POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Master Degree in Music and Acoustic Engineering

Synthetic Speech Detection Through Emotion Recognition: a Semantic Approach

by:
Emanuele Conti

matr.:
896658

Supervisor:
Paolo Bestagini

Co-supervisor:
Clara Borrelli
Davide Salvi

Academic Year
2020-2021

Abstract

The recent years have been characterized by huge advancements in the development of artificial intelligence techniques. These have opened the possibility of generating synthetic videos, audios, and images in such a realistic way that they are hardly distinguishable from real ones for a human eye or ear. This is the case of deepfakes, which are synthetic media in which a person in an existing image, video, or audio is replaced with someone else.

Although these newly created media have been largely employed for recreational and artistic purposes, it has shortly become evident that the misuse of this type of content may lead to serious consequences, in particular when impersonation is involved. Indeed, many problems related to synthetic media like deepfakes have arisen, such as spreading of fake news, fraud cases, reputation ruining, and even falsification of proofs in front of a court of law. For these reasons, it is becoming more and more urgent to develop techniques able to distinguish real content from synthetic one and avoid uncontrolled deepfake spread.

Since the majority of the deepfakes are videos containing visual and audio components, realistic speech generation is fundamental in order to achieve realistic synthetic content. Moreover, in some relevant cases, audio appears to be the only spoofed component. Some examples are fake speech recordings used as proof in front of a court of law or to fool a voice interface. The problem of recognizing synthetic speech is utterly delicate: speech synthesis techniques are becoming further sophisticated due to the increasing complexity of the underlying deep learning models.

In this work, we address the problem of synthetic speech detection using high-level semantic-based features. In particular, we extract embeddings from a neural network-based speech emotion recognition system and feed them into the final synthetic speech detector with a novel transfer-learning approach. Indeed, we believe that the synthesis algorithms, although definitely capable of synthesizing low-level characteristics of a particular human voice, fail to recreate more complex aspects, such as the emotional ones, which are instead natural in a real, or bonafide, speech.

In order to deeply evaluate the performances of the proposed system, we have built an ad-hoc dataset as the aggregation of multiple real and fake speech datasets. The total amount of processed data is huge, as well as the variety of the synthesis algorithms involved. Moreover, we used different datasets during the training and evaluation phase, in order to reach independence between the two and, therefore, to carry out a robust evaluation.

We have designed 4 experiments, with the goal of evaluating the general performances of the system, its robustness to noise, the quality of the semantic-based emotional features we extracted, and their compatibility with the synthetic speech detection task. The results of the system evaluation are promising and lead to some general observations and suggestions for possible future developments.

Sommario

Gli anni recenti sono stati caratterizzati da grandi progressi nello sviluppo delle tecniche basate sull'intelligenza artificiale. Tali progressi hanno reso possibile generare video, audio o immagini sintetiche così realistiche da essere difficilmente distinguibili da quelle reali per l'occhio o l'orecchio umano. E' il fenomeno dei deepfakes, ovvero media sintetici in cui una persona in un'immagine, video o audio esistente è rimpiazzata con qualcun altro.

Sebbene queste nuovi media artefatti siano stati impiegati per lo più per usi artistici e di intrattenimento, è diventato presto chiaro come il cattivo uso di questo tipo di contenuti potesse portare a conseguenze gravi, soprattutto quando si manifesta un furto di identità. In effetti, diversi problemi sono sorti riguardo a questi contenuti multimediali sintetici, come la diffusione di fake news, casi di frode e danneggiamento della reputazione, e addirittura falsificazione di prove davanti al giudice. Per queste ragione, sta diventando sempre più urgente sviluppare tecniche che siano capaci di discriminare i contenuti reali da quelli sintetici ed evitare la diffusione incontrollata di deepfakes.

Poiché la maggioranza dei deepfakes è costituita da video, contenenti dunque sia la componente visiva che quella audio, la generazione di parlato realistico è fondamentale per ottenere un contenuto sintetico verosimile. Oltretutto, in alcuni casi rilevanti, l'audio è l'unica componente falsificata. Alcuni esempi sono registrazioni di parlato falso usate come prova davanti ad una corte di giustizia o per raggirare un'interfaccia vocale. Il problema di riconoscere il parlato sintetico è assolutamente delicato: le tecniche di sintesi stanno diventando sempre più sofisticate a causa della complessità crescente dei modelli di apprendimento profondo, o deep learning, su cui sono basate.

In questo lavoro, affrontiamo il problema della rilevazione di parlato sintetico attraverso l'uso di caratteristiche semantiche di alto livello della voce. In particolare, estraiamo embeddings da un sistema di riconoscimento delle emozioni basato su una rete neurale, ed utilizziamo questi embeddings come input per il rilevatore di parlato sintetico, con un originale approccio transfer-learning. Siamo infatti convinti che gli algoritmi di sintesi vocale, sebbene capaci di riprodurre le caratteristiche di basso livello di una particolare voce umana, falliscano nel ricreare aspetti più complessi, come ad esempio quello emozionale, che sono invece naturali in una voce reale.

Per valutare in profondità le prestazioni del sistema proposto, abbiamo costruito un dataset ad-hoc mettendo insieme diversi datasets contenenti parlato reale e/o sintetico. Il numero totale di tracce utilizzate è ingente, così come la varietà degli algoritmi di sintesi utilizzati. Inoltre, abbiamo usato dataset diversi durante la fase di allenamento e quella di test, al fine di avere due set indipendenti e, perciò, di condurre una valutazione delle prestazioni più robusta.

Abbiamo progettato e realizzato 4 esperimenti, con gli obiettivi principali di valutare le prestazioni del modello in generale, la sua resistenza al rumore, la qualità delle caratteristiche semantiche che abbiamo estratto e la loro compatibilità con l'operazione di rilevamento del parlato sintetico. I risultati della valutazione del sistema sono promettenti, e conducono ad alcune osservazioni ed a suggerimenti per possibili sviluppi futuri.

Ringraziamenti

Ci tengo innanzitutto a ringraziare Paolo, Clara, Davide e Fabio per la direzione e l'aiuto che mi hanno fornito durante tutto lo svolgimento di questo lavoro.

Ringrazio i miei genitori, Stefania e Olmo per tutto il supporto che mi hanno dato durante tutti questi lunghi anni. Ringrazio la Nonna per avermi ascoltato e sopportato in numerose situazioni con i suoi 'eeeeh' e le sue spalle larghe. Ringrazio il nonno Giorgio e la zia per avermi dato la possibilità di stare dove sono e vivere serenamente quest'ultimo anno e mezzo. E ringrazio Anastasiya, per la sua infinita pazienza, per essermi stata vicina anche nei momenti peggiori, e per tante altre cose.

Ci tengo inoltre a ringraziare, in particolare, tutti coloro hanno contribuito a rendere quest'ultimo periodo di vita, credo esaltante per nessuno, più lieve. Andrea, Gabri, i due Stefani, Gaia, Ale, Paolo, Da, Edo, Jenny, Jaco, Anto e tutti gli altri.

Infine, un sentito grazie va allo stipite della porta dello studio, a causa del quale sono stato fortunatamente costretto a rimanere fermo e dedicarmi completamente alla scrittura di questo lavoro. Finalmente, giungo anche io a tagliare questo agognato e finora sfuggente traguardo. Il nome sul frontespizio pare sia il mio, ma il merito va sicuramente ripartito fra tutti coloro che ho citato, e, probabilmente, qualcuno in più.

Salut!

E.C.

Contents

List of Figures

List of Tables

Introduction

Even if we do not always realize it properly, Artificial Intelligence (AI) is increasingly becoming part of our lives. When Google decides which results of a search appears in the web page first, when a virtual assistant understands our speech and matches it with an answer or action, when Facebook decides the list of friends to suggest, when the advertisement of the exact product you need magically appears in the banner of an unknown site, most likely an AI-based algorithm is performing the task.

Multimedia contents are no exception to this trend. On the contrary, the possibility to come across videos, audios or images generated by means of AI algorithms increases day by day. This is enhanced by the impact of social media in our lives, which makes the propagation of content an immediate, zero-effort action.

Let us consider for a moment the audio field, which is the scope of this work. We bump into artificially-generated synthetic speech almost every day: at the station when trains are announced, in audiobooks, or when getting an answer from a virtual assistant, as Alexa or Siri. Not only, but the main high-tech companies have nowadays their proprietary software to generate synthetic speech [12, 13].

With the term deepfake (a portmanteau of "deep learning" and "fake") we refer to a particular category of AI-generated multimedia contents, videos in the great majority of the cases, that have been edited in order to alter the identity of the depicted person by means of facial or speech manipulation. This can be done by swapping the face, restyling the voice, or modifying what the person is saying [14, 15, 16], almost always through the use of deep learning techniques. Thanks to the constant development of new technologies and the massive evolution of neural networks, deepfake generation is nowadays an effortless operation. Furthermore, it is becoming increasingly difficult to distinguish the manipulated material from the original one.

Despite deepfakes are used for entertainment purposes in most of the cases [17], their misuse may also lead to unpleasant situations, primarily when the generated content does not have the approval of the involved people. Deepfakes have been used with malicious intents in several cases, especially in sectors such as mass and social media. Some examples concern the spreading of fake news [18], creation of revenge porn videos [19], and fraud cases [20], which led to some ethical considerations regarding the use of artificial intelligence [21]. Furthermore, social media are one of the contributing causes of the hyper-connected society in which we live. In this excess of stimuli, we are used to consume contents very quickly and, often, without asking ourselves questions about the quality, integrity or origin of the content itself.

Concerning audio modifications, the rapid progress of Text-to-speech (TTS) synthesis and Voice Conversion (VC) techniques has increased the possibility of impersonating one person's voice. Many examples of misuse are worth to be taken into consideration. Let us just picture a scenario in which a party to the dispute is able to produce high-fidelity, synthetic speech and use this recording as proof in a court of law. Evidently, the spoof audio could highly influence the decision of the court, thus the final sentence [22]. Synthetic speech generation is not only dangerous in social situations, but it represents also a threat to interfaces and biometric authentication systems based on voice.

To address these problems, it has become of paramount importance to develop techniques capable of determining whether a given multimedia content is authentic, namely *bonafide*, or it is synthesized by the means of an algorithm, namely *spoof* [23]. We call Synthetic Speech Detection (SSD) a system which is capable of performing this discrimination.

In the past few years, the research community has moved in this direction and has already proposed numerous approaches that analyze both video [24, 25] and audio [26, 27] material.

The audio component is crucial to generate a well-counterfeited speech and, therefore, accurate and convincing synthetically generated videos. Moreover, in some cases, speech is the only element in analysis: let us consider again the example of phone-tapping that can be used as proof in front of a law court. Hence, in this work, we focus on the audio field.

Several works have been done in the recent years. For example, [28] and [3] use the bicoherence feature to cluster speech segments coming from real and spoof speech. In [29], the authors exploit long-term features to discriminate between fake and real audio tracks. The method in [30] uses capsule neural networks for the same purpose. The authors of [31] feed linear filter banks into a Resnet to generate embeddings that are used as input of a final neural network classifier. Recently [32] proposed a method for audio deepfake detection based on long-term and short-term predictor features. Many of the past works make use of low-level speech features in order to perform the classification. The majority of recent works employs neural networks, either as end-to-end models or as extractors of features to feed the final classifier with.

In this work, we propose a method to identify whether a given speech signal is bonafide or spoof exploiting another task which involves AI, i.e. Speech Emotion Recognition (SER). To perform this classification, we leverage semantically meaningful audio embeddings, as the use of semantic features has already proved successful for both audio and video deepfake analysis [33, 34, 35]. In particular, as suggested in [33], we assume that synthesis algorithms are able to reconstruct low-level characteristics of voice, but fail to recreate more complex aspects, such as the emotional ones, which are in all respects natural in a bonafide speech.

Speech Emotion Recognition (SER) is a field that has been increasingly investigated in recent years and refers to the problem of automatically recognizing the emotion perceived by the talking person from the analysis of its recorded speech. Many different networks have been designed for this purpose, both considering audio (speech) only [6, 11, 5], and multi-modal approaches [36, 37]. In this work, we propose a novel transfer-learning method, feeding the semantic features extracted from a SER network as input of the Synthetic Speech Detection (SSD) classifier.

The method is focused on the detection of TTS and mixture TTS/VC deepfakes, while does not take into account pure VC algorithms. This is because we exploit speech semantic information to detect anomalies, and pure VC spoofed speeches do include such content, being generated from a real voice and then altered with style transfer techniques.

We performed 4 large-scale data-driven experiments on 123 effective hours of speech recordings. We built this ad-hoc dataset as an aggregation of different datasets containing bonafide and/or spoof speech tracks. Since different datasets are used in the training and evaluation phase, this choice guarantees us a robust evaluation of the performance, as the success of the classification is independent from prior knowledge of the spoofing attacks and from dataset-specific parameters. The experiments aim to verify the performances of the proposed model, its robustness to noise and the quality of the extracted semantic features in order to complete the SSD classification. The results of the experiments show promising performances, with balanced accuracy reaching a value of 0.912 on the cross-corpus setup with clean data.

The work is organized as follows. In Chapter 1, we give an overview of the main state-of-the-art techniques for both the Synthetic Speech Detection (SSD) and the Speech Emotion Recognition (SER) task. Then, we analyze some late works involving semantic-based features for SSD classification, as in our case. In Chapter 2, we provide to the reader the main notions about the features we use as input of the system, namely the log-mel spectrogram, and about the main machine and deep learning techniques we employ. In Chapter 3, we give a mathematical formulation of the problem and we formally describe the proposed system. In Chapter 4, we describe in details the ad-hoc dataset we have adopted and the experiments performed. Furthermore, we give the details about the training parameters choice, the baseline to compare with and the evaluation metrics. In Chapter 5, we analyze in details the results obtained in each experiment by means of the evaluation metrics. Finally, in Chapter 6, we pull the strings of the work, and suggest possible future developments and improvements.

1

State of the Art

In this chapter, we introduce the state-of-the-art related to this work. We start by describing some remarkable works concerning audio deepfake detection and SSD, our final goal. Then, we continue with an overview of the main works related to SER, since we are going to employ this kind of classifier as a feature extractor for our system. Finally, we conclude this section with a few works specifically related to semantic-based speech detection.

1.1 Synthetic Speech Detection

In many contexts, speech is nowadays used to perform biometric recognition. An Automatic Speaker Verification (ASV) system is a systems capable of performing this task [38]. Any ASV system is prone to error and to spoof. In particular, a malicious user can try to violate the system by the means of three types of attack [39]:

- Impersonation, which consists into manually trying to imitate the

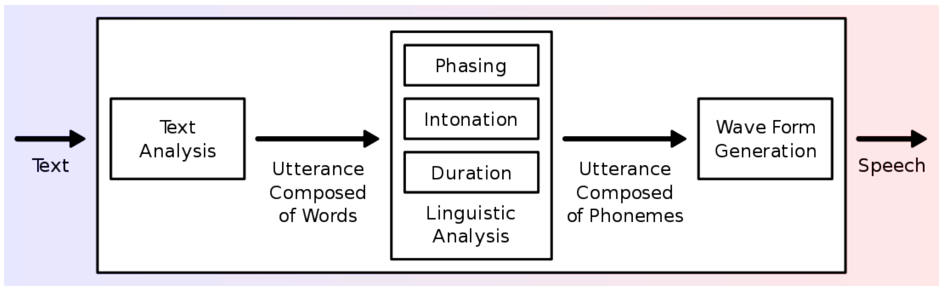


Figure 1.1: General architecture of a Text-to-speech system taken from [1].

voice of the service legitimate user.

- Replay, which happens when an attacker records a speech track of the legitimate user and reproduces it in front of the ASV system.

- Speech synthesis, which we generically intend as spoofing the speech of the legitimate user by means of computer-based techniques able to generate a similar voice. The main techniques are the generation of synthetic speech directly from text, namely Text-to-speech (TTS) algorithms, and the generation of synthetic speech from an original voice of a different speaker through parametric conversion, namely Voice Conversion (VC) algorithms.

The violation of ASV and biometric recognition systems is not the only issue that arises with synthetic speech. Indeed, as we discussed in the Introduction, the increased popularity of deepfakes has clearly shown that the diffusion of fake speech online may lead to serious consequences and has a dangerous social impact. We remind that with the term deepfakes we intend a category of multimedia content obtained altering the content of an original video and/or audio content, in order to change the identity or the intentions of a person that appears in it.

Since deepfakes videos can be altered at both image and audio level, both audio and video spoofing detection capability is particularly interesting. In this work, we focus on the audio field, i.e. on speech synthesis and SSD. We will call bonafide speech the speech utterances or tracks generated by real speakers, while we will call synthetic or spoof or deepfake speech the ones generated by the means of various TTS or VC algorithms.

We now analyze the two main techniques to generate spoof speech utterances. The TTS algorithms consist in synthesizing speech directly from text. Figure 1.1 shows the main blocks and the work flow of a general TTS system. We can distinguish between two main phases [40]:

- A Natural Language Processing (NLP) phase, which first analyses the syntax and morphology of the text and split it into words. Then, words are in turn associated to their phonetic transcription. The text is also divided in prosodic units (phrases, sentences), and target prosodic characteristics are assigned to these units, as pitch contours, phasing, phoneme durations, rhythmic patterns and velocity. Phonetic transcriptions and prosody characteristics form the symbolic linguistic representation and compose the output of the NLP module (Utterance Composed of Phonemes in Figure 1.1).

- A waveform generation phase, which is the key phase of TTS algorithms. This second block takes as input the symbolic linguistic representation and outputs the generated speech.

  Among the many techniques that have been developed for this purpose [41], we can focus on a few families that has proved very successful.

  Formant synthesis uses a tunable filterbank to estimate the vocal tract transfer function through modelling the formants, namely the fixed frequency peaks characteristic of the human voice. Linear prediction synthesis is very similar to formant synthesis but uses the linear prediction estimation to model the spectral envelope of a speech signal.

  Articulatory synthesis directly models of the human vocal tract to simulate the air flow passing through it.

  Concatenative synthesis follows a data-driven approach. It generates speech by connecting prerecorded speech units, called phones, by the means of diphones. A diphone is a transition unit modeling co-articulation between two consecutive phones.

  Although it is possible to generate waveforms with a variety of TTS techniques, Deep Learning (DL) based techniques have recently become increasingly popular due to the naturalness of the produced speech. Many techniques have been proposed recently, some of the most popular ones are based on generative models such as Generative Adversarial Network (GAN), Encoder-Decoder [42] and Wavenet [43, 44]. These are able to produce hardly distinguishable, high-quality synthetic speech.

A VC algorithm consists in converting a non-target speaker voice into an imitation of a target speaker voice, keeping the the same word content. Figure 1.2 shows the scheme of a general VC system. We can divide each VC algorithm in 3 main distinct phases, the first two pertaining to the model training and the third to the effective conversion:

- The analysis phase, in which the source and target speaker-dependent feature extraction is performed. Various feature can be extracted. Common choices are Mel Frequency Cepstrum Coefficients and linear spectral frequencies [45].

- The mapping phase, in which the features extracted in the previous step from the source speaker are mapped to the features extracted from the target speaker. Many classic signal processing algorithms are capable of performing this task: we can cite vector quantization and frequency warping among the other techniques [2]. Other mapping algorithms are based on statistical models like Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs). Nonetheless, as in the TTS case, the state-of-the-art algorithms make large use of Neural Networks (NNs).

- The conversion phase, in which an arbitrary speech from a source speaker is converted into a speech with the same content from the target speaker. This phase basically performs a voice style-transfer from target to source speaker exploiting the mapping obtained in the previous phase.

So far, we have briefly described the main techniques used for audio synthesis. In the prosecution of this section, we describe extensively the main approaches that have been used to perform Synthetic Speech

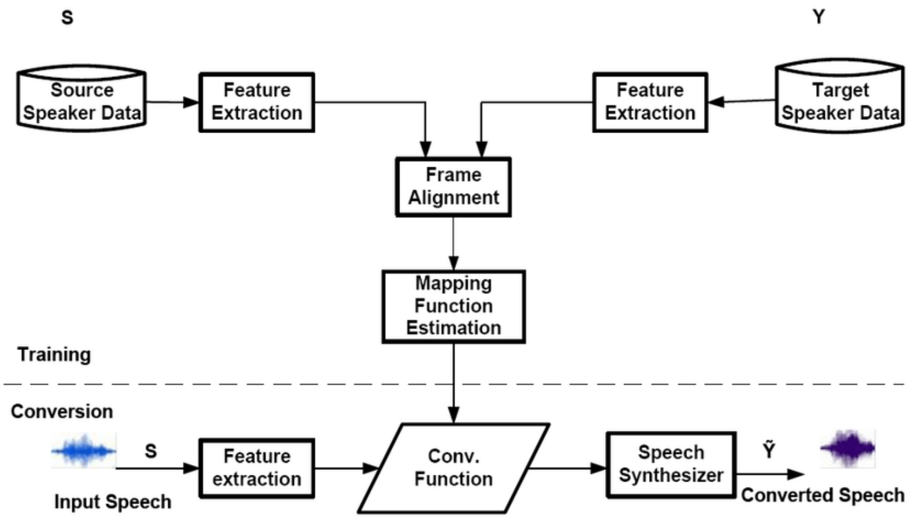


Figure 1.2: General Voice Conversion system. The figure is courtesy of [2].

Detection (SSD), starting from the signal processing-based ones and concluding with the most advanced and recent ones making an extensive use of Artificial Neural Network (ANN)s.

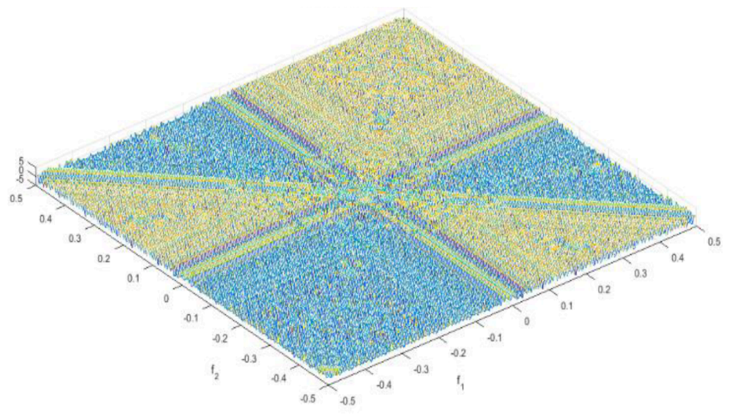
1.1.1 Traditional techniques

Early studies about SSD are based on traditional signal processing and Machine Learning (ML) techniques.

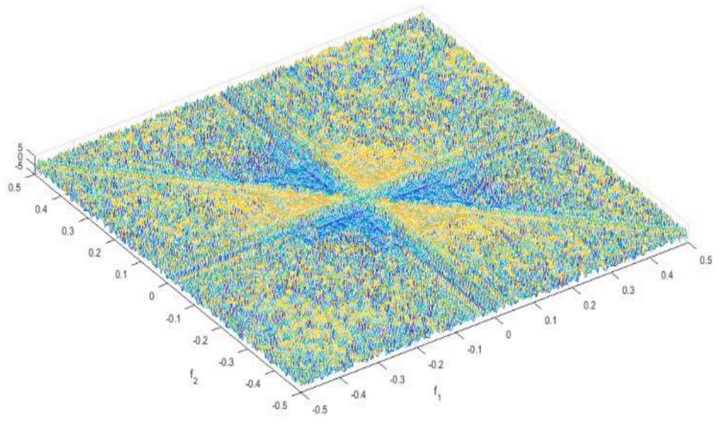
In [46], features based on pitch patterns are used to distinguish between human and synthetic speech.

Another early study analyzes long-term temporal information of speech starting from spectra, in order to detect temporal artifacts caused by the frame-wise processing of synthetic speech signal [47]. These works adopted their own databases and evaluation metrics.

A joint effort to develop a shared, exhaustive dataset and common evaluation metric has been done in 2015, with the first version of the Automatic Speaker Verification Spoofing and Countermeasures (ASVspoof) database and challenge [48]. The ASVspoof 2015 database consists of bonafide and spoofed speech tracks, which are generated via TTS and VC algorithms, divided in known and unknown attacks. This division has the purpose to train the model on known algorithms and test it on unknown algorithms. In the proposed challenge, it was observed that better



(a) Phase spectrum of bicoherence estimated from a cloned voice.



(b) Phase spectrum of bicoherence estimated from a bonafide speech (ground truth).

Figure 1.3: Phase spectrum of bicoherence estimated from a cloned and a ground truth voice respectively. The image is taken from [3].

features generally have a greater influence on performances with respect to more complex ML classifiers, and that long-term features normally guarantee better performances on evaluation with respect to short-term features.

The Constant-Q Cepstral Coefficients (CQCC) are features extracted using the constant-Q transform. They are considered a perceptually-based alternative to classical approaches based on spectra and Fourier transform for time-frequency analysis. In [26], a GMM binary classifier is fed with CQCC and Mel Frequency Cepstral Coefficients (MFCC). The CQCC features provided the best state-of-the-art performance at the time, not only in the ASVspoof challenge [48], but also with the

RedDots [49] and AVSpoof datasets [50, 51, 29].

Other features that proved to be effective are magnitude-based features, including Log Magnitude Spectrum and Residual Log Magnitude Spectrum, and phase-based features, including Group Delay Function, Pitch Synchronous Phase, Instantaneous Frequency Derivative. In [52], a MLP network is fed and trained with these features in turn, providing an extensive comparison and analysis of their performances for the SSD task.

SSD classification in the ASVspoof 2015 challenge has also been performed with features extracted using subband processing, like Linear Frequency Cepstral Coefficients (LFCC) [48]. The reason for subband processing is that artifacts of TTS are not equally distributed in all the subbands, but they normally concentrate in some parts of the spectrum.

Another family of features used for SSD is the one based on bispectral analysis to detect artifacts and third-order correlation between frames. In [53], the first 4 statistical moments are calculated from phase and magnitude of bicoherence, namely the normalized bispectrum. A simple logistic regressor is then fed with the resulting vector and performs the classification. The classifier is trained to detect between bonafide speech and 4 well-known TTS algorithms and has shown very promising results.

In [32], bicoherence, Short-term and Long-term (STLT) prediction traces, and a combination of bicoherence and STLT are used as features. RF and SVM classifiers has been tested. The work proposes 3 different experiments which differs for the final classifier: a binary classifier, a closed-set setup in which the classifier distinguish between the different types of spoofing algorithms and the bonafide speech, and an open-set setup, similar to closed-set but with the addition of a possible outcome for unknown spoofing algorithms.

In [3], quadrature phase coupling in the estimated bicoherence, Gaussianity test statistics, and linearity test statistics are used as features for the classification.

1.1.2 Deep learning-based techniques

Most of the techniques developed for SSD in the recent years are deep learning-based.

DL-based techniques are often used for feature learning. We define as feature or representation learning an ensemble of techniques that allows an AI model, often an Artificial Neural Network (ANN), to automatically extract the feature needed to complete the task (typically classification or regression) directly from the raw data. A feature learning model followed by GMM, SVM or other machine learning classifiers has proved to provide an efficient and performing solution for the SSD problem [54].

In [55], a network consisting of a MLP is trained for SSD. Then, the outputs of the last hidden layer are used as features for the final classifier. The latter calculates from these features the Mahalanobis between test utterances and the clusters of each different class. The shortest distance correspond to the class predicted for the utterance.

Similarly, in [54], delta filterbank spectra (DFB), delta and double delta (first and second derivative in time) Mel-frequency cepstral coefficients (DMCC), delta and double delta linear prediction cepstral coefficients (DLPCC) and product spectrum-based cepstral coefficients (DP-SCC) features are fed into a deep NN consisting of a MLP with 5 hidden layers, which is the feature learning model. The bottleneck features, i.e. the embeddings of the last hidden layer, are extracted from the ANN are then used to train a GMM model.

In [56], the features extraction is performed by a MLP-based filterbank. The spectrum is extracted and passed into a filterbank, whose various output bands train different 2-layer MLP networks. The features extracted from each MLP first hidden layer are then used to train a GMM as final classifier.

More recently, end-to-end techniques based on ANN have emerged for a large variety of audio processing applications [57]: with this kind of approach, both the representation learning from raw speech and the classification are implemented through DL techniques. SSD makes no exception to this trend.

In [58], a Convolutional Neural Network (CNN) model learns the representation from the raw speech signal and a single hidden layer MLP model learns the final classifier from the features provided by the CNN.

CNNs and RNNs are both used in [59]. A CNN is used as a convolutional feature extractor, then a RNN is used to capture correlations in the time domain. The final classification layer, which follows the RNN layer, consists in the classical fully-connected layer with softmax output. Many different features were analysed as input to this DL system, including Teager Energy Operator Critical Band Autocorrelation Envelope, Perceptual Minimum Variance Distortionless Response, and spectrograms.

Similarly, in [60], the representation learning model consists in a CNN applying convolution first in the time and then in the frequency domain to the raw speech signal, and then in a Long Short-term Memory (LSTM) recurrent model in order to catch long-term dependencies on the signal sequence. The final classifier is again learnt by a MLP.

In [61], the end-to-end network consist of 5-layer MLP with sigmoid activation. The work considers a close set problem, therefore, the output softmax layer has $k + 1$ nodes, where k is the number of spoofing algorithms considered. The inputs are dynamic acoustic features, in particular frame level spectrum-based cepstral coefficients (SBCC) and CQCC. The main novelty which this work introduces is the usage of Human Log-likelihoods (HLL) to replace Log-likelihood Ratios (LLR) as a scoring method for spoofing detection. Instead of averaging the outcome score of the single frames in the utterance as in LLR scores, the HLL score takes into consideration just the output probability of the human node (i.e. the node that predicts the probability of the data being bonafide) and enhances the importance of the frames with a very low score. When a certain number of frames with very low human score is observed in an utterance, the latter is classified as spoof. This scoring method is particularly interesting when the artifacts are concentrated in specific temporal frames of the synthetic speech signal, and has effectively proved to be more suitable than LLR scoring method [29].

1.2 Speech Emotion Recognition

In this section, we briefly analyze the main state-of-the-art works in the context of Speech Emotion Recognition (SER). This is relevant for the discussion since a SER system provides the semantic features that feed into our SSD classifier.

Sentiment analysis, also called opinion mining, is a set of natural language processing techniques used to determine automatically the general sentiment in the input data [62]. Sentiment analysis was born on textual input data to analyze the positiveness of the reaction of people to an event. With the diffusion of the internet and social media, this discipline has become of paramount interest [63]. In fact, it is evident that the ability to learn people's opinion and trends interests a lot of sectors: marketing [64], business intelligence [65] and political consensus management are just some example.

With the rapid development of the technologies and the interest in this field, sentiment analysis has begun to further spread from textual to multimedia input in general. Overall, analyzing sentiment of video and audio contents has become particularly interesting in the perspective of Human Computer Interaction (HCI) [66].

In this work, we are interested in the sentiment analysis concerning speech signals, which is commonly called Speech Emotion Recognition (SER). In fact, as said, we are going to use a model trained for SER as a feature extractor to train the synthetic speech detector by using a transfer learning technique.

SER can be declined as a regression or a classification task:

- A SER classifier maps the emotion contained in a speech utterance into a set of discrete, categorical emotions. Happiness, anger, sadness and neutrality represent a commonly used minimal set of emotion labels for the classification task. The majority of the SER models are classifiers.

- A SER regressor maps the emotion contained in a speech utterance into a point in a characteristic space. The most common space defined in these work is the Russel's valence-arousal space [67]. The

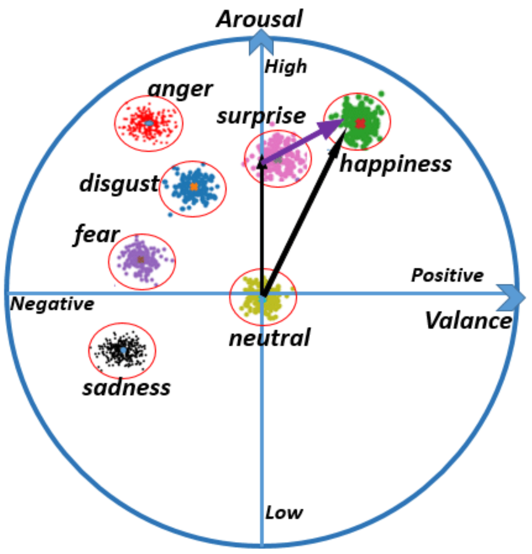


Figure 1.4: Clusters of the main emotions in a representation of the valence-arousal space. The image is taken from [4].

arousal is defined as the level of excitement in the speech utterance. The valence represents instead the level of positivity in the speech utterance. For example, a happy utterance will be probably characterized by high valence and medium or high arousal, while an angry segment will be characterized by negative low valence and high arousal. An extensive distribution of the emotions in the valence-arousal space is shown in Figure 1.4.

The main issue with the SER regressor consists in the utterances labeling. Existing SER datasets are for the most part labeled by human annotators. While it is relatively easy for them to discriminate between a finite amount of discrete emotions, it is quite hard to annotate a precise number for each dimension of the arousal space instead.

Another distinction that we need to make is between simulated, induced and natural databases.

In the simulated databases, the speech utterances are recited by professional actors or performers. The dialogs may be scripted or improvised. The latter are supposed to mimic human emotions in a slightly more realistic way.

In the induced databases, an artificial situation is induced in order to

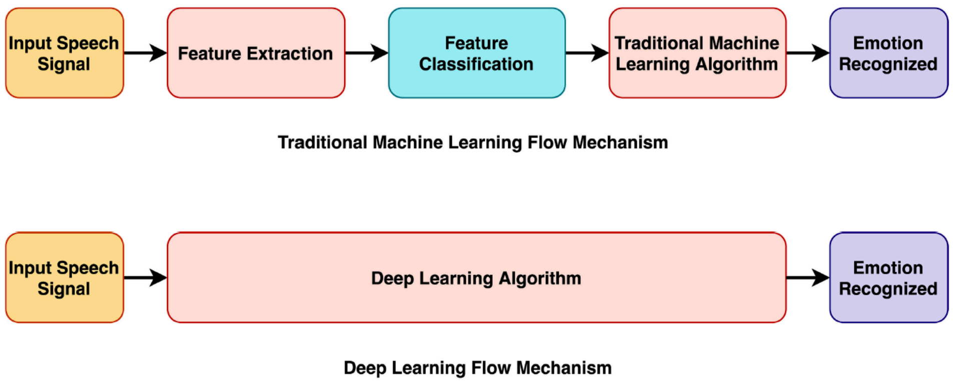


Figure 1.5: Traditional Machine Learning vs Deep Learning flow for SER. The figure is taken from [5].

enhance a particular emotion. The performers or speakers are unaware of the stratagem.

In the natural databases, speech utterances are obtained selecting emotional speech recorded in fully realistic situations.

Although the last two categories provide more realistic emotional speech utterances, the great majority of the databases are simulated, since it is the easiest technique to obtain a SER database.

1.2.1 Traditional techniques

Many techniques have been developed for SER in the years. The first techniques developed were the so called classical techniques. These techniques do not involve DL and ANN. Conversely, these systems normally consist of three fundamental blocks: the signal preprocessing block, the feature extractor and selector, and the classifier or regressor.

The signal preprocessing block is actually optional in both traditional and DL-based techniques. The preprocessing may consists of a denoising step, since noise can affect the system generalizing capability. Decompression, resampling, scaling and filtering of the input signal may also be performed whether needed, namely often in the case of cross-corpus setup.

Figure 1.5 shows a comparison between the workflows of the main SER paradigms, the one based on classical techniques and the DL-based one, after the preprocessing step.

The traditional SER system first perform the feature extraction. Numerous features have been proposed in different works, based on both short period characteristics (for example, pitch, energy, formants in short frames) and long period characteristics (statistical moments as mean and variance).

There are two main categories of speech features utilized in SER:

- Prosodic features, as pitch, rhythm, spoken words rate and intensity, which can be perceived by humans. Prosodic features are calculated on medium-long units as words or even entire sentences.

- Spectral features, which are obtained by transforming the waveform signal, segmented into short windows of 20 to 50 milliseconds duration, into the frequency domain using the Fourier transform. The most commonly used spectral feature is MFCC, which represents the short term power spectrum of the signal in the Mel perceptual scale [68].

Then, the selection of the main relevant extracted features may be performed to reduce the number of input variables.

The last block consists of a statistical or machine learning model, the classifier, which can be linear or non linear. The most common choices are GMM, SVM, HMM, K-nearest Neighbors (KNN) and Bayesian classifiers [5].

An early study on SER [69] compares the result obtained with different types of long period and prosodic features. A Kernel Regressor and a KNN perform the classification on the basis of the extracted features.

In [70], MFCC are extracted from short frames and their statistical moments are calculated to obtain a compact utterance-level representation. A KNN algorithm is then used as a classifier for the model.

MFCC, MFCC-low (a variant of MFCC limited to low frequencies) and plain pitches are adopted as features in [71]. A GMM performs the utterance-level classification. Similarly, in [72], MFCC are extracted from the SAVEE database [73] and fed to a HMM classifier.

Features as energy, pitch, linear predictive spectrum coding, MFCC and mel-energy spectrum dynamic coefficients are adopted in [74]. A

SVM model performs the classification. The accuracy of the classification is evaluated for five different combination of the features mentioned above. A similar work also adopted subsets of slightly different short period features as fundamental frequency, energy, zero crossing rate, linear predictive coding and MFCC. The classifier is again a SVM [75].

In [76], features are combinations of pitch, energy, zero-crossing rate and MFCC. The final learner is a Naive Bayes classifier.

1.2.2 Deep Learning-based techniques

As for SSD, even in the case of SER task the DL-based techniques have brought remarkable advancements in the last few years and are rapidly replacing the classical techniques.

As shown in Figure 1.5, with respect to the traditional case, in the DL-based SER systems the feature extraction and the classification/regression are normally both performed by DL layers and structures connected together. Albeit the main DL scheme consists of an end-to-end DL algorithm recognizing emotion directly from the raw speech signal or from spectrograms, some structures, as for example the ones based on Transfer Learning or Multitask Learning, are more complex and slightly differ from the scheme.

The techniques based on data-driven learning approach primarily differentiate in the composition of the NN model. Some of the main networks employed are MLPs, CNNs, RNNs and LSTMs, Deep Belief Networks (DBNs), Autoencoders (AEs) and combinations of the previous.

In [77], a regressor based on a RNN-LSTM structure is proposed. The system recognizes the emotion applying a mapping from the track frames to a 3D space composed by activation, valence, and time dimensions. The input features are various Low-level Descriptors (LLDs) as pitch, formants and cepstral coefficients. The regressor consists of a LSTM network composed by 8 blocks of 4 cells each.

A work presented at Interspeech 2014 adopts an end-to-end DL model consisting of an Extreme Learning Machine (ELM) with a single hidden layer. ELM is a feed-forward MLP network which uses Moore-Penrose generalized inverse instead of backpropagation to set its weights [78].

In [79], one of the first SER works involving CNNs networks is proposed. After a denoising step, a 1D CNN automatically learns the best representation from the raw speech signal, while the following LSTM structure has the function to capture the temporal dependencies between frames to build an utterance-level coherent representation. This works has highly contributed to the SER state-of-the-art.

In [80], the SER network consists of a 2D 3-layers CNN representation learner followed by a fully-connected classification layer. The input features of the network are spectrograms images of the speech signal.

In [81], the performances of 1D and 2D CNN structures are compared. Both networks are composed by 4 CNN layers with max pooling, followed by a LSTM and a fully connected layer. The input of the 1D CNN is the raw signal, while the input of the 2D is the Mel spectrogram calculated on the signal. The experiments are performed on IEMOCAP [82] and Emo-DB databases [83], in both speaker-dependent and speaker-independent cases. 2D convolution performed better in all the examined cases.

An extensive use of autoencoder-based structures is made in [84]. The system employs in turn denoising AE, variational AE, adversarial AE and adversarial variational Bayes AE as features extractors and feeds these feature to a classifier composed by a 3-layer fully-connected NN. The classification is performed calculating the probability of each class on fixed-length frames of one second, and averaging these probability on the entire speech track. The adversarial variational Bayes AE produces the best performance in terms of unweighted accuracy.

In [85], a Conditional Variational AE, an evolution of Variational AE which is able to generate a desired output injecting conditions into the latent representation, is employed as a feature extractor starting from log-mels input features. The final classification is then performed by a double layer LSTM followed by a fully-connected layer.

Many advanced techniques have been exploited in order to improve SER model accuracy. The most frequently used are Transfer Learning (TL), Multitask Learning and attention mechanisms.

TL is a ML technique that consists in relocating knowledge obtained solving a task in order to solve another task. TL techniques normally

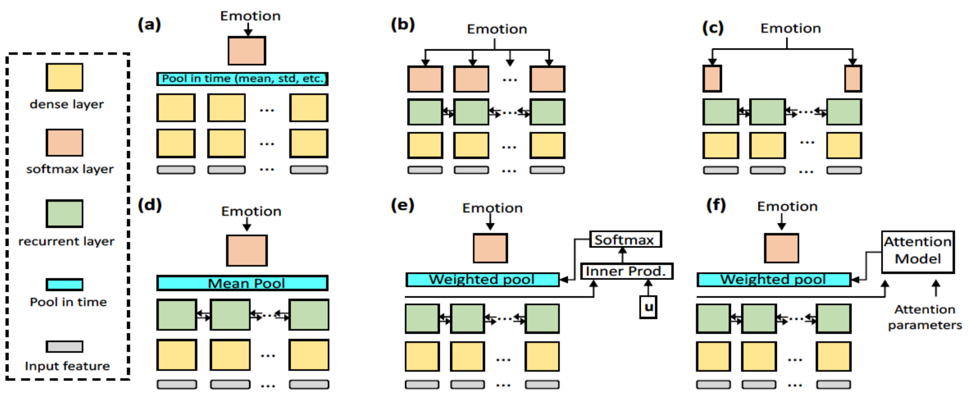


Figure 1.6: Scheme of a MLP/RNN network with different techniques for temporal aggregation. a) Pooling in time. b) Frame-wise training. c) Final-frame training. d) Mean-pooling in time. e) Weighted pooling using an attention model based on logistic regression. f) Generalized attention model. The figure is taken from [6].

train a source model or exploit a pretrained model on a similar, but different task, and then use this model as a feature extractor for the target task [86].

TL has been extensively used in SER. In [87], TL is applied between gender, speaker and emotion recognition tasks, in the hypothesis that the great majority of paralinguistic tasks are closely related. The knowledge transfer is performed by a Progressive Neural Network (ProgNet) [88]. The ProgNet trains the network on multiple tasks alternatively. When the network learns a single task, the layers previously trained for a different task are frozen, but the intermediate representations of these layers are used as inputs to learn the current task.

Image processing is another field from which knowledge is drawn on and transferred to SER. In particular, complex pretrained CNN models as AlexNet [89] are widely used due to their well-known effectiveness to solve image processing issues. In [90], SER is declined as a problem of image classification. The input of the system are RGB images of the spectrogram generated from the raw speech tracks. These input are fed into AlexNet with two different methods. In the first method, AlexNet is used as feature extractor for a classical SVM classifier. In the second method, the AlexNet is fine-tuned to directly provide the emotional outcome.

Multitask Learning [91] is a ML technique where a shared representation is used to learn more than one task at the same time. As in the case of TL, if the tasks have a high compatibility, it is possible to improve the generalisation of the model by mutually transferring the knowledge obtained from the learning of one task to the others.

In [92], a multitask approach is developed using, in particular, gender and naturalness as auxiliary tasks for SER. A set of LLD feeds the classifier composed by an LSTM learning the three task together at frame-level. Then, 4 statistical functions are applied to the emotional features learned at frame-level to produce utterance-level emotional features. Finally, the concatenation of the statistical function feeds an ELM network to produce the desired emotional outcome. The proposed method was experimented in an in-the-wild, cross-corpus setup composed of multiple existing database, and produces significant results in this context.

The great majority of the SER regressors learn each dimensional attribute separately. In [93], a novel framework learns the arousal, valence and dominance dimensions space together, i.e. considering them as correlated, in a multitask learning fashion. Two models are implemented, both consisting in a MLP with two hidden layers. The first model has both hidden layers in common, while in the second model the last hidden layer actually consists of 3 separate hidden layers, one for each task. The accuracy of the model improves significantly with respect to the single task learning, particularly in the experiment with cross-corpus setup.

An attention mechanism [94, 95] is a technique that trains a DL model in order to pay attention primarily to portions of the given samples. These portions are considered relevant for the task based on the attention weights calculated from the whole input. In particular, emotions are observed to be concentrated in particular, salient parts of the utterance and not equally distributed in the whole utterance. Supported by their capability of improving the system performances, attention mechanisms have recently become a popular technique within many context, including SER [96].

In [6], the first part of the network is composed by a MLP and a LSTM and produces the frame-wise emotional outcomes. Then, based on all the frame-wise single outputs, the model calculates the utterance-level outcome as a weighted sum of the frame-wise outputs, where the weights are determined by additional trained parameters of the attention model. In this case, the attention model is a simple logistic regression model. One of the main advantages of this model is the automatic removal of the silent frames, as the attention mechanism learns to assign them a weight very close to zero. Both LLD and high-level statistics has been adopted as input features. A comparison with the same network with different pooling strategies shows that this method significantly improves the accuracy of the prediction.

In [11], the input features are 3D log-mel spectrograms, where the 3 channels are log-mel spectrograms and their deltas and delta-deltas spectrograms, namely the discrete time derivative of first and second order of log-mel spectrograms. This inputs are processed by a 2D CNN with 6 layers that extract high-level features and a bidirectional LSTM layer that deals with temporal dependencies. As in [6], the outputs of the LSTM are the frame-wise emotional features. Then, an attention layer is implemented to weight the frames according to their importance in solving the task. The attention layer produces as output the weighted sum of the frame-level emotional features, which represent the utterance-level emotional features. Finally, these features are reduced to the 4 emotional outcomes by a 2-layers fully connected structure. The system is trained separately on the IEMOCAP and EmoDB databases. The comparison with the ELM-based network analyzed in [78] and with the same structure, but with only two dimensions (i.e. without the delta and delta-delta channels) shows the effectiveness of both the 3D structure and the attention mechanism integration.

1.3 Semantic-based Synthetic Speech and Deepfake Detectors

The idea behind this work is to build a synthetic speech detector model based on semantic features. We look for high-level features which express inborn meaning and are intrinsic in the signal, in our case the speech. As we explained in the Introduction, our supposition is that deepfakes generators, and in particular TTS algorithms, are capable of synthesizing low-level characteristics of voice, but fail to recreate more complex aspects.

This technique has already been successfully experimented in both audio and video deepfake detection.

In [34], the SSD is performed using eye blinking detection. The authors observed that this semantic feature is not well presented in spoofed videos. During the preprocessing phase, the face is identified and contoured in each frame. Moreover, landmarks, which are key points in the face structure such as tips of the eyes and nose, are assigned. Then, a rectangular region containing the landmarks of the eyes is extracted. The network is eventually fed with this sequence of rectangular regions. The network is composed by a CNN layer based on VGG pretrained model for feature extraction, a LSTM layer for sequence learning and a MLP fully-connected layer to obtain the final binary prediction.

In [33], a deepfake detection framework based on both video and speech is proposed. The task is performed using emotions in the valence-arousal space as high-level, semantic descriptors for the deepfake classifier. The inputs of the system are LLDs for both video, in particular OpenFace features based on Facial Action Coding System (FACS) [97], and speech, including MFCC and Geneva Minimalistic Acoustic Parameter Set (GeMAPS) [98]. The audio-video Emotion Recognition model receives as input both video and speech LLDs time series. The model consists in a LSTM layer followed by a fully connected layer. 4 networks with this same structure are trained separately for video and speech, each outputting a time serie representing the evolution of valence or arousal for video or speech. Two final deepfake classifiers are trained and tested on audio and video separately. The first one is a statistical method based on mean, standard deviation and correlation between the two signals. The second one uses again a DL approach, the model consisting of a lightweight LSTM-based model. The experiments reach promising results on dataset DFDC [99]. The model with both classifiers based on LSTM outperforms the model based on a statistical approach.

In [35], specific modality embeddings and semantic-based emotional embeddings are learned for audio and video. The networks are equivalent for audio and video. The audio/video modality embeddings networks are composed of 2 CNN layers and a MLP. The two emotional embeddings extractors for the two modalities consists of a Memory Fusion Network (MFN), a structure based on LSTM, attention mechanism and a gated memory component [100]. The final classification is performed calculating the distance between audio and video specific modality embeddings and semantic-based emotional embeddings. If the sum of the two distances overcomes a threshold τ learned during the training, the utterance is classified as spoof.

1.4 Conclusive Remarks

In this chapter we have analyzed the main state-of-the-art works related to SER and SSD. We have then focused on a recent trend, i.e. spoofed speech detection based on semantic features, since this is the same niche in which the method we propose is collocated.

2 Theoretical Background

In this chapter, we describe the main theoretical background and tools that we use in this work. This will be useful to provide the reader with the basic knowledge to understand the next sections with the maximum profit.

2.1 The mel Spectrogram

The mel spectrogram is a representation of the frequency content of a signal as it varies with time, in which the frequencies are converted to the mel scale. The mel scale [101] is a scale of pitches perceived by listeners to be equal in distance from one another. Being the mel scale derived by psychoacoustic studies, more than one formula for Hertz-to-mel conversion was proposed. We report in equation (2.1) the most commonly adopted, which we will use later in this work.

$$m = 2595 * \log_{10}(1 + f/700) \tag{2.1}$$

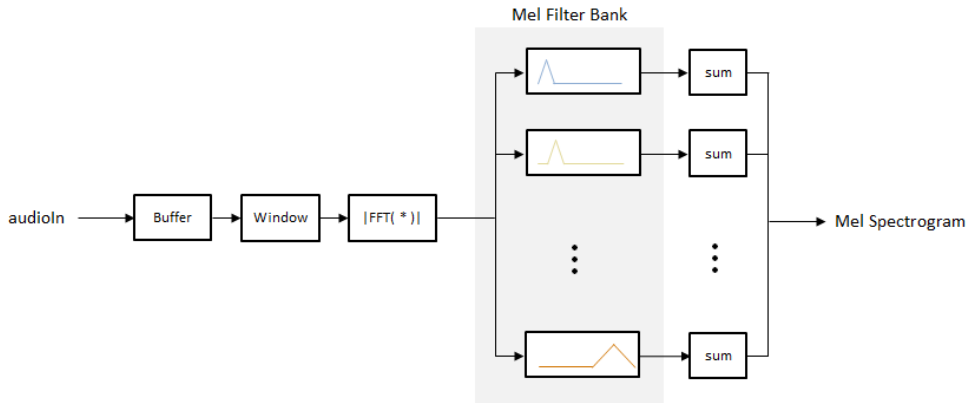


Figure 2.1: Block scheme for the computation of the mel Spectrogram. The Image is taken from [7].

Figure 2.1 shows the block scheme for the computation of the mel spectrogram. To calculate the short-time discrete Fourier transform the signal is divided into short overlapping frames, which are windowed, normally with a Hamming window, and then transformed in the frequency domain. Formally, the Discrete Fourier Transform (DFT) $X(m, \omega)$ is computed for each frame as

$$STFT\{x[n]\}(m, \omega) = X(m, \omega) = \sum_{n=-\infty}^{\infty} x[n] \cdot w[n-m] \cdot e^{-j\omega n} \tag{2.2}$$

where $x[n]$ is the speech signal and $w[n]$ is the chosen window. Each DFT representation of frame passes through a mel filterbank. The outputs of each mel filter bank are summed into a coefficient, and the coefficients are concatenated to form the mel spectrogram. Figure 2.2 shows an example of mel spectrogram.

2.2 Supervised Learning

Supervised Learning (SL) is a ML technique which provides an information system with the capability of learning a function that maps an input to an output based on examples input-output pairs [102]. More specifically, SL algorithms infer a function from labeled training data consisting of a set of training examples for which input and output are both known [103]. The inferred function can then be used to predict the output from any unseen instance of input. The precision of this inference on unseen

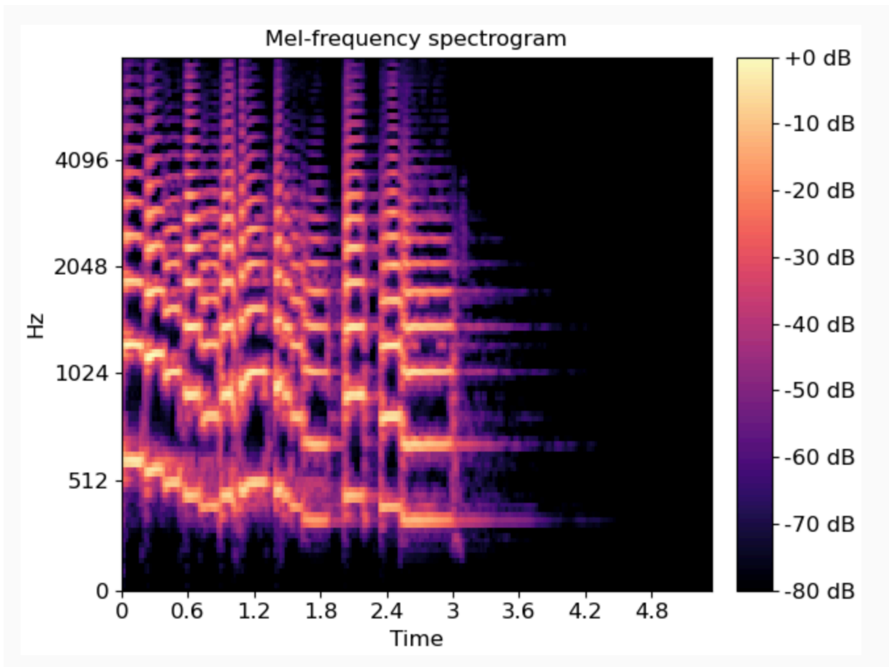


Figure 2.2: An example of mel Spectrogram.

test data, which can be described according to various metrics, measures the goodness of the trained model.

A SL task can be implemented by the means of a huge amount of different algorithms. The algorithm has to generalize from the training data, i.e. it has to extract the cues necessary to solve the task (namely perform the mapping) for unseen data. Therefore, the algorithm must avoid learning by heart the training data, as this phenomenon, called overfitting, leads to poor performances on the unseen data.

SL is divided into two main categories according to the output types of the mapping function:

- *Classification* is the task of learning qualitative categories from the input. The output of the model is defined in a discrete space [104]. Here, we formalize the problem of binary classification, which can be easily extended to the multiclass problem iterating the procedure. Given a training set of observations $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, the binary classification problem consists into estimating a function $f(x_i) = y_i$.

- *Regression* is the task of learning a quantitative outcome variable from the input. The output of the model is defined in a continu-

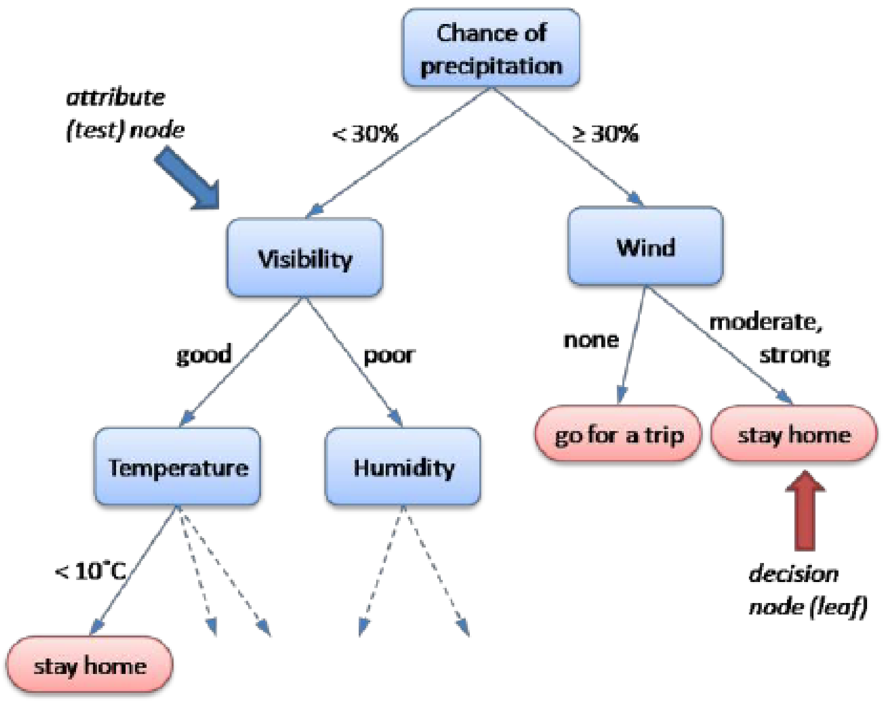


Figure 2.3: A toy example of decision tree for a trip decision based on weather conditions taken from [8]. The task can be declined as a classification problem. The decision nodes (in blue) splits the input according to a certain parameter. The leaves (in red) corresponds to the possible decisions.

ous space [104]. Formally, given a set of observations $x_1, x_2, ..., x_n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, the binary classification problem consists into estimating a function $f(x_i) = y_i$.

In this work, we focus on the classification task and, in particular, on two types of classifiers: the RF classifier and the SVM classifier.

2.2.1 Random Forest Classifier

For the sake of a better understanding of RF classifiers, we need to introduce the concept of decision tree. The decision tree classifier [105] is a categorical classifier which splits the input according to a certain parameter. In the most common case of classification, the decision tree splits the features into the subspaces composed by multiple classes. A decision tree consists of two entities, decision nodes and leaves. The decision nodes are responsible for splitting the data according to conditional decision rules, while the leaves are the final outcomes, i.e. the prediction. Figure 2.3

shows a toy example of decision tree for a classification problem.

The decision rules are inferred from training data. The training stage consists of two steps: induction, in which we build the trees setting its decision boundaries, and pruning, in which we simplify the tree structure in order to reduce complexity and, therefore, overfitting.

The RF classifier is an ensemble classifier that builds and trains a multitude of decision trees in parallel. When an input data is tested, each decision tree predicts a class, and the RF classifier selects the final outcome according to the majority vote of the decision trees [104].

2.2.2 Support Vector Machine Classifier

A SVM classifier is a supervised ML model, with associated learning algorithm, that performs linear and nonlinear classification [106, 107].

The linear Support Vector Machine (SVM) classifier discriminates between two or more classes of data by defining optimal hyperplanes in the multidimensional feature space. The support vectors are the non-outliers data points in the training set that are closest to the hyperplane and influence its position and orientation. In fact, the optimal hyperplane is defined as equidistant from the support vectors and maximizing the distance with them, since this leads exactly to a maximization of the margin between the two classes.

In some cases, the feature space is not linearly separable. This means it is not possible to efficiently divide the feature space with an hyperplane, therefore the optimal SVM classifier cannot be linear. The most common solution in these cases employs a transformation called the kernel trick. It consists in using a kernel function to transform the feature space into a higher-dimensional one, such that this new feature space is linearly separable [108]. Each dot product is transformed into an application of the nonlinear kernel function.

Formally, given a training set of observations $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, the SVM binary classifier learns a function f_x such that

$$\begin{cases} f(x_i) \geq 0 \implies y = 1 \\ f(x_i) < 0 \implies y = -1 \end{cases} \tag{2.3}$$

2.3 Deep Learning

Deep learning is a branch of ML which makes use of NNs for feature learning [109]. With feature or representation learning [110] we intend a set of techniques and models that have the objective to discover automatically the representation needed for the task (in our case, classification) from raw data.

An ANN, or simply NN, is a computational block loosely inspired by the behaviour of the human brain. It consists of a set of several artificial neurons, which correspond to the neurons of the human brain in the comparison. Neurons are linked with other neurons through connections called edges to transmit a numerical signal [111]. Neurons are organized in layer sequentially: The input flows through the first layer, whose output is the input of the second one, and so on. The edges have weights which are updated during the learning stage. In a nutshell, the latter consists in updating the weights of the network in order to minimize the error rate between the predicted and desired output. This is normally done through an algorithm called backpropagation [112], which calculates the gradient of a defined cost function with respect to the weights, and propagates the error from the outputs to the inputs with different optimization methods, the most known being Stochastic Gradient Descent. The training is normally performed on hundreds of epochs. In each epoch, all the training data pass through the network and the cost function is calculated at the output. Then, the weights are updated propagating the error back until the input layer. Normally, the training stops when the cost function does not decrease anymore for a certain amount of epochs.

Deep learning has become a trend in the past decade due to its impressive performances and many possible fields of application, such as computer vision [113], natural language processing [114] and speech

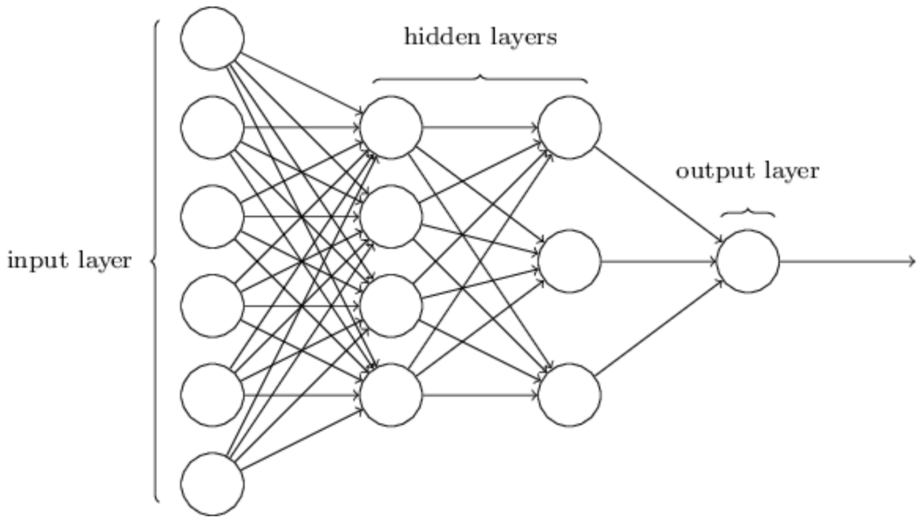


Figure 2.4: An example of Multilayer Perceptron (MLP) with two hidden layers and an output layer consisting of a single perceptron.

recognition [115] among the others. In this section, we aim to explain the main deep learning NNs and models we employ in this work, without the claim to be exhaustive.

2.3.1 Multilayer Perceptron

A Multilayer Perceptron (MLP) [116] is a class of feedforward ANN. Any MLP consists of an input layer which receives the signal to analyze, an arbitrary number of hidden layers responsible for the learning of the function from input to output, and an output layer that produces the final prediction outcome. The perceptron is the equivalent of the neuron in the generic ANN. The output signal y_j of each perceptron is the result of applying a a non-linear function, called activation function, to the sum of all the perceptron input signals. More formally,

$$y_j = g_j\left(\sum_{n=1}^{N} w_{ij} \cdot x_i\right) \tag{2.4}$$

where g_j is the activation function of the target perceptron j, x_i are the inputs of the perceptron and w_{ij} is the learning weight corresponding to the input x_i.

MLP are normally fully-connected, i.e. the perceptrons of a layer are connected to all the perceptrons of both the previous and next layer.

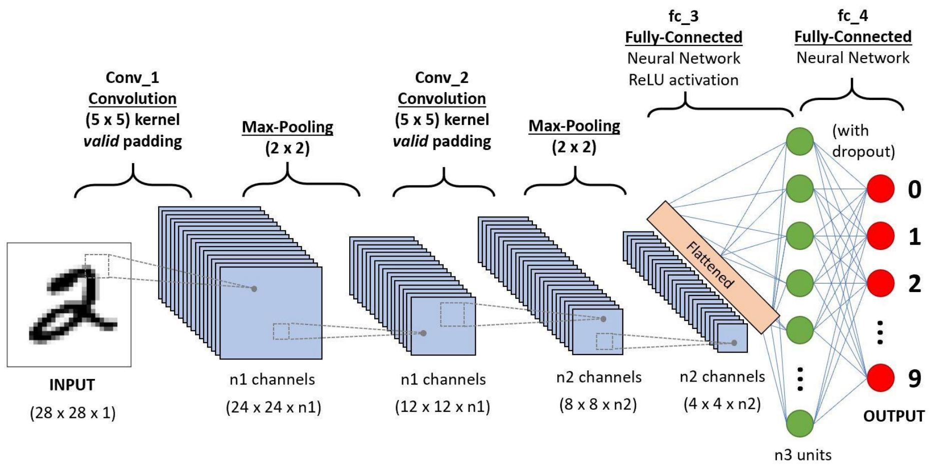


Figure 2.5: An example of Convolutional Neural Network (CNN) for image processing. Figure is courtesy of [9].

All connections between the neuron are weighted. The model learns the weights at the training stage from the error on the predicted output, through backpropagation [117] or similar algorithms. In a nutshell, the edges between neurons create paths between the input and the output layers. These paths are responsible for carrying information which helps learning the desired output from the inputs. At the end of the learning, the paths which contains information useful to solve the task are composed by edges with significant weights, while the other paths are composed by edges with weights around zero.

Figure 2.4 shows an example of MLP with two hidden layers and an a single perceptron as output.

2.3.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) [118, 119] is a type of deep NN, often employed to solve tasks related to images and multimedia signals in general. CNNs are inspired by the behaviour of the visual cortex. Each neuron of the CNN operates with a limited scope, the receptive field, exactly as each neuron in the animal brain operates in a restricted region of the visual field [120]. As in the case of MLP, a CNN is composed of an input layer, a variable number of hidden layers and an output layer.

Each hidden layer is normally composed by a convolutional layer, a non-linearity layer or activation function and an optional pooling layer [118]. The convolutional layer performs the convolution of the input with a series of filters which contain the weights (and biases) that are learned during the training phase. The filters are shared on the entire layer, i.e. on all the visual field. The result of each convolution is a feature map. In the case of 2D input, i.e. in the image processing case among the others, the output feature map y of a convolutional layer is computed as

$$y(t,f) = (x * w)[t,f] = \sum_{m}\sum_{n} x[m,n] \cdot w[t-m, f-n] \tag{2.5}$$

where x is the input image or 2D input feature map, w is the kernel or filter matrix and $*$ is the operation of convolution.

This feature map y passes through the non-linear activation function of the layer, often a Rectified Linear Unit (ReLU). This function is designed to break the linearity of the system allowing the generalization of more complex tasks. Eventually, a pooling layer may decrease the dimension of the feature map in order to reduce the complexity. The pooling layer combines the outputs of near neurons' clusters into a single output, normally through averaging the output or selecting the maximum.

The hidden layers usually represents hierarchical levels of abstraction from input to output [118]. If we consider the image processing task, the first layers are able to detect simple patterns in the receptive field like lines, curves or edges. Going gradually further from the input, the network level of abstraction grows. Thus, the last layers have the ability to detect more complex pattern, like faces, objects or words, in all the visual field.

The output structure normally consists of a fully-connected MLP [119]. This last block aims at learning the mapping function from the refined representation provided by the last convolutional hidden layer to the final outcome prediction.

Figure 2.5 shows an example of CNN for an image processing task.

2.3.3 Recurrent Neural Networks and Long short-term memory networks

The Recurrent Neural Network (RNN) [121] is a broad class of NN that is designed to deal with temporal sequences. Some of the main application areas of RNNs are natural language processing and text classification [122], machine translation [123] and speech recognition [124].

A RNN is composed of an ANN, normally a MLP. Figure 2.6 shows the basic scheme of a RNN. The RNN takes as input not only the current input instance, but also a background information preserving the description of past inputs instances, called hidden state [121]. In other words, the output of some of the hidden layers is saved into the hidden state and recursively fed back as input for the successive input instance.

The success of this NN structure is due to its capacity to model a function based both on past and present. However, a main issue arises with the training of this type of network. In fact, in the training phase, the cost function has to flow from output to input to update the weights (backpropagation). Therefore, in the RNN case, the error must flow not only back to the input layer, but also back in time, since part of the input is coming from the hidden state. The weights are normally initialized to random values in the proximity of zero. This means the gradient flowing back has an exponential decay, being multiplied for values near to zero at every step back in time. Therefore, the update of the weights shortly becomes negligible and the network becomes consequently very hard to train. The problem is exacerbated by the fact that RNNs normally have many time steps in order to deal with long-term temporal dependencies. This issue is known as the vanishing gradient problem [125].

LSTM [126] is a type of RNN which is frequently chosen in order to deal with the vanishing gradient problem. The LSTM network decouples the cell state and the output, basically allowing the gradient to flow back unchanged, since the cell state allows information to just flow through it substantially unmodified. In this way, the vanishing gradient problem is substantially solved [127]. This makes LSTM the preferred choice between the RNN models, especially when dealing with long-term

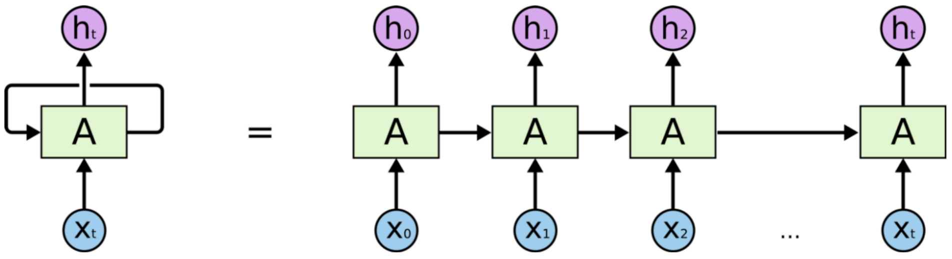


Figure 2.6: Basic scheme of the Recurrent Neural Network (RNN) (left) and of the same RNN unrolled in time (right). The input sequence x_t is sent to the network sequentially. The network block A produces the output h_t, which depends not only on the input, but also on the state of the network block A at time t. The state information transmission is represented by the arrows going from A at time step t to A at time step $t + 1$. The figure is courtesy of [10].

dependencies in time sequences.

2.3.4 Attention layers

In the context of deep learning, attention is a set of methods designed with the purpose of reproducing human cognitive attention [94, 95]. In fact, when the input data of a NN consist of a lot of information, as in the case of multimedia, the need of focusing just on the relevant information arises. For example, in the case of emotion recognition, not all frames are equivalently relevant to perform the task: therefore, it is possible to insert in the model an attention layer. The latter implements an attention mechanism with the objective to weight the frames according to their effective importance in solving the task and then to aggregate the weighted frame-level features vectors in a single utterance-level context vector. In the simple case of additive self-attention, a trainable vector learns and decides which frames are more and which are less relevant with respect to the task, and generates the normalized importance weights vector from the input vectors itself [6]. The utterance-level context vector is then computed as the weighted sum of input vectors, where the weights are calculated as explained.

2.3.5 Transfer Learning

Transfer Learning (TF) [86] is a ML technique that recycles knowledge learnt in solving a ML task by applying it to solve another task with some degree of similarity with the former. The usage of Transfer Learning (TF) provides two possible overall advantages:

- It can improve the learning speed, both by providing a higher performance starting point and/or by facilitating a faster improvement of performances in the training phase.
- It can improve the final learning performance, i.e. allowing to reach better performance at the model convergence.

We can divide TL into two main categories, according to the used approach:

- It is possible to select a task related to the desired task and to built a custom model for this related task in order to use it as a feature extractor for the desired task.
- It is possible to select an existing model pretrained on a similar task as a feature extractor for the desired task. The pretrained model it is often chosen among powerful existing model developed by well-known research institution, as in the example Oxford VGG [128] and Alexnet [89]. In fact, these model are trained with huge amount of resources and are able to provide extremely powerful generalization.

In both cases, the pretrained model may be optionally fine-tuned with very low learning rate. Finally, the model is employed as a starting point for the classification task. This can be done according to different techniques, the most common being embeddings extraction. By the way, this techniques happens is exactly the one we use in this work. This method consists of extracting embeddings from a chosen intermediate layer of the neural network model used for the similar task (no matter if pretrained or built and trained exactly for the purpose), and use these embeddings as input features for the desired task.

2.4 Conclusive Remarks

In this section, we have explained the main means and techniques that will be used later in this work. In particular, we explained the main preprocessing we adopted for our input features, and the main ML and DL techniques and structure adopted to build our SER and SSD models. Some of these techniques and tools are quite complex and, therefore, it is not easy to provide an exhaustive background. The interested reader is invited to refer to the cited papers whether he/she wants to deepen the knowledge in the field.

3 Proposed System

In this chapter, we formulate the problem we are focusing on in this work, i.e. to detect whether a speech recording belongs to a real person or it has been synthetically generated by means of some synthesis technique. Then, we propose a possible solution to this problem, consisting in a method based on semantic, high-level features that differ between the two classes of speech signals.

3.1 Problem formulation

Given a target speech audio signal x under analysis, the objective of this thesis is to estimate its class

$$y \in \{\mathrm{BF}, \mathrm{SPOOF}\}, \tag{3.1}$$

where BF indicates that the speech signal is authentic, *bonafide*, whereas SPOOF corresponds to synthetic *spoofed* audio tracks. We consider as bonafide those recordings that have been generated by using a micro-

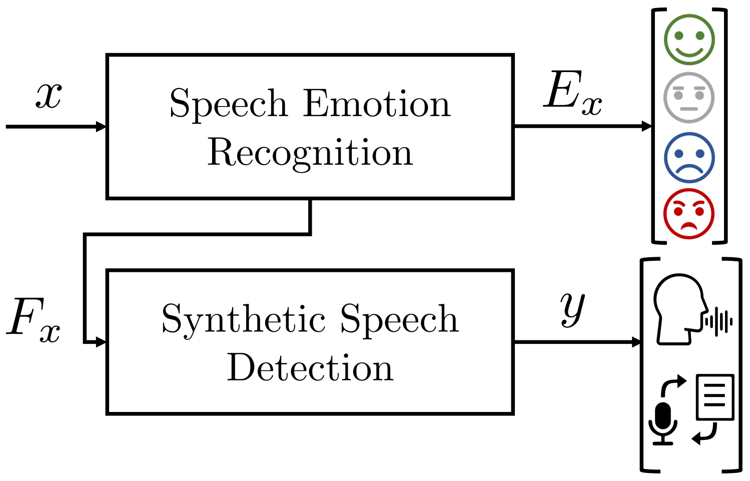


Figure 3.1: Architecture of the proposed system. A given speech signal x is processed by the SER system to predict its emotional class E_x. At the output of the SER attention layer, we extract an emotional feature vector F_x and use it as input for the Synthetic Speech Detector, which determines the signal class y.

phone while a real human is speaking. These tracks may undergo some global audio processing operations (e.g. compression, normalization, etc.) but have not been altered in any malicious way. We consider as spoofed the tracks that have been generated by the means of some synthesis technique. As explained in Section 1.1, the main techniques used for spoofing are Text-to-speech (TTS) and Voice Conversion (VC). In this work, we focus on the TTS spoofing algorithms, which generates synthetic speech directly from text.

It is important to further underline the generality of the objective. We want to find a method based on semantic characteristics of the speech signal, which have to be intrinsically different between the two classes. Therefore, the method aims to be sufficiently powerful to provide a correct estimation of the class not only in the case of known TTS synthesis algorithms, but for each type of unseen TTS algorithm.

3.2 System architecture

The pipeline of the system architecture is reported in Figure 3.1. The proposed method is composed of two main blocks. The first block is the

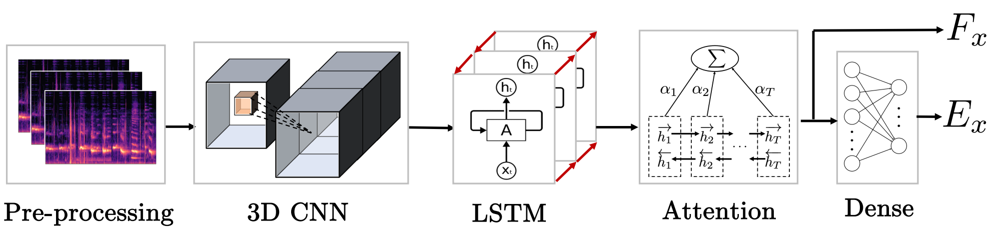


Figure 3.2: Emotional feature extraction architecture, as proposed in [11]. Here we show where we extract the set of emotional features F_x.

Speech Emotion Recognition (SER) system exploiting the architecture recently proposed in [11]. It estimates the speech emotion E_x and extracts a set of features F_x from the input signal x. The second block is the Synthetic Speech Detection (SSD) system that associates a class y to the input features F_x. In the following, we provide further details about each block.

3.2.1 Speech Emotion Recognition

The first part of the proposed pipeline extracts a set of features F_x able to express the emotional content of the speech audio signal x under analysis. Motivated by the state-of-the-art performances provided by deep-learning methods, we decided to explore data-driven neural networks for the purpose, rather than using hand-crafted feature extraction.

The considered emotional features are computed making use of the 3D-Convolutional Recurrent Neural Network (CRNN) proposed in [11], whose architecture is shown in Figure 3.2. Anyhow, we describe the latter in details later in this section. The authors address the problem of speech emotion recognition as a classification problem using a categorical approach, i.e. N possible emotion classes are considered [129]. Therefore, given a speech utterance x, the output of the network is

$$E_x \in \{e_1, e_2, ..., e_N\}, \tag{3.2}$$

where e_i is the i-th emotion class (e.g., happy, sad, angry, etc.).

As reported in [11], the input signal x must be preprocessed before being fed to the following neural network. We do so by computing the spectrum of x through a Short-time Fourier Transform (STFT), calculated according to equation (2.2). The STFT is then converted in the mel-frequency domain by means of a mel filter bank, as explained in 2.1, and then a logarithmic transform is applied to the STFT magnitude. This returns a log-mel spectrogram defined as

$$\mathbf{S}_{\text{mel}} \in \mathbb{R}^{M \times K}, \tag{3.3}$$

where M is the number of windows in the time domain and K is the number of mel bins. Figure 3.3a shows an example of log-mel spectrogram computed on a speech signal.

Then, we compute the first and second discrete derivatives of $\mathbf{S}_{\text{mel}}$ along the time dimension, obtaining $\Delta\mathbf{S}_{\text{mel}}$ and $\Delta\Delta\mathbf{S}_{\text{mel}}$. Figure 3.3b and figure 3.3c shows examples of Δ and $\Delta\Delta$ log-mel spectrogram respectively. The $\Delta\mathbf{S}_{\text{mel}}$ and $\Delta\Delta\mathbf{S}_{\text{mel}}$ are computed on the same speech signal of Figure 3.3a.

By stacking the log-mel spectrogram and its derivatives along a third dimension, we obtain the final 3D matrix $\mathbf{X}$ defined as

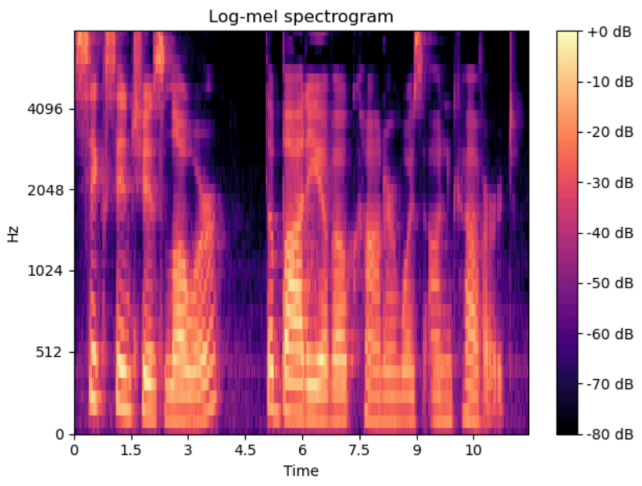
$$\mathbf{X} = [\mathbf{S}_{\text{mel}}, \Delta\mathbf{S}_{\text{mel}}, \Delta\Delta\mathbf{S}_{\text{mel}}] \in \mathbb{R}^{M \times K \times 3}. \tag{3.4}$$

This matrix is then standardized by means of z-score normalization. For example, the log-mel spectrogram $\mathbf{S}_{\text{mel}}$ becomes

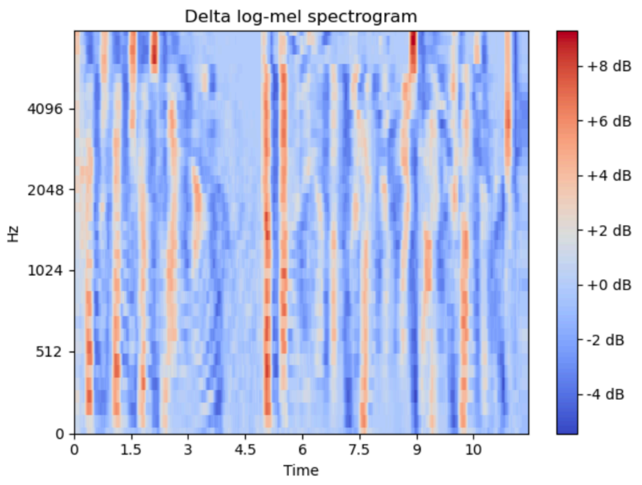
$$\mathbf{S}_{\text{mel,std}} = \frac{\mathbf{S}_{\text{mel}} - \mu}{\sigma + \epsilon} \tag{3.5}$$

where $\mathbf{S}_{\text{mel,std}}$ is the standardized log-mel spectrogram, $\mathbf{S}_{\text{mel}}$ is the input log-mel spectrogram, μ and σ are the average and standard deviation calculated on all log-mel spectrograms of the SER classifier's training set, and $\epsilon = e^{-5}$ is a constant to prevent divisions by 0. The same transformation is performed also on $\Delta\mathbf{S}_{\text{mel}}$ and $\Delta\Delta\mathbf{S}_{\text{mel}}$.

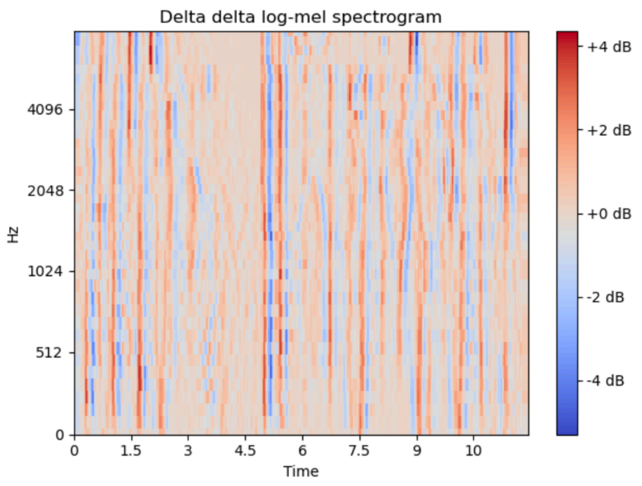
The standardized 3D log-mels is then processed by the NN structure described in [11]. First, the input is processed by a series of 6 convolutional layers. The first has $n_{\text{cnn1}} = 128$ feature maps, while the remaining have $n_{\text{cnn}} = 256$ feature maps. The filters or kernels have size of $k_{\text{size}} = 5 \times 3$, the dimensions corresponding to the time axis and to the frequency axis respectively. The output of each layer is processed by an activation function consisting in a Leaky ReLU function with negative slope $\alpha_{\text{relu}} = 0.01$. A single pooling layer with size $p_{\text{size}} = 2 \times 2$ and stride



(a) An example of log-mel spectrogram $\mathbf{S}_{\mathrm{mel}}$ of a speech signal.



(b) An example of Δ log-mel spectrogram $\Delta\mathbf{S}_{\mathrm{mel}}$ of a speech signal.



(c) An example of $\Delta\Delta$ log-mel spectrogram $\Delta\Delta\mathbf{S}_{\mathrm{mel}}$ of a speech signal.

$p_{\text{stride}} = 2 \times 2$ is set between the first and second convolutional layers in order to reduce the number of parameter upstream of the other layers.

Then, a linear time-distributed layer processes the output of the last convolutional layer in order to reduce the number of parameters to $n_{\text{lin}} = 768$ for each time step, and, therefore, the computational cost. The operation can be performed without loss of information, as demonstrated in [130].

The output of the linear layer is fed into a bidirectional Long Short-term Memory (LSTM) layer to catch the long-term dependencies in the time sequence. Since the LSTM contains $n_{\text{lstm}} = 128$ output cells, we obtain a 256-dimension representation h_t at each time step t by concatenating the two outputs generated by the LSTM forward and backward unrolling in time.

At this point, the output of the LSTM layer h is fed into an attention layer. The attention layer is implemented in order to produce discriminative utterance-level features from window-level features. In particular, we calculate the normalized importance weight α_t of each time step as

$$\alpha_t = \frac{\exp(W \cdot h_t)}{\sum_{t=1}^{T} \exp(W \cdot h_t)} \tag{3.6}$$

The matrix W is learned during the training phase. The eventual output of the attention layer is computed as a weighted sum on the LSTM outputs h_t according to the weights α_t, namely

$$c = \sum_{t=1}^{T} \exp(\alpha_t \cdot h_t) \tag{3.7}$$

The final output c consists of a unique vector with dimension $n_{\text{att}} = 256$ containing the *utterance-level emotional representation.* This representation is computed aggregating the frame-level emotional representations of all the frames in the utterance.

Finally, the reduction from the utterance-level feature vector c to the emotional outcomes is performed by two fully-connected layers. The first fully-connected MLP has $n_{\text{fc}} = 64$ output units and ReLU activation function. The second and last fully-connected layer is a simple softmax layer with $n_{\text{out}} = 4$ output units, corresponding to the probabilities of the classes angry, happy, sad and neutral. Looking back at Equation (3.2), this means we choose $N = n_{\text{out}} = 4$ as number of emotion classes and the described softmax output as E_x.

Adopting a transfer-learning strategy, we extract the feature vector F_x of dimensionality M from an intermediate network layer. More formally, we can express the feature extraction block as a function $\mathcal{F}_{\text{ser}}$ such that

$$F_x = \mathcal{F}_{\text{ser}}(x), \quad F_x \in \mathbb{R}^M \tag{3.8}$$

In particular, as it is shown in Figure 3.2, we consider the output of the final attention layer, i.e. we select $F_x = c$ and $M = n_{att} = 256$, which the authors present as the *utterance-level emotional representation*, as feature vector for our final SSD model. As we will show, the only exception to this choice is represented by the experiment described in Section 4.5.4.

The selected feature vector does not simply have good discriminative power for its original task (i.e., estimating the *quality* of the emotion) but also for the synthetic speech detection task (i.e., estimating the *intensity/quantity* of the emotion expressed). This is because TTS deepfake algorithms reach excellent results in terms of speech naturalness, but still fail in modeling the emotional properties of the human voice correctly. We can therefore exploit this weakness together with neural networks' ability to create powerful and flexible embeddings, using F_x as input of a classifier trained for deepfake detection.

3.2.2 Synthetic Speech Detection

In the second part of the proposed pipeline, the features F_x are first rescaled and then fed to a binary classifier in order to finally estimates the class y to which the input signal x belongs. The scaling consists again of a z-score normalization (as in Equation (3.5)), but this time average and standard deviation of F_x are calculated on the whole training set. It is worth noticing that the proposed work does not necessarily fix the classification algorithm, hence any supervised classification method can be chosen. As shall be evident from Chapters 4 and 5, the classifier can be as simple as a random forest, still achieving promising results. Formally, the SSD final classifier has the objective to find the mapping function $\mathcal{F}_{\text{ssd}}$ such that

$$y = \mathcal{F}_{\text{ssd}}(F_x), \quad y \in \{\text{BF}, \text{SPOOF}\} \tag{3.9}$$

3.3 Conclusive Remarks

In this chapter, we have formally addressed the problem tackled in this thesis, also reporting the proposed scheme for its solution. In the continuation of this work, we provide further details about our choices, defining the evaluation setup, the implementation details of the system we adopted and then showing the results of our experiments.

We remark that the method is focused on the detection of TTS and mixture TTS/VC deepfakes, while does not take into account pure VC algorithms. This is because we exploit speech semantic information to detect anomalies and pure VC fakes do include such content, being generated from a real voice and then altered only with style transfer techniques.

4
Evaluation Setup

In this chapter, we provide all the details related to our experimental campaign aimed at validating the method proposed in Section 3. We report information about the used datasets, the training policies and the evaluation metrics. Moreover, we clearly introduce all the performed experiments, describing each setup in details.

4.1 Evaluation Datasets

In this section, we present the datasets that have been jointly used for the training and evaluation stages of both the SER and deepfake detection methods. The considered datasets include both real and synthetic speech samples, for a total amount of 123 hours of audio recordings effectively employed. The choice of using multiple datasets prevents overfitting and allows to test the robustness of the proposed system in real-world conditions.

- **ASVspoof 2019** [131] is the main source of data for this work. It is a speech audio dataset created to develop antispoofing techniques for Automatic Speaker Verification (ASV). It contains short tracks of bonafide and spoofed speech data.

The dataset is split in two different use case scenarios: the Logical Access (LA) and the Physical Access (PA). The ASVspoof LA partition is designed against attacks that inject the signals directly into the system; this partition contains spoofed data coming from 17 different TTS and VC algorithms, comprehending both traditional and DL-based synthesis techniques. The PA partition, instead, is a collection of bonafide tracks recorded and reproduced, and it is the source of data designed against replay attacks.

Given the task at hand, we consider the LA partition, which is further divided in *train*, *dev* and *eval* subsets. As already mentioned in Chapter 3, since the proposed system exploits emotional content for discrimination, we select only samples generated using TTS or TTS/VC hybrid methods. Therefore, our *train* and *dev* partitions include speech samples generated with 4 different algorithms (named A01, A02, A03, A04), leading to a total of 35192 tracks, of which 5128 are bonafide and the remaining are equally distributed between the spoofing algorithms. In *eval* partition samples from 10 algorithms (A07, ..., A16) are kept, for a total of 56495 tracks, of which 7355 are bonafide and the remaining are equally distributed between the spoofing algorithms. We remark that all algorithms present in the *eval* set are not seen in neither *train* nor *dev* set.

- **LibriSpeech** [132] is an open-source dataset containing about 1000 hours of bonafide speech. From this corpus we considered the subset *train-clean-100*, which counts 28539 speech track for about 100 hours of recording. These tracks are coming from speakers with low error rate on automatic transcription, thus are easily intelligible. We decided to use this dataset to increase the number of bonafide speech tracks in the training phase, thus giving the model a strong boost in the generalization power of real speech.

- **LJ Speech** [133] is an open-source speech dataset containing 13100 audio clips of length variable between 1 and 10 seconds. The source is a single speaker reciting pieces from non-fiction books. We used this database to have solid amount of additional bonafide data in the *eval* set coming from a dataset distinct from the ones used in the training phase.

- **Cloud2019** corresponds to the dataset proposed in [27]. It includes 11785 tracks from different TTS modern cloud services: Amazon AWS Polly (PO), Google Cloud Standard (GS), Google Cloud WaveNet (GW), Microsoft Azure (AZ) and IBM Watson (WA). Different speaker profiles have been used for the generation. This dataset provides additional in-the-wild synthetic data to the *eval* set, since it is not used in the training phase and, moreover, the TTS generation algorithms are never seen in the training phase too.
- **Interactive Emotional Dyadic Motion Capture (IEMOCAP)** [82] contains approximately 12 hours of recordings, including video, speech and motion capture of faces, all annotated with the expressed emotions. Speech tracks are segments of scripted or improvised dialogues performed by actors emphasizing a particular emotion. We used the IEMOCAP dataset in order to train the SER model described in Chapter 3. From the improvised dialogues subset, which is the one considered in [11], we analyzed all the tracks labeled with the four classes of our SER system (i.e., angry, happy, sad, neutral) for a total of 2280 tracks. Moreover, we used the same partition of the dataset to provide bonus bonafide data for the evaluation stage of the deepfake classifier.

4.2 Training parameters

Both feature extraction and classification stages are data-driven, hence a training stage is required. Please notice that the two components are trained separately and on different training sets.

We have trained the Speech Emotion Recognition (SER) system following the specifics proposed in [11]. This means that, for the SER

Table 4.1: Composition of train, development and test set for the training stage of synthetic speech detector block

	Bonafide	**Spoofed**	**TOT**
Train	ASVspoof2019 *train* LibriSpeech	ASVspoof2019 *train*	46319
Dev	ASVspoof2019 *dev*	ASVspoof2019 *dev*	17412
Test	ASVspoof2019 *eval* IEMOCAP LJSpeech	ASVspoof2019 *eval* Cloud2019	83660

training alone, no other preprocessing has been performed to the tracks except the log-mels and deltas calculation and z-score standardization. Specifically, we have used the improvised dialogues from the IEMOCAP dataset and we have considered the classes *angry*, *happy*, *sad* and *neutral*, hence $N = 4$. Since the IEMOCAP dataset is divided in 5 dialogue sessions, we have selected sessions 1 to 4 for training and session 5 for development and testing. We have used Adam optimizer with learning rate $l_r = 10^{-5}$ and categorical cross-entropy as loss function. Since in our system this network acts as feature extractor, we are not going to present the results relative to the speech emotion estimation task. Nonetheless, this training stage have led to a recall consistent with the results shown in [11].

The dimension of the feature vector extracted from the SER system F_x is $M = 256$, corresponding to the feature dimension of the attention layer output. The only exception is the experiment illustrated in details in Section 4.5.4. The features F_x are then standardized and fed into the final Synthetic Speech Detection (SSD) supervised classifier.

Regarding the SSD main experiment, that we describe in details in Section 4.5.1, we have trained two different SSD supervised classifiers, based linear SVM and RF algorithms respectively. As we show in Chapter 5, the RF classifier performs better than the SVM. For this reason, we have selected the RF classifier to perform the other experiments, i.e. the ones with Voice Activity Detector (VAD) (Section 4.5.2), noise injection (Section 4.5.3) and concatenation of emotional outcomes in time (Section 4.5.4).

The weights for the SER network are fixed during all phases (SER training excluded) and the network is used only to extract the feature vector F_x from the input x. In other words, the SER model has been trained apart before all the other steps, and then treated as a pretrained model in a transfer learning fashion [86].

The composition of training, development and test dataset for the SSD is presented in Table 4.1. The train set has been balanced adjusting the learning weights associated with classes such that these are inversely proportional to the number of samples per class in the training set itself.

The hyperparameters for both the SSD classifiers have been selected using a grid search on the validation set, using balanced accuracy as a metric. For the RF, the considered parameters are the criterion of split quality and the number of learners. In particular, we tested as quality criterion function both Gini impurity and information gain. Regarding the number of learners, we consider $N_{\text{RF}} = [10, 30, 100, 300]$. For the SVM, we have considered different values of the kernel regularization parameter $C_{\text{SVM}} = [0.1, 0.3, 1, 3]$ with a linear kernel.

4.3 Baseline

As baseline for this work, we use a popular CNN architecture named VGGish [134]. This architecture has been firstly proposed for audio event classification and trained on an extensive dataset of audio tracks [135]. Due to its generalization capacity, it has often been used as an embedding extractor for other audio analysis tasks.

In our case, we add a final fully-connected layer to the standard VG-Gish embedding extractor architecture and we fine-tune it in order to learn the task at hand, i.e., synthetic speech detection, using binary cross-entropy as loss function. The training set used for the baseline is the same used for the proposed method.

4.4 Metrics

To evaluate the performances of the proposed binary classifier and the goodness of the performed experiment, we adopt different metrics.

The first metric is the balanced accuracy. The balanced accuracy is a metric often used in performance evaluation of binary classifiers. To define the concept of balanced accuracy, we need to introduce the concepts of True Positive Rate (TPR) and True Negative Rate (TNR). The TPR, or sensitivity, or recall is defined as the ratio between the true positive and the total amount of positive samples, i.e.

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

where TP are the true positives and FN are the false negatives. The TNR, or specificity, is defined as the ratio between the true negatives and the total amount of negative samples, i.e.

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

where TN are the true negatives and FP are the false positives. We will also consider TPR and TNR as stand-alone evaluation metrics when analyzing single bonafide speech sets or spoofing algorithms.

Now we can define the balanced accuracy as the average between the TPR and TNR, namely

$$\mathrm{BA} = \frac{\mathrm{TPR} + \mathrm{TNR}}{2} \tag{4.3}$$

For the sake of clarity, we here remark that we consider *bonafide* speech as the negative class and *spoof* speech as the positive class. We adopt the balanced accuracy metric when we need a single scalar value in order to concisely evaluate the performance of the system. We remark that, when we compute the balanced accuracy on multiple spoofing algorithms and bonafide source datasets, we first compute the overall TPR as the average between the TPRs of the single algorithms and the overall TNR as the average between the TNRs of the single real speech datasets. The final balanced accuracy is normally computed as by Equation (4.3).

Another metric we will adopt consists in the confusion matrix. In the context of binary classification, the confusion matrix is a table with two rows and two columns that are filled with the number of false positives, false negatives, true positives, and true negatives. The structure we adopt for the confusion matrix is

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

We remark that, when we compute the confusion matrix on many spoofing algorithms and bonafide source datasets, we aggregate all the observations and calculate the confusion matrix based on this aggregation. For this reason, algorithms with different numbers of observation have a different weight on the final result. The confusion matrix allow us to monitor the performance of the system more precisely with respect to balanced accuracy with 4 scalar numbers.

The last metrics we will adopt is the ROC curve [136]. The ROC curve is created by plotting the TPR against the False Positive Rate (FPR) at various probability threshold settings. The FPR is defined as the ratio between the false positive and the total amount of negative samples, i.e.

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

We remark that, when we compute the ROC curve on many spoofing algorithms and bonafide source datasets, we aggregate all the observations and plot the ROC curve based on this aggregation. Therefore, algorithms with different numbers of observation have a different weight on the final result. An important scalar parameter related to the ROC curve is the Area Under Curve (AUC), which measure the aggregate performance of a classifier considering all the possible probability thresholds. An AUC equal to 0.5 corresponds to a random guess, whereas an AUC equal to one corresponds to the perfect performance. The ROC curve is a standard metric for evaluating the performance of a binary classifier, and it is capable of describing extensively the performances of the model. Furthermore, it provides an immediate graphical comparison between different evaluation setups.

4.5 Experiments

In this section, we describe the 4 main sets of experiments we performed in order to robustly evaluate the proposed system.

4.5.1 Experiment I: Standard input preprocessing and transformation

The first and main set of experiments, which constitutes the standard framework we propose, consists of preprocessing each track that we will shortly explain in details, obtaining the log-mel spectrograms. Then, the log-mels are fed into the proposed system illustrated in Chapter 3. The final SSD system consists of a RF or SVM classifier.

In this section, we describe the standard setup and preprocessing we apply to each of the input tracks. We preprocess all tracks to make them as uniform as possible in order to avoid detecting dataset-specific artifacts.

All tracks are converted to mono standard Waveform Audio File Format (WAV) format and, if necessary, downsampled at the chosen standard sampling frequency $F_s = 16$ kHz.

Then, we filter all speech signals using a Butterworth band-pass digital filter with order 6, lowcut frequency $F_l = 250$ Hz and highcut frequency $F_h = 3.6$ kHz. Then, we normalize each track dividing all samples by the infinity norm calculated on the single track itself.

It is worth noticing that the filtering and normalization are applied to all but the IEMOCAP tracks used for the training phase of the SER model, since we comply precisely with the preprocessing steps explained in [11].

The input of the Speech Emotion Recognition block is computed starting from a time-frequency transform, as detailed in [11]. Each track of the datasets is reduced to have a common length $L_{\text{cut}} = 3$ s. In particular, longer tracks are cut and shorter tracks are zero-padded to reach this exact length. Then, we compute the STFT of x using a Hamming windows of length $L_w = 25$ ms and a hop-size $L_h = 10$ ms. Only the magnitude of the STFT is considered. The spectrum is then processed using a bank of mel-spaced filters, as already mentioned in the previous section, and further scaled using the natural logarithm function. In our implementation we consider $N_w = 300$ windows and $K = 40$ mel bins. Thus, we have obtained the log-mel spectrogram $\mathbf{S}_{\text{mel}}$. Then, as detailed in section 3, the first and second discrete derivatives of $\mathbf{S}_{\text{mel}}$ in the time dimension, namely $\Delta\mathbf{S}_{\text{mel}}$ and $\Delta\Delta\mathbf{S}_{\text{mel}}$, are computed and then stacked together as shown by Equation (3.4).

The log-mel spectrogram and its derivatives are finally standardized with the mean and standard deviation calculated on the train set of the SER system, as shown by Equation (3.5).

The operations described in this section constitutes the standard pre-processing of the input tracks for the proposed system. Therefore, this means that these operations are applied to all tracks in every experiment that we perform.

4.5.2 Experiment II: Voice Activity Detector

In order to verify the effectiveness of the SER model in providing vocal emotional-based features, we perform a second set of experiments employing a Voice Activity Detector (VAD). The VAD is a digital signal processing system with the function to detect the presence or absence of human speech. We use the VAD implemented in [137, 138]. The voice activity is detected on short windows of $L_{\text{vad}} = 10$ ms length. For each window, the VAD outputs a binary outcome, consisting in the presence or absence of speech signal. An important parameter of the VAD is the aggressiveness of the filtering, namely an integer in the range $A_{\text{vad}} = [0 - 3]$ that indicates how much assertive the VAD is in detecting non-speech signal [138]. In other words, the greater the A_{vad}, the more the number of short frames that are classified as non containing vocal activity.

We performed two different experiments involving the VAD:

- The first experiment, which does not modify the single data, consists of filtering out from all sets the data with total voice activity $V_{\text{vad}} \leq 80\%$. The voice activity percentage is calculated as the percentage of utterance's short windows for which the VAD gives a positive outcome, i.e., as we have described above, the frame is predicted to contain vocal activity. The tracks shorter than $t = 2.7$ seconds are filtered out too, in order to avoid low voice activity due to massive zero-padding. The aggressiveness is set to $A_{\text{vad}} = 1$ for this experiment. We repeated both training and evaluation with this subset of data.

- The second experiment, instead, detects the short frames without voice activity and substitutes these frames with an equivalent amount of zero-padding samples. The aggressiveness is set to $A_{\text{vad}} = 1$ and $A_{\text{vad}} = 3$ (the maximum). We also perform the complementary of this experiment, i.e. the short frames with voice activity are substituted with zero-padding and just the silent frames are preserved. The aggressiveness is again set to $A_{\text{vad}} = 1$ and $A_{\text{vad}} = 3$ (the maximum). The main objective of this experiment is to verify if and how much the background and preprocessing noises influence the classification unwantedly. Therefore, we do not expect an improvement of results with respect to Experiment I. Instead, obviously, we expect the experiment replacing the silent frames with zeros to reach better performances than the experiment replacing the vocal frames with zeros.

4.5.3 Experiment III: Dataset augmentation with noise injection

In order to test the robustness against possible audio degradation, a third version of the dataset is created using data augmentation techniques. To do so, we add Gaussian white noise to the speech tracks considering two different approaches. In both cases, the noise is added before the standard preprocessing filtering and normalization.

For the train and validation sets, we perform noise injection according to a double-layer probability distribution. The first layer injects the Gaussian white noise randomly between 30 dB and 15 dB of power SNR with probability $p_1 = 0.8$. The second layer randomly injects white noise between 15 dB and 10 dB of power SNR, with probability $p_2 = 0.3$.

For the test set, instead, the power SNR takes the fixed values of

SNR = $[25, 20, 15, 10]$ dB. In other words, training data contain a wide variety of noise in order to generalize on a wide range of different noise levels, whereas test data are obtained in a controlled scenario to enable results analysis.

4.5.4 Experiment IV: Emotional outcome in time

We perform an additional experiment in order to verify the effectiveness of synthetic speech detection using directly the outputs of the SER network, i.e. E_x according to the notation introduced in Chapter 3, and their evolution in time. Indeed, we want to better understand the level of compatibility between the SER and SSD task.

In order to conduce this experiment, we filter out from all sets the speech tracks shorter than 8 seconds. In this experiment alone, the features extracted from the SER system are not the output of the attention layer as explained in Section 3, but the emotional outcomes extracted in time. In particular, the input feature of the SSD system is calculated as follow:

- A sliding rectangular window of 3 seconds length and 0.5 seconds hop-size is applied to the signal.
- For each 3 seconds length windowed signal obtained, the SER system predicts $E_x(t)$, which is the emotional outcome calculated on the window going from time t to time $t + 3$ seconds.
- The $E_x(t)$ outcomes calculated as above are concatenated in a linear vector. For this experiment, all tracks are cut at $L_{\text{cut}} = 8$ s, i.e. the last window produces the outcome $E_x(t = 5)$. The vector obtained in this way is used as the input feature vector of the SSD classifier, i.e. it corresponds to F_x according to the notation introduced in Chapter 3 and, more precisely, in Figure 3.1. Therefore, in this experiment alone, the final feature vector F_x consists of $M = 44$ elements and it is built by concatenating the 11 4-classes emotional outcomes in time $E_x(t)$. We call this feature vector *emotional outcome in time*. More formally, we have

$$F_x = [E_x(t = 0), E_x(t = 0.5), ..., E_x(t = 5)], \quad F_x \in \mathbb{R}^M \tag{4.6}$$

Since most of the datasets used, in particular the ASVspoof dataset, consist of very short utterances, it is worth noticing that this experiment is conduced on a limited amount of training data. In particular, the final training set for this experiment is composed of 544 data and the validation set is composed of 546 data, manually balanced in both cases between the bonafide and spoof classes. However, as we will discuss in details in Chapter 5, the experiment confirms the goodness of the choice of the semantic-based emotional features in order to perform the SSD classification.

4.6 Conclusive Remarks

In this chapter we have described the evaluation setup of this work. In this sense, we have reported all the details of the chosen dataset. We have then introduced the training methodology and parameters as well as the metrics adopted to evaluate the performances for all the experiments. Finally, we have reported the setup of each experiment we performed. The next chapter is devoted to the analysis of the results achieved through these experiments.

5
Simulations and Tests

In this chapter we present the results relative to the Synthetic Speech Detection (SSD) task. Results are obtained evaluating 4 different set of experiments, which have been described in Section 4. We remind that the SER feature extractor is the same in all the experiments performed.

In Experiment I, we describe the results of the framework we propose, selecting the best SSD classifier and its hyperparameters. We also compare these results with the ones of the baseline. In Experiment II, we perform 3 experiments involving a Voice Activity Detector (VAD) to verify the effectiveness of the SER model in providing vocal emotional-based features. In Experiment III, we perform experiments with noise injection at different Signal-to-noise Ratio (SNR) levels in the test set to analyze the influence of noise in the performances. We then augment the train and development set too as countermeasure, and analyze the difference in the performances. In Experiment IV, we use directly the evolution of the SER labels in time as input for the final SSD classifier, in order to analyze the level of compatibility between the SER and SSD

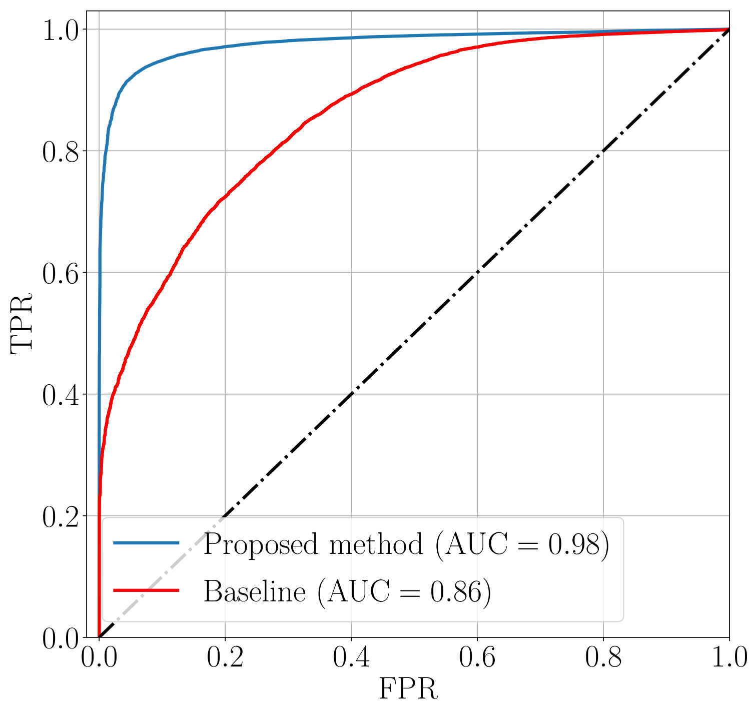


Figure 5.1: ROC curves for the proposed method and the considered baseline in the Experiment I setup. The evaluation set corresponds here to the ASVspoof *eval* set.

tasks.

5.1 Experiment I: Main framework

In this section, we analyze the results obtained performing the experiment described in Section 4.5.2. We compare the results with the baseline described in Section 4.3. We also compare the results obtained with two different algorithms for the SSD classifier stage, namely Random Forest (RF) and Support Vector Machine (SVM).

Figure 5.1 shows the Receiver Operating Characteristic (ROC) curves of the proposed system with the SSD classifier based on RF algorithm against the selected baseline. We point out that, in this case, the train set is the original dataset illustrated in Table 4.1, while the test set for both systems is composed of the ASVspoof 2019 *eval* partition. As it is evident from the plot, our method significantly outperforms the baseline. The ROC curve of the proposed method is always above the one of the baseline, as attested also by the AUC that reaches a value of $\mathrm{AUC}_{\mathrm{RF}} = 0.98$

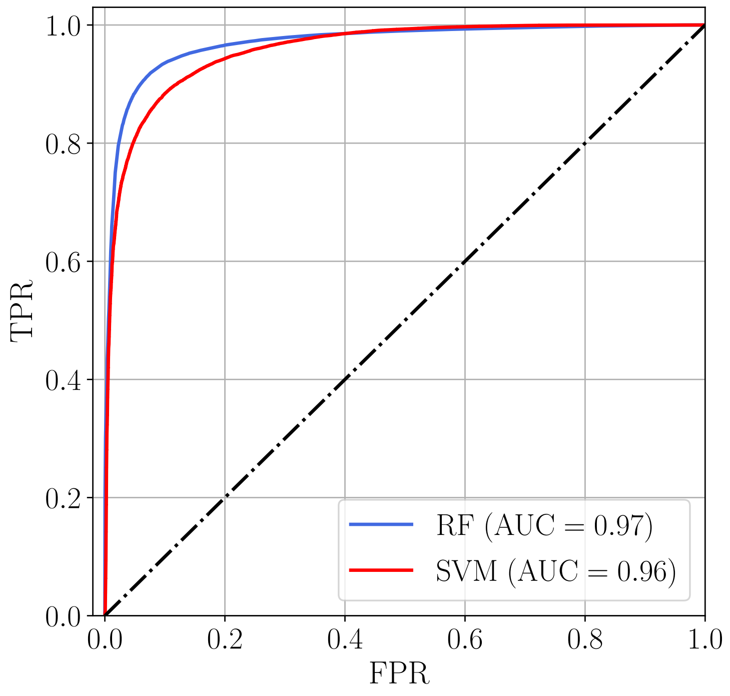


Figure 5.2: ROC curves of SVM and RF classifiers in the Experiment I setup.

against the $\text{AUC}_{\text{base}} = 0.86$ of the baseline. This first experiment confirms that the proposed semantic-based approach allows achieving higher discrimination capability when compared to more traditional CNN-based methods.

Figure 5.2 shows the ROC curves of the proposed system for two different SSD classifier, based on RF and SVM algorithms. We have performed grid search on both classifiers in order to select the best hyperparameters, as showed in Section 4.2. For the RF classifier, these have been found to be information gain as quality criterion function and a number of learners $N_{\text{RF}} = 300$. For the SVM classifier, the best value of the regularization parameter corresponds to $C_{\text{SVM}} = 1$, which is a rather standard one, thus indicating no particular overfitting. We have selected the best parameters as the ones that give the best balanced accuracy on the *dev* set.

The figure shows that the RF model slightly outperforms the SVM model, as it is confirmed by the fact that the ROC curve corresponding to RF is almost always above the ROC curve corresponding to SVM. The small but significant difference between the AUCs confirms this trend. The balanced accuracy reaches the value of $\mathrm{BA}_{\mathrm{RF}} = 0.912$ for the RF classifier, beating again the SVM classifier that reaches a value of $\mathrm{BA}_{\mathrm{SVM}} = 0.885$.

Table 5.1 shows a more extensive comparison between the True Negative Rate (TNR) of each bonafide evaluation set and the True Positive Rate (TPR) of each spoofing algorithm. While the SVM classifier slightly outperforms the RF in the majority of ASVspoof *eval* algorithms, RF outperforms SVM in the Cloud2019 algorithms, with the only relevant exception of Amazon Polly (PO). Most importantly, the evidence shows that the RF classifier outperforms the SVM when considering all the bonafide evaluation sets, particularly Ljspeech. For these reasons, as well as the high computational time required for the SVM classifier training, we select RF as best choice to perform the remaining experiments.

In all experiments, we have performed grid search on the RF classifier with the same hyperparameters detailed in Section 4.2. However, we explicitly point out only the cases in which the grid search selects different values for at least one of the best hyperparameters with respect to this case. In all other cases, the best hyperparameters are implied to be information gain as quality criterion function and a number of learners $N_{\mathrm{RF}} = 300$.

As showed in Table 5.1, the RF classifier reaches remarkable performances in this experimental setup. Algorithm A14 from ASVspoof and Amazon Polly (PO) TTS from Cloud2019 are the only cases where the accuracy is below 0.8. About algorithm A14, this is probably due to the fact that algorithm A14 is a mixed TTS/VC system that has been built starting from a very efficient VC system [39]. Hence real emotional qualities are probably still present in the audio tracks, affecting the efficiency of the proposed system. For all the other spoofing algorithms, the TPR value is close to or greater than 0.9. This classifier constitutes the touchstone for the remaining experiments: from now on, we will refer to it as the main or reference classifier of Experiment I.

Table 5.1: The table shows a comparison between TNR of all the bonafide evaluation sets and TPR of all the spoofing algorithms for RF and SVM classifier. The names of the bonafide datasets and spoofing algorithms are coherent with definitions in Section 4.1.

	Bonafide (TNR)			**Spoof (TPR)**														
Classifier	**BF**	**LJS**	**IEM**	**A07**	**A08**	**A09**	**A10**	**A11**	**A12**	**A13**	**A14**	**A15**	**A16**	**PO**	**AZ**	**GS**	**GW**	**WA**
RF	0.970	0.941	0.943	0.948	0.988	1.000	0.900	0.895	0.890	0.831	0.763	0.927	0.898	0.483	0.812	0.99	0.921	0.855
SVM	0.959	0.774	0.910	0.977	0.969	1.000	0.966	0.978	0.953	0.940	0.940	0.980	0.922	0.560	0.632	0.994	0.838	0.681

5.2 Experiment II: Voice Activity Detector

This section analyzes the results of the set of experiments described in Section 4.5.2, i.e. the ones involving the Voice Activity Detector (VAD).

The first experiment of the set consists in filtering out the tracks shorter than 2.7 s and the tracks with low voice activity $V_{\text{vad}} \leq 80\%$ from all sets. As explained in Section 4.5.2, V_{vad} corresponds to the percentage of utterance's short frames for which the VAD gives a positive outcome, i.e. the frame is predicted to contain vocal activity. We have repeated both training and evaluation with the subset of tracks with high voice activity $V_{\text{vad}} \geq 80\%$. Figure 5.3 shows the comparison between the ROC curve obtained as explained and the ROC curve relative to the reference classifier of Experiment I. The curves are very similar to each other. We impute the slight decrease of the AUC to the reduction of the training set numerosity due to the elimination of short and low voice activity tracks. Nevertheless, the similarity between the curves shows that the performance obtained does not depend significantly on the amount of silence in the data. This experiment provides another proof of the quality of the extracted features.

The second and main experiment of the set consists of substituting silent frames with zero-padding and vice versa, as explained in Section 4.5.2, in all the dataset tracks. For the sake of brevity, we call *Silenceout* the experiments in which the frames without voice activity are substituted with zero-padding, whereas we call *Silencein* the experiments in which, complementary, the frames containing voice activity are substituted with zero-padding (and thus only the silent frames are

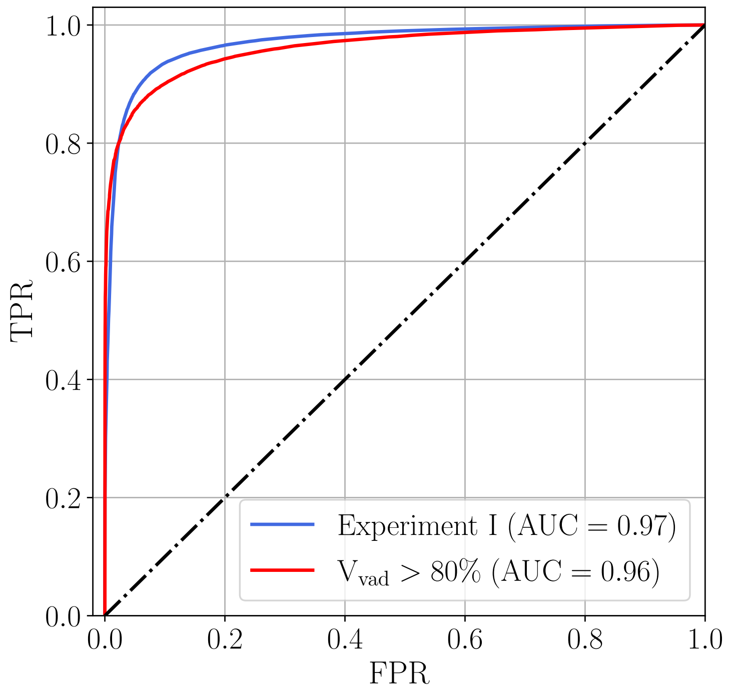


Figure 5.3: ROC curves of RF classifier showed in experiment I and same classifier trained and tested eliminating short tracks and tracks with voice activity $V_{\text{vad}} \leq 80\%$.

maintained).

Figure 5.4 shows the comparison between the ROC curve of the main classifier from Experiment I and the curves of *Silenceout* and *Silencein* experiments at different levels of VAD aggressiveness $A_{\text{vad}} = \{1, 3\}$. Both training and evaluation have been repeated for each experiment. We point out that, in both the *Silenceout* and *Silencein* experiments with $A_{\text{vad}} = 1$, the grid search on the RF classifier has selected Gini impurity as quality criterion function, differently from the reference experiment illustrated in Section 5.1.

We start from the comparison between the main experiment and the *Silenceout* experiments with $A_{\text{vad}} = 1$ and $A_{\text{vad}} = 3$. First, we notice that the value of the aggressiveness does not change significantly the performances, since the results of the two *Silenceout* experiments show practically coincident curves and the same AUC values. Then, we notice a significant degradation of the ROC curves and AUC with respect to the reference classifier discussed in Experiment I. We impute this decrease

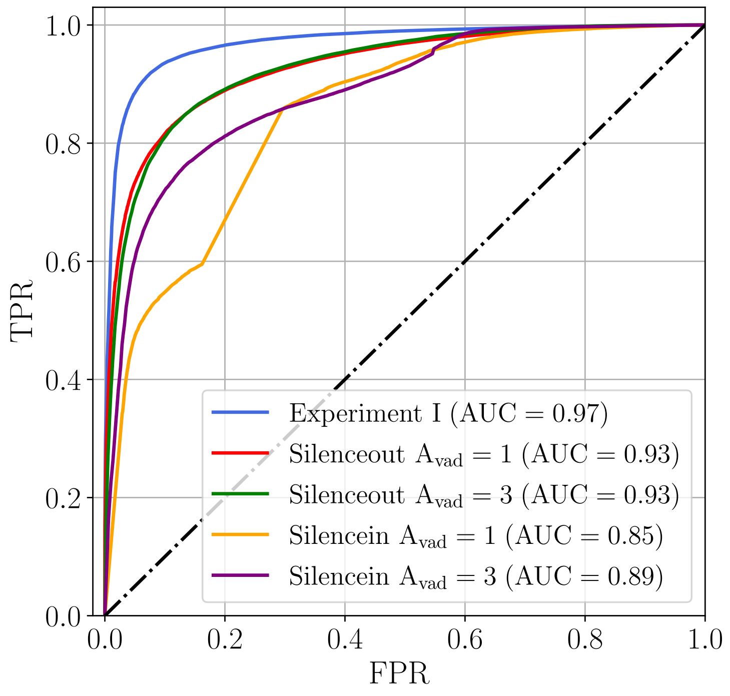
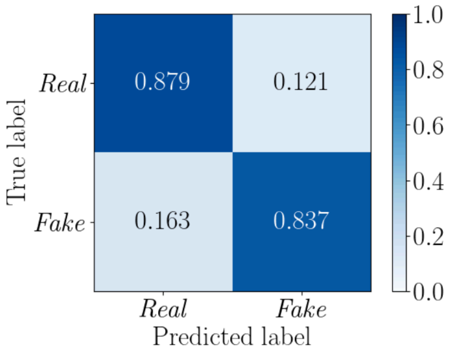


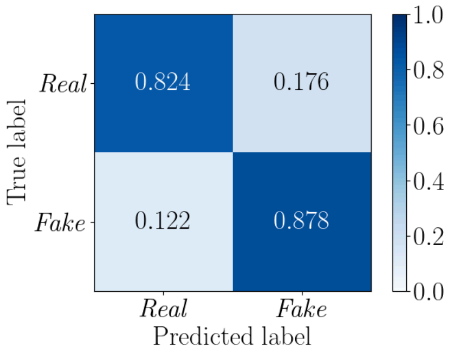
Figure 5.4: ROC curves of *Silenceout* and *Silencein* experiments against the ROC of Experiment I.

of performances to the discontinuity that is caused by the substitution of the frames without voice activity with zero-padding. This operation may lead to degradation of the quality of the original track, therefore of the SER ability to recognize emotion, therefore, eventually, of the performances.

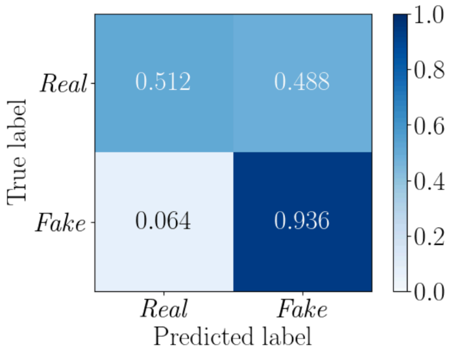
As expected, and as it is evident from Figure 5.4, the *Silencein* experiments reach poorer performances overall with respect to the *Silenceout*s. Moreover, differently from the *Silenceout* experiments, we notice a further significant deterioration of ROC and AUC when the aggressiveness passes from $A_{\text{vad}} = 3$ to $A_{\text{vad}} = 1$. This difference is probably due to the fact that the maximum filtering aggressiveness $A_{\text{vad}} = 3$ leads the VAD to consider as silent frames a small portion of the frames containing vocal activity instead. These frames may help the *Silencein* model in predicting the correct outcome. For this reason, we should consider the experiment with $A_{\text{vad}} = 1$ as the only one effectively eliminating all frames with vocal activity. In this case, the AUC has a very significant decrease from $\text{AUC}_{\text{Silenceout}} = 0.93$ to $\text{AUC}_{\text{Silencein}} = 0.85$. However, the



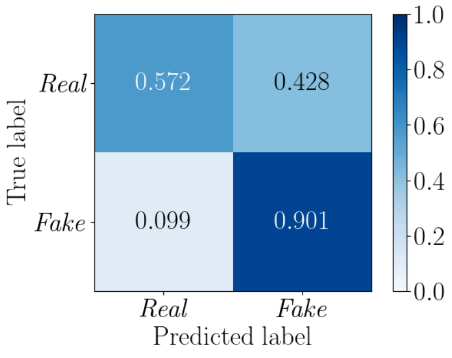
(a) Confusion matrix of *Silenceout* experiment with $A_{\text{vad}} = 1$.



(b) Confusion matrix of *Silenceout* experiment with $A_{\text{vad}} = 3$.



(c) Confusion matrix of *Silencein* experiment with $A_{\text{vad}} = 1$.



(d) Confusion matrix of *Silencein* experiment with $A_{\text{vad}} = 3$.

Figure 5.5: Confusion matrix for the *Silenceout* and *Silencein* experiments with aggressiveness $A_{\text{vad}} \in \{1, 3\}$.

silent frames and their background noise seem to provide features sufficiently informative in order to perform a non-casual binary classification.

Figure 5.5 shows the confusion matrices for the 4 examined cases, i.e. the *Silenceout* and *Silencein* experiments with different VAD aggressiveness. The matrices are computed on all the samples of the evaluation set. We observe that in the *Silenceout* experiments the TPR and TNR reach similar values. On the other hand, in both the *Silencein* experiments the classifier is highly unbalanced, i.e. it reaches a high TPR at the expense of a very poor TNR. In other words, both the *Silencein* classifiers are unable to generalize well the bonafide speech data.

Table 5.2 shows a more extensive comparison between the TNR of

Table 5.2: The table shows a comparison between the main RF classifier showed in 5.1 and the *Silenceout* and *Silencein* experiments with varying aggressiveness A_{vad}. The TNRs of all the bonafide evaluation sets and the TPRs of all the spoofing algorithms are computed for each case considered. The names of the bonafide datasets and spoofing algorithms are coherent with the definitions given in Section 4.1.

		Bonafide (TNR)			**Spoof (TPR)**														
Classifier	A_{vad}	**BF**	**LJS**	**IEM**	**A07**	**A08**	**A09**	**A10**	**A11**	**A12**	**A13**	**A14**	**A15**	**A16**	**PO**	**AZ**	**GS**	**GW**	**WA**
Experiment I	/	0.970	0.941	0.943	0.948	0.988	1.000	0.900	0.895	0.890	0.831	0.763	0.927	0.898	0.483	0.812	0.993	0.921	0.855
Silenceout	1	0.852	0.890	0.901	0.904	0.929	0.996	0.873	0.867	0.862	0.755	0.846	0.930	0.853	0.368	0.354	0.973	0.750	0.665
Silenceout	3	0.784	0.846	0.831	0.935	0.908	0.996	0.922	0.910	0.904	0.741	0.930	0.962	0.872	0.526	0.552	0.988	0.941	0.781
Silencein	1	0.954	0.232	0.696	0.991	0.664	0.985	0.991	0.989	0.988	0.972	0.984	0.981	0.908	0.988	0.846	0.974	0.767	0.923
Silencein	3	0.986	0.318	0.695	0.987	0.663	0.987	0.971	0.992	0.991	0.803	0.960	0.957	0.885	0.989	0.330	0.987	0.939	0.937

each bonafide evaluation set and the TPR of each spoofing algorithm. In particular, we focus on the bonafide sets of the *Silencein* experiments. We have already pointed out that the classifier results are unbalanced in the direction of the spoofing data, i.e. with both values of aggressiveness the TPR is quite good whereas the TNR is quite poor (Figures 5.5c and 5.5d). Taking the analysis further, we notice that the *Silencein* classifiers are able to classify with a good rate only the bonafide coming from the ASVspoof evaluation set, which are indicated with the abbreviation BF. Going back to Table 4.1, we observe that a partition of the ASVspoof dataset is used in the training phase. We suppose that the *Silencein* classifier predicts the outcome based on noise content and/or preprocessing parameters which are intrinsic characteristics of the ASVspoof dataset. For this reason, it fails to classify bonafide data coming from different datasets. On the contrary, the Experiment I main classifier and the *Silenceout* classifiers generally appear to be able to classify both bonafide and spoof data coming from different datasets with good score.

This experiment gives also the opportunity to underline the importance of using a cross-corpus approach. This is particularly true for a task such as SSD detection, in which we must be able to classify correctly in-the-wild data potentially coming from any possible kind of source and not from a controlled environment. In fact, thanks to this cross-corpus analysis, we can conclude that our main model is effectively using good semantic-based features to perform the classification, and it is only marginally influenced by preprocessing parameters and by noise, at least until the Signal-to-noise Ratio (SNR) of the speech signal is in a certain range. In this direction, in Section 5.3 we perform some experiments augmenting the dataset with noise injection in order to define more precisely this range and, more generally, to test the performances of the model when the SNR degrades.

5.3 Experiment III: Dataset augmentation with noise injection

In this section, we evaluate the set of experiments described in Section 4.5.3, i.e. the ones in which an injection of noise in the data is performed.

Table 5.3: Results of the evaluation of the proposed system for different datasets and TTS algorithms using clean and augmented training sets. Bonafide and spoof dataset names are coherent with definitions in Section 4.1.

SNR	**Train**	**Bonafide (TNR)**			**Spoof (TPR)**														
[dB]	**Augm.**	**LJS**	**IEM**	**BF**	**A07**	**A08**	**A09**	**A10**	**A11**	**A12**	**A13**	**A14**	**A15**	**A16**	**PO**	**AZ**	**GS**	**GW**	**WA**
∞		0.941	0.943	0.970	0.948	0.988	1.000	0.900	0.895	0.890	0.831	0.763	0.927	0.898	0.483	0.812	0.993	0.921	0.855
25		0.947	0.944	0.996	0.911	0.883	0.992	0.875	0.861	0.803	0.740	0.710	0.872	0.736	0.421	0.558	0.966	0.851	0.610
20		0.965	0.943	0.999	0.814	0.699	0.917	0.800	0.783	0.539	0.632	0.547	0.687	0.439	0.304	0.264	0.819	0.639	0.331
15		0.982	0.942	0.999	0.565	0.421	0.587	0.576	0.542	0.202	0.446	0.216	0.303	0.129	0.138	0.032	0.361	0.238	0.060
10		0.988	0.934	0.999	0.342	0.224	0.223	0.355	0.334	0.128	0.314	0.084	0.093	0.080	0.093	0.000	0.051	0.038	0.020
∞	✓	0.854	0.828	0.865	0.975	0.994	1.000	0.957	0.973	0.940	0.910	0.877	0.965	0.941	0.603	0.832	0.996	0.966	0.920
25	✓	0.857	0.829	0.894	0.969	0.970	1.000	0.952	0.969	0.922	0.863	0.870	0.956	0.901	0.584	0.768	0.994	0.942	0.803
20	✓	0.861	0.829	0.904	0.947	0.926	0.999	0.939	0.961	0.892	0.824	0.834	0.927	0.837	0.533	0.522	0.978	0.907	0.697
15	✓	0.797	0.823	0.842	0.927	0.884	0.995	0.923	0.946	0.845	0.809	0.783	0.868	0.758	0.497	0.259	0.955	0.845	0.617
10	✓	0.656	0.807	0.800	0.886	0.817	0.984	0.907	0.916	0.843	0.836	0.764	0.829	0.748	0.466	0.268	0.887	0.767	0.676

Table 5.3 shows the performances of the proposed binary classifier for the two training configurations, namely with clean and augmented train set. The TNR is computed for each dataset of bonafide speech tracks and the TPR is compued for each TTS algorithm.

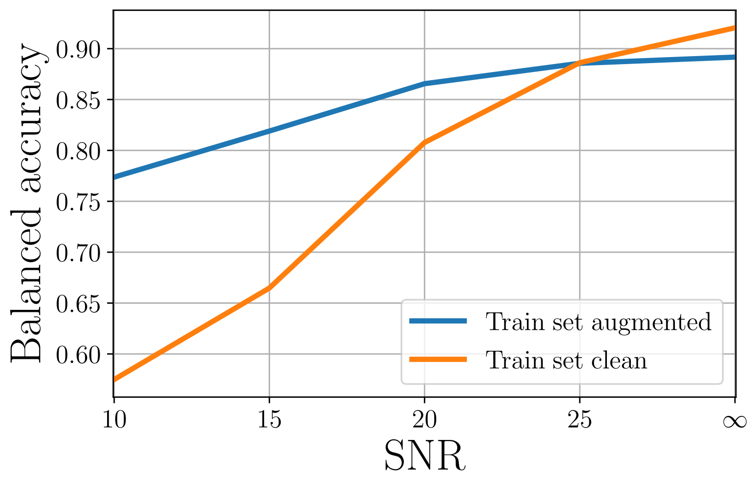


Figure 5.6: Balanced accuracy values at different arbitrarily injected SNR levels.

The first row of Table 5.3 correspond to the results of the reference classifier discussed in Experiment I. In this case, neither the train nor the test set has not been augmented with noise, hence SNR $= \infty$, since the SNR values are computed on the injected noise.

In the following rows in the first half of Table 5.3, the system trained on clean data (i.e. the reference classifier) is tested in noisy conditions with different fixed SNR levels, as described in Section 4.5.3. We can observe that, as the noise level increases, the performances of the synthetic speech detector progressively degrade. The classifier tends to consider the input always as bonafide speech, and we can observe a remarkable decrease of the general TPR for all the tested spoofing algorithms. The TPR degradation is particularly evident when we decrease the SNR to 15 dB or less. This is due to degradation of speech quality and intelligibility in the track, which in turn leads to degradation of the semantic feature extraction. Furthermore, we notice that the TPR of some algorithms, as Microsoft Azure (AZ) and IBM Watson (WA), appears to degrade significantly faster.

This behavior encourages the use of data augmentation strategy on the training set. We have described in details our data augmentation strategy in Section 4.5.3. The results of this second training configuration are presented in the second half of Table 5.3. We point out that, in this case, the grid search on the RF classifier has selected Gini impurity as quality criterion function, differently from the reference experiment illustrated in Section 5.1.

When the training set is augmented, the presence of noise does not greatly affect the detector performance. Even with low SNR values, the balanced accuracy values are good on the majority of spoofing algorithms and bonafide speech datasets.

To further analyze the effects of training data augmentation, in Figure 5.6 we report the behavior of the balanced accuracy on the entire dataset for different SNR values. One curve corresponds to performances obtained training on the clean dataset, the second one corresponds to the ones obtained training on the augmented dataset. We remind that, as described in 4.4, we compute the balanced accuracy by taking the average between the average TPR of all spoofing algorithms and the average TNR of the three bonafide speech datasets. Although the TNR remains above 80% in practically all the augmented cases, we notice the classifiers reach better performance for the classification of clean test data when it has been trained on the clean training set rather than on the augmented training dataset. This suggests that the noise content in the original bonafide tracks, albeit marginally, helps the classifiers with clean training set in predicting the correct outcome. A possible explanation for this behavior is the following. Original bonafide signals, being real recordings of speech, may contain some environmental noise, that is absent in the synthetically generated speech signals. This difference may help the classifier in discriminating between the two classes when the training and evaluation set are not augmented but, on the other side, it leads to less generalization ability and robustness of the system. In fact, the classifier trained with augmentation evidently outperforms the one with clean train set, in direct proportion to the decrease in SNR, reaching a difference of almost 20% in the noisiest experiment, i.e., for SNR = 10 dB.

Figure 5.7 confirms this trend by showing the ROC curves obtained with the proposed method considering clean (solid) and augmented (dashed) train sets. We can observe that, when the SNR is high, the training on clean data is more advantageous than using the augmented training set.

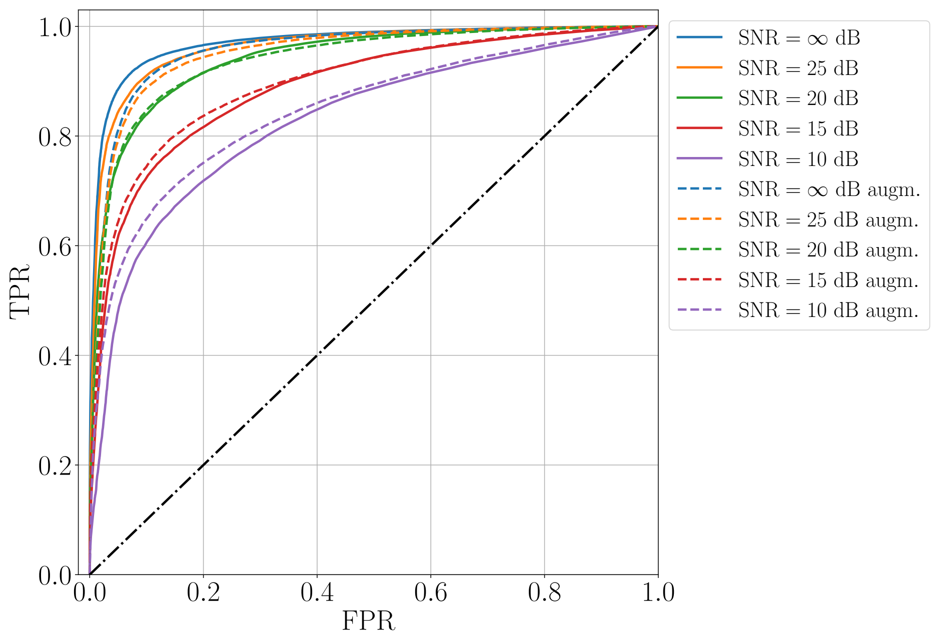


Figure 5.7: ROC curves for the proposed method with clean (solid) and augmented (dashed) train sets at different arbitrarily injected power SNR levels.

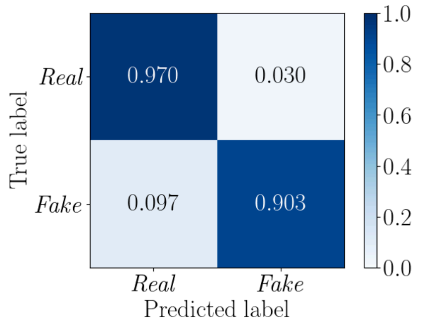
As the test set SNR level decreases, the ratio between true positives and false positives generally lowers, mostly because of the degradation of the TPR, as it is evident from Table 5.3. However, this ratio drops remarkably more in the case of the classifier trained on clean data with respect to the one trained on augmented data.

In general, the results of this experiment show that data augmentation constitutes a good approach to improve the robustness of the model to noise. This is very interesting for us, since the SSD classifier must be able to analyze in-the-wild data, potentially coming from any kind of source.

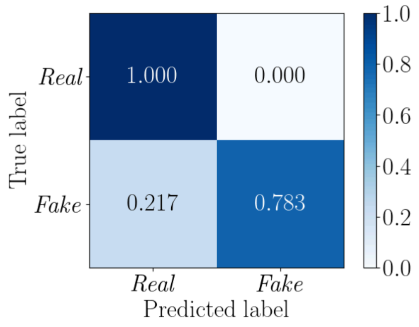
5.4 Experiment IV: Emotional outcome in time

In this section, we evaluate the experiment described in Section 4.5.4, i.e. the one in which the *emotional outcome in time* is used as feature vector for the final SSD classifier. For brevity, we will call this classifier *emoconcat*, stemming from 'emotion' and 'concatenation'.

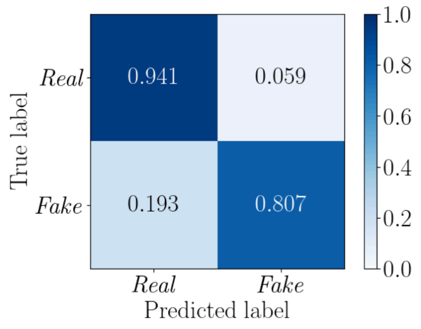
We start the analysis from the balanced accuracy for this experiment,



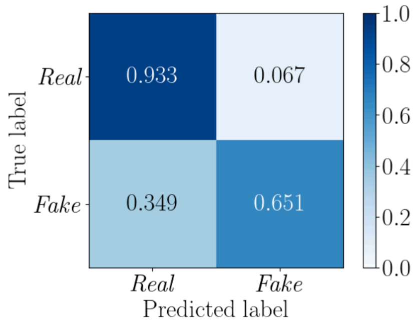
(a) Confusion matrix of the reference Experiment I classifier computed on the ASVspoof *eval* set.



(b) Confusion matrix of the *emoconcat* classifier computed on the ASVspoof *eval* set.



(c) Confusion matrix of the reference Experiment I classifier computed on the aggregated samples of all datasets excluded ASVspoof.



(d) Confusion matrix of the *emoconcat* classifier computed on the aggregated samples of all datasets excluded ASVspoof.

Figure 5.8: Confusion matrix of the reference classifier and the *emoconcat* classifier computed on the ASVspoof dataset and the aggregation of the other datasets samples separately.

that reaches the value of $\text{BA}_{\text{emoconcat}} = 0.871$ against the $\text{BA}_{\text{main}} = 0.912$ of the RF reference classifier in Experiment I. This is a greatly relevant result, not only because it proofs the compatibility between the SER and SSD tasks, but also because it is obtained with a training set of small dimensions compared to all the other experiments performed, as explained in details in Section 4.5.4.

Figure 5.8 shows the confusion matrices of the main classifier from Experiment I and the *emoconcat* classifier computed on the ASVspoof dataset and the aggregation of the other datasets samples separately. We notice that the *emoconcat* classifier succeeds in classifying the great majority of bonafide from both ASVspoof and the other datasets, reaching TNRs similar or greater than the reference classifier. On the other hand, the classification of the spoof data reaches not completely satisfying performances, with the TPR values of $\text{TPR}_{\text{asvspoof}} = 0.783$ on ASVspoof and $\text{TPR}_{\text{cloud2019}} = 0.651$ on the aggregated samples from all the remaining algorithms in the test set. This decrease in the general TPR can be imputed to the small numerosity of the train set for this experiment, and possibly to the reduction of the features size.

Table 5.4: The table shows a comparison between TNR of all the bonafide evaluation sets and TPR of all the spoofing algorithms for RF and SVM classifier. The names of the bonafide datasets and spoofing algorithms are coherent with definitions in Section 4.1.

	Bonafide (TNR)			**Spoof (TPR)**														
Classifier	**BF**	**LJS**	**IEM**	**A07**	**A08**	**A09**	**A10**	**A11**	**A12**	**A13**	**A14**	**A15**	**A16**	**PO**	**AZ**	**GS**	**GW**	**WA**
Reference	0.970	0.941	0.943	0.948	0.988	1.000	0.900	0.895	0.890	0.831	0.763	0.927	0.898	0.483	0.812	0.993	0.921	0.855
Emoconcat	1.000	0.929	1.000	0.979	0.969	1.000	0.733	0.938	0.783	0.750	0.628	0.837	0.659	0.409	0.269	0.945	0.863	0.715

Table 5.4 shows the TNR for all the bonafide datasets and the TPR for all the spoofing algorithms. The results are reported for both the reference classifier from Experiment I and the *emoconcat* classifier, in order to provide a comparison between the two. The *emoconcat* classifier shows good or acceptable performances in all cases (again, also considering the limited amount of data in the training set for this particular case), with the only exception of algorithms PO and AZ from Cloud2019. However, we remark that PO is one of the most elusive algorithms, as demonstrated

by the results showed in all sections. Also AZ created some problem to the classifiers, due to its susceptibility to noise, as analyzed in Section 5.3. We conclude that this case is substantially different from the *Silencein* case analyzed in Section 5.2. In fact, the *emoconcat* classifier fails just in classifying two particular, hard-to-get algorithms. Instead, the *Silencein* classifier fails in classifying the majority of the bonafide data not coming from the ASVspoof dataset, as we detailed in Section 5.2. Finally, given the promising performances, it would be interesting to repeat this experiment using a larger training set. In fact, the training set we have used is limited by the little amount of long enough tracks present in the ASVspoof dataset, as we detailed in Section 4.5.4.

5.5 Conclusive Remarks

In this chapter, we have analyzed the results of four different experiments in order to evaluate the model performances, the quality of the semantic-based features and the robustness of the model to noise.

The evaluation of the experiments results has given us generally positive response about the goodness of proposed method. In particular, we have observed that the noise augmentation performed in Experiment III constitutes a good strategy to reach better performances, especially in case of noisy test data. Moreover, it helps to strengthen the classifier by reducing the effect of the environmental noise difference in power between bonafide and spoof train data.

The experiment II, involving the VAD, instead, has been useful to verify that the performances of our SSD classifier are effectively based on qualitative semantic-based features and just marginally influenced by preprocessing parameters and by noise (at least when the latter is limited in power).

Finally, with experiment IV, using as features directly the concatenation of the emotional outcomes in time and nonetheless obtaining remarkable results, we have proved the high compatibility between the two tasks. In this sense, it would be also interesting to further investigate the performances using the same feature extraction method but a larger training set.

In the next chapter, we pull the strings of this work and analyze possible future works and improvements related to it.

6 Conclusions and Future Works

The rapid and uncontrolled diffusion of the multimedia contents in the hyper-connected society of social media has created an increasing need of security. The possibility to produce spoof but very realistic contents has increased exponentially with the late progresses of AI-based technologies. In principle, we would like to have a general, accurate, automatic technique to discriminate authentic, bonafide content from spoof content, i.e. deepfakes. However, truth be told, the struggle between the creators and analysts of fake content is in continuous evolution, driven by the development of the technologies.

In this work, we have proposed a system with the objective to discriminate bonafide from Text-to-speech (TTS) spoof data. The problem has been declined as a binary classification problem. The system is composed of two main blocks. The first one, the emotional feature extractor, takes advantage of a state-of-the-art Speech Emotion Recognition (SER) neural network based on CNN, RNN and attention mechanism and trained on an authentic speech signal dataset annotated with the emotion expressed by the speaker. By applying a transfer learning approach to this network, we have been able to create an embedding space that is meaningful not only for the original task, i.e. SER, but also for the task at hand, i.e. Synthetic Speech Detection (SSD). The second component is a supervised classifier that takes as input these emotional embeddings and predicts whether the given speech track is bonafide or spoof.

In order to perform a robust evaluation, we have tested the proposed system on an ad-hoc dataset, which is in turn composed of multiple existing dataset of bonafide and spoof speech. We have performed 4 different set of experiments in order to evaluate primarily the performances of the system, its robustness to noise, the quality of the chosen semantic features and their compatibility with the SSD task.

The interesting idea behind semantic emotional features is their generality. If the inability of synthesis algorithms to recreate complex semantic aspects of the real human voice (such as the emotional ones) subsists, this extends not only to a particular algorithm, but to the great majority (if not all) of them. For this reason, we think it would be interesting to further develop the use of semantic features in order to perform the SSD classification task.

The results of the experiments demonstrate that the proposed methodology is promising: the balanced accuracy reaches a value of $\mathrm{BA}_{\mathrm{cc}} = 0.912$ on the cross-corpus setup. However, many experiments may still be performed to further improve and validate the proposed methodology.

The first suggestion goes in the direction of strengthening the SER model: an immediate idea would be to train the SER classifier in a cross-corpus setup too. Another possible improvement of the SER robustness may consists on extending the analysis of the influence of noise. A suggestion in this direction is to possibly augment the SER training set too.

Secondly, it would be interesting to train different models on top of the emotional features provided by the SER. It would be interesting, in particular, to build an ad-hoc deep learning model for the final SSD classifier.

Finally, a possible future work consists in implementing a parallel classifier able to detect speech tracks spoofed with Voice Conversion (VC) algorithms. Since VC algorithms generate spoof speech tracks applying style transfer to real speech tracks, it is hard to find semantic, high-level features to perform discrimination of VC speeches from bonafide. This is because, in the case of VC, the semantic information does not need to be generated from scratch by the algorithm, but can be simply maintained from the original speech track. A VC-specific classifier would constitute an optimal pair with the proposed system. A suggestion of a possible classifier focused on VC algorithms alone is provided by [139]. The work exploits statistic analysis of common shift applied to the spectral slope of consecutive speech frames spoofed with VC techniques. A generic track could be analyzed by a framework composed by the two systems in parallel, and would be classified as bonafide just in case of negative response from both classifiers.

Bibliography

[1] "Speech synthesis." `https://en.wikipedia.org/wiki/Speech_synthesis`.

[2] D.-Y. Huang, L. Xie, Y. S. W. Lee, J. Wu, H. Ming, X. Tian, S. Zhang, C. Ding, M. Li, N. Q. Hy, *et al.*, "An automatic voice conversion evaluation strategy based on perceptual background noise distortion and speaker similarity.," in *SSW*, pp. 44–51, 2016.

[3] 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.

[4] A. Bhattacharjee, T. Pias, M. Ahmad, and A. Rahman, "On the performance analysis of apis recognizing emotions from video images of facial expressions," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 223–230, 2018.

[5] R. A. Khalil, E. Jones, M. I. Babar, T. Jan, M. H. Zafar, and T. Alhussain, "Speech emotion recognition using deep learning techniques: A review," *IEEE Access*, vol. 7, pp. 117327–117345, 2019.

[6] S. Mirsamadi, E. Barsoum, and C. Zhang, "Automatic speech emotion recognition using recurrent neural networks with local attention," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2227–2231, IEEE, 2017.

[7] "Mel spectrogram scheme." `https://se.mathworks.com/help//audio/ref/melspectrogram.html`.

[8] V. Podgorelec, "Improved mining of software complexity data on evolutionary filtered training sets," *WSEAS Trans. on Inf. Sci, and App*, vol. 6, 2009.

[9] "A comprehensive guide to convolutional neural networks." `https://towardsdatascience.com/a-comprehensive-guideto-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`.

[10] "Understanding lstm networks." `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[11] 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.

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

[13] "Amazon polly." `https://aws.amazon.com/polly/`.

[14] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of RGB videos," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. L. Moreno, *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," *arXiv preprint arXiv:1806.04558*, 2018.

[16] H. Tachibana, K. Uenoyama, and S. Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[17] "People are using creepy, cutting-edge ai technology to splice nic cage into every movie they can think of." `https://www.businessinsider.com/nicolas-cage-inserted-movies-fakeapp-ai-technology-2018-1?r=UK`.

[18] E. Howcroft, "How faking videos became easy and why that's so scary," *Bloomberg: New York, NY, USA*, 2018.

[19] C. Wang, "Deepfakes, revenge porn, and the impact on women," *Forbes*, Nov. 2019.

[20] BBC News, "'deepfake' app causes fraud and privacy fears in china." `https://www.bbc.com/news/technology-49570418`, 2019.

[21] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar, *et al.*, "The malicious use of artificial intelligence: Forecasting, prevention, and mitigation," *arXiv preprint arXiv:1802.07228*, 2018.

[22] M. Reynolds, "Courts and lawyers struggle with growing prevalence of deepfakes," 2020.

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

[24] S. Agarwal, H. Farid, O. Fried, and M. Agrawala, "Detecting deep-fake videos from phoneme-viseme mismatches," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

[25] F. Matern, C. Riess, and M. Stamminger, "Exploiting visual artifacts to expose deepfakes and face manipulations," in *IEEE Winter Applications of Computer Vision Workshops (WACVW)*, 2019.

[26] M. Todisco, H. Delgado, and N. W. Evans, "A new feature for automatic speaker verification anti-spoofing: Constant q cepstral coefficients.," in *Odyssey*, vol. 2016, pp. 283–290, 2016.

[27] 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.

[28] E. A. AlBadawy, S. Lyu, and H. Farid, "Detecting AI-synthesized speech using bispectral analysis," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[29] M. R. Kamble, H. B. Sailor, H. A. Patil, and H. Li, "Advances in anti-spoofing: from the perspective of asvspoof challenges," *APSIPA Transactions on Signal and Information Processing*, vol. 9, 2020.

[30] A. Luo, E. Li, Y. Liu, X. Kang, and Z. J. Wang, "A capsule network based approach for detection of audio spoofing attacks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[31] T. Chen, A. Kumar, P. Nagarsheth, G. Sivaraman, and E. Khoury, "Generalization of audio deepfake detection," in *Odyssey Speaker and Language Recognition Workshop*, 2020.

[32] 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.

[33] 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 Workshops (CVPRW)*, 2021.

[34] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing ai created fake videos by detecting eye blinking," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–7, IEEE, 2018.

[35] T. Mittal, U. Bhattacharya, R. Chandra, A. Bera, and D. Manocha, "Emotions don't lie: An audio-visual deepfake detection method using affective cues," in *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 2823–2832, 2020.

[36] D. Deng, Y. Zhou, J. Pi, and B. E. Shi, "Multimodal utterance-level affect analysis using visual, audio and text features," *CoRR*, vol. abs/1805.00625, 2018.

[37] S. Poria, E. Cambria, D. Hazarika, N. Mazumder, A. Zadeh, and L.-P. Morency, "Multi-level multiple attentions for contextual multimodal sentiment analysis," in *IEEE International Conference on Data Mining (ICDM)*, 2017.

[38] A. E. Rosenberg, "Automatic speaker verification: A review," *Proceedings of the IEEE*, vol. 64, no. 4, pp. 475–487, 1976.

[39] X. Wang, J. Yamagishi, M. Todisco, H. Delgado, A. Nautsch, N. Evans, M. Sahidullah, V. Vestman, T. Kinnunen, K. A. Lee, *et al.*, "Asvspoof 2019: A large-scale public database of synthesized, converted and replayed speech," *Computer Speech & Language*, vol. 64, p. 101114, 2020.

[40] T. Dutoit, *An introduction to text-to-speech synthesis*, vol. 3. Springer Science & Business Media, 1997.

[41] M. Rashad, H. M. El-Bakry, I. R. Isma'il, and N. Mastorakis, "An overview of text-to-speech synthesis techniques," *Latest trends on communications and information technology*, pp. 84–89, 2010.

[42] J. Latorre, J. Lachowicz, J. Lorenzo-Trueba, T. Merritt, T. Drugman, S. Ronanki, and V. Klimkov, "Effect of data reduction on sequence-to-sequence neural tts," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7075–7079, IEEE, 2019.

[43] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[44] S. Ö. Arık, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman, *et al.*, "Deep voice:

Real-time neural text-to-speech,” in *International Conference on Machine Learning*, pp. 195–204, PMLR, 2017.

[45] E. Helander, H. Silén, T. Virtanen, and M. Gabbouj, “Voice conversion using dynamic kernel partial least squares regression,” *IEEE transactions on audio, speech, and language processing*, vol. 20, no. 3, pp. 806–817, 2011.

[46] P. L. D. Leon, B. Stewart, and J. Yamagishi, “Synthetic speech discrimination using pitch pattern statistics derived from image analysis,” in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[47] Z. Wu, X. Xiao, E. S. Chng, and H. Li, “Synthetic speech detection using temporal modulation feature,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7234–7238, IEEE, 2013.

[48] 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.

[49] K. A. Lee, A. Larcher, G. Wang, P. Kenny, N. Brümmer, D. v. Leeuwen, H. Aronowitz, M. Kockmann, C. Vaquero, B. Ma, *et al.*, “The reddots data collection for speaker recognition,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[50] S. K. Ergünay, E. Khoury, A. Lazaridis, and S. Marcel, “On the vulnerability of speaker verification to realistic voice spoofing,” in *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–6, IEEE, 2015.

[51] M. Todisco, H. Delgado, and N. Evans, “Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification,” *Computer Speech & Language*, vol. 45, pp. 516–535, 2017.

[52] X. Xiao, X. Tian, S. Du, H. Xu, E. S. Chng, and H. Li, "Spoofing speech detection using high dimensional magnitude and phase features: The ntu approach for asvspoof 2015 challenge," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[53] E. A. AlBadawy, S. Lyu, and H. Farid, "Detecting ai-synthesized speech using bispectral analysis.," in *CVPR Workshops*, pp. 104–109, 2019.

[54] M. J. Alam, P. Kenny, V. Gupta, and T. Stafylakis, "Spoofing detection on the asvspoof2015 challenge corpus employing deep neural networks.," in *Odyssey*, pp. 270–276, 2016.

[55] N. Chen, Y. Qian, H. Dinkel, B. Chen, and K. Yu, "Robust deep feature for spoofing detection—the sjtu system for asvspoof 2015 challenge," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[56] H. Yu, Z.-H. Tan, Y. Zhang, Z. Ma, and J. Guo, "Dnn filter bank cepstral coefficients for spoofing detection," *Ieee Access*, vol. 5, pp. 4779–4787, 2017.

[57] Z. Zhang, N. Cummins, and B. Schuller, "Advanced data exploitation in speech analysis: An overview," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 107–129, 2017.

[58] H. Muckenhirn, M. Magimai-Doss, and S. Marcel, "End-to-end convolutional neural network-based voice presentation attack detection," in *2017 IEEE international joint conference on biometrics (IJCB)*, pp. 335–341, IEEE, 2017.

[59] C. Zhang, C. Yu, and J. H. Hansen, "An investigation of deep-learning frameworks for speaker verification antispoofing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 684–694, 2017.

[60] H. Dinkel, N. Chen, Y. Qian, and K. Yu, "End-to-end spoofing detection with raw waveform cldnns," in *2017 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4860–4864, IEEE, 2017.

[61] H. Yu, Z.-H. Tan, Z. Ma, R. Martin, and J. Guo, "Spoofing detection in automatic speaker verification systems using dnn classifiers and dynamic acoustic features," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4633–4644, 2017.

[62] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[63] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[64] M. Rambocas, J. Gama, *et al.*, "Marketing research: The role of sentiment analysis," tech. rep., Universidade do Porto, Faculdade de Economia do Porto, 2013.

[65] S. Chaturvedi, V. Mishra, and N. Mishra, "Sentiment analysis using machine learning for business intelligence," in *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, pp. 2162–2166, IEEE, 2017.

[66] M. Chen, P. Zhou, and G. Fortino, "Emotion communication system," *IEEE Access*, vol. 5, pp. 326–337, 2016.

[67] P. Kuppens, F. Tuerlinckx, J. A. Russell, and L. F. Barrett, "The relation between valence and arousal in subjective experience.," *Psychological bulletin*, vol. 139, no. 4, p. 917, 2013.

[68] M. B. Akçay and K. Oğuz, "Speech emotion recognition: Emotional models, databases, features, preprocessing methods, supporting modalities, and classifiers," *Speech Communication*, vol. 116, pp. 56–76, 2020.

[69] F. Dellaert, T. Polzin, and A. Waibel, "Recognizing emotion in speech," in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, vol. 3, pp. 1970–1973, IEEE, 1996.

[70] S. Demircan and H. Kahramanlı, "Feature extraction from speech data for emotion recognition," *Journal of Advances in Computer Networks*, vol. 2, no. 1, pp. 28–30, 2014.

[71] D. Neiberg, K. Elenius, and K. Laskowski, "Emotion recognition in spontaneous speech using gmms," in *Ninth international conference on spoken language processing*, 2006.

[72] G. Vyas, M. K. Dutta, K. Riha, J. Prinosil, *et al.*, "An automatic emotion recognizer using mfccs and hidden markov models," in *2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 320–324, IEEE, 2015.

[73] P. Jackson and S. Haq, "Surrey audio-visual expressed emotion (savee) database," *University of Surrey: Guildford, UK*, 2014.

[74] Y. Pan, P. Shen, and L. Shen, "Speech emotion recognition using support vector machine," *International Journal of Smart Home*, vol. 6, no. 2, pp. 101–108, 2012.

[75] T. Seehapoch and S. Wongthanavasu, "Speech emotion recognition using support vector machines," in *2013 5th international conference on Knowledge and smart technology (KST)*, pp. 86–91, IEEE, 2013.

[76] S. K. Bhakre and A. Bang, "Emotion recognition on the basis of audio signal using naive bayes classifier," in *2016 International conference on advances in computing, communications and informatics (ICACCI)*, pp. 2363–2367, IEEE, 2016.

[77] M. Wöllmer, F. Eyben, S. Reiter, B. Schuller, C. Cox, E. Douglas-Cowie, and R. Cowie, "Abandoning emotion classes-towards continuous emotion recognition with modelling of long-range dependencies," in *Proc. 9th Interspeech 2008 incorp. 12th Australasian Int. Conf. on Speech Science and Technology SST 2008, Brisbane, Australia*, pp. 597–600, 2008.

[78] K. Han, D. Yu, and I. Tashev, "Speech emotion recognition using deep neural network and extreme learning machine," in *Fifteenth annual conference of the international speech communication association*, 2014.

[79] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5200–5204, IEEE, 2016.

[80] A. M. Badshah, J. Ahmad, N. Rahim, and S. W. Baik, "Speech emotion recognition from spectrograms with deep convolutional neural network," in *2017 international conference on platform technology and service (PlatCon)*, pp. 1–5, IEEE, 2017.

[81] 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.

[82] 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.

[83] F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier, and B. Weiss, "A database of german emotional speech," in *Ninth European Conference on Speech Communication and Technology*, 2005.

[84] S. E. Eskimez, Z. Duan, and W. Heinzelman, "Unsupervised learning approach to feature analysis for automatic speech emotion recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5099–5103, IEEE, 2018.

[85] S. Latif, R. Rana, J. Qadir, and J. Epps, "Variational autoencoders for learning latent representations of speech emotion: A preliminary study," *arXiv preprint arXiv:1712.08708*, 2017.

[86] L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264, IGI global, 2010.

[87] J. Gideon, S. Khorram, Z. Aldeneh, D. Dimitriadis, and E. M. Provost, “Progressive neural networks for transfer learning in emotion recognition,” *arXiv preprint arXiv:1706.03256*, 2017.

[88] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” *arXiv preprint arXiv:1606.04671*, 2016.

[89] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[90] M. N. Stolar, M. Lech, R. S. Bolia, and M. Skinner, “Real time speech emotion recognition using rgb image classification and transfer learning,” in *2017 11th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pp. 1–8, IEEE, 2017.

[91] R. Caruana, “Multitask learning,” *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[92] J. Kim, G. Englebienne, K. P. Truong, and V. Evers, “Towards speech emotion recognition" in the wild" using aggregated corpora and deep multi-task learning,” *arXiv preprint arXiv:1708.03920*, 2017.

[93] S. Parthasarathy and C. Busso, “Jointly predicting arousal, valence and dominance with multi-task learning.,” in *Interspeech*, vol. 2017, pp. 1103–1107, 2017.

[94] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *arXiv preprint arXiv:1508.04025*, 2015.

[95] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *arXiv preprint arXiv:1506.07503*, 2015.

[96] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[97] R. Ekman, *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS).* Oxford University Press, USA, 1997.

[98] F. Eyben, K. R. Scherer, B. W. Schuller, J. Sundberg, E. André, C. Busso, L. Y. Devillers, J. Epps, P. Laukka, S. S. Narayanan, *et al.*, "The geneva minimalistic acoustic parameter set (gemaps) for voice research and affective computing," *IEEE transactions on affective computing*, vol. 7, no. 2, pp. 190–202, 2015.

[99] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, and C. C. Ferrer, "The deepfake detection challenge dataset," *arXiv preprint arXiv:2006.07397*, 2020.

[100] A. Zadeh, P. P. Liang, N. Mazumder, S. Poria, E. Cambria, and L.-P. Morency, "Memory fusion network for multi-view sequential learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[101] P. Pedersen, "The mel scale," *Journal of Music Theory*, vol. 9, no. 2, pp. 295–308, 1965.

[102] R. Stuart and N. Peter, "Artificial intelligence-a modern approach 3rd ed," 2016.

[103] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning.* MIT press, 2018.

[104] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[105] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

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

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

[108] S.-i. Amari and S. Wu, “Improving support vector machine classifiers by modifying kernel functions,” *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.

[109] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.

[110] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[111] B. Yegnanarayana, *Artificial neural networks.* PHI Learning Pvt. Ltd., 2009.

[112] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, “Backpropagation: The basic theory,” *Backpropagation: Theory, architectures and applications*, pp. 1–34, 1995.

[113] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, “Deep learning for computer vision: A brief review,” *Computational intelligence and neuroscience*, vol. 2018, 2018.

[114] M. M. Lopez and J. Kalita, “Deep learning applied to nlp,” *arXiv preprint arXiv:1703.03091*, 2017.

[115] L. Deng, G. Hinton, and B. Kingsbury, “New types of deep neural network learning for speech recognition and related applications: An overview,” in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8599–8603, IEEE, 2013.

[116] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, no. 5-6, pp. 183–197, 1991.

[117] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[118] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, Ieee, 2017.

[119] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0588–0592, IEEE, 2017.

[120] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[121] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[122] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[123] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," *arXiv preprint arXiv:1611.02344*, 2016.

[124] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, Ieee, 2013.

[125] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[126] Y. Yu, X. Si, C. Hu, and J. Zhang, “A review of recurrent neural networks: Lstm cells and network architectures,” *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[127] M. Sundermeyer, R. Schlüter, and H. Ney, “Lstm neural networks for language modeling,” in *Thirteenth annual conference of the international speech communication association*, 2012.

[128] L. Wang, S. Guo, W. Huang, and Y. Qiao, “Places205-vggnet models for scene recognition,” *arXiv preprint arXiv:1508.01667*, 2015.

[129] G. Colombetti, “From affect programs to dynamical discrete emotions,” *Philosophical Psychology*, vol. 22, no. 4, pp. 407–425, 2009.

[130] T. N. Sainath, V. Peddinti, B. Kingsbury, P. Fousek, B. Ramabhadran, and D. Nahamoo, “Deep scattering spectra with deep neural networks for lvcsr tasks,” in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[131] 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.

[132] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: an asr corpus based on public domain audio books,” in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210, IEEE, 2015.

[133] K. Ito and L. Johnson, “The lj speech dataset.” `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[134] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, “CNN architectures for large-scale audio classification,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[135] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[136] C. Marzban, “The roc curve and the area under it as performance measures,” *Weather and Forecasting*, vol. 19, no. 6, pp. 1106–1114, 2004.

[137] “Webrtc.” `https://webrtc.org/`.

[138] “Webrtc.” `https://github.com/wiseman/py-webrtcvad/`.

[139] F. Alegre, A. Amehraye, and N. Evans, “Spoofing countermeasures to protect automatic speaker verification from voice conversion,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3068–3072, IEEE, 2013.