's output, do not necessarily make sense for humans but are meaningful and representative from the machine point of view. From this idea comes the concept of *embedding*, a low-dimensional representation of data analogous to a feature vector, extracted with an ANN by taking one of its layers output.

Due to this incredible power, DL has lately become the state-of-the-art for most tasks and is adopted in this work too. We will now outline the main characteristics of the ANNs involved in our research.

1.3.1 Multilayer Perceptron

A Multi Layer Perceptron (MLP) [53] is a feedforward ANN connecting the nodes (or neurons) of multiple layers in a directed graph. Each neuron performs a weighted sum of the input values and applies a function, called *activation function* to get the output

$$y_j = f_j(\sum_{n=1}^{N} w_{ij} x_i), \tag{1.7}$$

where y_j is the output of neuron j in a layer, f_j is the the activation function, x_i is the input i to the node and w_{ij} is the learning weight corresponding to that input. The MLP's architecture consists of a minimum of three layers, an *input layer*, one or more *hidden layers* and an *output layer*. Except for the input neurons, f_j is generally non-linear in order to allow the network to capture complex structures inside the given data. These ANNs are also called fully-connected neural networks since each node in one layer connects to every other node in the following layer, as shown in Figure. 1.6.

1.3.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) [54] is a type of ANNs specialized in processing grid-like data, such as images. For this reason, it is widely employed in the field of Computer Vision. Nonetheless, this network

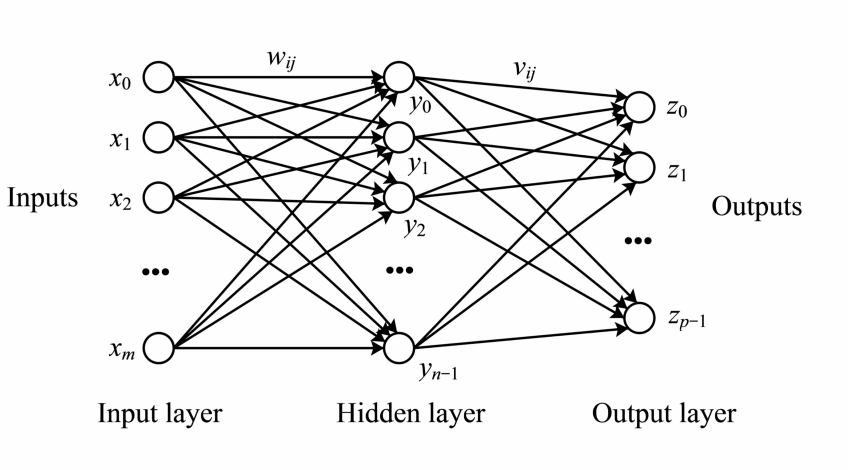


Figure 1.6: Graph of a MLP with one hidden layer [6].

frequently processes audio signals, when converted into two-dimensional representations, like the Mel Spectrogram [55] previously presented.

As for MLP, a CNN is composed of an input layer, a variable number of hidden layers and an output layer. Typically it alternates between two types of hidden layers: *convolutional* and *pooling*. Each convolutional layer performs a mathematical operation called *convolution* between the input matrix and one or more filters, also called *kernels* (corresponding to the layer's neurons), such as

$$y(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m)(j-n), \tag{1.8}$$

where $y(i,j)$ is the feature map, I and K are respectively the input and kernel matrices and $*$ is the convolution operator. In other words, the kernel slides over the input matrix with a *stride* s — meaning that it moves by s positions at time — covering at each step different regions, called *receptive fields*, and performing convolution with them. The result, called *feature map*, is then fed to a non-linear activation function. The role of pooling layers, instead, is to downsample feature maps in order to reduce the number of coefficients to process and introduce spatial-filter hierarchies by making successive convolutional layers focus on increasingly larger windows.

Depending on their values, filters encode specific aspects of the input data which become higher-level representations by going deeper through the CNN. The first layers, for example, will be able to detect low-level information as straight lines, edges and other simple geometrical shapes. In contrast, the last layers will eventually capture complex patterns like faces or objects. The goal of this network is to find the best filters values, i.e, the weights, applying the backpropagation algorithm.

These networks can process many channels in parallel, as it is common to have a multi-channel input, like an RGB image. In this case, the kernel is a stack of matrices, one for each channel, denoting its *depth*. After

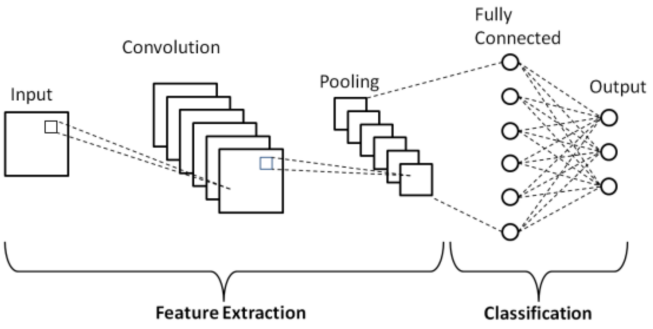


Figure 1.7: Graph of a CNN ending with a fully connected layer [7].

convolution, the output feature map of a layer is obtained by summing the results of each channel. Similarly, the number of kernels in one layer identifies the number of its output channels.

CNNs usually end with one or more fully connected layers aimed at mapping the last feature map to a prediction. The connection of the two networks is carried out by a process called *flattening*, which consists of unfolding the bi-dimensional CNN output to match the MLP input shape, as it is shown in Figure 1.7.

CNNs generally refer to the bi-dimensional scenario just described, but they can be adapted to process data of different dimensions too. One-dimensional CNNs, for example, are particularly suited to work with sequential data, like audio signals or word sequences [56]. They behave as described above, but deal with one-dimensional vectors instead of matrices.

1.3.3 Recurrent Neural Network

A Recurrent Neural Network (RNN) [57] is a type of ANNs best suited for processing sequential data or time series. Hence, some of the main application areas of RNNs are Natural Language Processing (NLP), machine translation, and speech recognition. In fact, they are provided with a "memory", which can capture variable-length time dependencies from the input.

A RNN unit builds its memory by iterating over the input sequence and generating at each time step t a *state vector* h_t which depends on the current input x_t and the previous state vector h_{t-1}, as shown in Figure 1.8. In this way, every state vector includes the information of the previous ones and the final output will encode the temporal dependencies of the whole sequence.

During training, the RNN does not perform the standard backpropagation algorithm, but a variant of it called *back propagation through time (BPTT)*. It approximates the full gradient by unrolling state vectors up to a finite number of time steps and applies standard backpropagation over this subsequence. Due to its recursive behavior BPTT suffers from the

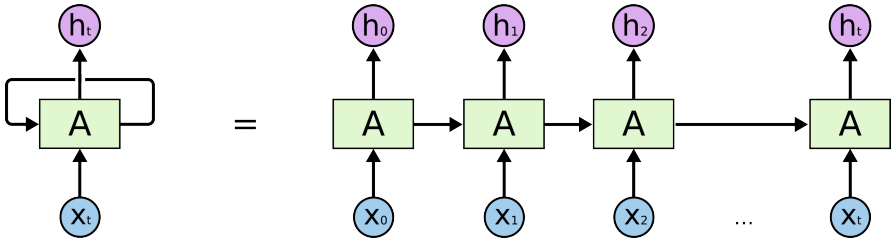


Figure 1.8: Scheme of a RNN unit (left) and its unrolled representation (right) [8].

vanishing gradient and *exploding gradient* problems. When the gradient is too small, it continues to become smaller, updating the weights until they become insignificant. On the contrary, when the gradient is too large, the weights grow too much, generating an unstable model. One solution to this can be using an advanced RNN unit, the *Gated Recurrent Unit (GRU)* [58], explicitly designed to mitigate this problem. It makes use of two vectors, the *update gate* and the *reset gate*, to control how much and which information should be passed to the output. It can model many longer-term dependencies since it can train the gates to keep information from long ago or remove irrelevant information to the prediction.

1.3.4 Self-attention

Self-attention [59] is a DL mechanism first introduced in the field of NLP for helping to build a more meaningful representation of each word considering its semantic role in the *context* of a sentence. This method has also become popular for audio signal processing, dealing with sequential data. Still, for the sake of clarity, it is appropriate to introduce it in the context of NLP.

Embeddings make it possible to represent each word in a sequence as a vector, which encodes its meaning but is blind to the context-related concept. As an example, the word "lie" always has the same embedding vector regardless of the sentence where we use it. Still, depending on the context, it could be a verb or a noun and could even represent very different concepts. Self-attention, instead, transforms the standard vectors into contextualized representations, whose values also depend on the relationships with the other elements in the sequence.

Figure 1.9 shows the whole process. Three different linear projections are applied to the input embeddings, resulting in the *queries*, *keys* and *values* matrices. These representations allow focusing on different aspects of the input embeddings. Then, the scalar product is performed between queries and keys to extract similarity coefficients for each pair of word embeddings. The result is normalized and passed to a softmax activation function, which returns a similarity score for each word. Finally, the contextualized embeddings are obtained as a linear combination of the values of the vectors using softmax scores as weights.

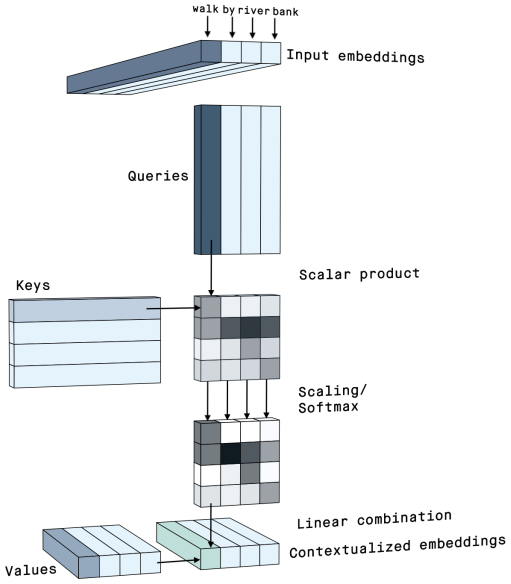


Figure 1.9: Self-attention mechanism for the input sentence "Walk by river bank" [9].

Multi-head attention enhances this model by repeating the same process multiple times using different linear projections and then concatenating the results. Each projection set can focus on computing different relationships between the words and create specific contextualized embeddings.

1.3.5 Time Delay Neural Network

Time Delay Networks (TDNNs) [60] are types of ANNs, often adopted for speech processing tasks [61], effective in modeling long range temporal dependencies, as RNNs do, but with training times comparable to standard feed-forward ANNs. They work in a similar way to CNNs, so much so that they are seen as their precursor. The input to the network is a set of t frames in the form of feature vectors containing m feature values each. Hence, they can be collected into a $m \times t$ matrix where each column identifies a frame and each row a feature. The Mel Spectrogram and MFCCs are well suited to this type of representation, therefore, they are mostly employed as input. As for CNNs, the layers of a TDNN perform convolution between the input matrix and a trainable weight matrix (the kernel or filter). The difference is that in this case, the receptive field embraces the whole dimension m, thus focusing on each entire fragment t. The length of the receptive field, instead, defines the *context* or *delay* since it sets how many past and future frames the convolution takes into consideration with respect to the current frame, which is at the center of the receptive field. Another difference is that the stride is always set to one, making the kernel move step-by-step over the temporal axis. In the latest TDNN architectures, the filters may not pass over contiguous

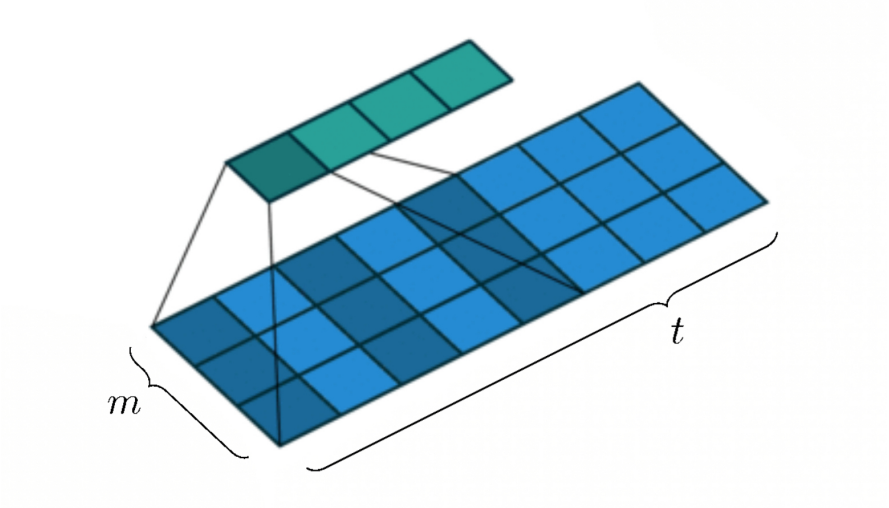


Figure 1.10: TDNN convolution with dilation. The blue matrix is the input, its shady regions represent the receptive field and the green vector is the output.

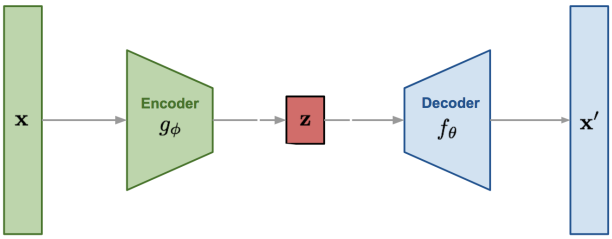


Figure 1.11: Architecture of an autoencoder [10].

frames during convolution but consider context frames further away from the target one [62]. This operation, referred as *dilation* or *subsampling*, allows the network to focus in a broader context that may be less related to the target frame than the one extracted from its neighbors as shown in Figure 1.10.

1.3.6 Autoencoder

The autoencoder [63] is a DL implementation of the classical encoder-decoder architecture, where the encoder generates a compressed representation of the input through a lossy process and the decoder decodes it back to its original version. Figure 1.11 shows its complete pipeline. Both the encoder and decoder are ANNs. The former learns a function g_ϕ to map the input x into a lower-dimension representation z, defined over the so called *latent space*. In contrast, the latter learns a function f_θ to output a reconstruction x' of the original input from the latent vector z. The training process is carried out in an unsupervised manner, without ground-truth labels. The parameters ϕ and θ of both ANNs are updated during training in order to reduce dissimilarity between the true input and its reconstruction.

The autoencoder belongs to the class of *generative models* since, once trained, by randomly sampling a vector z from the latent space, its decoded output can be a new realistic sample, not present in the original data, but showing the same distinctive features. In fact, each dimension of the latent space can be interpreted as a high-level descriptor of the input data. For this reason, the latent space can be interpreted as an embedding space, and thus the trained encoder can be used on its own for the embedding extraction process.

1.4 Conclusive Remarks

In this section we explained the main theoretical concepts involved in this work. In particular, we have focused on input data representations and feature extraction by analyzing different aspects of it. Then we introduced ML and DL and the models that we use for our method.

2
State of the Art

In this chapter we present the state-of-the-art in the field of the proposed work. In particular we focus on ASV, audio DF generation techniques, prosody modeling and DF detection, which are the four main topics on which our research is based.

2.1 Automatic Speaker Verification

Biometric recognition and authentication enable a system to ensure a person's identity and uniqueness based on specific physiological and behavioral characteristics, unlike traditional methods. For this reason, this practice is becoming increasingly popular, bringing several benefits as to not having to remember passwords or carry unlocking devices. The most common types of recognition biometrics are the face, iris, fingerprint and voice. While the first ones deal with physical traits of the human body, the last one also encompasses traits learned and acquired along with life and environment. For example, an individual's accent changes according to the geographical area in which they were born and raised, and speech style is influenced by personality and habits. At the same time, a voice may vary based on factors such as emotion, age, health status, distance from the interlocutor, referred as to *speaker-variability*. Finally, the recording conditions may vary based on the device used, environmental characteristics and signal-to-noise ratio, referred to as *channel-variability*. For this reason, speech represents one of the most complex and variable and one of the most complete types of biometrics.

We can group recognition techniques by task and by functioning when it comes to automating this process. Tasks can be different depending on the goal of the recognition system. *ASV* aims at determining whether

two given voices belong to the same individual or not. This is equivalent to finding out whether a given unseen input voice x and a voice v_i, known to the system, belong to the same identity i

$$f(x, v_i) = \begin{cases} \text{same identity} & \text{if } x, v_i \in i. \\ \text{different identity} & \text{otherwise.} \end{cases} \tag{2.1}$$

On the other hand, we define *Speaker Identification (SI)* the task of recognizing which of the known identities $I = 1...n$ the input voice x is more similar to

$$f(x, v) = i, \tag{2.2}$$

where i is one of the known identities I and $v = v_1...v_n$ are the voices associated to them.

Finally, the most general task of *Speaker Recognition (SR)* is given by the concatenation of SI and ASV and consists of ensuring that a given unseen input voice x corresponds to one of the known identities $I = 1...n$ and is who it claims to be

$$f(x, v) = \begin{cases} \text{identity } i \text{ verified} & \text{if } x, v \in i. \\ \text{authentication failed} & \text{otherwise.} \end{cases}, \tag{2.3}$$

where i is one of the known identities I and $v = v_1, ..., v_n$ are the voices associated to them. Therefore, SI and ASV can work orthogonally. The former selects among the known identities the one that most closely resembles the input, while the latter checks that both belong to the same person. The result can be the authentication of the person or not.

In terms of functioning, there can be two types of recognition systems: *text-dependent* and *text-independent*. The former requires that the text of the two compared speech utterances is known to the system, while the latter does not. While text-independent systems are suitable for a more flexible scenario, they represent a more complex problem and usually achieve lower performance than their counterparts. In our work we will concentrate on text-independent ASV system. The following subsections will discuss the state-of-the-art approaches to this task.

2.1.1 Traditional Techniques

Many ASV methods rely on extracting hand-crafted spectro-temporal features and passing them to a binary classifier to determine whether the two input identities match or not. For instance, [35] combines temporal pitch information and cepstral coefficients and uses them to represent the short-time spectral envelope of speech. These features are used to train a Hidden Markov Model (HMM) which proves to be suitable for the text-independent scenario classification. The authors show that including temporal features yields better results for both English and Cantonese Chinese languages. Similar approaches rely on prosodic features, which

model the component of speech related to speaking style and habits learned over a period of time. These features are mostly based on the analysis of the temporal contours of f_0 and E_0. In this regard, [36] implies that each speaker has their own prosodic signature. As a result, any sentence repeated by a speaker in the same context show a consistency of prosody, not exhibited across different speakers. We can exploit this trait to distinguish the voices of different people. The authors of [64] enhance a speaker verification system by adding two new features: jitter, i.e., the cycle-to-cycle variation of fundamental frequency and shimmer, i.e., the variability of the peak-to-peak amplitude in decibel. The result is a more discriminative feature set and demonstrates that jitter and shimmer can provide complementary information to both spectral and prosodic components.

Other approaches extract representations from data through ML methods. One of the earliest examples is the GMM-UBM [65]. Gaussian Mixture Model (GMM) is a linear combination of Gaussians Probability Density Functions (PDFs) which can be used to model the distribution of the speaker's data [66]. In this way, each known identity is associated to a GMM. Instead, Universal Background Model (UBM) is a type of GMM that models the distribution of the entire speech data in a speaker-independent fashion. The accept or reject result of authentication comes from comparing the unseen utterance with these two distributions. Later variants of this model achieve better performance by deriving the GMM of each speaker from the already trained UBM, using Bayesian adaptation.

In [67] the authors aim at solving the problem of variable-length speech samples, which results in variable-length feature vectors. For this purpose they introduce *supervectors*, obtained by concatenating the parameters of the GMM, which are fixed in number. This allows extracting a fixed-dimensional representation out of a single variable-length utterance. The previous model suffers from speaker and channel variability. Hence, Joint Factor Analysis (JFA) was introduced to solve this issue [68]. It consists on splitting each supervector V, into speaker-dependent s and channel-dependent c components such that $V = s + c$. The training phase is JFA-based and the supervectors do not include channel-dependent information. On the contrary in the testing stage, such information is acquired from test utterance and the obtained supervector is ranked using the linear dot product.

In 2009 [69] proposed a model known as *i-vectors* approach that has been the state-of-the-art for ASV until the diffusion of DL. Its power is due to the low dimensional representation capability and the availability of robust countermeasures against channel and speaker variabilities. This approach can be seen as a dimensionality reduction of the GMM supervectors and aims to reduce the performance loss of JFA caused by channel-dependent content being ignored during the training phase. In fact, it involves a single variability space T where GMM speaker- and

session-dependent supervectors $m_{s,h}$ are defined as

$$m_{s,h} = V + Tw_{s,h}, \tag{2.4}$$

where V is the GMM-UBM supervector and $w_{s,h} \sim \mathcal{N}(0, 1)$ is called total factor. The total factors, being hidden variables, are not observable, but can be estimated by their posterior expectation. The estimates are used as input features to a classifier and are known as identity-vectors or i-vectors for short.

In general, once the features have been extracted for an ASV task, computing the probability that the two audios belong to the same person is typically performed using two types of similarity measures: Cosine Distance (CD) and Probabilistic Linear Discriminant Analysis (PLDA). CD is calculated between the feature vectors x_i and x_t extracted from the two input and target utterances to compare and gives a measure of similarity. It consists on the normalized dot product between the two vectors:

$$CD(x_i, x_t) = \frac{x_i \cdot x_t}{\|x_i\| \cdot \|x_t\|}. \tag{2.5}$$

On the other hand, PLDA is a probabilistic approach to Linear Discriminant Analysis (LDA) which finds orthogonal axes for minimizing intra-class variability and maximizing inter-class variability. In fact, when a biometric trait has low intra-class variability, it demonstrates permanence and repeatability. On the contrary, if it has high inter-class variability, we can successfully use it to distinguish between people. Although these two are the most widely used techniques for ASV classification, many approaches make use of more common ML classifiers such as SVM.

2.1.2 Deep Learning Methods

The advent of DL has made it possible to implement many new state-of-the-art approaches, which very often achieve better performance than previous ones. In ASV tasks DL is typically involved in the feature extraction or classification stages. In the first case traditional ASV decision making techniques (e.g. CD, PLDA) are applied to DL extracted features. In the second case traditional ASV features (e.g. i-vectors) represents the input to an ANN.

Several works aim to extract features similar to i-vectors using ANNs. In [70] the authors propose a 4 hidden layers MLP in which the output of the last one is selected as feature vector and called deep-vector or *d-vector* for short. The classification probabilities are then computed by CD comparison. This method achieves similar performance to the i-vector one and exceeds it primarily for noisy environments. However, combining the two models significantly outperforms it in almost all possible operating points and noise conditions.

The work of [71] extends the d-vector method by presenting a MLP with a multi-task learning approach. Assuming each speaker has their own

style on each syllable or word, the authors provide speaker ids and texts as targets for the training stage. Finally, as for the d-vector approach, the features are extracted at the output of the last MLP's hidden layer and are called *j-vector*. The results show that this method outperforms the d-vector one.

In 2018, [72] proposed a novel feature extraction method, which is still at the basis of the state-of-the-art best performing techniques for ASV tasks. The extracted features are fixed-length and are called *x-vectors*. Similar to the previous methods it is still based on considering the output of the last hidden layer of an ANN, which in this case is a TDNN. The network consists of three 1D convolutional layers with different temporal contexts, a statistical pooling layer that computes the input sequence's mean and standard deviation, and two fully connected layers. The inputs of the TDNN are 24-dimensional filter banks with a frame-length of 25 ms and mean-normalized over a sliding window of up to 3 seconds. Furthermore, a Speech Activity Detector (SAD) is used to filter out non-speech frames. The first layers operate at frame-level and the deeper the network goes, the more the temporal context increases. For example, the second layer has a temporal context of 9, while the third of 15 frames. On the contrary, the statistics pooling layer aggregates frame-level outputs so that the subsequent layers deal with global information. After training, the x-vectors are extracted and used as input to a PLDA classifier. This approach has been shown to capture speaker characteristics across the entire recording and outperform the i-vector model, mainly when applied data augmentation.

In the context of multi-speaker conversation ASV, [73] extends the x-vectors architecture. This variant uses a slightly wider temporal context by adding another 1D convolutional layer and interleaves dense layers between the frame-level layers. Besides, differently from the original model, it passes vectors of 30 MFCCs extracted from each frame as input to the TDNN.

The described x-vectors approaches give equal importance to all the hidden vectors extracted since they apply statistic pooling to all frame-level outputs without distinction. However, some frames could contain more speaker-discriminative information than others. For this reason [74] replaces the statistic pooling layer with an attention-layer to assign different importance to frames based on their content. This new layer extracts from the outputs of the previous ones a weighted mean and a standard deviation vector. It updates its weights to maximize speaker classification performance for the entire system. This model uses a 23-dimensional MFCCs feature vector extracted from each frame as input. The results show that this approach exceeds previous baselines, especially when the number of attention heads increases, capturing different aspects of a speaker's speech.

In [75] the authors propose a new pooling method for deep speaker embeddings extraction, called attentive statistic pooling, believing that

both higher-order statistics and attention mechanisms are effective at discriminating between speakers. The vector of averages and standard deviations output from this layer thus considers frame-level features, scaled by attention.

Finally, a recent model, ECAPA-TDNN presented in [15] enhances to the x-vector approach just discussed, and places greater emphasis on channel attention, propagation, and aggregation using Squeeze-Excitation [76] and Res2Net [77] blocks, extra skip connections and channel-dependent attentive statistics pooling. This system brings significant improvements over previous models and, being an integral part of our work, will be discussed later.

Rather than reinforcing the representation of a speaker's voice through robust and discriminative deep features, DL can be used for scoring and comparison instead of traditional techniques such as PLDA and CD. For instance, in [78] an ANN replaces the UBM in the GMM-UBM model while [79] introduces SincNet, an end-to-end CNN architecture for classification purpose. Since a comparison of two vectors is required in ASV, the loss function to use for training the ANN is the contrastive loss that calculates the distance between the network output for a positive example and an example of the same class and contrasts it with the distance from the negative examples.

2.2 Audio Deepfake Generation Techniques

The advent of DL has been revolutionizing many aspects of life for a few years now [80, 22, 23, 24]. Many fields have greatly benefited from this new technology and new scenarios and possibilities are opening up. DF technology is one of these and has lately been in the spotlight arousing great fascination, but also concerns and alarm. It refers to a category of synthetic multimedia content, such as videos, audios or photos, generated through AI techniques that depict individuals in actions and behaviors that do not belong to them [81, 82]. Such forgeries can make a person say or do something they never said or did. Thanks to their realism they pass it off as true and jeopardize their reputation. The term *deepfake* comes from the nickname of a Reddit user who first posted online in 2017 some digitally generated videos of famous actresses with their faces exchanged on pornographic content. Revenge porn is one of the most common uses, in which the victim becomes the protagonist of a porn video and this happens especially for famous actresses for whom there is enough media material available online [26]. Indeed, politicians are also attractive targets for counterfeit videos [25]. Often DF videos are used to discredit their image, pilot elections or spread fake news [83, 84]. Likewise, DF voices have proved equally dangerous by fooling ASV systems into accessing the victim's personal information and committing fraud or by providing support for voice phishing attacks [28, 29]. If initially generating DFs was very complicated and required specific knowledge, today they are

easily accessible thanks to applications such as FaceApp [85], ZAO [86], Wombo [87] and VoiceApp [88]. Using these you can create a DF in a few seconds without any computer science knowledge. A further reason that has led to the spread of DFs is the massive amount of data and helpful material to train their generation models available online as a result of the proliferation of mass media. This has led to even more realistic counterfeit media almost impossible to distinguish from the original ones.

Regarding the audio domain, which is our case study, there are different methods to generate counterfeit speech: impersonation, replay speech, Text-to-Speech (TTS) and Voice Conversion (VC) techniques. The first two do not require the use of DL. Impersonation is performed by a professional voice actor who can realistically emulate a target voice. The impersonator usually tries to imitate both timbre and prosodic aspects such as pronunciation, accent, etc. Replay techniques consist of cutting, concatenating and clipping different recordings of the victim's voice to modify the message conveyed by the uttered sentence. On the contrary, TTS and VC are techniques that largely benefit from DL and represent the most powerful tools to generate spoof speech. In the following sections we will review the state-of-the-art of these last two methods since they are considered in our work.

2.2.1 Text-to-Speech

TTS is a technique able to generate a target speaker voice from a given input text that defines its linguistic content. It has been around for many years, but while the earliest methods generated unrealistic and low-quality results [89, 90, 91], this technique has undergone significant improvements with the advent of DL. It is possible to train an ANN on the vocal characteristics of a speaker so that it is then able to replicate them while changing the linguistic content.

The first breakthrough in this regard was *WaveNet* [92] presented in 2016. WaveNet is an ANN for generating raw audio waveforms capable of emulating the characteristics of many different speakers in a much more natural way than previous methods. It relies on the computation of the joint probability $p(x)$ of a waveform $x = x_1, ..., x_T$ which can be written as:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1}). \tag{2.6}$$

To do so it uses a CNN made of causal convolutions that ensure the preservation of temporal order. In addition, it does not employ pooling layers to increase the receptive field but dilated convolutions in which the filter slides through the signal values with a specific step. In this way, the output, a categorical distribution over the next x_t value, maintains the same length as the input. This architecture was the state-of-the-art until the release of *Tacotron* [16] in 2017 by Google.

Tacotron is a seq2seq model, which includes an encoder, an attention-based decoder, and a post-processing net [93, 94]. Unlike WaveNet, this model takes text characters as input and outputs the corresponding raw Spectrogram, which is then fed to a vocoder to synthesize speech. Instead of working at sample-level as WaveNet does, it works at frame-level and, therefore, can generate audio in much less time. The encoder extracts high-level representations from the input text while the decoder generates the Spectrogram from these representations, applying attention to each step. Finally, the task of the post-processing net is to generate the waveform from the Spectrogram. For this purpose, the authors use Griffin-Lim, but any other synthesizer can be employed.

The following year Google unveiled a new version of the model called Tacotron 2 [11], which retains the same seq2seq structure of encoder, decoder and synthesizer. As it is shown in Figure 2.1 the main difference

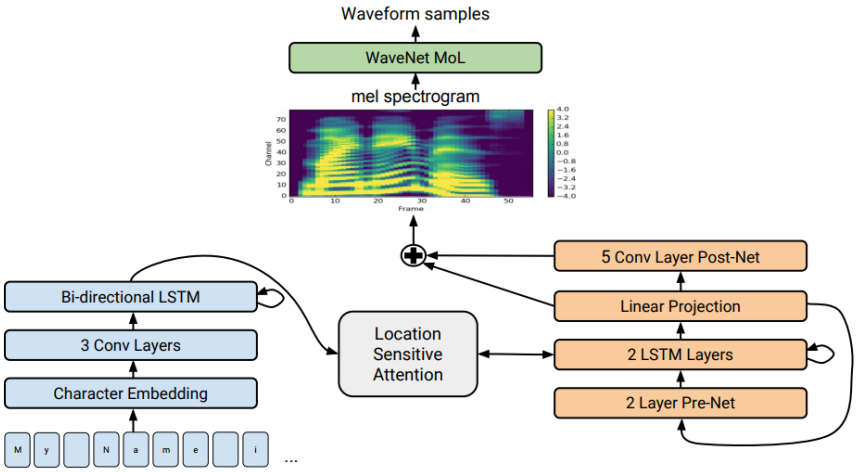


Figure 2.1: Tacotron 2 architecture [11].

concerns the use of a modified version of WaveNet (which takes a Mel Spectrogram as input) as a post-processing net instead of Griffin Lim, so as to take the best of the previous two approaches. This new version of Tacotron achieves results that are almost indistinguishable from natural speech and is therefore still one of the state-of-the-art TTS models.

The above-mentioned powerful end-to-end speech synthesizer models have enabled the production of large-scale commercial products, such as Google Cloud TTS services (both Standard and Wavenet) [95], Amazon AWS Polly [96], Microsoft Azure [97] and IBM Watson [98]. In this way, the creation of digital voices that are indistinguishable from real ones is now within everyone's reach.

2.2.2 Voice Conversion

VC consists of modifying a source voice to resemble that of a target speaker while keeping the linguistic and paralinguistic content unchanged.

Therefore, the input and output audio signals share the same text and prosodic characteristics, while they differ in timbres, which identify two different speakers. VC techniques often involve high-level speech features such as voice timbre and prosody. This makes it particularly suitable for the task of expressive speech synthesis.

The earliest VC models [99, 100, 101] were based on spectrum mapping using parallel training data, where speech samples from both the source and target speaker uttering the same linguistic content are required during training. However, it is not easy to find large-scale paired source and target speaker utterances in the real world, so lately much more sophisticated models that do not impose this constraint have been developed. These are mostly Generative-Adversarial-Network (GAN) [102] based approaches, able to learn a mapping from source to target speaker without relying on parallel data and are therefore called non-parallel.

CycleGAN-VC [103], unveiled in 2018, is an adaptation to the VC task of a network originally aimed at unpaired image-to-image translation [104]. It can capture sequential and hierarchical structures while preserving linguistic information, simultaneously using adversarial loss for inverse mapping and cycle-consistency loss for direct mapping. The first measures how similar the target and generated audio signals are, while the second is responsible for preserving the contextual information of the input audio. During training, they induce the network to find <input, output> pairs with the same contextual information and desired timbre. However, the results show a perceptual gap between the real target and converted speech.

An improved version of the previous model, *CycleGAN-VC2* [105], was presented to fill this gap. To mitigate the over smoothing problem caused by the cycle-consistency loss the authors introduce an additional discriminator and use adversarial losses twice for each cycle. Moreover, instead of using a 1D-CNN generator as for CycleGAN it employs a network architecture called a 2-1-2D CNN, built as a stack of 2D, 1D and 2D convolutions. This results in downsampling and upsampling, which is good at effectively capturing the wide-range structures. As a final enhancement, the discriminator, called patchGAN [106], analyzes the data section by section and determines their authenticity. This requires fewer parameters than considering the entire audio at once and therefore eases the training. Although CycleGAN-VC2 generates good results, a significant limitation of the method is that it performs a one-to-one VC. This means that it can only generate audio belonging to the one identity on which it is trained. Including multiple speakers would require training new generators and classifiers independently, even though they all represent speech and thus have a certain amount of common latent features that can be shared across different domains.

To overcome this limitation *Star-GAN* [107] was presented as a non-parallel many-to-many VC method capable of simultaneously modeling the identities of multiple speakers. It has a similar structure to Cycle-GAN,

but with adaptations that allow the speaker's identity to be considered as well. This time the objective of the generator is not only to create a realistic sequence of acoustic features from a given input but also to make it as similar as possible to the considered identity. On the other hand, the role of the discriminator is to produce a probability that the generated audio is true. At the same time, a new component called classifier computes the probability that the audio belongs to the considered identity. The loss function thus includes a new component concerning Cycle-GAN, called Domain Classification Loss, which induces the network to maintain speaker identity consistency. This allows the model to learn multiple identities simultaneously and switch between them while maintaining some shared attribute domains.

Lately, some new techniques [108, 109, 110] are pushing the envelope even further to make VC faster and more flexible. These are called one-shot methods and implement an any-to-any mapping that requires neither the input nor the output speakers to have been seen during training. They need only a reference utterance for the target speaker and the input audio to be transformed to do this. This allows any target voice to be modeled without needing a lot of data at hand and makes VC accessible to everyone, as there is no need to train the model on any identity.

2.3 Prosody Modeling

Prosody is the part of speech consisting of suprasegmental aspects such as rhythm, intonation, stress, style, etc., which can enrich with different nuances of meaning the linguistic content of a sentence. Generally, this aspect is responsible for giving the voice expressiveness and reflects the style and emotional state of the speaker. For this reason, prosody modeling is a very active field of research, especially concerning speech synthesis. While VC techniques can transfer the expressive cues of the input audio to the output, TTS ones cannot, since they generate audio from text and can result in flat and unexpressive voices. For this reason the TTS techniques benefit most from prosodic reinforcement to their model. Another challenging task in this regard is prosody labeling. Given the difficulty in formalizing prosody, there are no clear and shared labels to define it and the annotation process is difficult. For this reason many methods seek a criterion to categorize different nuances of prosody, which we can also include in expressive speech synthesis pipelines. Moreover, as described previously, ASV systems can also rely on and benefit from modeling prosodic aspects. Indeed, since the human voice is described by a set of features that are both timbral and physiological as well as emotional and adaptive, including a prosodic representation helps explain it more in-depth.

2.3.1 Prosody Labeling

Prosody labeling refers to the task of assigning an audio file one or more labels that describe and identify its prosodic content. Unfortunately, there is no single standard for transcription or symbolism of prosody as there is, for example, for phonetic representations.

A first attempt to meet this need was *ToBi* [111], a standard approach to label english prosody, designed in 1992 by a group of researches with expertise in prosodic analysis and speech technology. The system consists of parallel tiers, reflecting the multiple components of prosody, each comprising symbols for prosodic events and their timestamps. The highest level one is the tonal tier, closely resembling traditional intonational analysis. Then, the break index tier adds to the tonal information a representation of the rhythmic structure as well as the nature of pauses and lenghtening between adjacent words. In fact, in addition to tonal makeup, utterances can differ in the way words are grouped or separated by non-tonal means. Finally, according to ToBi there is one last type of variability in speech: hesitations, disfluencies, breaths, laughter, false starts, and other spontaneous dynamics. Their onsets and offsets are marked in the various tier. While this method offers a wide range of annotations, it also allows transcribers to express uncertainty in some circumstances to avoid forcing decisions into all-or-nothing choices.

In 2010, the ToBi standard was digitized by a publicly available tool called *AuToBi* [112], able to automatically hypothesize the presence and type of prosodic events in a spoken utterance. If provided with both the audio track and its word segmentation AuToBi can perform different tasks: pitch accent detection and classification, intonational and intermediate phrase detection, phrase accent classification and boundary tone/phrase accent classification. It first extracts pitch, intensity and spectral information from the speech signal. These acoustic contours are then aligned to the word-defined regions. Then, a series of different classifiers, one for each task, is run on the acoustic contours of every word and the prosodic events are predicted.

Another approach called *SLAM* [113] performs the automatic labeling of intonation assuming that a specific dictionary of elementary contours can be derived for each linguistic unit. To do so, it analyzes for each unit the variations in the contour of f_0 that are considered relevant to the description of the speech prosody. The method is based on the French language, but can be easily adapted to other stressed languages.

2.3.2 Expressive Speech Synthesis

Now that the synthesized voices are almost perceptually flawless, more emphasis is being placed on the emotional side to make them even more natural. This is why prosody modeling is becoming increasingly crucial since, in order to provide true human-like speech, a voice generation system must learn to reproduce this aspect. TTS systems in particular

would benefit from that as they only take a textual input without explicit prosody specifications for the generation process.

The first attempt in this direction was a tool presented in 1994, called *ProTran*, aimed at prosody transplantation for improving synthetic speech [114]. This technique consists of including pitch and duration values taken from a reference audio with the desired prosody in the phonetic transcription of the phrase to be synthesized. The obtained enriched phonetic transcription, containing linguistic and prosodic annotations, is the input to a TTS system.

Given the complexity in annotating prosody with handcrafted labels, underlined in the previous section, [115] treats emotional speech synthesis with an unsupervised clustering approach to produce automatic expression annotations. The method extracts prosodic features such as f_0, voicing probability, jitter, shimmer, logarithmic harmonic to noise ratio and their statistics from audiobook tracks. Then, it uses a hierarchical k-means algorithm to place expressively similar utterances in equal or nearby clusters. Finally, a HMM-based speech synthesis is trained incorporating the expressive cluster assignments as context features.

Many works aim to augment the Tacotron model [16, 11], being a state-of-the-art for TTS techniques, through the inclusion of explicit prosodic aspects and controls. For example, the authors of [116] present a fully automatic method that makes Tacotron prosody-aware. This is done through a small set of naturally disjoint and interpretable prosodic observations, called *prosody info*, evaluated at utterance level. During training, the system is provided with measured prosody info vectors, embedded in a 2-dimensional latent space (one dimension represents pace and the other represents pitch) and concatenated with each vector in the Tacotron encoder output sequence. Then, a prosody-info prediction module is separately trained to extract info vectors from the phonetic sequence output by the encoder. In this way at inference time the model does not require external prosodic controls but automatically extracts them from the encoder output and enables to modify them in order to obtain the desired result. This mechanism is proven to improve the expressivity of synthesized voices while preserving or even improving its quality and naturalness.

Similarly [117] presents an extension of the Tacotron architecture, which aims to learn syllable-level discrete prosodic representations from speech data in order to transfer or control the output prosody. The authors change the text-encoder from phoneme-level to syllable-level since it is more appropriate for working with prosody. Then, a continuous prosodic representation based on prosodic features, such as f_0, intensity, and duration is extracted for each syllable and discretized through a Vector-Quantised Variational Auto-Encoder (VQ-VAE). Finally, as for the previous approach, the discrete representations are concatenated with text encoder output, generating speech with a desired prosody.

In 2017 a research group from Google introduced a method to control

the prosodic style of a synthesized voice [12]. They achieved this by including the concept of *style tokens* to the Tacotron architecture. In practice, they added to the model a new encoder, called style encoder, which takes K style tokens as input, and outputs fixed-length embedding vectors fed to a style attention layer as shown in Figure 2.2. On the decoder side text and style attention heads are computed in parallel and their context vectors are combined through a weighted sum. The whole set of tokens can be interpreted as a latent representation of the utterance's style, obtained as a weighted sum of independent prosodic styles represented by each token. Therefore, the authors demonstrate how, by modifying single tokens, learned in an unsupervised way, one can modify independent stylistic voice features, e.g., sloppy style, robotic-sounding style, high-pitched voice etc.

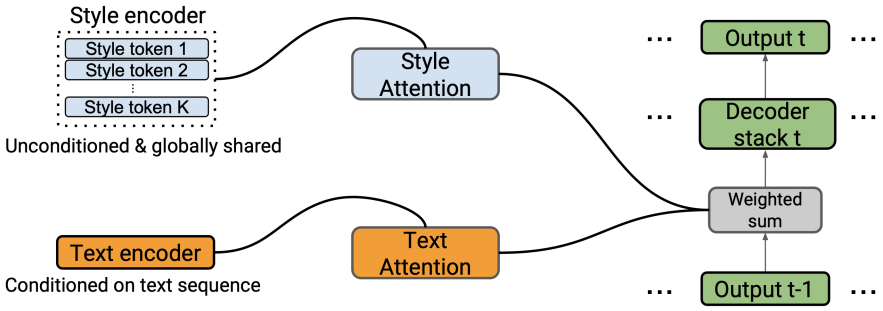


Figure 2.2: Expressive Tacotron Architecture presented in [12].

In [17] a similar approach is presented relying on the extraction of prosody embeddings used to condition Tacotron synthesis. Compared to the previous case, the difference lies in how these prosody embeddings are extracted and transferred between input and output. In fact, in this case the authors add to the model a prosodic encoder that receives an audio with a target prosody and represents it as a vector that takes part of the attention context. As a result, the synthesized audio will show the same expressive style of the target one. The authors show that the results match the prosody with fine temporal detail even when the target and reference speakers are different. In this way, the transfer of prosody between utterances is almost a speaker-independent operation. We will further analyze this model in the following sections, being it part of the proposed system.

The work presented in [118] encompasses the two previous approaches since it uses the same prosody encoder presented in [17], but then the prosody embeddings are fed as input to a global style token layer similar to the one in [12].

However, these methods using fixed length prosodic representations cannot retain all the temporal information, allowing to change the expressive style at utterance-level (e.g., a happy or sad mood), but not at a specific moment of speech. To solve this problem [119] introduces two types of prosody control methods, speech-side and text-side, which

involve the use of variable-length prosody embeddings to enable sequential prosody control. Speech-side ones have the same length as the reference speech, while text-side ones have the same length as the input text. To do this, the authors align and downsample the prosody embeddings to match the number of decoder time steps or the number of encoder time steps, respectively. The method outperforms [118] and interestingly shows that it is able to transfer the prosody of a song to another speaker, resulting in a vocal conversion of a song.

2.4 Deepfake Detection

As previously anticipated, audio DFs represent a great threat to ASV systems as they can be used to fool them. In this respect, there are two ways of attacking an ASV system: Physical Attack (PA) and Logical Attack (LA). The former involves the reproduction of the counterfeit victim's voice through an audio device. In this way the recognition system sensor that captures the audio receives a recorded and played back sound wave, not directly uttered by the speaker. This is why such an attack is said to occur at sensor level. The latter consists of injecting the counterfeit audio signal after the sensor pretending it was captured by it.

Given the threat posed by these attacks on ASV systems, there is an urgent need to address a DF detection task, where a speech audio x must be labeled as $REAL$ or DF

$$f(x) = \begin{cases} REAL \\ DF \end{cases} \tag{2.7}$$

This verification can be incorporated as the last block of an ASV system to ensure the authenticity of the input voice and foil malicious attacks.

In recent years, audio DF detection has become a hot topic in the forensic research community, as it tries to keep up with the rapid evolution of major counterfeiting techniques, namely TTS and VC, described in Section 2.2. To this end, there are international challenges with the purpose of sustaining the research, where participants compete to implement the best anti-spoofing systems. The Automatic Speaker Verification and Spoofing Countermeasures (ASVspoof) challenge is the most famous one and played a key role in advancing research on spoofed speech detection to protect ASV systems from manipulation. Held for the first time in 2015 [120] it is repeated every two years presenting participants with new tasks, data, or baseline methods to compare against. Until 2019 [121] it proposed competitions for LA and PA, considering the spoof speech detection task in relation to an ASV system. However, in the latest version, held in 2021 [122], a task exclusively finalized to audio DF detection has been added. Other than ASVspoof, similar challenges are emerging such as the brand new Audio Deep synthesis Detection (ADD) challenge launched in 2022 to further accelerate research on deep synthesis detection

and manipulated audio [123]. It includes new algorithms of TTS and VC and, differently from ASVspoof, some challenging attacking situations in realistic scenarios, e.g., diverse background noises and disturbances or several small fake clips hidden in a real speech audio.

In general, DF detection methods start from similar assumptions, be it an image, audio or video and can be divided into two main groups. The first one focuses on low-level aspects, looking for artifacts introduced by the generators at the pixel or sample level, while the second one focuses on higher-level features representing semantic aspects.

2.4.1 Artifacts-based Approaches

Very often, DF synthesis processes leave traces in the form of artifacts, which serve as hidden fingerprints that we can exploit to determine the authenticity of the given data. For example, the authors of [31] perform detection of AI-generated fake images by looking for artifacts through high-frequency component analysis. They show how the energy distribution analysis in the high-frequency domain leads to very good classification results in both supervised and unsupervised scenarios, even when only a few annotated training samples are available.

On the other hand, [124] addresses the same problem considering a set of local features specifically designed to model the convolutional traces that could be left in DF images. This extraction is performed as a reverse engineering problem at the last computational layer of the generation network and such features are extracted in an unsupervised way though the Expectation Maximization (EM) algorithm. The method considers five different types of DF generation techniques, all relying on GANs, and not only it is able to classify an image as fake but also to predict the most likely technique used to generate it.

The work of [125] aims to secure ASV systems against PAs through channel pattern noise analysis. In fact, in PAs converge different types of noise introduced by intermediate recording and playback devices. Therefore, the authors train a SVM to capture this manipulation fingerprint and thus detect spoof attacks.

In [13] the authors assume that a real recording has greater non-linearity than a counterfeit one. This is motivated primarily by two elements: the neural networks used in spoofing speech generation fail to introduce enough non-linearity into the process, and typical non-linearity due to microphone processing blocks are absent in synthesized audio. Then, they use specific features to analyze these aspects, such as bicoherence, i.e., a normalized version of 2-dimensional Fourier transform of the third-order cumulants. Figure 2.3 shows the marked difference between the phase spectra of the bicoherence estimated from the spoof and real speech. It is evident from the graphs how flat and monotonous the bicoherence phase of spoofed speech is compared to real speech.

Bicoherence is also employed in [126] along with a number of features

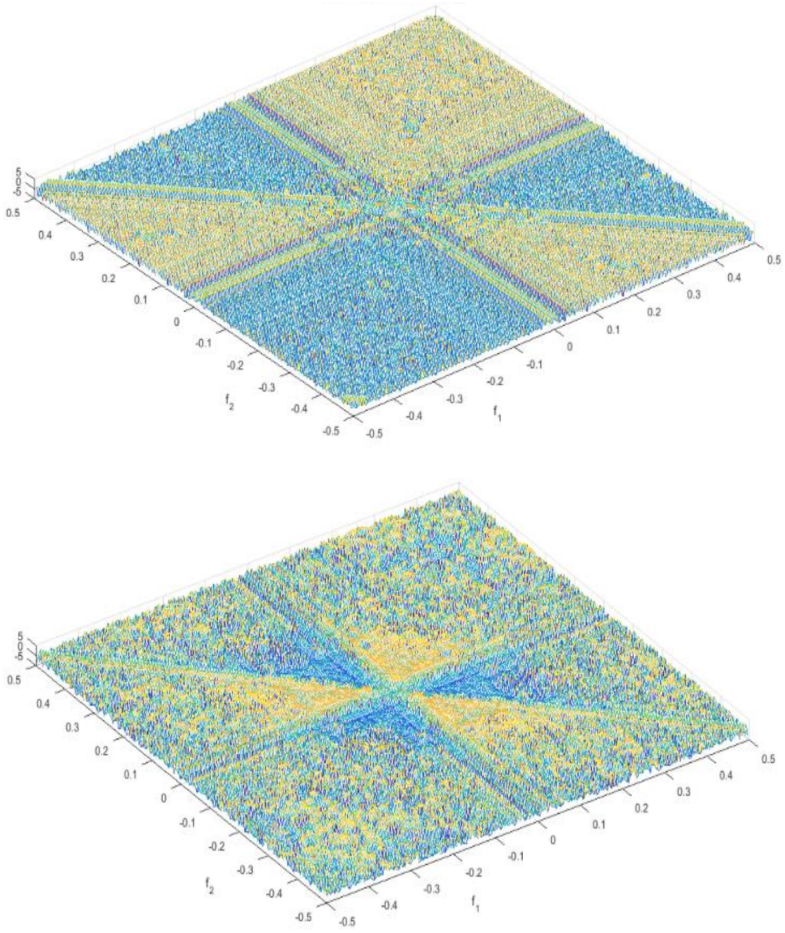


Figure 2.3: Phase spectra of bicoherence estimated from a spoof (on top) and real (on bottom) speeches [13].

based on the idea of modeling speech as a auto-regressive process. The authors investigate whether these features complement and benefit each other. They conduct tests in both closed-set and an open-set scenarios. In the former, the method detects whether the speech is real or synthetic and, if so, the generation technology. In the second, it is able to detect whether a fake speech was generated through a never-before-seen algorithm. The research shows how the combined use of these features provides an accuracy gain in some situations.

2.4.2 Semantic-based Approaches

Some DF detection methods rely on more semantically meaningful features and exploit high-level inconsistencies to discriminate them, assuming their weakness in emulating the finest aspects of human nature.

Sometimes, for example, synthetic processes are unable to replicate spontaneous and involuntary physiological activities such as breathing, pulse or eye movements, which are intrinsic to human beings. The work of [33] leverages just that, detecting the lack of natural eye blinking in syn-

thesized videos. The model employs a Long-term Recurrent Convolutional Neural Network (LRCN) obtained as the combination of a CNN and a RNN. In this way it is able to capture both the phenomenological and temporal regularities in the eye blinking process, as the CNN generates single-frame predictions while the RNN analyzes temporal coherence.

Other times a mismatch between some interrelated semantic aspects is present in DFs. For example [127] looks for dissimilarities between the audio and visual modalities in DF videos assuming that manipulation of either one leads to dis-harmony such as loss of lip-sync, unnatural facial and lip movements, etc. The system takes both audio and video signals as input and via the Modality Dissonance Score (MDS) labels the data as real or fake. The MDS is modeled after contrastive loss, which has traditionally been used to discover lip sync problems in video.

On the other hand the authors of [128] perform face-swap DF detection by comparing two different estimates of the subject's head pose, one involving all facial landmarks and one only the central region. The results show that the two estimates are close for original faces, while there is a significant difference for face-swap DFs since the central region belongs to a different person.

Similarly, [129] approaches the same problem combining a static biometric based on facial recognition with a temporal, behavioral biometric based on facial expressions and head movements. The first is extracted using VGG [130], a state-of-the-art method for face recognition tasks. For the second, the authors adapt a network called Facial Attributes-Net (FAb-Net) [131], to distinguish spatio-temporal behavior features between individuals. In fact, the original FAb-Net captures frame-based movements and facial expressions well, but is identity agnostic. Finally, for each video they analyze through a cosine-similarity metric how much these two representations are coherent between them and in this way perform classification.

Other methods leverage semantic aspects that are very difficult to formalize and replicate, such as human emotional behaviors. For example, the work presented in [34] exploits the lack of emotionality in synthetic voices generated via TTS techniques to recognize them. To do this they take the Convolutional Recurrent Neural Network (CRNN) originally proposed in [132] for a Speaker Emotion Recognition (SER) task and consider as emotional embeddings the output of the final attention layer. These representations are fed to a SVM classifier that learns the mapping between the emotional content of an audio and its authenticity.

Finally, the work presented in [14] leverages the inconsistency between emotions conveyed by audio and visual modalities in a joint audio-visual DF, as shown in Figure 2.4. The authors rely on a valence-arousal diagram used to describe and represent emotions. Valence denotes an emotion's positivity or negativity level, while arousal its intensity. The model extracts low-level descriptors from audio and video signals and uses them to compute valence and arousal values through a supervised classifier.

Finally, depending on the coherence of these two values it determines if the video is authentic or a DF.

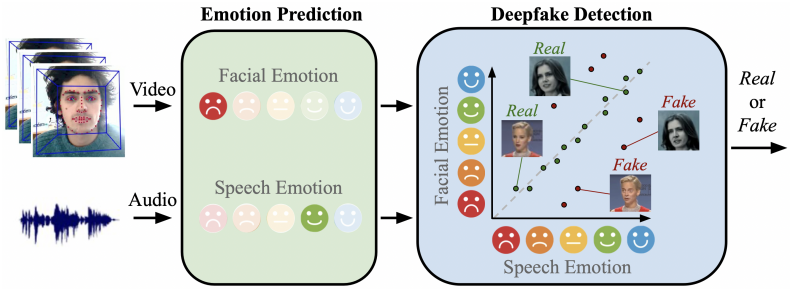


Figure 2.4: DF detection method exploiting audio-visual emotion analysis proposed in [14].

2.5 Conclusive Remarks

In this chapter, we have first reviewed the main state-of-the-art works related to ASV, showing how voice can be used as a biometric. Afterwards, we have introduced the main techniques for DF generation, with a focus on synthetic voices and showed how prosody modeling can make them even more expressive and natural. Finally, we have examined how audio DFs pose a threat to ASV systems and illustrated the main techniques to detect them.

3
Proposed System

In this chapter we better formalize the problem we are going to address, namely, audio DF detection. Next, we present our proposed method, that leverages the extraction and concatenation of high-level semantic features from an input audio signal.

3.1 Problem Formulation

The problem we want to tackle in this thesis is that of audio DF detection. Formally, given a discrete-time input speech signal $\mathbf{x}$ sampled with sampling frequency F_s, the goal is to predict the associated label y such that

$$y \in \{\mathrm{REAL}, \mathrm{DF}\}, \tag{3.1}$$

where REAL identifies authentic speech samples, while DF corresponds to speech that has been synthetically generated. We consider as REAL audio signals those containing the speaking voice of a real human, directly captured by a microphone. Consequently, any artifacts and compressions they may contain are not considered as forgeries, but as audio processing operations. On the other hand, we consider as DF any audio that has been altered or fully synthesized. In particular, our work focuses on DFs generated using the two main speech synthesis techniques, TTS and VC, discussed in Section 2.2.

3.2 System Architecture

We propose a method for audio DF detection named *ProsoSpeaker*. Figure 3.1 shows its complete pipeline. Our approach leverages the difficulty

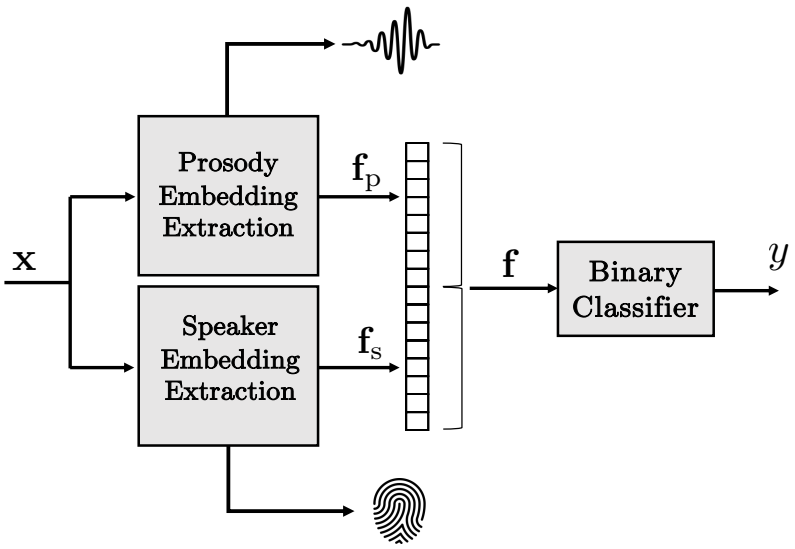


Figure 3.1: Pipeline of the proposed system.

of DFs in generating complex semantic aspects of voice naturally. Hence, the proposed *ProsoSpeaker* method relies on a rich set of high-level features that we extract from the input x. We will refer to each one of them as *speaker* ($\mathbf{f}_\mathrm{s}$) and *prosody* ($\mathbf{f}_\mathrm{p}$) embeddings. Then, we use this representation as input to a fast supervised classifier, which outputs for each input $\mathbf{x}$ a prediction of the label y.

3.2.1 Speaker Embedding Extraction

The principle of VC algorithms is to operate on pristine speech signals and modify their frequency content to match a target identity. We believe that this kind of forgeries could leave traces in the speaker timbre quality that we can leverage to perform synthetic speech detection. We propose to do so through a feature set that describes each voice's unique fingerprint compactly, extracting the spectro-temporal characteristics of the analyzed spokesperson, i.e., timbre specific properties or pitch contour of the voice. This feature set, that we indicate with $\mathbf{f}_\mathrm{s}$, is extracted exploiting a state-of-the-art network, called ECAPA-TDNN [15], originally proposed for an ASV task, as mentioned in Section 2.1.2. The proposed speaker embeddings can spot voice anomalies and allow us to discriminate between real and synthetic tracks generated through VC engines, as we will prove in the results section. The ECAPA-TDNN model enhances the typical x-vectors architectures [72, 73, 133] and significantly outperforms state-of-the-art TDNN based systems. Figure 3.2 shows an overview of its pipeline. The design allows projecting variable-length utterances into fixed-length representations. Being a TDNN, the model involves several 1D convolutions with $C = 1024$ channels that gradually increase the temporal context. Each of them comes with a non-linear ReLU activation function and a Batch Normalization (BN).

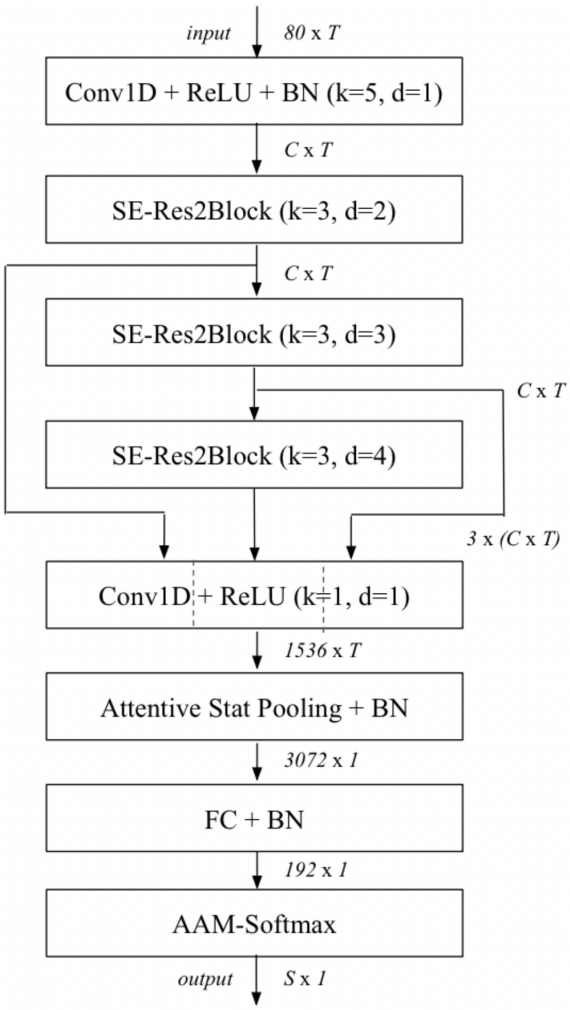


Figure 3.2: Architecture of the ECAPA-TDNN model described in [15]. k is the kernel size and d the dilation coefficient of the convolutional layers or SE-Res2Blocks. S is the number of training speakers, while the channel and temporal dimensions of the intermediate feature maps are denoted as C and T, respectively.

The ECAPA-TDNN architecture resembles the x-vectors one, widely used for ASV tasks, but enhances it with several improvements. The first one builds on the work of [75], which presents a new pooling method for extracting deep speaker embeddings, called attentive statistical pooling. It exploits both the effectiveness of higher-order statistics and attention mechanisms for calculating content-based weights at frame-level in an ASV temporal pooling layer. ECAPA-TDNN extends this temporal attention mechanism to be channel-dependent. The network focuses more on speaker characteristics that do not activate on identical or similar temporal instances, e.g., vowel-specific versus consonant-specific properties. Furthermore, by extending self-attention to global properties of the utterance, they enlarge the temporal context of the pooling layer. In this way the network better adapts to global audio properties such

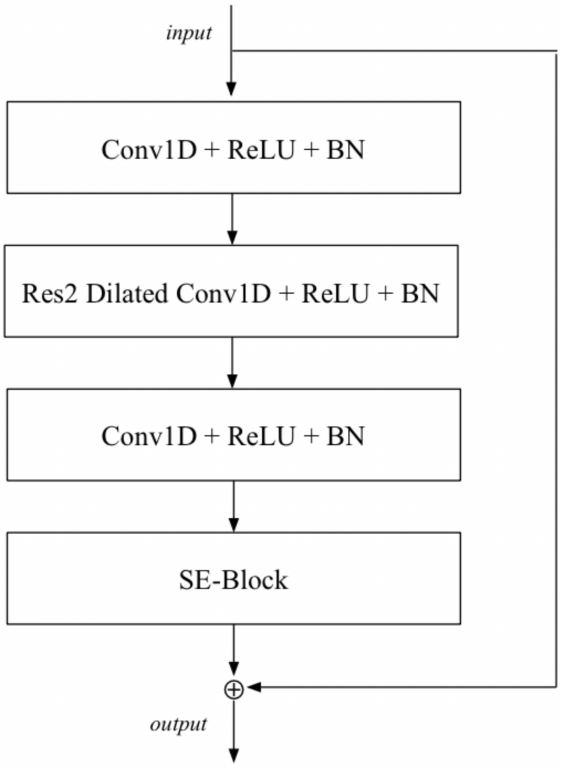


Figure 3.3: Insight into the architecture of a SE-Res2Block. [15]

as noise or recording conditions. The second type of improvements draws from recent trends in face verification and computer vision, such as *Squeeze-Exitation (SE) Blocks* [76], and *ResBlocks* [77]. The former models interdependencies between channels, while the latter involves skipping connections between layers and is widely employed in very deep ANNs since it solves the problem of vanishing gradient and exploits multi-layer information. ECAPA-TDNN exploits SE blocks to rescale the frame-level features, concerning global properties of the recording. However, to combine them with the benefits of residual connections without increasing the total number of parameters, it implements a SE-Res2Block. Figure 3.3 shows its structure. Two 1D convolutional layers with a unit kernel size serve to reduce and restore the feature size, respectively. A Res2Net module [134] improves the dilated 1D convolutional layer in the middle, making it capable of processing multi-scale features by building residual hierarchical connections within. Finally, a SE block scales each channel. The dimension of the convolutional bottleneck is 128, while the scale dimension in the Res2Block is 8. As a final computer vision influence, the last layer, the Additive Angular Margin (AAM)-Softmax, which computes the speaker probabilities during training, is widely used for face recognition tasks. The third and final enhancement concerns the aggregation of features extracted from different layers. Traditional x-vectors architectures only consider the output of the last frame-layer to

compute pooled statistics. On the other hand, it can be advantageous to correlate the deeper and therefore more complex features with the more superficial ones, strongly related to the speaker's identity. To do so, ECAPA-TDNN concatenates for each frame the features coming out from the three SE-Res2Blocks. In addition, the shortcut present in each SE-Res2Block reinforces this aspect, by further enabling the exploitation of multi-layer information.

During training, for each considered speaker, the model outputs the probability that the identity of the input voice belongs to them, through the AAM-Softmax layer. On the contrary, at inference time it measures the similarity of the identities of two different audios through CD between their embeddings, extracted after the fully-connected layer. We use this model as an embedding extractor by first training it for the original task (i.e., ASV) and then feeding the considered input $\mathbf{x}$ to the trained network. We then assume the output of the network as embedding representation discarding the final classification layer, hence adopting a transfer-learning strategy. In particular, the variable length signal $\mathbf{x}$ is first pre-processed, i.e., transformed in time-frequency domain applying a Short Time Fourier Transform (STFT) with window length W_s and hop size H_s. From the resulting Spectrogram $\mathbf{X}$ we compute a set of MFCCs, i.e.,

$$\mathbf{X}_{\mathrm{MFCC}} = \mathrm{MFCC}(\mathbf{x}) \in \mathbb{R}^{M_\mathrm{s} \times B}, \tag{3.2}$$

where M_s corresponds to the number of time windows and B is the total number of Mel-frequency cepstrum coefficients. This feature map is then used as input to the trained ECAPA-TDNN network, which projects it into the fixed-length speaker embedding $\mathbf{f}_\mathrm{s}$ of length N_s.

3.2.2 Prosody Embeddings Extraction

Complementary to the aspects described by the speaker embeddings, we believe that high-level prosodic aspects, like speech signal variations in rhythm, intonation and style, constitute another aspect we can leverage to discriminate deepfake speech tracks. The rationale behind this choice lies in the subtractive definition of prosody introduced in Section 1.1.3, which describes it as *the variation in speech signals that remains after accounting for variation due to phonetics, speaker identity, and channel effects*. It follows that speaker identity and prosody are two disjoint and complementary properties of speech. For this reason, we believe that a representation that includes both the aspects is capable of describing speech signals more comprehensively. Moreover, prosody measures an intrinsic human voice characteristic that we assume TTS synthesis algorithms struggle at recreating. In fact, despite the recent advances, synthetic prosody has different quality and intensity w.r.t. to human speech, and this difference can be captured using a set of prosody embeddings. The obtained results later prove this assumption. To extract this feature set, that we indicate as $\mathbf{f}_\mathrm{p}$ in Figure 3.1, we experimented with two different

extraction approaches based on the main prosodic modeling methods presented in Section 2.3. The first one employs hand-crafted features typically used to describe prosody, while the second one makes use of DL and extracts features learned by the network.

3.2.2.1 Handcrafted Features

Inspired by several state-of-the-art approaches to prosodic modeling, we extract a representation of prosody based on handcrafted features from an audio signal. As already mentioned in Section 1.1.3, most approaches derive them from three speech properties: the temporal contour of f_0, which encodes aspects related to intonation, such as the rising pitch in a question; the temporal contour of energy E_0, which encodes aspects related to emphasis and emotional depth; the duration of voiced, unvoiced and pause segments, associated with the speaking rate.

We tested two different approaches to feature extraction. The first divides each audio into overlapping segments using fix-length windows. The second divides each audio into non-overlapping segments of variable length in which the presence of voice is detected. We will refer to the features extracted by the first method as *windowed features* $\mathbf{f}_{\mathrm{w}}$ and the others as *voiced features* $\mathbf{f}_{\mathrm{v}}$. Before applying either method, we preprocess the audio. As a first step we eliminate the initial and final silence by analyzing the first and the last instant in which voice is detected. Subsequently, we either trim or zero-pad the signal to 4 seconds. Finally, we normalize the audio and shift the average f_0 to 150 Hz to achieve sex invariance.

Windowed Features

This method assumes that pauses in speech also play an important role in prosodic representation, together with the voiced component. For this reason, it partitions the audio into segments of constant length that may contain both voiced and unvoiced speech material. To do so, it employs triangular windows of length W_{w} and with a hop size H_{w}. Then, for each window it estimates the temporal contours of f_0 and E_0 with fine time detail. Finally, it extracts a set of 23 features we deemed relevant from the two contours, computed in each segment through minimum (min), maximum (max) and the first four statistical moments, average (avg), standard deviation ($stdev$), skewness ($skew$) and kurtosis ($kurt$):

- Features directly calculated on the values of f_0 for each segment: $avg(f_0)$, $stdev(f_0)$, $max(f_0)$, $min(f_0)$, $max(f_0){-}min(f_0)$, $skew(f_0)$, $kurt(f_0)$.
- Features calculated on the Tilt of a linear estimation of f_0 for each segment: $avg(tilt(f_0))$, $stdev(tilt(f_0))$, $max(tilt(f_0))$, $min(tilt(f_0))$, $skew(tilt(f_0))$, $kurt(tilt(f_0))$.

- Features calculated on the Mean Square Error (MSE) of a linear estimation of f_0 for each segment: $avg(MSE(f_0))$, $stdev(MSE(f_0))$, $max(MSE(f_0))$, $min(MSE(f_0))$, $skew(MSE(f_0))$, $kurt(MSE(f_0))$.

- Features directly calculated on the values of E_0 for each segment: $avg(E_0)$, $stdev(E_0)$, $skew(E_0)$, $kurt(E_0)$.

In this way we extract from each audio a matrix $M_\mathrm{w} \times 23$ where M_w is the number of windows and hence of audio fragments. To change from this two-dimensional representation to a one-dimensional vector we compute the same six statistics (avg, $stdev$, $skew$, $kurt$, min, max) for each feature along the temporal dimension M_w. As a result we obtain a vector of 138 features describing each audio.

To take explicitly into account the distribution of the pauses and of the voiced and unvoiced segments we extract an additional vector of features in a global fashion from the whole utterance. It is composed of 25 features:

- Voiced rate: voiced segments per second.

- Features based on the duration of voiced segments D_v: $avg(D_v)$, $stdev(D_v)$, $skew(D_v)$, $kurt(D_v)$, $max(D_v)$, $min(D_v)$.

- Features based on the duration of unvoiced segments D_u: $avg(D_u)$, $stdev(D_u)$, $skew(D_u)$, $kurt(D_u)$, $max(D_u)$, $min(D_u)$.

- Features based on the duration of pauses P: $avg(P)$, $stdev(P)$, $skew(P)$, $kurt(P)$, $max(P)$, $min(P)$.

- Features based on duration ratios: $\frac{P}{D_v+D_u}$, $\frac{P}{D_u}$, $\frac{D_v}{D_v+D_u}$, $\frac{D_u}{D_v+D_u}$, $\frac{D_v}{P}$, $\frac{D_u}{P}$.

To obtain the final representation of the prosody of an input speech signal we concatenate the two feature vectors described above into a single one of length N_w, which we denote as $\mathbf{f}_\mathrm{w}$.

Voiced Features

This method assumes that prosody depends on the sequence of words or syllables. Therefore, it is appropriate to only consider such voiced units for feature extraction, discarding the pauses between them. By analyzing the temporal contour of f_0, this method identifies voiced segments that plausibly represent the single words in the audio signal. To do so it estimates the temporal contours of f_0 with fine time detail and selects as voiced segments those portions of audio containing speech voice sounds that are at least D_v long, as depicted in the upper graph of Figure 3.4a. Then, for each segment it extracts a set of 36 features:

- f_0 and E_0 statistical features: the same set of 23 features of the windowed case described above.
- Duration of each voiced segment.
- f_0 and E_0 approximation coefficients: a set of 12 coefficients we derive from a Legendre polynomial expansion of the temporal contour of f_0 and E_0, respectively, as described in [135]. These coefficients can accurately approximate the two contours as

$$f(t) = \sum_{i=0}^{N} a_i L_i(t), \tag{3.3}$$

where $f(t)$ is either the frequency or energy contour, $L_i(t)$ is the ith Legendre polynomial, and N its order. Therefore, by setting N equal to 5 we end up with 6 coefficients. Figure 3.4 visually demonstrates the approximation of both f_0 and E_0 temporal contours for each voiced segment.

Consequently, the method extracts from each audio a matrix $N_\text{v} \times 36$, where M_v is the number of voiced segments, which is variable and generally much lower than M_w, used in the windowed case. To change from this two-dimensional representation to a one-dimensional vector we compute the six statistics (avg, $stdev$, $skew$, $kurt$, min, max) for each feature along the temporal dimension M_v. As a result we obtain a vector of N_v features, which we denote as $\mathbf{f}_\text{v}$, describing the prosody of an input speech signal.

3.2.2.2 Prosody Encoder

In order to extract a learned representation of audio prosody we rely on the reference encoder described in [17]. The authors originally introduced this module in Tacotron's architecture [16, 118] to improve the naturalness of the synthesized voices, enhancing their prosody controls.

The original Tacotron model, as already mentioned in section 2.2.1, performs a generative end-to-end TTS synthesis and generates speech starting from a text input, depending on the speaker's identity considered in the training phase. The core of Tacotron is a seq2seq model, comprising an encoder, an attention-based decoder, and a post-processing net [93, 94]. The model takes as input a sequence of characters from which the encoder extracts a high-level representation, successively transformed in Spectrogram by the decoder. Finally, a vocoder produces the corresponding waveform from the resulting Spectrogram. As a first step, the character sequence is encoded in an embedding where a one-hot vector represents each character. Then, each embedding undergoes a series of transformations, collectively called pre-net and is input to a CBHG-encoder, shown in Figure 3.5. CBHG stands for 1D convolution bank - highway network - bidirectional GRU and comes from the work of

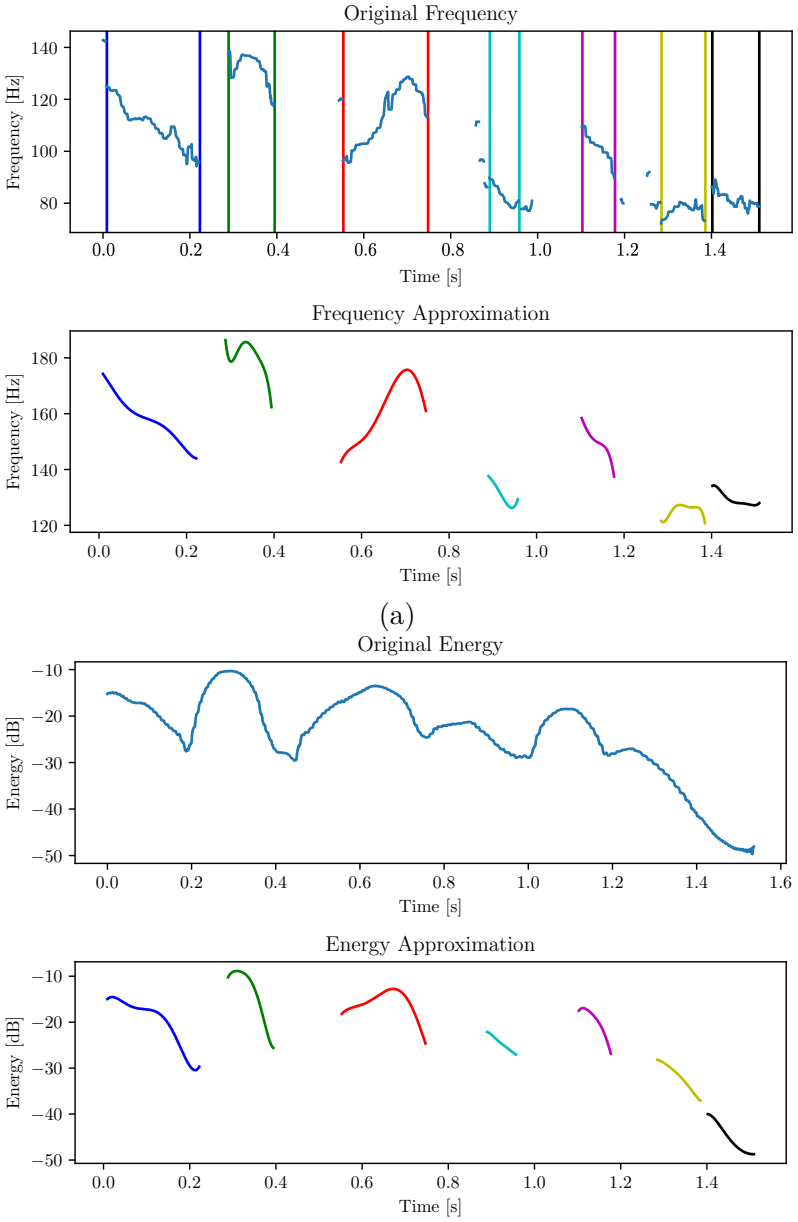


Figure 3.4: (a) and (b) show the comparison between the original temporal contour and the one approximated in each voice segment by Legendre polynomial for f_0 and E_0, respectively. In particular, in the upper plot of (a) is highlighted the division into voiced segments, represented by the colored sections. The text of the considered audio is “He will address the nation this evening" and intuitively each voiced segment represents a word.

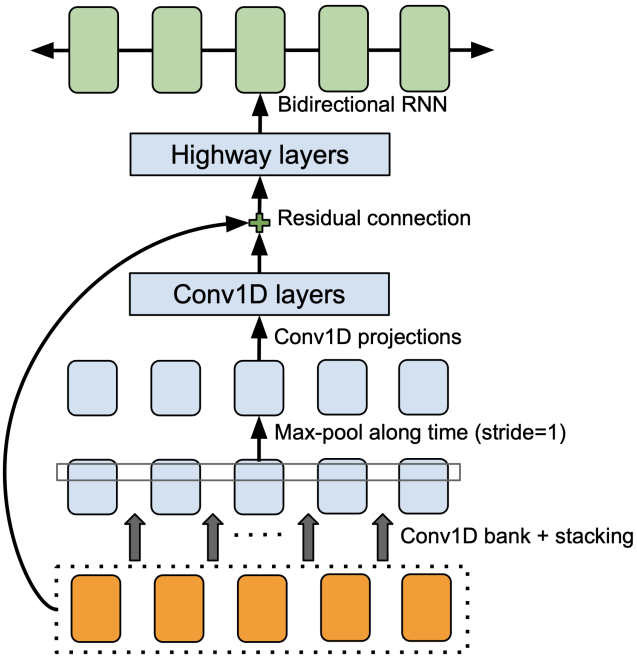


Figure 3.5: CBHG-encoder architecture [16].

[136] in machine translation. CBHG is a particularly effective module for extracting representations from sequences. Its architecture first convolves the input with K sets of 1D convolutional filters to explicitly model local and contextual information. Then, it stacks together the convolution's outputs and max-pools them along time to increase local invariances. Successively, it feeds the processed sequence to some fixed-width 1D convolutions and adds the outputs to the original input sequence via residual connections. Next, a multi-layer highway network derives high-level features from the convolutional outputs. Finally, a bidirectional GRU extracts sequential features from the context both forward and backward. From this transcript embedding Tacotron generates multiple Spectrogram frames simultaneously, using a content-based *tanh* attention decoder, where a static recurrent layer produces the attention request at each decoder time step. Once the Spectrogram is ready, a pre-trained vocoder transforms it into audio. In the original model [16] the authors employ Griffin-Lim, but we can use any other. Later implementations of the model use WaveNet [92] for this purpose [11]. During training, the first decoder step is conditioned on an all-zero frame and the model is trained on <text, audio> inputs, considering their 80-band Mel Spectrograms as targets.

The authors of [17], make two main improvements to the Tacotron's architecture as shown in Figure 3.6. First, they adapt it to the multispeaker scenario by conditioning the decoder also on the speakers's identity in the training set. Second, they include an explicit representation of prosody to make the output voices more expressive and be able to modify or transfer their style. This is done by broadcast-concatenating

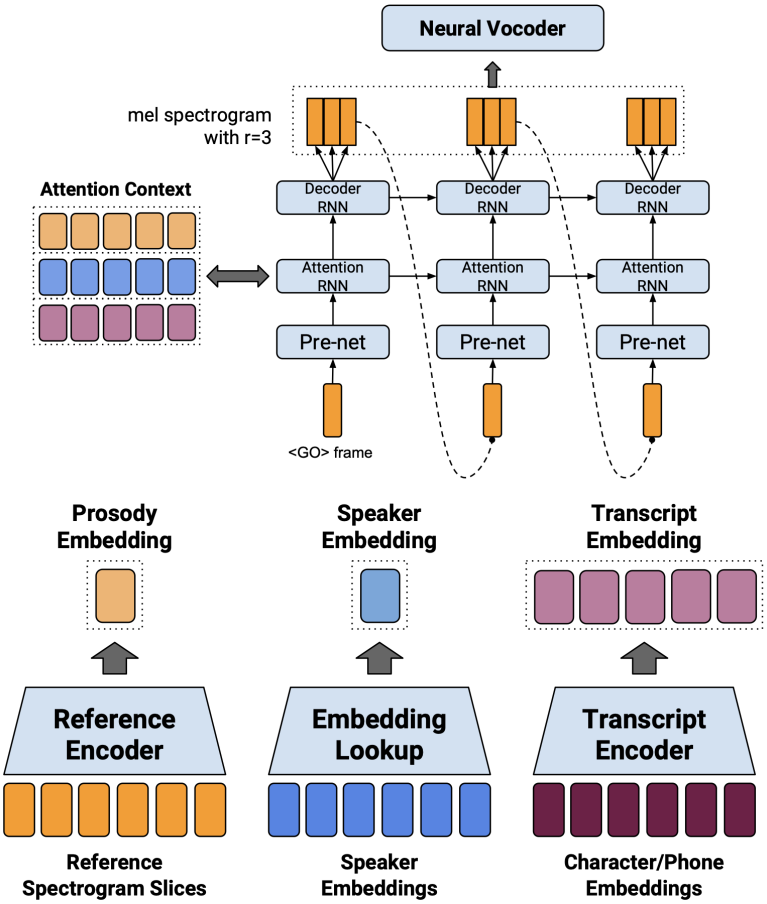


Figure 3.6: Pipeline of the prosody enhanced Tacotron presented in [17].

both a speaker embedding and a prosody embedding to the transcript one. Therefore, the matrix used as context for the attention layer has dimensions $L_{\mathrm{T}} \times (d_{\mathrm{t}} + d_{\mathrm{s}} + N_{\mathrm{e}})$, where L_T is the length of the encoded representation of the transcript, d_{t} is the transcription embedding dimension, d_{s} speaker embedding one, and N_{e} the prosody embedding one. When the encoder is conditioned on this learned embedding the synthesized audio is able to match the prosody of a reference signal with fine time detail even when the reference and synthesis speakers are different. Figure 3.7 shows an insight into the architecture of the reference encoder. It is a 6-layer CNN where each layer includes 3×3 filters with 2×2 stride, same padding and ReLU activation with BN. Compared to the downsampling rate the number of filters in each layer doubles by half: 32, 32, 64, 64, 128, 128. The $L_{\mathrm{r}} \times d_{\mathrm{r}}$ reference signal is downsampled 64 times in both dimensions. Then, a single 128-width GRU [137] compresses the sequence produced by the CNN layers down to a single fixed-length vector. The final output of the GRU is taken as the pooled summarization of the sequence. Finally, a fully-connected layer followed by a tanh activation function projects the output to the desired dimensionality, allowing to compute the N_{e}-dimensional embedding. This design sufficiently bottlenecks the input information to force the encoder to learn a compact

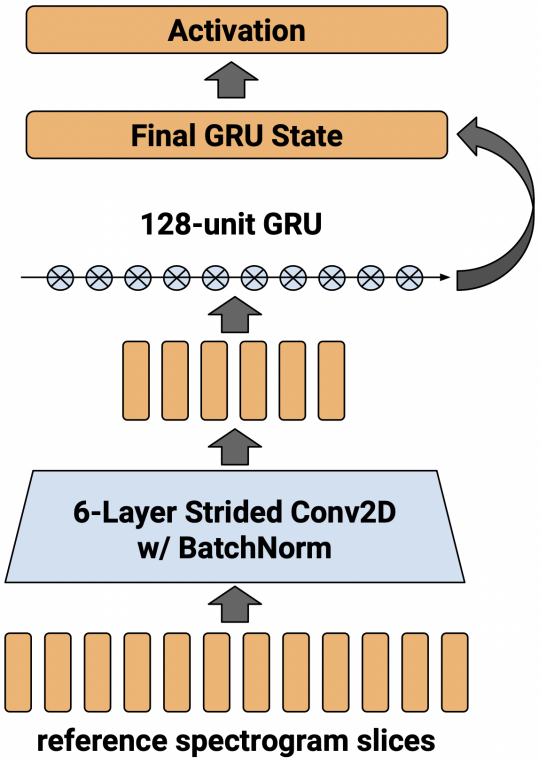


Figure 3.7: Architecture of the prosody encoder included in [17].

representation of prosody.

For this work, we train Tacotron and prosody encoder jointly by synthesizing target audio signals, provided as input to both, and using the reconstruction error as loss function. On the contrary, the target audio for prosody can be any utterance whose style is to be transferred to the output at inference time. Once the prosody encoder is trained, we use it as an embedding extractor, feeding as input the Mel Spectrogram of the input signal $\mathbf{x}$

$$\mathbf{X}_{\text{mel}} = \text{MelSpec}(\mathbf{x}) \in \mathbb{R}^{M_p \times K}, \tag{3.4}$$

where M_p is the number of time windows and K corresponds to the total number of frequency bins, extracted with window size W_p and hop size H_p. We denote the output of the prosody encoder as $\mathbf{f}_\text{e}$, of length N_e.

3.2.3 Classifier

As shown in Figure 3.1, the final part of the *ProsoSpeaker* pipeline is a supervised binary classifier. We concatenate the two embeddings $\mathbf{f}_\text{s}$ and $\mathbf{f}_\text{p}$ obtaining a final feature vector fed to the classification stage and defined as

$$\mathbf{f} = [\mathbf{f}_\text{s}, \mathbf{f}_\text{p}] \in \mathbb{R}^{N_\text{s}+N_\text{p}}, \tag{3.5}$$

where $\mathbf{f}_{\mathrm{p}}$ can be either $\mathbf{f}_{\mathrm{w}}$ or $\mathbf{f}_{\mathrm{v}}$ or $\mathbf{f}_{\mathrm{e}}$ depending on the prosody extraction method used and N_{p} is its length which can be can be N_{w}, N_{v} or N_{e}, respectively. The supervised classifier is trained to predict the class y of the input speech $\mathbf{x}$. We decide to adopt a simple classification front-end because we mostly rely on the discriminative capacity of the rich proposed feature set. Moreover, it is worth noting that any supervised classifier algorithm can be used at this stage, as our pipeline is classifier-independent.

3.3 Conclusive Remarks

In this chapter we have formalized the problem that this thesis attempts to solving and presented the pipeline of the method proposed to tackle it. In particular, we have explained the various approaches tested to extract high-level semantic features and then the classifier we use to label the input audio signals as real or deepfake.

4
Experimental Setup

This chapter describes the environment of the experimental setup we used to validate our method. We provide details about the datasets involved, the training and feature extraction procedures and the evaluation metrics. Then, we describe the baselines against which we compare the method and the experiments we performed to evaluate it.

4.1 Datasets Description

In this section we introduce the datasets involved in the training and testing phases of the presented work, that in total count almost 800000 tracks. We considered multiple datasets containing tracks of both REAL, i.e, authentic, and DF, i.e., synthetic, classes, aiming at testing the generalization properties of the proposed method. In the following we provide further details for each dataset, that will be later useful in interpreting the experimental results.

ASVspoof 2019 [121] is the official dataset of the ASVspoof 2019 challenge, in which participants competed to implement the best antispoofing system for ASV, as mentioned in Section 2.4. It is a speech audio dataset with both real and synthetic tracks belonging to the same speakers. It contains two partitions, one for PA and one for LA, further divided into *train*, *dev* and *eval*. The former is designed against attacks that physically reproduce the counterfeit voice of the victim through an audio device, while the latter against those that directly inject the spoofed signal into an ASV system. We considered the LA partition since it includes spoofing attacks generated through TTS, VC and TTS/VC hybrid techniques.

Each partition comprises authentic signals along with speech samples generated with 19 different synthesis algorithms. The *train* and *dev* partitions have been created using the same set of synthesis algorithms (named $A01$, $A02$, ..., $A06$), while the *eval* partition includes samples generated with different techniques ($A07$, ..., $A19$). We use *train* and *dev* partitions for training and fine-tuning the proposed method, while *eval* partition is used in test.

ASVspoof 2021 [122] is the dataset from the 2021 version of the ASVspoof challenge. While *train* and *dev* are the same as in the 2019 version, *eval* comprises a new partition, so called DF, specifically designed for the problem we are addressing. This has been built by processing with different lossy codecs the data from ASVspoof 2019 LA *eval* set and additional sources. However, unlike the 2019 version, the labels are not yet fully released at the time of writing so the synthesis and compression algorithms are unknown.

LibriSpeech [138] is an open-source dataset containing about 1000 hours of authentic speech from different speakers. From this corpus we considered the subset *train-clean-100*. We decided to include audio tracks from this dataset in the training set to increase the number of authentic tracks, helping the model to generalize better.

LJSpeech (LJS) [139] is a dataset containing short audio tracks of REAL speech recorded from a single speaker reciting pieces from non-fiction books. This dataset is part of the testing set.

Cloud 2019 [140] is a collection of TTS generated audio signals proposed in [140]. It includes tracks from different state-of-the-art speech generators available as cloud services: Amazon AWS Polly (PO), Google Cloud Standard (GS), Google Cloud WaveNet (GW), Microsoft Azure (AZ) and IBM Watson (WA). We include this dataset in the test set as DF signals. These algorithms are different from those considered in the training set, and thus provide additional in-the-wild synthetic data to the test set.

Interactive Emotional Dyadic Motion Capture (IEMOCAP) (IEM) [141] is a dataset originally designed for the SER task. The data were recorded during scripted and improvised conversations by 10 actors. It contains video and audio signals annotated with information about the speakers' facial expressions and hand movements. We include this dataset in the test set as authentic signals.

VoxCeleb is an audio-video dataset containing short clips extracted from celebrity interviews on YouTube. It counts more than 7000 different speakers, embracing a wide range of ethnicities, ages and accents, for a total of more than one million utterances. The dataset consists of two versions, VoxCeleb1 [142] and VoxCeleb2 [143], each with its own train/test split. The first contains about 100000 audio tracks, while the second contains about 1000000.

Table 4.1: Composition of the training, development and testing sets for the front-end binary classifier.

Dataset	N. Tracks	REAL	DF	Train	Dev	Test
ASVspoof 2019	121 461	✓	✓	✓	✓	✓
ASVspoof 2021	611 829	✓	✓			✓
LibriSpeech	28 539	✓		✓		
LJSpeech	13 100	✓				✓
Cloud2019	11 888		✓			✓
IEMOCAP	10 039	✓				✓
Total	796 856	86 778	710 078	53 919	24 844	718 093

Blizzard 2013 [144] is a dataset including the speech of a professional narrator reading the text of a collection of classic novels in an expressive and emotional storytelling style.

In Table 4.1 we report the train and testing split used for the front-end binary classifier and the type of speech signal, REAL or DF, included in each dataset.

4.2 Evaluation Metrics

To evaluate the performance of our method and analyze its behavior, we relied on well-known metrics. The first is *balanced accuracy*, often used in binary classification problems when dealing with unbalanced datasets, i.e., when the classes do not share the same number of samples. In order to define it we must first introduce two other metrics from which it is calculated: True Positive Rate (TPR), also known as sensitivity, and True Negative Rate (TNR), also known as specificity. We refer with positive and negative to the two classes for the problem at hand, which in our case are respectively REAL and DF. TPR is the ratio between the True Positive (TP), which is the number of correctly classified positive samples, and the total amount of positive samples

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{4.1}$$

where False Negative (FN) is the number of positive samples incorrectly classified as negatives. On the other hand, TNR is the ratio between the True Negative (TN), which is the number of correctly classified negative samples, and the total amount of negative samples

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}, \tag{4.2}$$

where False Positive (FP) is the number of negative samples incorrectly classified as positives. Finally we can define the balanced accuracy as

$$\text{Bal. Acc.} = \frac{\mathrm{TPR} + \mathrm{TNR}}{2} \tag{4.3}$$

The closer to 1 the balanced accuracy, the higher the performance of the model. Another metric we adopt is the *confusion matrix* which allows the visualization of the classification performances. It either displays TP, FP, TN, FN values or their normalized percentage. It is defined as

$$\begin{bmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{bmatrix} \tag{4.4}$$

Another widely used metric we employ is the *Receiver Operating Characteristic (ROC)* curve with the associated *Area Under the Curve (AUC)*. The ROC curve is a graph showing the performance of a classification model at all classification thresholds. It can be graphically represented by plotting at different classification thresholds the TPR against the False Positive Rate (FPR) defined as

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{4.5}$$

AUC is defined as the two-dimensional area underneath the entire ROC curve. The AUC represents the separability ability of the model, e.g, how much the model is able to distinguish between the classes. The closer to 100 the AUC, the better the model is at predicting TP and TN. The last metric we employ is *Equal Error Rate (EER)*, typically used to evaluate performance of biometric security systems. It individuates the point on a ROC curve at which the FPR is equal to the False Negative Rate (FNR), defined as

$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} \tag{4.6}$$

Therefore, the number of correctly and incorrectly classified samples is equal between the two classes at this operating threshold. The closer to 0 the EER value, the the higher the accuracy of the biometric system. Figure 4.1 shows an example of a ROC curve with a visual understanding of what AUC and EER represent.

4.3 Features Extraction & Training Details

Our system involves two different types of feature extraction methods, namely windowed and voiced features and three independent training stages, namely for ECAPA-TDNN network, for the prosody encoder and for the final binary classifier.

To extract the windowed features we use M_{w} triangular windows of length $W_{\text{w}} = 0.5$ s, with a hop size $H_{\text{w}} = 0.1$ s. After the concatenation between their statistics vector and the global features extracted at utterance-level we obtain a final feature vector $\mathbf{f}_{\text{w}}$ of length $N_{\text{w}} = 163$. On the other hand, we extract voice features on a variable number M_{v} of segments that contain voice traces. In particular we include in this analysis all the segments in which the speech voice sounds detected have

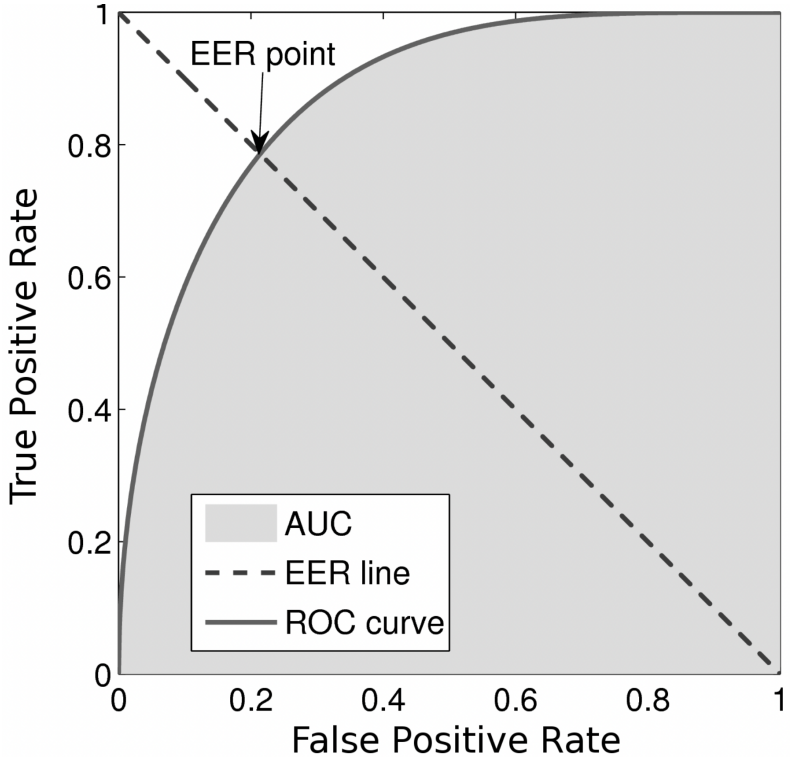
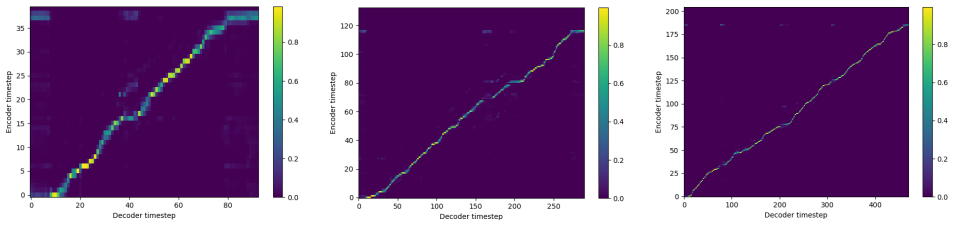


Figure 4.1: Example of a ROC curve, with the associated AUC and EER values [18].

a minimum length of $D_{\text{v}} = 60$ ms as suggested in [135]. After computing the statistics over all segments, we obtain a final feature vector $\mathbf{f}_{\text{v}}$ of length $N_{\text{v}} = 216$.

Regarding the speaker embedding extractor, we use a version of ECAPA-TDNN available at [145], which uses Additive Margin Softmax Loss and is trained on VoxCeleb 1 and VoxCeleb 2 datasets. As mentioned in Section 3.2.1, the input waveform $\mathbf{x}$, to be used as input to ECAPA-TDNN network, must be first transformed in its MFCC representation $\mathbf{X}_{\text{MFCC}}$. For this operation we consider $B = 80$ MFCCs extracted with $W_{\text{s}} = 25$ ms windows with hop size $H_{\text{s}} = 10$ ms, leading to a $M \times 80$ representation, where the number of windows M depends on the length of the audio. The final embedding vector $\mathbf{f}_{\text{s}}$ has dimension $N_{\text{s}} = 192$. For the prosody learned features, we train the prosody encoder on Blizzard 2013 dataset, for 200000 epochs, following the training procedure detailed in [17]. For computational issues, we modify only one parameter value, the mini-batch size, that in our training process is equal to 8. Before feeding it to the encoder, the input signal $\mathbf{x}$ is transformed into a Mel Spectrogram $\mathbf{X}_{\text{mel}}$ using window length $W_{\text{p}} = 50$ ms and hop size $H_{\text{p}} = 12.5$ ms. The number of frequency bins used is $K = 80$. This lead to a final input of dimension $M \times 80$. The resulting embedding vector $\mathbf{f}_{\text{p}}$ has length and $N_{\text{p}} = 128$. In Figure 4.2 we show the learned attention alignments at different epochs when jointly training Tacotron and the prosody encoder. The graphs, showing how well the encoded input matches the decoder



(a) 50k epochs (b) 130k epochs (c) 200k epochs

Figure 4.2: Attention alignment graphs extracted at different training epochs of the expressive Tacotron model presented in [17]

output for each timestep, reflect a marked learning throughout the epochs as the line gets neater and well-defined.

The final set of concatenated features is $\mathbf{f}$ as a fusion of $\mathbf{f}_{\mathrm{s}}$ extracted via ECAPA-TDNN and $\mathbf{f}_{\mathrm{p}}$ represented by either prosody encoder output $\mathbf{f}_{\mathrm{e}}$ or by windowed/voiced features, $\mathbf{f}_{\mathrm{w}}$ and $\mathbf{f}_{\mathrm{v}}$, respectively. This vector is then standardized by removing the mean and scaling to unit variance, as described in Section 1.1.4 and acts as input to the binary classifier. The supervised classification algorithm we adopt is SVM classifier, following the training-development partition detailed in Table 4.1. To find the best set of hyper-parameters we performed a grid search on development partition using balanced accuracy as a metric. We considered the following parameters: $C \in [0.01, 0.1, 1, 10, 100]$, kernel coefficient $\gamma \in [1/N, 1/(N * \sigma_{\mathbf{f}}^2)]$, where N is dimensionality of the feature vector $\mathbf{f}$ and $\sigma_{\mathbf{f}}^2$ is the variance of $\mathbf{f}$ over the training dataset. In addition, we vary the kernel type between radial basis function kernel, polynomial kernel and sigmoid kernel. The best configuration proved to be $C = 100$, $\gamma = 1/(N * \sigma_{\mathbf{f}}^2)$ and using radial basis function kernel.

4.4 Baselines

To test the validity and novelty of our method, we compare its performances with those of three different baselines. They rely on various representations of the input data, allowing us to have an orthogonal approach to the problem. The first one is RawNet2 [146], a state-of-the-art end-to-end neural network that operates on raw waveforms. It has been first proposed for the ASVspoof 2019 challenge and included as a baseline in the ASVspoof 2021 challenge both for LA and DF tasks. We train the network for 100 epochs on the same training set we adopted for the proposed method, using a cross-entropy loss function with Adam optimization [147], a mini-batch size of 32 and a learning rate of 0.0001. The second and third baselines are variants of ResNet [77], a residual CNN that creates shortcuts between layers by skipping connections that help stabilize training. We consider two versions of the ResNet, with an almost identical architecture, but fed with different representations of the input audio track, as presented in [148]. The first one, referenced as Spec-

ResNet, takes as input the log-magnitude representation of the STFT of the considered audio, computed on hamming windows with window size = 2048 and 25% overlap. The second one, called MFCC-ResNet, is fed with the first 24 MFCCs of the input data, together with their first and second derivatives. Input transformation and training strategy for these two networks are implemented following [148]. We train both of them on the same training set we adopted for the proposed method by minimizing a cross-entropy loss function with different weights assigned to genuine and spoofed examples, to mitigate the imbalance in the training data distribution. We use Adam optimizer with learning rate = 0.00005 and train the networks for 200 epochs with a mini-batch size of 32.

4.5 Experiments

This section lists the experiments we conducted to test our method from different perspectives. We set the sampling frequency F_s to 16 kHz for all of them, hence if necessary down-sampling the audio tracks but never upsampling them. We note that we keep the same training set for all considered scenarios, as defined in Table 4.1, and only change the test set depending on the experiment. Likewise, we maintain the same classifier hyperparameters described in Section 4.3.

Embedding Comparison As a first experiment we compare the performance of the different approaches considered for prosodic extraction, presented in Section 3.2.2. To this regard we examine three different models, all following the pipeline described in 3.1, but only differing in the feature set passed to the classifier. In particular, in the first one we only consider the vector of windowed features $\mathbf{f}_{\mathrm{w}}$, in the second one only the vector of voiced features $\mathbf{f}_{\mathrm{v}}$ and in the third one only the embeddings $\mathbf{f}_{\mathrm{e}}$, extracted using the prosody encoder. Being a purely prosodic analysis, in all three cases we exclude speaker embedding $\mathbf{f}_{\mathrm{s}}$. To compare the models with each other and find the most accurate prosodic representation for the task of audio DF detection we test them on ASVspoof 2019 LA *eval* set. The purpose of this test is to choose the best performing one and include it in the system pipeline as $\mathbf{f}_{\mathrm{p}}$ to continue with subsequent experiments, while definitively discarding the others. Then, once the method is selected, we compare the two representation, $\mathbf{f}_{\mathrm{s}}$ and $\mathbf{f}_{\mathrm{p}}$. In particular, we inspect how much the two embeddings differ from each other to avoid computing redundant information.

Baseline Comparison In this experiment, we compare the results obtained using the proposed method with those of the three considered baselines on the LA *eval* partition of the ASVspoof 2019. The purpose is to validate the novelty and effectiveness of our method with respect to the state-of-the-art of the specific task we are addressing.

Ablation Study In this third experiment we further analyze the char-

acteristics and the importance of each embedding subset, namely the speaker embeddings $\mathbf{f}_p$ and the prosody embeddings $\mathbf{f}_s$, used in our method. Through an ablation study we test how they perform individually in different scenarios. In this analysis, we consider three distinct models, all based on the proposed architecture, differing only for the embeddings subset that the final SVM classifier receives as input. The first model, that we indicate with *Prosody Emb*, is fully-prosodic and it is based on $\mathbf{f}_p$ only. The second only considers the speaker information of $\mathbf{f}_s$ and it is indicated with *Speaker Emb*. The third model is the complete one, i.e., *ProsoSpeaker*, and it performs classification using the concatenation of $\mathbf{f}_p$ and $\mathbf{f}_s$. We then considered three test scenarios, depending on the synthesis techniques used to generate the synthetic speech signals of the test set. In the first scenario (a) we consider only speech tracks created with TTS techniques; in the second scenario (b) only speech tracks created with VC techniques; in the third scenario (c) both synthesis techniques are considered. All the tracks for the three scenarios are selected from ASVspoof 2019 dataset. Please notice that we consider the hybrid TTS/VC techniques into category (a) because their starting point is a fully-synthetic, non-human voice.

Generalization Capability This fourth experiment aims to test the generalization capabilities of the model. Therefore, we first verify the performances of the proposed detector singularly on each algorithm present in ASVspoof 2019 *eval* to check the classification performances consistency over different synthesis strategies. Then, we consider a more complex open-set scenario made of multiple datasets, unseen during training and external to the ASVspoof challenge corpora. Therefore, ProsoSpeaker detector is tested on ASVspoof 2019 LA *eval*, LJSpeech, IEMOCAP and Cloud2019.

Robustness Analysis In this fifth and final test we aim to verify the robustness of the proposed method to common signal manipulation. In fact, in a real-world scenario, we can perform many operations to hide the artifacts introduced by deepfake generation algorithms, like for instance lossy compression. Some signal information is lost by compressing an audio track, including traces that may help deepfake detectors determine the signal's authenticity. Since our method does not rely on low-level signal characteristics but analyzes semantic features, we hypothesize that compression should not affect its performance significantly. In practice, speaker and prosody embeddings should only partially impact this type of data augmentation and keep their discriminative potential. To test such aspect, we create three versions of the ASVspoof 2019 LA *eval* dataset using MP3 compression at different bitrates, namely 128 kBits/s, 64 kBits/s and 32 kBits/s, using SoX tool [149]. By testing our model on these compressed datasets, we monitor performance, checking for any signs of degradation. Finally, to further validate this aspect, we test the proposed method on the DF partition of ASVspoof 2021, which

comprises more than 600000 tracks from both REAL and DF classes, compressed with different codecs to simulate VoIP transmission. In this case, compared to the previous, we are not aware of which audio manipulation technique or parameters have been applied to the analyzed tracks. This represents the most complex real-world scenario in which our method must deal with a large amount of new data generated and post-processed with unseen techniques.

4.6 Conclusive Remarks

In this chapter we have described the environment and datasets on which we perform our experiments, along with the metrics used to evaluate the results and the training details. Afterward, we have introduced the baselines against which we compare our method and outlined the experiments conducted to analyze different aspects of the model. In the next chapter, we will report the results of these experimental setups.

5 Results

In this chapter we present the results of the tests outlined in Section 4.5. These experiments aim to analyze the proposed *ProsoSpeaker* method from different perspectives. Each test allows to evaluate a specific quality of the system and better understand its functioning concerning the audio DF detection task. In the first experiment, we compare the embeddings extraction methods. In the second, we test the performance of the *ProsoSpeaker* model in relation to that of the chosen baselines to prove the effectiveness and novelty of our work. Through the third experiment, we analyze the proposed method's behavior more in-depth to understand its strengths and functioning. The fourth one aims at verifying the generalization capabilities when confronted with unknown datasets. Finally, in the fifth and last experiment we introduce distortions in the test set and measure the robustness of the *ProsoSpeaker* method to this aspect.

5.1 Embeddings Comparison

Here we compare the classification performance of our model when we feed the classifier with different embedding sets, to understand which prosodic representation is more suitable for the task at hand. The considered embeddings are $\mathbf{f}_{\mathrm{w}}$, $\mathbf{f}_{\mathrm{v}}$ and $\mathbf{f}_{\mathrm{e}}$ and Figure 5.1 shows the results of the investigation, in terms of ROC curves. These are named as *windowed features*, *voiced features* and *prosody encoder*, respectively. There is an evident difference in performance between the handcrafted features $\mathbf{f}_{\mathrm{w}}/\mathbf{f}_{\mathrm{v}}$ and the learned ones $\mathbf{f}_{\mathrm{e}}$. The EER, AUC and balanced accuracy values shown in Table 5.1 confirm this trend. The prosody encoder extraction method widely outperforms the other two, with an improvement of $\approx 23\%$

Table 5.1: EER, AUC and balanced accuracy values for the three $\mathbf{f}_\mathrm{p}$ extraction methods on ASVspoof 2019 LA *eval* set.

$\mathbf{f}_\mathrm{p}$ **extraction**	**EER %**	**AUC**	**Bal. Acc. %**
Prosody Encoder	**15.13**	**91.99**	**85.05**
Windowed Features	37.49	67.54	53.94
Voiced Features	38.92	65.18	53.08

in EER, more than 26 in AUC and $\approx 31\%$ in balanced accuracy. The two handcrafted features extraction methods have the same performance, very almost identical ROC curves and similar metrics values. However, we note a slight improvement in the case of windowed features. These results confirm the power of DL, which, as highlighted earlier in Section 1.3, fully automates the crucial step of feature engineering. In fact, it allows learning from raw data the most effective representation, which does not necessarily make sense to humans but is meaningful and optimized from the machine's point of view. Thus, for complex tasks such as prosody modeling, DL approaches may derive learned features ($\mathbf{f}_\mathrm{e}$) that are much more effective than handcrafted ones ($\mathbf{f}_\mathrm{w}$, $\mathbf{f}_\mathrm{v}$). Given these results, from now on we will consider $\mathbf{f}_\mathrm{p} = \mathbf{f}_\mathrm{e}$ extracted by the prosody encoder, since, in addition to having largely exceeded the other methods, it already achieves good performance for the task we face.

As a second analysis, we compare the newly selected prosody embedding $\mathbf{f}_\mathrm{p}$ with the speaker embedding $\mathbf{f}_\mathrm{s}$ to see how much they differ from each other and avoid the computation of redundant information. To do so, we measure the sample Pearson correlation coefficient $r_{f_i f_j}$ for each pair of elements (f_i, f_j) of the vector $\mathbf{f} = [f_0, f_1, ..., f_{N-1}]$ over the test dataset, as explained in Section 1.1.4. The resulting matrix $\mathbf{R_{ff}}$ describes both cross-correlation between prosody and speaker embeddings $\mathbf{R}_{\mathbf{f}_\mathrm{s}\mathbf{f}_\mathrm{p}} = \mathbf{R}^T_{\mathbf{f}_\mathrm{p}\mathbf{f}_\mathrm{s}}$ both auto-correlations of each embedding vector $\mathbf{R}_{\mathbf{f}_\mathrm{p}\mathbf{f}_\mathrm{p}}$ and $\mathbf{R}_{\mathbf{f}_\mathrm{s}\mathbf{f}_\mathrm{s}}$. Figure 5.2 shows the results of this analysis computed in the ASVspoof 2019 *eval* partition in the form of a correlation matrix. The diagonal has been set to 0 for visualization purposes. There, we can identify two rectangular regions, one at the top left, corresponding to $\mathbf{R}_{\mathbf{f}_\mathrm{s}\mathbf{f}_\mathrm{s}}$, and one at the bottom right, corresponding to $\mathbf{R}_{\mathbf{f}_\mathrm{p}\mathbf{f}_\mathrm{p}}$. Although the elements of $\mathbf{f}_\mathrm{p}$ have a higher degree of internal correlation than those of $\mathbf{f}_\mathrm{s}$, with mean value $\mu(\mathbf{R}_{\mathbf{f}_\mathrm{p}\mathbf{f}_\mathrm{p}}) = 0.21$ and standard deviation $\sigma(\mathbf{R}_{\mathbf{f}_\mathrm{p}\mathbf{f}_\mathrm{p}}) = 0.08$, respectively, the cross coefficients present low values, with an average value of $\mu(\mathbf{R}_{\mathbf{f}_\mathrm{s}\mathbf{f}_\mathrm{p}}) = 0.07$. This means that the two embedding vectors do not strongly correlate with each other and do not share much information. The spectro-temporal and prosodic characteristics we are considering have turned out to be orthogonal to each other, benefiting our detector.

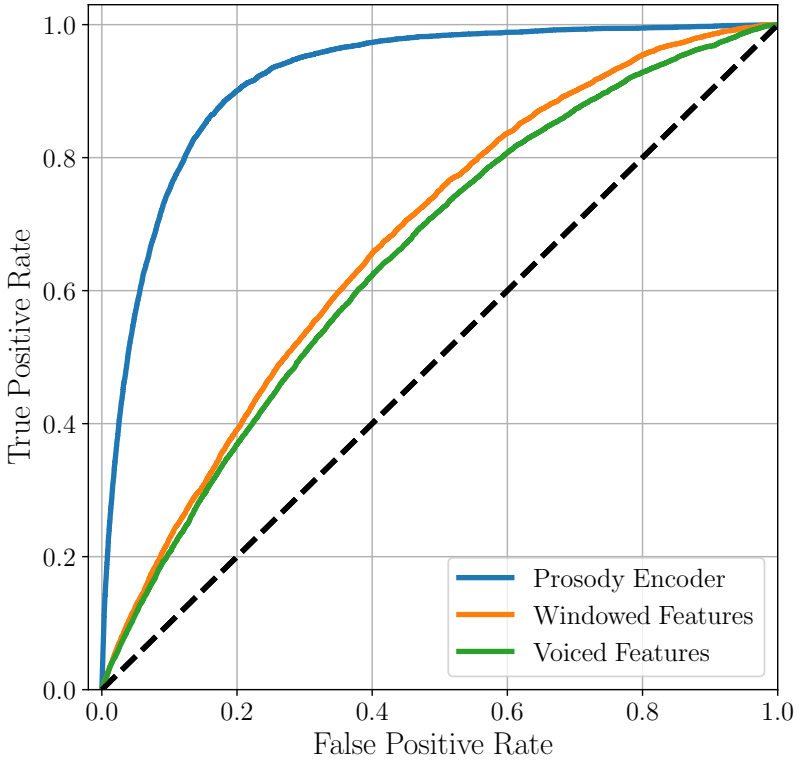


Figure 5.1: Comparison of ROC curves extracted from the three $\mathbf{f}_\mathrm{p}$ extraction methods on ASVspoof 2019 LA *eval* set.

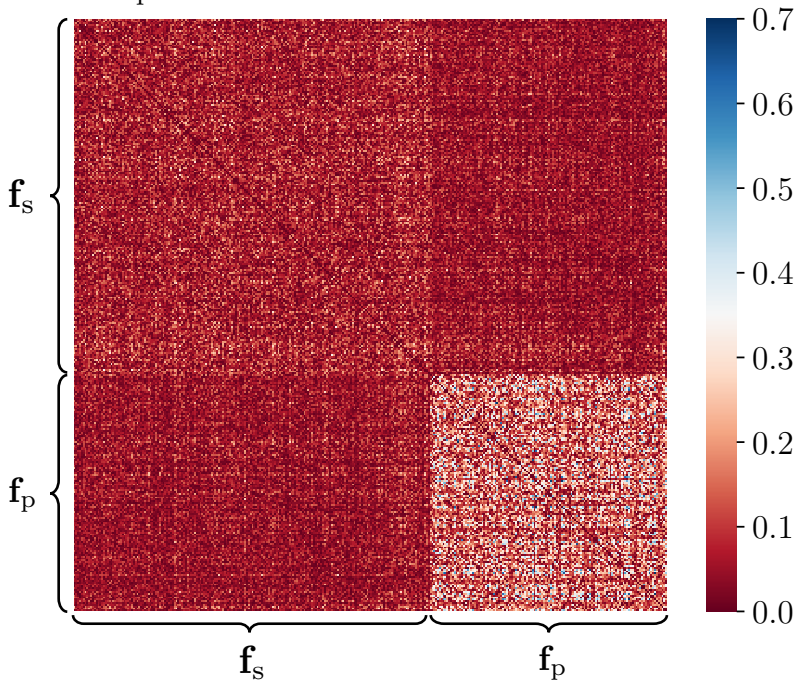


Figure 5.2: Cross-correlation matrix $\mathbf{R}_{\mathbf{ff}}$ of feature vectors $\mathbf{f}$ realizations of ASVspoof 2019 *eval* set.

5.2 Baseline Comparison

In this section we test the classification performance of our method and compare the results with those of the considered baselines on the same test

set of the previous experiment, the LA *eval* partition of the ASVspoof 2019 dataset. First, we test how well the method can separate the REAL and DF classes. Figure 5.3 shows the results on a histogram, by representing on the y-axis the number of audio signals, expressed as a percentage of the total, and on the x-axis the scores that *ProsoSpeaker* assigns to them. We choose REAL as the positive class, associated to a score of 1 and DF as the negative class, with a score of 0. In this sense, the closer an output score $s = ProsoSpeaker(\mathbf{x})$ is to either of these values, the greater the confidence with which the model assigns to the input signal $\mathbf{x}$ the corresponding class label. It is evident from the histogram that the method can separate the two classes effectively. It assigns to most DF audio signals a score lower than 0.2 and to most REAL ones a score higher than 0.8. Next, we calculate the metrics presented in Section 4.4 to test the performance of our method and compare it to the baselines, RawNet2, MFCC-ResNet and Spec-ResNet. Figure 5.4 shows the ROC curves of the three detectors and Table 5.2 shows the corresponding AUC, EER and balanced accuracy values. *ProsoSpeaker* detector outperforms all the three baselines in the considered metrics. In particular, the most remarkable improvement is seen over Spec-ResNet with a difference of about 15% for EER and balanced accuracy, while 10 for AUC. A significant gain is also shown concerning RawNet2, which is the best-performing baseline among those proposed for the ASVspoof 2021 challenge. Here, our method improves by almost 3% over EER and balanced accuracy, while 2 over AUC. This shows that our method can compete with and improve upon the proposed state-of-the-art for the audio DF detection task.

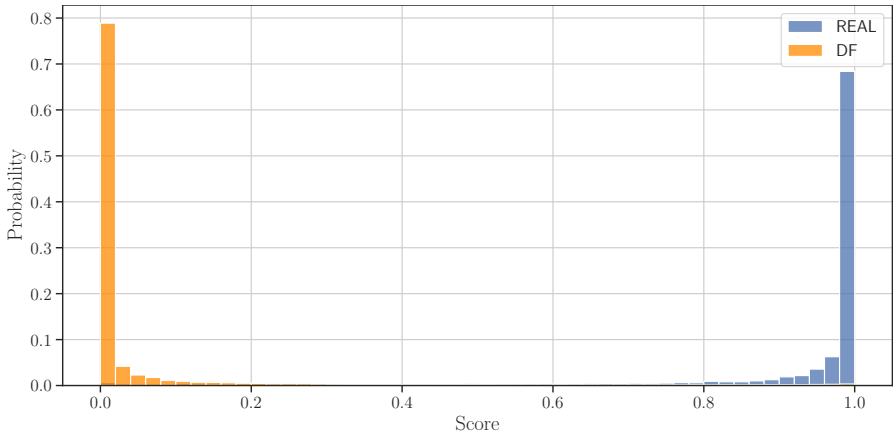


Figure 5.3: Histogram showing the distribution of the output scores of *ProsoSpeaker* computed on ASVspoof 2019 *eval* set.

5.3 Generalization Capability

In this third set of experiments, we aim to analyze the consistency and generalization ability of the proposed method by augmenting the consid-

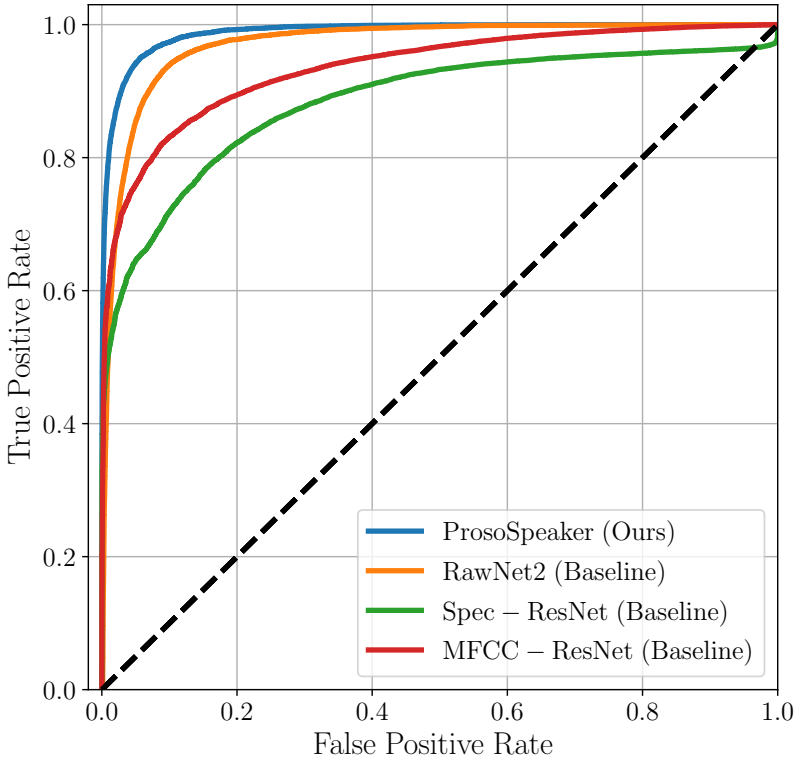


Figure 5.4: ROC curves for the proposed method and the considered baselines, evaluated on ASVspoof 2019 LA *eval* set.

Table 5.2: EER, AUC and balanced accuracy values for the proposed *ProsoSpeaker* method and the considered baselines, evaluated on ASVspoof 2019 LA *eval* set.

Model	**EER %**	**AUC**	**Bal. Acc. %**
RawNet2 (Baseline)	8.15	97.09	91.66
MFCC-ResNet (Baseline)	13.98	93.52	84.96
Spec-ResNet (Baseline)	18.75	88.31	79.50
ProsoSpeaker (Ours)	**5.39**	**98.85**	**94.43**

ered test set. First, we verify the performances of the proposed detector singularly on each algorithm present in ASVspoof 2019 *eval* to check the classification performances consistency over different synthesis strategies. Then, we want to assess *ProsoSpeaker*'s generalization capabilities across multiple datasets, unseen during training and external to the ASVspoof challenge corpora. Figure 5.5 shows the percentage of correct attribution values obtained for each synthesis algorithm included in ASVspoof 2019 *eval* set (A07, A08, ..., A13) and for LJSpeech, IEMOCAP and Cloud2019 (divided in PO, AZ, GS, GW, WA). The label AU corresponds to real speech samples distributed in ASVspoof 2019. The proposed method is successful in almost all the considered cases, with a percentage of correct attribution value always higher than 0.80. This means that *ProsoSpeaker* has good generalization capabilities, and we can consider it a reliable method. The only exception is represented by the TTS generator IBM

Watson, included in Cloud2019, where the accuracy is equal to 0.50. We believe this issue is due to the fact that the IBM TTS method is specifically trained considering a “prosodic-phonology" approach for generating expressive speech [150], hence deceiving our detection method.

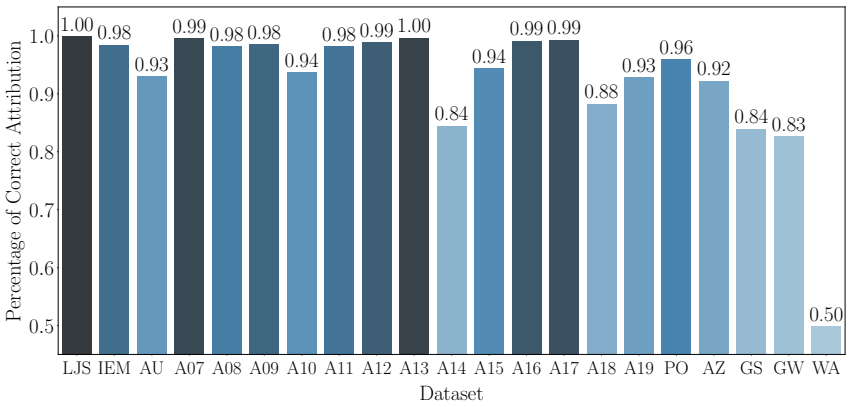


Figure 5.5: Bar plot of the percentage of correct attribution values of the proposed model on each partition of each considered dataset.

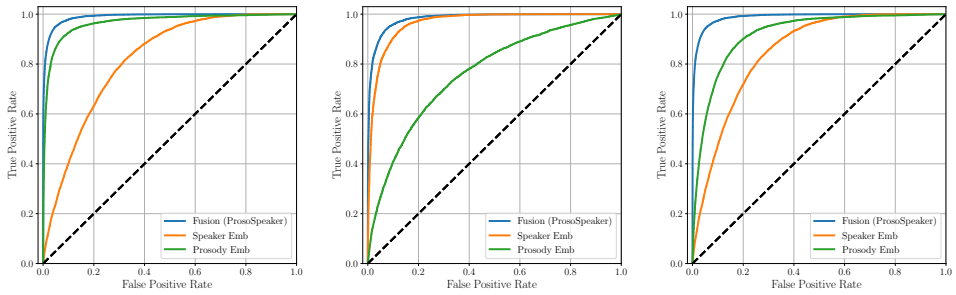
5.4 Ablation Study

In this analysis, we consider the three distinct models presented in Section 4.5, based on the proposed architecture, but differing for the embeddings subset fed to the final classifier. *Prosody Emb*, is the fully-prosodic one, *Speaker Emb* only considers the speaker information and the proposed method, *ProsoSpeaker*, is the fusion of the two. Table 5.3 and Figure 5.6 show the binary classification performances of this analysis, the first in terms of EER, AUC and balance accuracy obtained for the three models in the three considered test scenarios, (a) VC, (b) TTS and (c) ALL, the second showing the corresponding ROC curves. The predictions of the two partial models are orthogonal to each other and each performs better on a distinct scenario. In particular, prosodic embeddings $\mathbf{f}_p$ can discriminate speech signals generated with TTS algorithms well but are less effective with VC methods, while speaker embeddings $\mathbf{f}_s$ achieves better results in the VC case than TTS. This is further confirmed by the confusion matrices shown in Figure 5.7. They contain the percentages of correctly and incorrectly classified signals for all the models in the three scenarios, computed with a 0.5 threshold on the output scores. Note how the bottom row, concerning the REAL audio signals, remains unchanged for each type of embedding used for each matrix. Only the DF partition changes between different scenarios, while the REAL one stays the same. Clearly, the worst cases are when only $\mathbf{f}_s$ is used to classify TTS-generated audio (top-left) and only $\mathbf{f}_p$ to classify VC-generated audio (center). On the other hand, the complementary cases (top-center and

Table 5.3: EER, AUC and balanced accuracy values for the three models (*ProsoSpeaker*, *Speaker Emb*, *Prosody Emb*) tested on the three scenarios (TTS, VC, ALL).

	(a) TTS		
	EER %	**AUC**	**Bal. Acc. %**
ProsoSpeaker	**4.93**	**99.02**	**94.77**
Prosody Emb	8.58	96.68	90.64
Speaker Emb	26.21	81.64	74.62
	(b) VC		
	EER %	**AUC**	**Bal. Acc. %**
ProsoSpeaker	**6.70**	**98.29**	**93.28**
Prosody Emb	30.04	76.35	66.44
Speaker Emb	9.82	96.53	88.20
	(c) ALL		
	EER %	**AUC**	**Bal. Acc. %**
ProsoSpeaker	**5.39**	**98.85**	**94.43**
Prosody Emb	15.13	91.99	85.05
Speaker Emb	22.88	85.08	77.75

center-left) show very good performances, confirming the effectiveness of $\mathbf{f}_s$ in the VC scenario and $\mathbf{f}_p$ in the TTS scenario. Consequently, for both cases when all the algorithms are considered the performances worsen. Confirming what was demonstrated earlier with ROC curves and metrics, the last row of matrices, related to the use of $\mathbf{f}$, shows better results in all three scenarios, especially the more general one that includes all synthesis techniques. From these results we can confirm our initial hypothesis, i.e., that each one of the two speech generation techniques fails in reproducing one of the semantic features encoded by $\mathbf{f}_s$ or $\mathbf{f}_p$. On one side, TTS systems struggle at recreating natural sounding prosody, starting from a pure textual input, and hence prosody embeddings are effectively discriminating them from real speech. On the other hand, VC techniques manipulate an authentic speech sample to impersonate a target speaker, introducing artifacts in the timbre qualities that can be detected leveraging speaker embeddings. Nonetheless, the fusion of the two embeddings improves the predictions in all the considered scenarios, reaching an AUC = 0.99 in the case of the complete dataset. We can conclude that the concatenation of the two embeddings provides a more comprehensive and significant representation of the input speech signal, leading to higher binary classification performances.



(a) TTS (b) VC (c) ALL

Figure 5.6: ROC curves obtained for the three models using different embeddings (*ProsoSpeaker*, *Speaker Emb*, *Prosody Emb*) and tested on the three scenarios (TTS, VC, ALL).

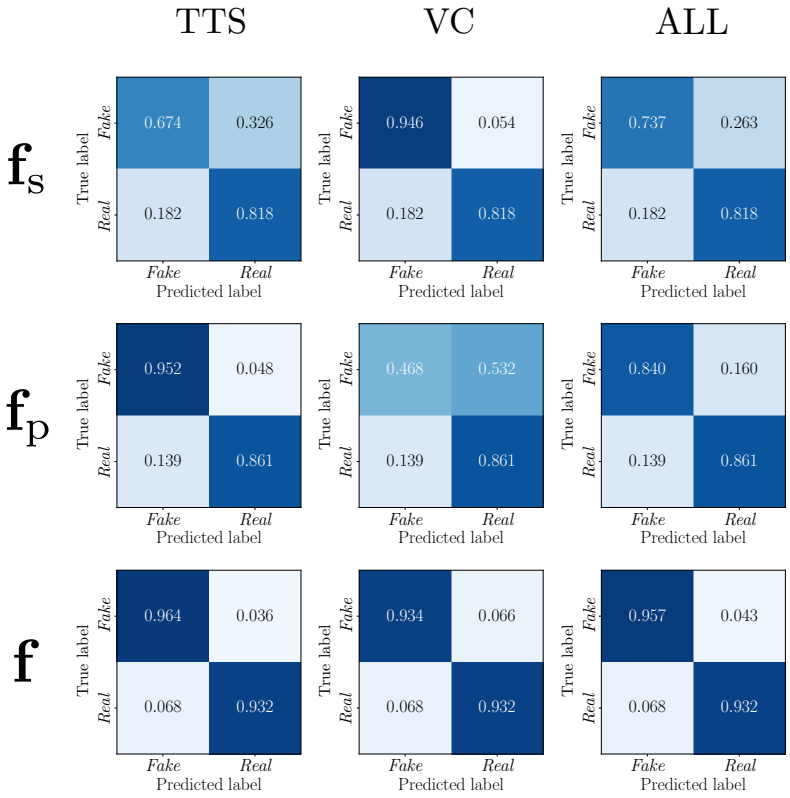


Figure 5.7: Confusion matrices obtained for the three models using different embeddings (ProsoSpeaker, Speaker Emb, Prosody Emb) and tested on the three scenarios (TTS, VC, ALL).

5.5 Compression Robustness

In these final tests we verify the robustness of *ProsoSpeaker* to different types of compressions. This is definitely the most real-world scenario, as compressions are commonly applied to DFs to make them even more difficult to detect. As a starting point, we test the model on the three versions of ASVspoof 2019 LA *eval* sets compressed at different bitrates described in Section 4.5. Figure 5.9 shows the correspondent AUC, EER and balanced accuracy values for different compression bitrates. The

detector's performance deteriorates as we increase the compression factor, observing AUC and EER values dropping by 2 and 4%, respectively, between the two extreme cases. Balanced accuracy decreases significantly when compression is firstly introduced, with a drop of 4% between the no-compression and 128 kBits/s cases. At the same time, it maintains stable values when the bitrate decreases, falling only by 1% between 128 and 32 kBit/s cases. We can conclude that, overall, the proposed system, thanks to its high-level semantic approach, is able to maintain its effectiveness even in presence of heavy signal compression.

Finally, we report the results computed over the DF partition of ASVspoof 2021 in which the synthesis and post-processing manipulation techniques are unknown. For this reason, together with the very high number of signals in this unseen dataset, this final scenario represents the most complex one. As for the 2019 case of Section 5.2, we test the abilities of the method in separating the two classes, REAL and DF. The histogram in Figure 5.8 shows the results. Overall the classes are well separated, however a considerable amount of REAL audio falls closely to the DF side, while the opposite is true to a lesser extent. This result is less successful than the ASVspoof 2019 case, representing a much more controlled and less complex case. Regarding the ROC curves and metrics, the *ProsoSpeaker* method's results are reported respectively in Figure 5.10 and Table 5.4, together with those obtained with the three considered baselines, namely RawNet2 and two versions of ResNet. Our method performs significantly better than all the others, with a difference of $\approx 7\%$ on both EER and AUC compared to RawNet2, which is the best performing baseline. The performance improvement on the baselines is even more significant than that obtained on ASVspoof 2019, proving great robustness of our method in a realistic and challenging scenario.

Therefore, we can conclude that our method outperforms the main state-of-the-art methods for all the considered scenarios and is also more robust at compression. The best performing baseline RawNet2 between the tests performed on ASVspoof 2019 and 2021 loses 16% in EER, 13.71 in AUC and 15.02% in balanced accuracy. On the other hand, ProsoSpeaker only loses 11.77% in EER, 7.26 in AUC and 14.27% in balanced accuracy.

Table 5.4: EER and AUC values for the proposed method and the considered baselines, evaluated on ASVspoof 2021 DF *eval* set.

Model	**EER %**	**AUC**	**Bal. Acc. %**
RawNet2 (Baseline)	24.15	84.19	76.64
Spec-ResNet (Baseline)	34.67	70.40	66.49
MFCC-ResNet (Baseline)	31.33	73.41	67.72
ProsoSpeaker (Ours)	**17.16**	**91.59**	**80.16**

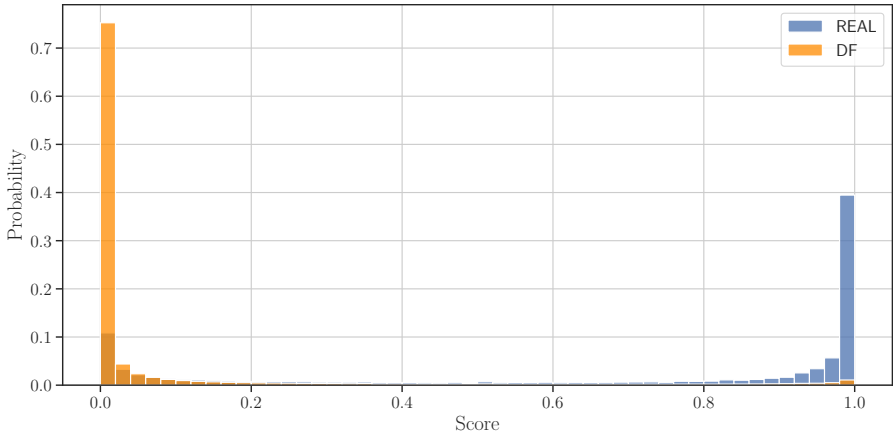


Figure 5.8: Histogram showing the distribution of the output scores of *ProsoSpeaker* computed on ASVspoof 2021 *eval* set.

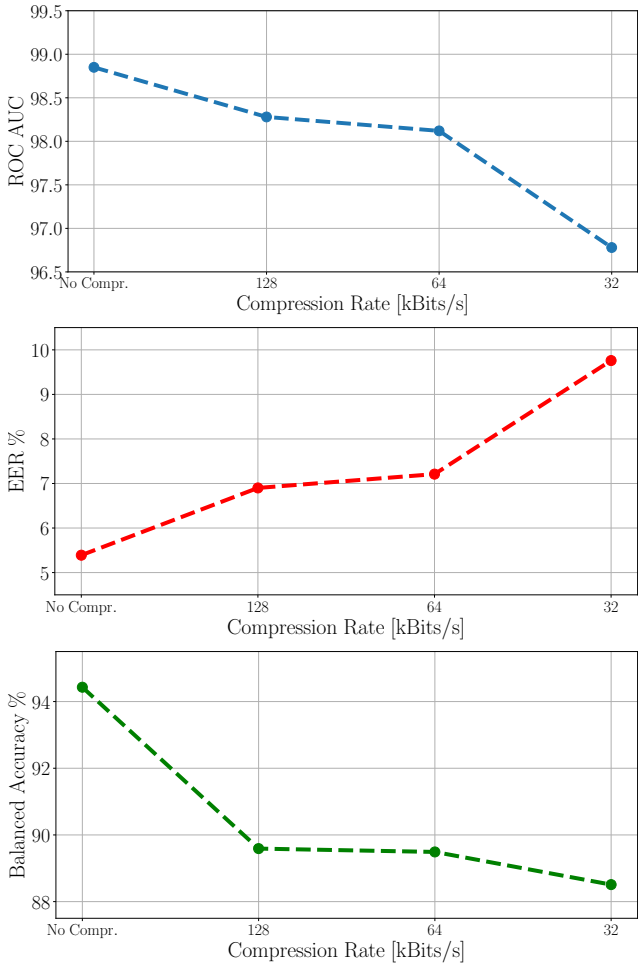


Figure 5.9: ROC AUC, EER and Bal. Acc. values computed on compressed versions of ASVspoof 2019 LA *eval* at different bitrates.

5.6 Conclusive Remarks

In this chapter, we have evaluated the proposed methodology through five experiments to analyze different aspects and confront them with

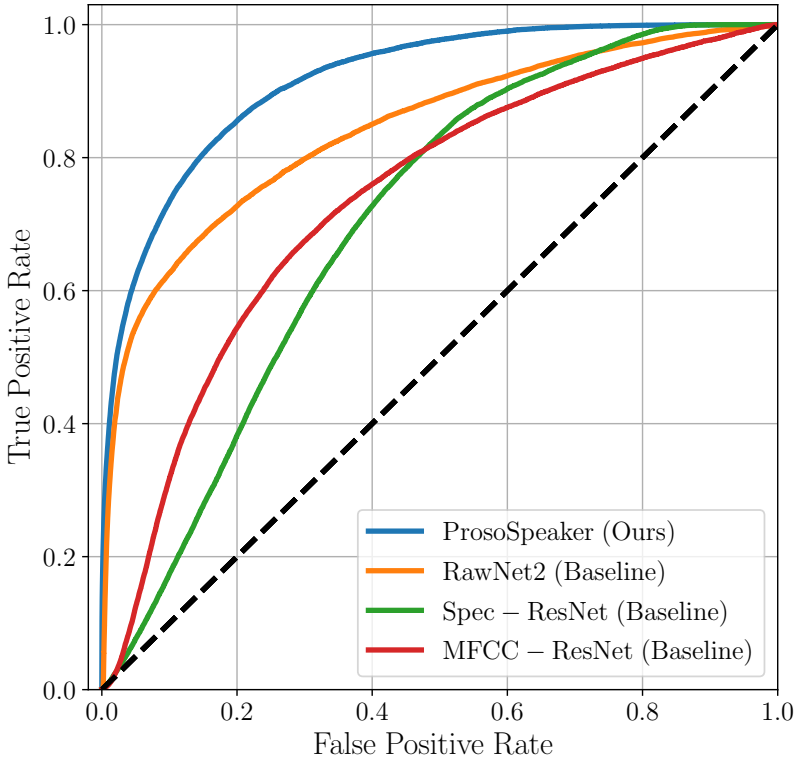


Figure 5.10: ROC curves of the proposed method and the considered baselines computed on ASVspoof 2021 DF *eval* set.

state-of-the-art methods. We have first compared the proposed prosody extraction methods and have chosen to incorporate the prosody encoder into the whole pipeline definitively. Through correlation coefficients, we have found that prosody and speaker embeddings do not share much information. Regarding the goodness of the model, we have first observed how *ProsoSpeaker* achieves excellent performance in a controlled environment such as ASVspoof 2019, outperforming all the considered baselines. Subsequently, through an ablation study on the same dataset we validated our initial hypothesis, i.e, that the speaker embeddings are effective in detecting VC-generated audio, while the prosody embeddings in that of TTS-generated audio. We have then found their combination beneficial, especially in a mixed scenario where both technologies are considered. Next, we found good generalization capabilities when testing the system on several unseen datasets. As a final qualitative analysis we have focused on robustness by testing the method first on versions of ASVspoof 2019 compressed at different bitrates and then on the DF partition of ASVpoof 2021 dataset, together with the baselines. In the first case, *ProsoSpeaker* showed good stability in all evaluation metrics and in the second it proved to be more robust to compression than the other methods.

In the next and final chapter, we will draw the summaries of our research and propose possible future developments of the model.

6
Conclusions and Future Works

Recent developments in DL have produced a new media content synthesis technology called deepfake, which can potentially have a damaging impact on people's lives. These generate counterfeited audio and video material to discredit people's reputations. This threat has raised the need to implement systems that can automatically detect such forgeries with high accuracy. As a result, the two areas of research dealing with generating and detecting DFs are in constant development, with one constantly trying to prevail over the other.

In this thesis we presented a method for audio DF detection, named *ProsoSpeaker*, capable of detecting spoofed signals generated with the two most common speech generation techniques: VC and TTS. Adopting a semantic approach, we based our system on the concatenation of high-level features, denoted as speaker $\mathbf{f}_\mathrm{s}$ and prosody $\mathbf{f}_\mathrm{p}$ embeddings. We used this representation as input to a fast supervised binary classifier that predicts whether the speech signal is authentic or synthetically generated. $\mathbf{f}_\mathrm{s}$ encodes spectro-temporal aspects of the voice associated with the speaker's identity and is extracted through a state-of-the-art model for ASV, called ECAPA-TDNN. On the other hand, $\mathbf{f}_\mathrm{p}$ encodes prosody, associated with the voice's tone and expression, and is extracted via a prosody encoder, initially included in Tacotron.

To perform a robust evaluation, we have tested *ProsoSpeaker* on multiple datasets containing both DF and REAL audio signals and used several evaluation metrics. In addition, we have performed several experiments to compare our work with state-of-the-art and test it in different scenarios. We inspected its generalization capabilities over unseen datasets, intrinsic functioning and robustness to real-world audio manipulation, as lossy compression. The results validated our initial hypothesis, that speaker

embeddings can detect DFs generated through VC algorithms, while prosody embeddings are more effective with TTS synthesis. Nonetheless, the fusion the two embeddings has proved to be the more effective strategy, achieving higher classification performances, good generalization capabilities and robustness to lossy compression. The obtained results validate the idea of exploiting semantic features to discriminate deepfakes and highlight some of the aspects on which speech generators still fail.

We can still apply many improvements to the proposed methodology. The first one focuses on improving the extraction of speaker and prosody embeddings, obtaining a more discriminative and robust representation of the two semantic features. For example, if we manage to reduce the degree of correlation between prosody embedding features we could produce a more significative representation of the speech signal, beneficing the method's perfomrnace. For example, suppose we manage to reduce the degree of correlation between the prosody and speaker embeddings. In that case, we could produce a more meaningful representation of the speech signal to benefit the method's performance. Additional experimental setups can be designed to individually validate the classification capabilities of the two embeddings and understand how to improve the performance of each. The final result will surely be even better once the two representations are optimized and totally uncorrelated. Another possible improvement to the model might be considering a third embedding that encodes an additional semantic aspect of the voice. For example, we could include a representation of the emotionality of the voice, as proposed in many works discussed in Section 2.4. Alternatively, we could also consider a vector of low-level features that can detect the traces left by the generation methods, as they may show signs of forgery that semantic features do not capture. This can take inspiration from the numerous artifacts-based approaches proposed in the state-of-the-art for audio DF detection presented in Section 2.4. In this way, the analysis would focus on two different levels at the same time, combining the advantages of the two main DF detection methods based on high-level semantic aspects or low-level artifact detection. Finally, we could consider the channel effects, not related to the quality of the voice, but the recording environment. The subtractive definition of prosody presented in Section 1.1.3 describes it as what remains in audio after accounting for speaker identity and channel effects. It follows that these three components may be able to describe audio in its entirety. Therefore, this the channel effects represent an additional level of analysis to detect DFs. For example, besides being flawed in the generation of realistic voices and their semantic aspects, voice synthesis techniques may recreate a non-natural sound environment that we can leverage to detect them. As a result of these updates we believe that the model may become more robust in recognizing audio DFs and its overall performance may further improve.

Bibliography

[1] "Mel spectrogram example." https://learn.vonage.com/blog/2019/03/20/building-a-machine-learning-model-for-answering-machine-detection-dr.

[2] "Mel spectrogram scheme." https://it.mathworks.com/help/audio/ref/melspectrogram.html.

[3] J.-G. Kim and B. Lee, "Appliance classification by power signal analysis based on multi-feature combination multi-layer lstm," *Energies*, vol. 12, no. 14, p. 2804, 2019.

[4] H. Wu, L. Wang, Z. Zhao, C. Shu, and C. Lu, "Support vector machine based differential pulse-width pair brillouin optical time domain analyzer," *IEEE Photonics Journal*, vol. 10, no. 4, pp. 1–11, 2018.

[5] "Artificial intelligence, machine learning, deep learning." https://master-iesc-angers.com/artificial-intelligence-machine-learning-and-deep-learning-same-context-different-concepts.

[6] C. Shao, "A quantum model for multilayer perceptron," *arXiv preprint arXiv:1808.10561*, 2018.

[7] V. H. Phung, E. J. Rhee, *et al.*, "A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets," *Applied Sciences*, vol. 9, no. 21, p. 4500, 2019.

[8] "Recurrent neural network." https://necst.it/exploring-boundary-accuracy-performances-recurrent-neural-networks.

[9] "Self-attention mechanism." https://peltarion.com/blog/data-science/self-attention-video.

[10] "Autoencoder architecture." https://blog.nikhiljay.com/floating-in-latent-space-fb0747aa667f.

[11] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4779–4783, IEEE, 2018.

[12] Y. Wang, R. Skerry-Ryan, Y. Xiao, D. Stanton, J. Shor, E. Battenberg, R. Clark, and R. A. Saurous, "Uncovering latent style factors for expressive speech synthesis," *arXiv preprint arXiv:1711.00520*, 2017.

[13] H. Malik, "Securing voice-driven interfaces against fake (cloned) audio attacks," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 512–517, IEEE, 2019.

[14] B. Hosler, D. Salvi, A. Murray, F. Antonacci, P. Bestagini, S. Tubaro, and M. C. Stamm, "Do Deepfakes Feel Emotions? A Semantic Approach to Detecting Deepfakes via Emotional Inconsistencies," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[15] B. Desplanques, J. Thienpondt, and K. Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," *arXiv preprint arXiv:2005.07143*, 2020.

[16] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[17] R. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. Weiss, R. Clark, and R. A. Saurous, "Towards end-to-end prosody transfer for expressive speech synthesis with tacotron," in *international conference on machine learning*, pp. 4693–4702, PMLR, 2018.

[18] R. Tronci, G. Giacinto, and F. Roli, "Dynamic score combination: A supervised and unsupervised score combination method," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pp. 163–177, Springer, 2009.

[19] T. Guardian, "Deep Blue computer beats world chess champion." `https://www.theguardian.com/sport/2021/feb/12/deep-blue-computer-beats-kasparov-chess-1996`.

[20] T. N. Y. Times, "Computer Wins on Jeopardy!: Trivial, Its Not." `https://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html`.

[21] T. N. Y. Times, "A Learning Advance in Artificial Intelligence Rivals Human Abilities." https://www.nytimes.com/2015/12/11/science/an-advance-in-artificial-intelligence-rivals-human-vision-abilities.html.

[22] I. Spectrum, "At last, a self-driving car that can explain itself." https://spectrum.ieee.org/at-last-a-self-driving-car-that-can-explain-itself.

[23] Lionbridge, "The Future of Language Technology: The Future of Machine Translation." https://www.lionbridge.com/blog/translation-localization/the-future-of-language-technology-the-future-of-machine-translation.

[24] T. Guardian, "Elon Musks brain chip firm Neuralink lines up clinical trials in humans." https://www.theguardian.com/technology/2022/jan/20/elon-musk-brain-chip-firm-neuralink-lines-up-clinical-trials-in-humans.

[25] T. Guardian, "The rise of the deepfake and the threat to democracy." https://www.theguardian.com/technology/ng-interactive/2019/jun/22/the-rise-of-the-deepfake-and-the-threat-to-democracy.

[26] Forbes, "Deepfakes, revenge porn, and the impact on women." https://www.forbes.com/sites/chenxiwang/2019/11/01/deepfakes-revenge-porn-and-the-impact-on-women/?sh=45b66a961f53.

[27] The New York Times, "Pennsylvania Woman Accused of Using Deepfake Technology to Harass Cheerleaders." https://www.nytimes.com/2021/03/14/us/raffaela-spone-victory-vipers-deepfake.html.

[28] Forbes, "Fraudsters Cloned Company Directors Voice In 35$ Million Bank Heist, Police Find." https://www.forbes.com/sites/thomasbrewster/2021/10/14/huge-bank-fraud-uses-deep-fake-voice-tech-to-steal-millions.

[29] Mimecast, "Why Deepfakes are Revolutionizing the World of Phishing." https://www.mimecast.com/blog/deepfakes-revolutionizing-phishing.

[30] L. Verdoliva, "Media forensics and deepfakes: an overview," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 910–932, 2020.

[31] R. Durall, M. Keuper, F.-J. Pfreundt, and J. Keuper, "Unmasking deepfakes with simple features," *arXiv preprint arXiv:1911.00686*, 2019.

[32] D. Cozzolino, A. Rössler, J. Thies, M. Nießner, and L. Verdoliva, "Id-reveal: Identity-aware deepfake video detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[33] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing ai created fake videos by detecting eye blinking," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2018.

[34] E. Conti, D. Salvi, C. Borrelli, B. Hosler, P. Bestagini, F. Antonacci, A. Sarti, M. C. Stamm, and S. Tubaro, "Deepfake Speech Detection Through Emotion Recognition: a Semantic Approach," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[35] Y. Cheng and H. C. Leung, "Speaker verification using fundamental frequency.," in *ICSLP*, 1998.

[36] L. Mary and B. Yegnanarayana, "Prosodic features for speaker verification," in *Ninth International Conference on Spoken Language Processing*, Citeseer, 2006.

[37] S. S. Stevens and J. Volkmann, "The relation of pitch to frequency: A revised scale," *The American Journal of Psychology*, vol. 53, no. 3, pp. 329–353, 1940.

[38] D. O'Shaughnessy, *Speech Communication: Human and Machine*. Addison-Wesley series in electrical engineering, Addison-Wesley Publishing Company, 1987.

[39] A. V. Oppenheim and R. W. Schafer, "From frequency to quefrency: A history of the cepstrum," *IEEE signal processing Magazine*, vol. 21, no. 5, pp. 95–106, 2004.

[40] F. Cummins, F. Gers, and J. Schmidhuber, "Comparing prosody across many languages," *Instituto Dalle Molle di Studie sullIntelligenza Artificiale, Lugano, Switzerland, Tech. Rep., IDSIA-07-99*, 1999.

[41] L. Mary and B. Yegnanarayana, "Prosodic features for speaker verification," in *Ninth International Conference on Spoken Language Processing*, Citeseer, 2006.

[42] R. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. Weiss, R. Clark, and R. A. Saurous, "Towards end-to-end prosody transfer for expressive speech synthesis with tacotron,"

in *international conference on machine learning*, pp. 4693–4702, PMLR, 2018.

[43] J. Weston, R. Lenain, U. Meepegama, and E. Fristed, "Learning de-identified representations of prosody from raw audio," in *International Conference on Machine Learning*, pp. 11134–11145, PMLR, 2021.

[44] E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke, "Modeling prosodic feature sequences for speaker recognition," *Speech communication*, vol. 46, no. 3-4, pp. 455–472, 2005.

[45] A. Batliner, E. Nöth, J. Buckow, R. Huber, V. Warnke, and H. Niemann, "Duration features in prosodic classification: why normalization comes second, and what they really encode," 2001.

[46] M. M. Ahsan, M. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique, "Effect of data scaling methods on machine learning algorithms and model performance," *Technologies*, vol. 9, no. 3, p. 52, 2021.

[47] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[48] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[49] W. S. Noble, "What is a support vector machine?," *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[50] Q. Wu and D.-X. Zhou, "Analysis of support vector machine classification.," *Journal of Computational Analysis & Applications*, vol. 8, no. 2, 2006.

[51] I. Syarif, A. Prugel-Bennett, and G. Wills, "Svm parameter optimization using grid search and genetic algorithm to improve classification performance," *Telkomnika*, vol. 14, no. 4, p. 1502, 2016.

[52] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[53] H. Ramchoun, M. A. J. Idrissi, Y. Ghanou, and M. Ettaouil, "Multilayer perceptron: Architecture optimization and training.," *Int. J. Interact. Multim. Artif. Intell.*, vol. 4, no. 1, pp. 26–30, 2016.

[54] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[55] B. Thornton, "Audio recognition using mel spectrograms and convolution neural networks," 2019.

[56] J. Zhao, X. Mao, and L. Chen, "Speech emotion recognition using deep 1d & 2d cnn lstm networks," *Biomedical Signal Processing and Control*, vol. 47, pp. 312–323, 2019.

[57] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[58] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[60] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[61] A. Waibel, "Modular construction of time-delay neural networks for speech recognition," *Neural computation*, vol. 1, no. 1, pp. 39–46, 1989.

[62] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth annual conference of the international speech communication association*, 2015.

[63] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[64] M. Farrús, J. Hernando, and P. Ejarque, "Jitter and shimmer measurements for speaker recognition," in *8th Annual Conference of the International Speech Communication Association; 2007 Aug. 27-31; Antwerp (Belgium).[place unknown]: ISCA; 2007. p. 778-81.*, International Speech Communication Association (ISCA), 2007.

[65] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1, pp. 19–41, 2000.

[66] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE transactions on speech and audio processing*, vol. 3, no. 1, pp. 72–83, 1995.

[67] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using gmm supervectors for speaker verification," *IEEE signal processing letters*, vol. 13, no. 5, pp. 308–311, 2006.

[68] N. Dehak, P. Kenny, R. Dehak, O. Glembek, P. Dumouchel, L. Burget, V. Hubeika, and F. Castaldo, "Support vector machines and joint factor analysis for speaker verification," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4237–4240, IEEE, 2009.

[69] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Tenth Annual conference of the international speech communication association*, 2009.

[70] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4052–4056, IEEE, 2014.

[71] N. Chen, Y. Qian, and K. Yu, "Multi-task learning for text-dependent speaker verification," in *Sixteenth annual conference of the international speech communication association*, 2015.

[72] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5329–5333, IEEE, 2018.

[73] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *ICASSP 2019-2019 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pp. 5796–5800, IEEE, 2019.

[74] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification.," in *Interspeech*, vol. 2018, pp. 3573–3577, 2018.

[75] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *arXiv preprint arXiv:1803.10963*, 2018.

[76] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[77] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[78] O. Ghahabi and J. Hernando, "Deep belief networks for i-vector based speaker recognition," in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 1700–1704, IEEE, 2014.

[79] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 1021–1028, IEEE, 2018.

[80] Forbes, "The Most Amazing Artificial Intelligence Milestones So Far." `https://www.forbes.com/sites/bernardmarr/2018/12/31/the-most-amazing-artificial-intelligence-milestones-so-far`.

[81] T. T. Nguyen, Q. V. H. Nguyen, C. M. Nguyen, D. Nguyen, D. T. Nguyen, and S. Nahavandi, "Deep learning for deepfakes creation and detection: A survey," *arXiv preprint arXiv:1909.11573*, 2019.

[82] M. Masood, M. Nawaz, K. M. Malik, A. Javed, and A. Irtaza, "Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward," *arXiv preprint arXiv:2103.00484*, 2021.

[83] S. Greengard, "Will deepfakes do deep damage?," *Communications of the ACM*, vol. 63, no. 1, pp. 17–19, 2019.

[84] M. Westerlund, "The emergence of deepfake technology: A review," *Technology Innovation Management Review*, vol. 9, no. 11, 2019.

[85] "FaceApp." `https://www.faceapp.com/`.

[86] "ZAO." `https://apps.apple.com/cn/app/zao/id1465199127`.

[87] "Wombo." `https://play.google.com/store/apps/details?id=com.womboai.wombo&hl=en&gl=US`.

[88] "VoiceApp." `https://apps.apple.com/nl/app/voiceapp-ai-voice-changer`.

[89] R. Gagnon, "Votrax real time hardware for phoneme synthesis of speech," in *ICASSP'78. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 175–178, IEEE, 1978.

[90] W. I. Hallahan, "Dectalk software: Text-to-speech technology and implementation," *Digital Technical Journal*, vol. 7, no. 4, pp. 5–19, 1995.

[91] B. G. Greene, J. S. Logan, and D. B. Pisoni, "Perception of synthetic speech produced automatically by rule: Intelligibility of eight text-to-speech systems," *Behavior Research Methods, Instruments, & Computers*, vol. 18, no. 2, pp. 100–107, 1986.

[92] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio.," *SSW*, vol. 125, p. 2, 2016.

[93] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[94] O. Vinyals, Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a foreign language," *Advances in neural information processing systems*, vol. 28, 2015.

[95] "Google cloud tts services." `https://cloud.google.com/text-to-speech`.

[96] "Amazon aws polly tts services." `https://aws.amazon.com/polly`.

[97] "Microsoft azure tts services." `https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech`.

[98] "Ibm watson tts services." `https://www.ibm.com/watson/services/text-to-speech`.

[99] S. Desai, A. W. Black, B. Yegnanarayana, and K. Prahallad, "Spectral mapping using artificial neural networks for voice conversion," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 5, pp. 954–964, 2010.

[100] S. Desai, E. V. Raghavendra, B. Yegnanarayana, A. W. Black, and K. Prahallad, "Voice conversion using artificial neural networks," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3893–3896, IEEE, 2009.

[101] T. Toda, A. W. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2222–2235, 2007.

[102] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[103] T. Kaneko and H. Kameoka, "Cyclegan-vc: Non-parallel voice conversion using cycle-consistent adversarial networks," in *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 2100–2104, IEEE, 2018.

[104] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[105] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, "Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6820–6824, IEEE, 2019.

[106] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[107] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, "Stargan-vc: Non-parallel many-to-many voice conversion using star generative adversarial networks," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 266–273, IEEE, 2018.

[108] H. Lu, Z. Wu, D. Dai, R. Li, S. Kang, J. Jia, and H. Meng, "One-shot voice conversion with global speaker embeddings.," in *INTERSPEECH*, pp. 669–673, 2019.

[109] S. Liu, J. Zhong, L. Sun, X. Wu, X. Liu, and H. Meng, "Voice conversion across arbitrary speakers based on a single target-speaker utterance.," in *Interspeech*, pp. 496–500, 2018.

[110] J.-c. Chou, C.-c. Yeh, and H.-y. Lee, "One-shot voice conversion by separating speaker and content representations with instance normalization," *arXiv preprint arXiv:1904.05742*, 2019.

[111] K. E. Silverman, M. E. Beckman, J. F. Pitrelli, M. Ostendorf, C. W. Wightman, P. Price, J. B. Pierrehumbert, and J. Hirschberg, "Tobi: A standard for labeling english prosody.," in *ICSLP*, vol. 2, pp. 867–870, 1992.

[112] A. Rosenberg, "Autobi-a tool for automatic tobi annotation," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[113] N. Obin, J. Beliao, C. Veaux, and A. Lacheret, "Slam: Automatic stylization and labelling of speech melody," in *Speech Prosody*, p. 246, 2014.

[114] B. COILE, "Protran: A prosody transplantation toll for text-to-speech application," in *ICSLP94*, 1994.

[115] F. Eyben, S. Buchholz, N. Braunschweiler, J. Latorre, V. Wan, M. J. Gales, and K. Knill, "Unsupervised clustering of emotion and voice styles for expressive tts," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4009–4012, IEEE, 2012.

[116] S. Shechtman and A. Sorin, "Sequence to sequence neural speech synthesis with prosody modification capabilities," *arXiv preprint arXiv:1909.10302*, 2019.

[117] G. Zhang, Y. Qin, and T. Lee, "Learning syllable-level discrete prosodic representation for expressive speech generation.," in *INTERSPEECH*, pp. 3426–3430, 2020.

[118] Y. Wang, D. Stanton, Y. Zhang, R.-S. Ryan, E. Battenberg, J. Shor, Y. Xiao, Y. Jia, F. Ren, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *International Conference on Machine Learning*, pp. 5180–5189, PMLR, 2018.

[119] Y. Lee and T. Kim, "Robust and fine-grained prosody control of end-to-end speech synthesis," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5911–5915, IEEE, 2019.

[120] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "Asvspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," in *Sixteenth annual conference of the international speech communication association*, 2015.

[121] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, "Asvspoof 2019: Future horizons in spoofed and fake audio detection," *arXiv preprint arXiv:1904.05441*, 2019.

[122] J. Yamagishi, X. Wang, M. Todisco, M. Sahidullah, J. Patino, A. Nautsch, X. Liu, K. A. Lee, T. Kinnunen, N. Evans, *et al.*, "Asvspoof 2021: accelerating progress in spoofed and deepfake speech detection," *arXiv preprint arXiv:2109.00537*, 2021.

[123] J. Yi, R. Fu, J. Tao, S. Nie, H. Ma, C. Wang, T. Wang, Z. Tian, Y. Bai, C. Fan, *et al.*, "Add 2022: the first audio deep synthesis detection challenge," *arXiv preprint arXiv:2202.08433*, 2022.

[124] L. Guarnera, O. Giudice, and S. Battiato, "Deepfake detection by analyzing convolutional traces," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[125] Z.-F. Wang, G. Wei, and Q.-H. He, "Channel pattern noise based playback attack detection algorithm for speaker recognition," in *IEEE International Conference on Machine Learning and Cybernetics (ICMLC)*, 2011.

[126] C. Borrelli, P. Bestagini, F. Antonacci, A. Sarti, and S. Tubaro, "Synthetic speech detection through short-term and long-term prediction traces," *EURASIP Journal on Information Security*, vol. 2021, no. 1, pp. 1–14, 2021.

[127] K. Chugh, P. Gupta, A. Dhall, and R. Subramanian, "Not made for each other-audio-visual dissonance-based deepfake detection and localization," in *International Conference on Multimedia (ACM)*, 2020.

[128] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head poses," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[129] S. Agarwal, H. Farid, T. El-Gaaly, and S.-N. Lim, "Detecting deepfake videos from appearance and behavior," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2020.

[130] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[131] O. Wiles, A. Koepke, and A. Zisserman, "Self-supervised learning of a facial attribute embedding from video," *arXiv preprint arXiv:1808.06882*, 2018.

[132] M. Chen, X. He, J. Yang, and H. Zhang, "3-d convolutional recurrent neural networks with attention model for speech emotion recognition," *IEEE Signal Processing Letters*, vol. 25, no. 10, pp. 1440–1444, 2018.

[133] H. Zeinali, S. Wang, A. Silnova, P. Matějka, and O. Plchot, "BUT system description to voxceleb speaker recognition challenge 2019," in *The VoxCeleb Challenge Workshop*, 2019.

[134] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 2, pp. 652–662, 2019.

[135] N. Dehak, P. Dumouchel, and P. Kenny, "Modeling prosodic features with joint factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2095–2103, 2007.

[136] J. Lee, K. Cho, and T. Hofmann, "Fully character-level neural machine translation without explicit segmentation," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 365–378, 2017.

[137] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[138] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[139] K. Ito and L. Johnson, "The LJ Speech Dataset." `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[140] A. Lieto, D. Moro, F. Devoti, C. Parera, V. Lipari, P. Bestagini, and S. Tubaro, ""Hello? Who Am I Talking to?" A Shallow CNN Approach for Human vs. Bot Speech Classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[141] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "IEMOCAP: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, no. 4, pp. 335–359, 2008.

[142] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: a large-scale speaker identification dataset," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2017.

[143] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2018.

[144] S. King and V. Karaiskos, "The Blizzard Challenge 2013," in *Blizzard Challenge Workshop*, 2013.

[145] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "SpeechBrain: A General-Purpose Speech Toolkit," *arXiv:2106.04624*, 2021.

[146] H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and A. Larcher, "End-to-end anti-spoofing with RawNet2," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[147] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

[148] M. Alzantot, Z. Wang, and M. B. Srivastava, "Deep residual neural networks for audio spoofing detection," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2019.

[149] "SoX Sound eXchange." `http://sox.sourceforge.net`.

[150] J. F. Pitrelli, R. Bakis, E. M. Eide, R. Fernandez, W. Hamza, and M. A. Picheny, "The IBM expressive text-to-speech synthesis system for American English," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1099–1108, 2006.