UNIVERSITY OF PADOVA

Department of Information Engineering

Master Degree in ICT for internet and multimedia

Synthetic Speech Detection Algorithms

Author:
Federica LATORA

Supervisor:
Prof. Simone MILANI

Co-Supervisor:
Daniele MARI

A.Y. 2021-2022

Padova, October 18, 2022

Ai miei genitori
A mia sorella Francesca
A nonno Pietro

Abstract

The recent diffusion of audio recording devices together with the rapid evolution of deepfake technologies have fostered the widespread of synthetic speech signals. Being extremely convincing and realistic can be used in many malicious applications, e.g., for fake news spreading over social media platforms, frauds or specifically in impersonation attacks, since speech signals are needed to unlock or control many devices. As a matter of fact, the development of efficient detection algorithms that verify the authenticity of audio recordings and help human listeners in discriminating fraudulent audio samples from real ones is therefore of paramount importance. Synthetic Speech Detection (SSD) algorithms are systems that estimate whether a speech signal under analysis has been synthetically created or has been authentically acquired by an audio recorder. However, this problem is getting challenging due to the constant development of new technologies and methods brought by deep learning for fake speech generation. For this reason, the study of new detection strategies is becoming increasingly urgent and necessary. In this thesis, some algorithms for the SSD task are proposed. The first approach uses the First Digit (FD) statistics computed on signal transform coefficients to detect peculiar characteristics of fake audio signals. The second method instead adopts Implicit Neural Representations (INRs) of speech signals, which are obtained with neural networks overfitted on each signal, to distinguish fake samples from bonafide ones. In both cases, it has been pointed out the fundamental role of silenced parts in synthetic speech detection. However, this thesis represents only a preliminary analysis, which we hope will help widening the perspectives of audio forensic research.

Sommario

La recente diffusione di dispositivi capaci di registrare segnali audio insieme al rapido sviluppo delle tecnologie deepfake hanno favorito una sempre più crescente diffusione di audio sintetici. Queste registrazioni di voci umane create sinteticamente sono sempre più convincenti e realistiche, a tal punto da poter essere utilizzate in modo malevolo, ad esempio per diffondere notizie false sui social media, per frodi o per attacchi di impersonificazione, poiché al giorno d'oggi i segnali vocali sono necessari per sbloccare e controllare numerosi dispositivi. È quindi di fondamentale importanza sviluppare efficienti algoritmi che verifichino l'autenticità delle registrazioni audio ed aiutino l'uomo a distinguere campioni audio potenzialmente fraudolenti da quelli reali. Definiamo come Synthetic Speech Detection (SSD) il problema di determinare se un certo segnale audio è stato creato sinteticamente oppure è reale. Questo problema sta diventando sempre più difficile da risolvere a causa del costante sviluppo di nuove tecnologie basate sul deep learning per la generazione di audio falsi. Per questo motivo, lo studio di nuove strategie per la rilevazione di audio falsi sta diventando sempre più urgente e necessario. In questa tesi vengono proposti alcuni algoritmi per far fronte al problema della SSD. Il primo approccio utilizza le statistiche First Digit (FD), calcolate su coefficienti estraibili dal segnale audio, per rilevare alcune caratteristiche tipiche di un audio falso che invece un audio reale non presenta. Il secondo metodo proposto adotta invece Neural Implicit Representations (NIRs) dei segnali vocali, che sono ottenute con reti neurali overfittate su ciascun singolo segnale, per distinguere i campioni falsi da quelli reali. In entrambi i casi è stato evidenziato il ruolo fondamentale delle parti silenziose nel rilevamento del parlato sintetico. Tuttavia, questa tesi rappresenta solo un'analisi preliminare, che speriamo possa aiutare ad ampliare le prospettive della ricerca nel campo dell'audio forense.

Contents

List of Figures

List of Tables

1 Introduction

In this chapter a brief overview of the problem and the contribution of this work are given. Details about the organisation of the thesis are provided.

***

Recent advancements in digital technologies such as smartphones, tablets and digital cameras have led to an exponential growth of multimedia content like audio recordings, images and videos. Moreover, the increasing social media usage has resulted in a massive generation of these content to be shared online and that can be accessed easily by anyone from any part of the world.

At the same time, the recent advances of Artificial Intelligence (AI) techniques have significantly increased the capability to produce realistic multimedia content and its quality. Therefore, it has become difficult to distinguish between real and fake content generated through highly advanced computer graphics and AI algorithms. These synthetically generated content like speech audio signals, images and videos can have useful applications in real life, but can also lead to various threats related to privacy and security through the so-called "Deepfake".
The term is a combination of the worlds "deep learning" and "fake", meaning that using deep learning techniques anyone can manipulate multimedia content or generate new fake ones to obtain even more credible audio signals, images and videos [85]. In a study conducted in 2021 by Statista, a leading company in data ranking and analysis, deepfake was placed among the five most diffused scenarios of AI-enabled cyberattacks worldwide (Fig. 1.1) [79].

Deepfakes caught people's attention and started to spread in autumn 2017, when a Reddit user under the pseudonym "Deepfake" used deep learning technology to generate a pornographic video by swapping the face of the original character for the face of some Hollywood actresses like Emma Watson, Katy Perry and Scarlett Johansson. The quality of these videos was high enough that it made it hard to distinguish them from real ones, thus this can be considered the initiator of deepfakes. Face manipulation is one of the most famous applications of deepfake and involves swapping one person's face in an image or video with the face of another person, while the original facial expressions, movements and surroundings remained unchanged [23]. An example of this deepfake application is reported in Figure 1.2.

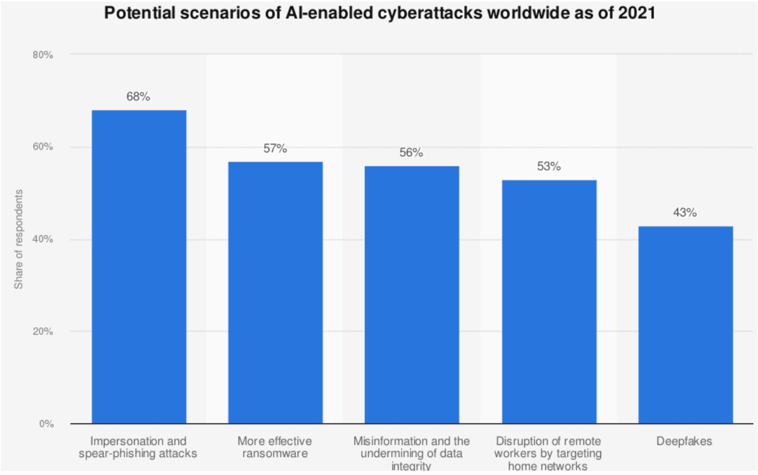


Figure 1.1: Types of AI-enabled cyberattacks 2021: 43% of survey respondents stated that AI can be used for deepfake attacks against their companies in the future. (from *Statista - The Statistics Portal*, 2022

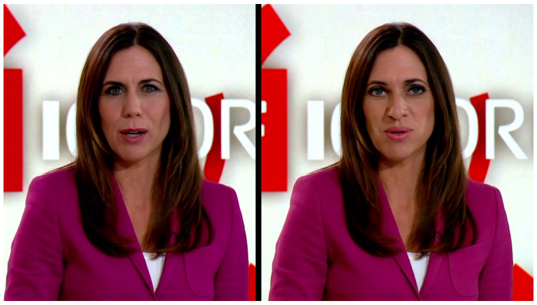


Figure 1.2: Example of a deepfake: (*left*) the original image; (*right*) the fake one.

Nevertheless, there's a whole range of useful deepfake applications as well, for example in the film industry. Indeed, the technology allows to revive a dead actor or to realistically dub films in different languages by matching the mouth movements of the actors with the spoken dialog. Similarly, this concept could also be applied to the advertising industry to use the face of a celebrity in more then one campaign. Photo shoots would no longer require the celebrities themselves to be present, but instead simply a person of similar stature.

However most of the times, deepfake technologies are used maliciously for example for pornographic or political purposes. Indeed they allow to counterfeit the identity of a person in a video by falsifying their face or voice [88] (for example to defame politicians [20] or innocent individuals [81]). Moreover, deepfake can be used to spread false information and fake news on the Internet or to attack the payment and authentication systems based on face or voice recognition.

Deepfake technology can do more than just swap faces. Essentially, any objects can be swapped as long as they have enough similarity in their basic features. For example, zebras can be transformed into horses, or summer photos can be turned into winter ones, or even natural photographs can be rendered into different artistic styles (Fig. 1.3). The only limit to the possibilities is a person's own creativity [104].

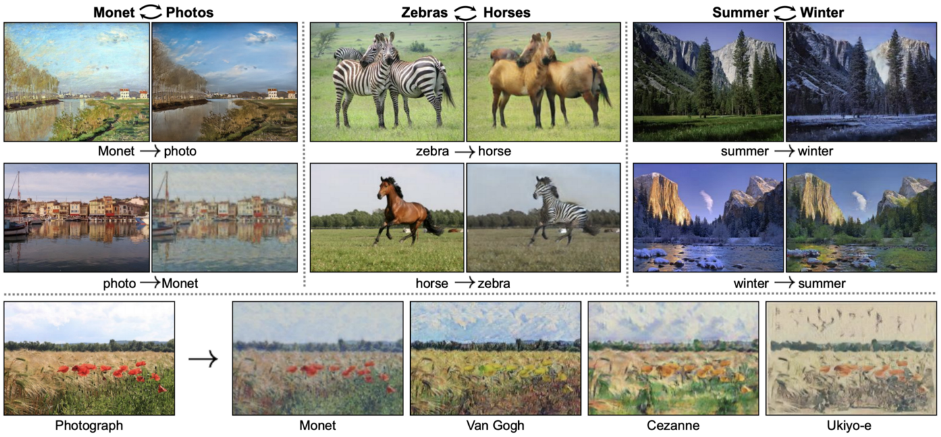


Figure 1.3: Another example of a deepfake in which an image is automatically "translate" into the other and vice versa: (*left*) Monet paintings and landscape photos; (*center*) zebras and horses; (*right*) summer and winter photos; (*bottom*) natural photograph and paintings rendered into the respective styles of famous artists.

In this work we put our attention on the application of deepfake to audio signals, which has only recently become a problem of significant interest to the research community. Many AI-generated tools have lately been developed with the ability to generate convincing voices [40], leading to a new technology known as Audio Deepfakes (ADs). However, while these tools were introduced to help people, they have also been used maliciously to manipulate public opinion for propaganda, defamation, or even terrorism [17]. Audio deepfakes are becoming widely accessible by everyone using only a simple smartphone or a personal computer [64]. Consequently, a massive amounts of voice recordings is broadcast daily over the Internet, but detecting fakeness from them constitutes a challenging task [43].

The problem of fake audio detection has become increasingly popular since there have been several criminal activities using fake audio in recent years. One of all, in 2019 an AI-based software was used to impersonate a CEO's voice and stole more than $243,000 via a telephone call [28]. Now more than ever, there is the need of fake audio detection techniques that can discriminate efficiently fraudulent audio samples from bonafide ones.

For this purpose, ADs have thus recently come to the attention of researchers, with several fake audio detection methods being developed to detect them, focusing on different types of acoustic features that are present in a real signal and, at the same time, are difficult to synthesize.

The majority of traditional strategies estimate fake audio characteristics from audio transform coefficients like Mel-Frequency Cepstral Coefficients (MFCCs) or Linear Predictive Coding (LPC) [30]. The general AD detection process is illustrated in Figure 1.4. Each audio clip should be preprocessed and transformed into suitable audio feature (as MFCC, LPC, etc.). These features are given as input to the detection model, which performs the training process. Then, the output is fed into a fully connected layer with an activation function to produce a prediction probability of real or fake class.

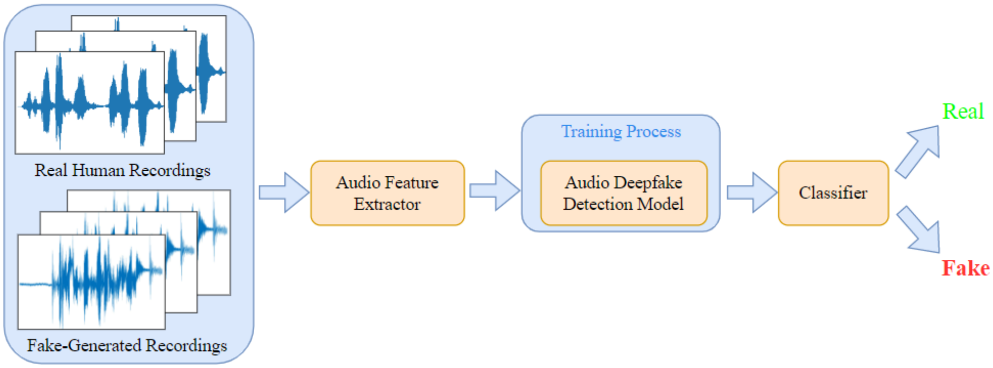


Figure 1.4: An illustration of the AD detection process.

Some other approaches are based on the effects of the physical acquisition environment on the signal (like reverberation, noise, etc.) [10, 37, 45] or on prosodic and emotional characteristics of the synthetic speech [13]. Other methods rely instead on statistics and symmetry properties of speech signals [71]. In the last years, more advanced methods that learn features representation by processing the analyzed audio with Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) architectures have become increasingly popular [103].

1.1 PROPOSED CONTRIBUTION

The objective of this work is to develop algorithms that aim at discriminating efficiently synthetic generated speech signals from real ones. Given a speech audio track, the goal consists in detecting whether the speech is synthetic (i.e., it has been generated through synthetic algorithms) or bonafide (i.e., it belongs to a real human speaker).

In order to validate the proposed methods on multiple kinds of synthetically generated speech signals, the publicly available ASVspoof 2019 dataset [83] was used. Some initial analyses have been carried out on it, highlighting some interesting insights related to its samples. According to these, it was possible to explain the outcomes of the proposed strategies.

The first suggested method is based on the First Digits (FD) statistics. This approach consists in extracting FD statistics from signal transform coefficients and shows how they can efficiently enable a robust detection. Indeed, it will be proved that the probability distribution of the FD statistics usually follows a pre-defined behavior that instead is completely altered whenever the signal is synthetic. Moreover, this work investigates the discriminative role in synthetic speech detection of silenced parts, which are present in many speech audio tracks of the ASVSpoof dataset [83].

The second strategy relies on Implicit Neural Representations (INRs) of speech signals. This method parametrizes each signal using a neural network that is trained to map 1D temporal coordinates of the input signal to its corresponding amplitude values. Each network is overfitted to its specific input and is able to reconstruct only this signal. It will be demonstrated that the trend of the training losses in terms of epoch vs reconstruction Mean Square Error (MSE), is different in most of the cases between fake and bonafide samples. Also in this case, it has been pointed out the essential role of silenced parts in synthetic speech detection.

1.2 Thesis Outline

In this thesis we describe the contributions just illustrated in the previous section by organizing them in three different chapters. In the following, we give some details about the organisation of the thesis. In Chapter 2 an overview of the theoretical background concerning both synthetic speech generation and synthetic speech detection is provided. Moreover, the ASVSpoof 2019 dataset [83], which is used to test the proposed methods, is presented. Chapter 3 describes how FD statistics extracted from MFCC coefficients can be used to detect synthetically generated speech signals. Then, chapter 4 illustrates how INRs of speech signals can be useful to discriminate between fake and bonafide samples. Finally, conclusions are drawn in Chapter 5.

2 Theoretical Background

In this chapter, the state of the art relative to synthetic speech generation and synthetic speech detection is introduced. Additionally, the ASVSpoof 2019 dataset is presented.

* * *

Thanks to the constant development of new technologies and neural networks, synthetic speech generation is nowadays an easy operation. Thus, it is becoming increasingly difficult to distinguish the synthetic audio material from original one. This can lead to dangerous consequences so much that in the last few years, also speech analysis research community has recognised the centrality of the synthetic speech detection problem. Consequently, many challenges have been organized to specifically addressed the problem of automatic speaker verification (ASV), like ASVSpoof 2019 [83] and ASVSpoof 2021 [98].

In this chapter, some backgrounds on state of the art algorithms for synthetic speech generation and synthetic speech detection are provided. This insight is useful to better understand the challenges that lie behind the synthetic speech detection problem. Finally, we provide a detailed description of the ASVSpoof 2019 [83] dataset, which has been chosen to evaluate the proposed methods.

2.1 Related Works

In the first part of this section, different approaches and latest trends in the field of synthetic speech generation are presented. Then, in the second part, we investigate the literature regarding the topic of this thesis, i.e., synthetic speech detection.

2.1.1 Synthetic Speech Generation

The aim of Synthetic Speech Generation (SSG) task is to automatically create speech samples which seem natural and perfectly comprehensible. Many techniques that achieve natural sounding results are presented in the literature, thanks to the recently advances of neural networks architectures.

People have tried to build machines to synthesize human speech dating back to the 12th century [93]. In the second half of the 18th century, an Hungarian scientist had constructed a speaking machine with a series of bellows, springs, bagpipes and resonance boxes to produce some simple words and short sentences [19]. The first speech synthesis system that built upon computer came out in the latter half of the 20th century [93].

The early computer-based speech synthesis methods include an articulatory synthesis, in which speech is produced by simulating the behavior of human articulator (like lips, tongue, glottis and moving vocal tract) [11]; a formant synthesis, in which speech is created on a set of rules that mimic the formant structure and other spectral properties of speech and that control a simplified source-filter model [4]; and a concatenative synthesis, which relies on the concatenation of pieces of speech that are stored in a database [25].

From 2010s, neural network-based speech synthesis has gradually become the dominant methods achieving much better voice quality [101, 102]. These synthetic speech generation methods can mainly be divided in two branches, i.e., text-to-speech and voice conversion algorithms. In the following, we first present text-to-speech methods, and then review some recent voice conversion techniques.

2.1.1.1 Text-To-Speech

Text-To-Speech (TTS) methods start from a textual representation of the speech and aim at producing the correspondent waveform signal. The first TTS approaches were largely based on waveform concatenation [6, 52], i.e., given a text as input, the output audio is produced by selecting the correct diphones from a large dataset of diphone waveforms and concatenating them so that intelligibility is ensured [25, 49]. The main problem of waveform concatenation is the complexity of modifying the voice timbral characteristics, for example to change the speaker, embed emotion in the voice or insert prosodic content in it.

Another proposed method to increase the naturalness of generated speech is the Statistical Parametric Speech Synthesis (SPSS). Given an input text, these models first process it into a sequence of phonemes and other linguistic features. Then, an acoustic model is built to learn and predict the mapping between the extracted linguistic features and some traditional acoustic features like duration, fundamental frequency, spectral envelope and excitation signal. Usually, this acoustic model is a Hidden Markov Model (HMM) trained on large datasets of acoustic features extracted from diphones and triphones [41]. Finally, it is defined a vocoder synthesizer to transform a spectral representation of the audio in the final raw waveform. Recently, the vocoder synthesizers are for example STRAIGHT [31], WORLD [47] and VOCAINE [1]. The simplicity of the SPSS approach makes it suitable for real-time scenarios, since it obtains good results with a reduced computational cost.

Additionally, neural networks have also been used to replace only portions of the SPSS systems to improve the results. Indeed, RNNs [89] have substituted HMMs in acoustic model and traditional vocoders have been replaced with neural vocoders. Some examples are WaveNet [70], which predicts samples of the waveform using convolutional layers in an auto-regressive setup, or LPCNet [86], which combines LPC analysis and RNNs to predict sample by sample a speech waveform.

Later, first end-to-end models have been proposed to overcome the problem of synchronisation between the acoustic and linguistic features. The even more powerful neural network has allowed to use more informative acoustic features, like mel-spectrograms and simpler linguistic features, like simple phoneme or characters. One example is Tacotron [92], based on seq2seq [76] architecture and attention mechanism. Given as input a sequence of characters, it produces the corresponding raw spectrogram, which is then transformed in a waveform using the Griffin-Lim algorithm [22]. Then a second version called Tacotron2 [69] was proposed. It improves the reconstruction of the waveform by predicting mel-spectrograms and using WaveNet as vocoder. The generated speech signals sound really natural and comprehensible, especially if compared to the one created with SPSS methods. Another examples of end-to-end TTS generation algorithms are Deep Voice [68], which roughly recalls the structure of SPSS systems and Deep Voice 3 [59], which consists of a fully convolutional network architecture.

2.1.1.2 Voice Conversion

Voice Conversion (VC) methods manipulate the voice signal to change the perceived identity of the speaker in the audio. Therefore, differently from TTS, the input is not text but a speech waveform.

VC pipelines are usually split in three blocks [72] a feature extraction step in which a suitable intermediate representation of the audio signal is extracted, a feature mapping step in which the modifications necessary to match the target characteristics are applied, and finally a reconstruction step in which the raw waveform is reconstructed. Each different VC method combines distinct techniques for each pipeline's block. For the feature extraction step, the chosen strategies are usually based on Pitch Synchronous Overlap and Add (PSOLA) [12] that represents the input as the parameters required by a vocoder synthesizer to reproduce it.

The use of a vocoder guarantees good quality in the final speech reconstruction, since these algorithms are well tested and efficient. On the other hand, it is not easy to adapt vocoder parameters to match the target voice characteristics and for this reason, alternative spectral representations are often adopted (like MFCC or LPC).

Regarding the mapping function, it can be implemented with parallel training, i.e., on the pairs of utterances of original and target speaker with the same content, or with non-parallel training data. Parallel training methods can be performed using Gaussian Mixture Model (GMM) [75] or more advanced neural network architectures [16]. More recently encoder-decoder architectures with attention mechanism has been proposed to implicitly learn the alignment between the input and the output [39]. On the other side, non-parallel training of the mapping function allows more flexibility on the choice of the training data. It can be performed by means of Generative Adversarial Networks (GANs) since the task is similar to image to image translation allowing similar techniques to be adopted [14].

2.1.2 Synthetic Speech Detection

Synthetic Speech Detection (SSD) can be defined as the task of estimating whether a speech signal under analysis has been synthetically created or it is bonafide. Since there is a wide variety of algorithms through which synthetic speech tracks can be generated, it is hard to find a general model suitable for all possible synthetic speech methods. Additionally, the continuous advances in deep learning allow the development of always new and better ways of generating fake speech. To try to overcome these difficulties, a series of detection algorithms has been proposed in literature to prevent the diffusion of fake speech recordings.

Traditional methods rely on the extraction of meaningful features from speech samples that will be used to discriminate between fake and real audio tracks like the Constant-Q Cepstral Coefficients (CQCC) [82], based on a perceptually inspired time-frequency analysis, Log Magnitude Spectrum, based on the magnitude, or Group Delay [95], based on the phase. Moreover, also other audio transform coefficients like Linear-Frequency Cepstral Coefficients (LFCC), Mel-Frequency Cepstral Coefficients (MFCC), Cepstral Mean and Variance Normalization (CMVN), Cochlear Filter Cepstral Coefficient (CFCC), Linear Prediction Cepstral Coefficient (LPCC) and many of their variations and combinations can be used [65]. Usually, one or a few of these features are used to train a Gaussian Mixture Model (GMM) or a Support Vector Machine (SVM) for classification. Taking the advantage of deep neural networks (DNNs) in classification tasks, multilayer perceptron (MLP) and CNN based classifiers have been used to replace the conventional back-end classifiers. On the other side, DNN structures have also been used at the front-end to facilitate feature extraction [60], followed by traditional classifiers.

Among the features listed above, CQCC has been found to be the best choice, which is also the baseline feature in the ASVspoof2019 challenge [91]. Recently, a set of subband CQCC features is introduced in [99] for better detection performance. Subsequently, in [15] eight hand-crafted features are further fused and followed by an MLP classifier. For deep learning based approach, Lavrentyeva et al. [35] proposed the use of Fast Fourier Transform (FFT), LFCC, and CMVN followed by a CNN for classification, while Li et al. [36] adopted a Res2Net structure and a squeeze-andexcitation (SE) block. Lavrentyeva et al. [35] and Li et al. [36] have achieved the state of the art performance on ASVspoof2019 dataset [83].

Neural network based techniques have proven very successful and effective for the SSD task. Some examples are [37], where the time frequency representations of the signals are fed to simple CNNs, and in [103] where the CNN is just exploited for feature extraction while a RNN is used for classification. In this case, several inputs have been tested, ranging from classic spectrograms to more complex novel features like Perceptual Minimum Variance Distortionless Response (PMVDR). In [80] instead, linear filter banks are fed into a Resnet to generate embeddings used as input of a neural network classifier, and in [30] long-term features are used to discriminate fake and real audio tracks.

Most recently methods to detect audio deepfakes that have been introduced in literature use the bicoherence matrix [3], long-short term features computed in an autoregressive manner [8], environmental cues [10], or even emotions [13].

Some approaches have also been directly applied to the raw input signal, i.e., in the time domain [77]. In particular, Rawnet2 [77] has achieved impressive results both for synthetic speech detection and user identification. This architecture is composed by a first layers made by a SincNet [63], i.e., a novel convolutional network that transforms the raw input with a band-pass filter bank for which the set of parameters is learnt during training. Then, three layers corresponding to three residual blocks are inserted. Finally, a Gated Recurrent Unit (GRU) and a fully connected layer are placed. Since this architecture has been proved to be successful for synthetic speech detection, it has been proposed as baseline in the recent ASVSpoof 2021 challenge [98].

2.2 Dataset

In order to test the proposed methods on different synthetically generated speech signals, we work on the publicly available ASVSpoof 2019 dataset [83, 91]. This dataset has been created for different tasks, from spoofing detection to countermeasures to replay attacks. Since we are focusing on the SSD problem, we only use a part of the dataset called logical access (LA) dataset.

This dataset is derived from the VCTK base corpus [97] which includes real speech data captured from 107 speakers (46 males, 61 females), and also contains synthetic speech tracks generated through 17 different speech synthesis techniques, ranging from the older based on waveform concatenation (WC), text to speech (TTS), voice conversion (VC), transfer function (TF) or non parallel voice conversion (NP) to novel ones based on CNNs approaches (NN). Here we describe synthetically each of them using the convention proposed in [91]:

- ***A01*** is a NN-based TTS system. It uses WaveNet [87], which is an efficient neural waveform generator.
- ***A02*** is a NN-based TTS system similar to A01 except that the WORLD vocoder [48] rather than WaveNet is used to generate waveforms.
- ***A03*** is a NN-based TTS system similar to A02 that can be easily built by using recipes in an open-source TTS toolkit called Merlin [94].
- ***A04*** A waveform concatenation TTS system based on the MaryTTS platform [66].
- ***A05*** is a NN-based VC system that uses a Variational Auto-Encoder (VAE) [24] and WORLD vocoder for waveform generation.
- ***A06*** is a transfer-function-based VC system [42]. This method analyzes the input voice signal following a source-filter model to replace a speaker voice into another speaker voice. The signal is synthesized using a vocoder and overlap-add technique.
- ***A07*** is a NN-based TTS system. The waveform is synthesized using the WORLD vocoder, and it is then processed by WaveCycleGAN2 [78], a time-domain neural filter that transforms output waveform of the vocoder into a natural-sounding waveform.

- ***A08*** is a NN-based TTS system similar to A01. However, A08 uses a neural-source-filter waveform model [90], which is much faster than WaveNet.
- ***A09*** is a NN-based TTS system [100] that uses Vocaine vocoder [2] to generate waveforms.
- ***A10*** is an end-to-end NN-based TTS system [27] that applies transfer learning from speaker verification to the neural TTS system Tacotron 2 [69]. The synthesis is performed through WaveRNN neural vocoder [29].
- ***A11*** is a neural TTS system that is the same as A10 except that it uses the Griffin-Lim algorithm [21] to generate waveforms.
- ***A12*** is a neural TTS system based on WaveNet. It produces high-quality waveforms.
- ***A13*** is a combined NN-based VC and TTS system that directly modifies the input waveform to obtain the output synthetic speech of a target speaker [34].
- ***A14*** is another combined VC and TTS system. It uses the STRAIGHT vocoder [32] for waveform reconstruction.
- ***A15*** is another combined combined VC and TTS system similar to A14. However, A15 generate waveforms through speaker-dependent WaveNet vocoders rather than the STRAIGHT vocoder.
- ***A16*** is a waveform concatenation TTS system that uses the same algorithm as A04. However, A16 is built given a different training set.
- ***A17*** is a NN-based VC system that uses the same VAE-based framework as A05. However, rather than using the WORLD vocoder, A17 uses a generalized direct waveform modification method [34] for waveform generation.
- ***A18*** is a non-parallel VC system [33] that uses a vocoder to generate speech from MFCCs.
- ***A19*** is a transfer-function-based VC system using the same algorithm as A06. However, A19 is built given different training set.

The dataset is divided in training set $\mathcal{D}_{\text{ASV tr}}$, development set $\mathcal{D}_{\text{ASV dev}}$ and evaluation set $\mathcal{D}_{\text{ASV eval}}$. The three partitions are disjoint in terms of speakers, and the recording conditions for all source data are identical (the sampling frequency is equal to 16000Hz).

In Table 2.1 are reported more specific details about the dataset. In particular, the training set $\mathcal{D}_{\text{ASV tr}}$ includes real speech from 20 (8 male, 12 female) subjects and synthetic speech generated from 6 methods (i.e., from A01 to A06). The development set $\mathcal{D}_{\text{ASV dev}}$ contains real speech from 10 (4 male, 6 female) subjects and synthetic speech generated with the same 6 methods used in $\mathcal{D}_{\text{ASV tr}}$ (i.e., from A01 to A06). The evaluation set $\mathcal{D}_{\text{ASV eval}}$ contains real speech from 48 (21 male, 27 female) speakers and synthetic speech generated from 13 methods (i.e., from A07 to A19). While the training and development sets contain speech signals produced with the same algorithms, the evaluation set also contains recordings generated with different and new ones. Notice that however A16 and A19 actually coincide with A04 and A06, respectively with small changes in the algorithms parameters. Consequently, $\mathcal{D}_{\text{ASV eval}}$ only shares 2 synthetic speech generation methods with $\mathcal{D}_{\text{ASV tr}}$ and $\mathcal{D}_{\text{ASV dev}}$, whereas 11 methods are completely new.

		$\mathcal{D}_{ASV\ tr}$	$\mathcal{D}_{ASV\ dev}$	$\mathcal{D}_{ASV\ eval}$	Category
Samples	**Real**	2580	2548	7355	
	Synthetic	22800	22296	63882	
	Total	25380	24844	71237	
Speakers	**Real**	20	10	48	
Synthetic Algorithms	**A01**	✓	✓		NN
	A02	✓	✓		VC
	A03	✓	✓		VC
	A04 = A16	✓	✓	✓	WC
	A05	✓	✓		VC
	A06 = A19	✓	✓	✓	VC
	A07			✓	NN
	A08			✓	NN
	A09			✓	VC
	A10			✓	NN
	A11			✓	NN
	A12			✓	NN
	A13			✓	NN
	A14			✓	VC
	A15			✓	VC
	A17			✓	VC
	A18			✓	VC

Table 2.1: ASVSpoof2019 dataset: training, development and evaluation compositions per number of samples, speakers, and synthetic algorithms. In particular, in the "Category" column specifies which method is used by the algorithm to synthetically generate speech signals, where NN = network, VC = vocoder and WC = waveform concatenation.

3 First Digit Features

In this chapter is investigated how first digit statistics extracted from MFCC coefficients can efficiently enable a robust synthetic speech detection.

***

In the following we present a strategy to address the problem of synthetic speech detection, i.e., discriminate fraudulent audio samples that have been synthetically generated from bonafide ones. The proposed procedure relies on First Digits (FD) statistics computed on signal transform coefficients [5], whose applications have been widely exploited in other multimedia contents [7]. More precisely, we show that FD statistics are successful in detecting fake audio samples generated by a set of algorithms and are extremely useful in highlighting the statistics of silenced parts. Finally, we design a simple classifier that can efficiently compete with more complex detectors in discriminating fake audios from bonafide ones. Even if the suggested strategy does not rely on large neural network architectures, it still obtains an accuracy above 90% in most of the examinated cases.

In this chapter we initially introduce the theoretical background related to the FD statistics and then we explain its application for detecting traces left in the signals by the synthetic algorithms. In the final part of this chapter, we apply this strategy to the ASVSpoof 2019 dataset [83] and we show the results achieved.

3.1 Background

First Digit (FD) law, also known as the Benford's law law or Significant Digit law, affirms that the statistical frequencies of the leading significant digits of a large dataset coming from real-life measurements (e.g., population numbers, stock prices, death rates, physical and mathematical constants, etc.), follow a peculiar distribution illustrated in Figure 3.1.

More precisely, the probability value of the d-th digit is computed as follows:

$$p(d) = log_{10}\left(1 + \frac{1}{d}\right) \tag{3.1}$$

where d is the FD in base 10.

Nowadays, first digit law is a well-defined probabilistic problem and it has been observed over a vast number of natural measurements. [62]. Additionally, it has also been noticed that this rule plays an important role in the detection of data altered by humans.

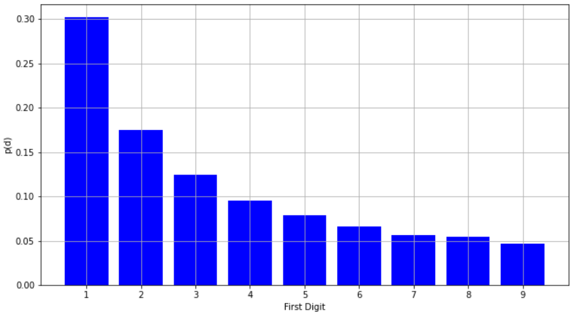


Figure 3.1: Benford's law FD pmf computed in base 10

Indeed, FD statistics from manipulated data do not perfectly follow Benford's law: whenever numbers are manipulated maliciously, the Benford's distribution is destroyed (i.e., FD frequencies diverge from their theoretical values) [18]. Consequently, this rule has been successfully applied to many forensics problems such as fake accounts, false financial reports and frauds detection [84] but also to multimedia forensics, with the aim to detect image tampering [56] and AI-generated images [7].

3.2 FD Features for Synthetic Speech Detection

In this section, we demonstrate that fake audio recordings do not respect the first digit law. In particular, in the first part we explain the rationale behind the proposed method. After that, we provide a formal definition of the FD analysis for SSD and report all the technical details about the detection strategy we propose.

3.2.1 Proposed Method

First digit law has been successfully used to detect multiple compressed data [45, 54] and more recently also to identify GAN generated images [7]. Starting from this point, it is possible to demonstrate that any synthetic signal generated by a set of FIR filters with limited support fits Benford's law with a different accuracy with respect to a natural signal.

The suggested method aims at detecting any possible relevant traces in the signal left from the synthetic algorithms and using them to distinguish between natural and fake audio recordings. Actually, this can be done by studying the statistics of quantized Mel-Frequency Cepstral Coefficients (MFCC), which are then fed to a classifier.

Initially, given as input a speech signal $x(t)$, we have obtained its representation in the frequency domain by computing the MFCC coefficients $m_w(f)$, where f is the considered frequency and w is the index of the frame. We have choosen the MFCC coefficients since they highlight the more meaningful frequency elements in speech signals [65].

By looking at the considered dataset, we have noticed that many samples sometimes contained long sequences of zeros that resulted in zero-valued MFCCs coeffiecients. Since computing FD statistics requires processing non-zero signals, we have decided to remove from the input data these zero values. We have verified on both training and test sets that this operation did not compromise the final results.

In order to obtain more informative features, MFCC coefficients were rescaled (to obtain different first digits quantization steps) with different quantization step Δ as

$$m_{w,\Delta}(f) = \frac{m_w(f)}{\Delta}. \tag{3.2}$$

Given b as integer representation base (e.g. 10 for decimal), the first digits related to $m_{w,\Delta}(f)$ were obtained as

$$d_{w,\Delta}(f) = \left\lfloor \frac{|m_{w,\Delta}(f)|}{b^{\lfloor log_b |m_{w,\Delta}(f)| \rfloor}} \right\rfloor. \tag{3.3}$$

We can notice that the FD can only assume values in $\{1, 2, \ldots, b-1\}$.

The probability mass function (pmf) for each distinct cepstral coefficient and for each quantization step can be computed as

$$p_{f,\Delta}(d) = \sum_{w=1}^{n_w} \frac{\mathbf{1}_d(d_{w,\Delta}(f))}{n_w} \tag{3.4}$$

where n_w is the number of windows in the signal (whose value depends on the duration of the audio and on the window overlap) and $\mathbf{1}_d(d_{w,\Delta}(f))$ is the indicator function for digit d, i.e.,

$$\mathbf{1}_d(x) = \begin{cases} 1 & \text{if } x = d \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

On the contrary, as it was already said previously, the pmf of natural audio signal must follow the generalized Benford's law defined as

$$\hat{p}_{f,\Delta}(d) = \beta log_b\left(1 + \frac{1}{\gamma + d^{\delta}}\right) \tag{3.6}$$

where β is a scale factor, γ and δ parameterize the logarithmic curve and $d \in \{1, 2, ..., b-1\}$ is one potential value that the considered FD in base b can assume.

The approximation accuracy between the pmf and the generalized Benford's law highly varies if we are considering bonafide w.r.t. forged data [57, 58]. As a matter of fact, such accuracy was measured using different distance and divergence measures to quantify the proximity of $p_{f,\Delta}(d)$ w.r.t. $\hat{p}_{f,\Delta}(d)$. In the rest of this section, we will omit indexes Δ and f for the sake of simplicity although in the creation of the final set of features multiple values of f and Δ were considered.

The first divergence metric that we have computed is the Shannon divergence

$$D^{JS}(p|\hat{p}) = D^{KL}(p|\hat{p}) + D^{KL}(\hat{p}|p). \tag{3.7}$$

which can be considered as a symmetrized version of the Kullbak-Leibler divergence $D^{KL}(p|\hat{p})$. Additionally, since such metric proves to be unstable for biased pmfs, we have computed also Renyi $D^{R}_{\alpha}(p|\hat{p})$ and Tsallis $D^{T}_{\alpha}(p|\hat{p})$ ($\alpha \in [0,1]$) divergences as

$$D^{R}_{\alpha}(p|\hat{p}) = \frac{1}{1-\alpha}\Big(logS_{\alpha}(p,\hat{p}) + logS_{\alpha}(\hat{p},p)\Big) \tag{3.8}$$

$$D^{T}_{\alpha}(p|\hat{p}) = \frac{1}{1-\alpha}\Big(2 - S_{\alpha}(p,\hat{p}) - S_{\alpha}(\hat{p},p)\Big) \tag{3.9}$$

where

$$S_{\alpha}(p,q) = \sum_{d=1}^{b-1} \frac{p(d)^{\alpha}}{q(d)^{\alpha-1}} \tag{3.10}$$

Since Shannon, Renyi and Tsallis divergences can be highly correlated for certain values of the parameter α (in our specific case $\alpha = 0.3$ is used), we also added the Mean Square Error (MSE)

$$D^{MSE}(p,\hat{p}) = \frac{1}{b-1}\sum_{d=1}^{b-1}(p(d) - \hat{p}(d))^2. \tag{3.11}$$

We have adopted also this metric in some preliminary tests where the divergences of original and fake audios were compared. We have discovered that the three divergences often agree, i.e., in the original sample they are always smaller than those in the forged sample or vice-versa. This statement does not always hold for MSE.

Finally, we have obtained a total number of features n_f equal to

$$n_f = n_d n_c n_b n_q \tag{3.12}$$

where n_d is the number of divergences, n_c is the number of chosen cepstral coefficients, n_b is the number of basis for the first digit extraction and n_q is the number of different Δ parameters.

With this procedure, it is possible to prove that Benford's law is not respected anymore by fake speech signals since the modification introduced by the synthetic algorithms redistributes data among the bins of the quantizer. Indeed, the final pmf presents some oscillating probability values that deviate from the ideal distribution. That's why we have computed the divergences between the empirically estimated $p_{f,\Delta}(d)$ and its ideal fitted version $\hat{p}_{f,\Delta}(d)$ to discover whether an audio recording is natural or it has been synthetically generated.

3.2.2 Problem Formulation

Let us consider a speech signal $x(t)$ sampled at sampling frequency Fs. Our goal is to identify whether it is bonafide or a fake. For this reason, we are in a binary classification problem in which we associate to each speech signal $x(t)$ a label

$$y = \begin{cases} 0 & \text{if } x(t) \text{ is bonafide} \\ 1 & \text{if } x(t) \text{ is fake.} \end{cases} \tag{3.13}$$

Therefore given an audio signal, the proposed procedure aims at obtaining and $\hat{y}$ i.e. an estimate of the ground-truth label y.

To reach this goal, we will proceed in two steps: a features extraction and a supervised classification. The features extraction phase returns for each speech signal the corresponding feature vector $\mathbf{f} = \mathrm{F}(x(t))$, where the function $\mathrm{F}(\cdot)$ assigns to the speech recording a more concise and informative representation. The classification block instead is responsible of assigning the correct label to a specific signal. It can be described by the function $\mathrm{C}(\cdot)$ that can assume two values: $\mathrm{C}(\mathbf{f}) = 0$ or $\mathrm{C}(\mathbf{f}) = 1$ depending on whether the signal is bonafide o fake.

3.2.3 Detection Method

The feature extraction block is shown in Figure 3.2. Given an audio signal $x(t)$, we have divided it in n_w windows and for each of them we have computed the MFCCs, which were then quantized with a quantization step Δ.

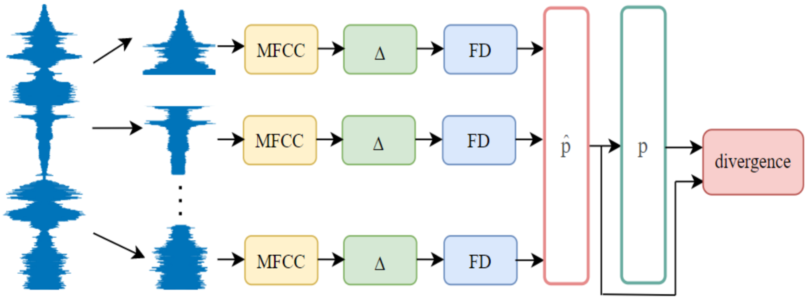


Figure 3.2: Feature extraction pipeline.

In agreement with (3.3) and given a base b, we have calculated the first digit statistic of the f-th quantized MFCC frequency sample from the w-th block. Then the estimated pmf

$p_{f,\Delta}(d)$ was computed according to (3.4). An example of these pmfs for both bonafide and fake signal is reported in Figure 3.3. It is possibile to see that the pmf associated to the fake signal presents some oscillating probability values that deviate more from the ideal distribution compared to the bonafide one. Other examples of pmfs computed with different frequency numbers, quantization steps and bases can be found in Appendix A.

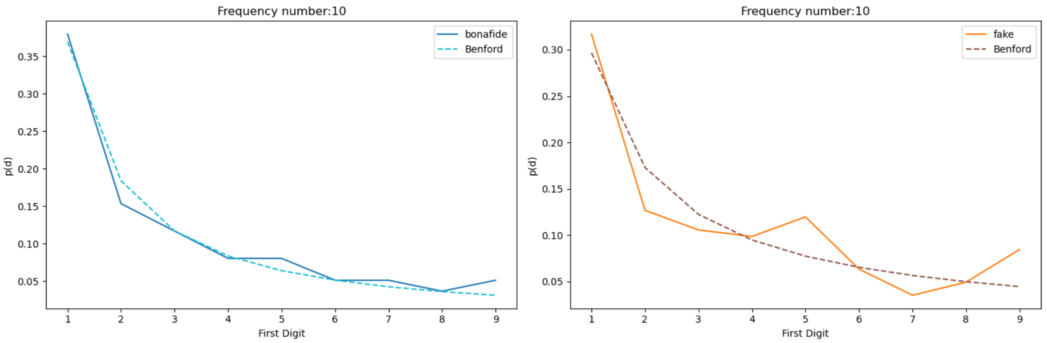


Figure 3.3: Pmf $\hat{p}$ for bonafide (*blue*) and fake (*orange*) speech signal compared to the ideal Benford curve (*dashed light blue* and *red* respectively) for frequency number 10 and base 10.

In the end, we have calculated the Shannon D^{JS} (3.7), Renyi D^{R} (3.8) and Tsallis D^{T} (3.9) divergences and finally the mean square error D^{MSE} (3.11).

Given a set $\mathcal{B}$ of bases, a set $\mathcal{F}$ of MFCC frequencies and a set $\mathcal{Q}$ of quantization steps Δ, we have concatenated all the computed values in a final feature vector as

$$\mathbf{f}_{\mathcal{B},\mathcal{F},\mathcal{Q}} = [D^{JS}, D^{R}, D^{T}, D^{MSE}]_{b\in\mathcal{B}, f\in\mathcal{F}, \Delta\in\mathcal{Q}}. \tag{3.14}$$

Consequently, this feature vector was fed to a basic supervised classifier, which in this case is a Random Forest classifier.

3.3 Application

In the following section we present the dataset preprocessing and the technical details related to our experiment. Finally, we show and discuss the results obtained on the ASVSpoof 2019 [83].

3.3.1 Dataset Preparation

In order to test the proposed method on different synthetically generated speech signals, we work on the ASVSpoof 2019 dataset [83]. The work by Muller et. al. [50] highlights the presence of a bias in the distribution of the lengths of leading and trailing silences in real and synthetic speeches. Given that, probably most detectors are just discriminating between fake and real samples by using this information. In order to avoid this problem, silent parts were removed from the signal, as suggested in [50] but this led to a big loss in performance.

For this reason, we have analyzed the effectiveness of FD features on the silent (without considering leading and ending silences) and voiced parts of the signals, independently. In this way, we understood which speech elements were the most discriminative and whether the proposed approach was reliably effective. For this purpose, we chose signal windows of 101 samples with energy $E(s,t)$ higher than -40 dB (assuming energy is normalized).

Since the number of silent values was sufficient to obtain meaningful statistics, only a few samples (less than 1%) were then removed. However, this is not a big issue since as shown in [50], the very low amount of silence in the audio track allows an easy detection of synthetic audio samples. Moreover, computing FD statistics on a limited amount of signal windows would lead to highly irregular statistics: this implies strong divergences/distances with respect to Benford's law (and therefore, a correct classification).

Starting from the original data, three datasets called respectively *Full*, *Silence* and *Voiced* have been created. In particular, *Full* contains the whole samples, *Silence* comprises the silent parts of the signals and *Voiced* includes the remaining samples.

3.3.2 Experimental Setup

Our feature vectors are built as described in (3.14) and the following parameter values were selected after some optimizations:

- In the computation of MFCCs, we select all the MFCC frequencies $\mathcal{F} = \{1, 2, ..., F\}$, where a filter bank of 26 filters was adopted: only coefficients from the second to the fourteenth frequency were considered. Computation was carried out on window sizes of 1024 samples with an overlap of 512 in the case of *Full* and *Voiced*. Overlap was set to 128 in the case of *Silence* to have a sufficient number of signal windows (and therefore stable FD statistics).
- The base for the first digit was chosen as $\mathcal{B} = \{10, 20\}$ since higher values would imply only a few samples (or no samples at all) for many FD values.
- The quantization factor Δ varied in the set $\mathcal{Q} = \{1, 2, 3, 4\}$.

At the end our feature vectors were composed by $n_f = 420$ features.

Since the amount of features was not huge, we avoided the adoption of complex neural network classifiers and instead we selected a low complexity classifier. For this reason, a simple random forest classifier was chosen as it proved well suited for tabular data processing and highly robust w.r.t. overfitting problems and unbalancing. We used the implementation provided with the open source e ScikitLearn Python library [55]. A grid search to tune the parameters is performed on the following set of values:

- number of decision trees, $n_{trees} \in \{10, 100, 500, 1000\}$
- split criteria, $criterion \in \{gini, entropy\}$

while all the other parameters are left as their default values.

For this experiment we consider only a portion of $\mathcal{D}_{\text{ASV tr}}$. Indeed, it originally contains 2580 natural and 22800 generated recordings, but for our binary classification problem we would prefer a balanced dataset. For this reason, we extract the same number of real and fake audio, i.e., all the 2580 natural audio and only 2580 audio generated by all the algorithms (specifically, 430 recordings for each algorithm). Then, we use the 80% of this new version of the training set to actually train the classifier, while the remaining 20% is used for parameters tuning. In this way, we ensure that every generated class has the same number of audio traces and that the total amounts of bonafide and synthetic data are balanced (the adopted classifier is a random forest which has low overfitting problems so the data used for training is plenty). Then we test the method on both development set $\mathcal{D}_{\text{ASV dev}}$, which consists of samples generated with the same algorithms used for synthesised the training samples, and evaluation set $\mathcal{D}_{\text{ASV eval}}$, which comprises also new synthesis algorithm.

3.3.3 Results

Here, we report the results obtained by applying a random forest classifier to the feature vectors described before to detect whether a speech signal is real or fake. For each features configuration, we run an indipendent grid search to assure a fair comparison between the various configurations.

In table 3.1 are reported the one-vs-one on-set results achieved by performing binary classification between bonafide and generated samples. In particular, we show the accuracy in detecting fake speech signals based on the used synthetic algorithm. Instead, in Table 3.2 are reported the one-vs-one off-set accuracy. In both cases, all the results are reported for *Silence*, while for *Full* and *Voiced* only the best results are shown.

Dataset	**Development**					
Algorithm	**A01**	**A02**	**A03**	**A04**	**A05**	**A06**
Silence Δ=1	0.944	0.962	0.961	0.819	0.949	0.471
Silence Δ=1-2	**0.953**	0.972	0.970	0.829	0.961	0.472
Silence Δ=1-3	0.951	0.972	0.972	0.836	**0.964**	0.466
Silence Δ=1-4	0.952	0.973	0.972	0.838	0.963	0.456
Silence b=10	0.945	0.959	0.961	0.830	0.924	0.468
Silence b=20	0.866	0.973	0.881	0.796	0.957	0.434
Full Δ=1-4	0.951	**0.982**	**0.949**	**0.871**	0.956	0.424
Voiced Δ=1-3	0.755	0.708	0.713	0.548	0.574	**0.532**

Table 3.1: On-set results and ablation studies for the proposed algorithm

Regarding the quantization parameter Δ, we incrementally concatenated the features computed with different Δs in order to measure how much each specific feature affected the final performance. We noticed that that using only one Δ value is usually not enough to achieve the best performance. With several trials, we discovered that in general a quantization step equal to 3 or 4 allow to maximize the performance.

With respect to the base value, we observed that keeping the features generated by both $b = 10$ and $b = 20$ (with all the selected values of Δ) was more effective than selecting only one base value. Indeed, the statistics generated for $b = 10$ were not very correlated with those obtained for $b = 20$, and therefore, merging them provides additional information to the system.

From table 3.1 and 3.2, we can see that the performance obtained on *Silent* is comparable to or even better to the one obtained for *Full* especially in the off-set tests. On the other hand, removing silences from the signals leads to very poor performance on *Voiced*. This means that synthetic algorithms are able to reconstruct realistic voices more easily, while the noise present in silent sequences instead cannot be easily modelled. Additionally, the slightly lower performance achieved on *Full* w.r.t. *Silence* might be explained by the presence of spoken parts in some bonafide samples that might deviate the FD statistic towards the ideal FD distribution.

The lower performance are obtained by algorithms A06, A17, A18, A19, which perform a voice conversion task starting from real audio samples as input and converting them into voiced samples for a desired speaker. However, also the algorithm A05 is a VC algorithm but in this case the achieved accuracy is higher. This happends because the task is carried out by a neural network that processes the entire sequence with the silence included leading to FD statistics that are detected by our approach.

Dataset	**Evaluation**												
Algorithm	**A07**	**A08**	**A09**	**A10**	**A11**	**A12**	**A13**	**A14**	**A15**	**A16**	**A17**	**A18**	**A19**
Silence Δ=1	0.946	0.948	0.955	0.947	0.947	0.952	0.953	0.931	0.876	0.860	0.597	0.615	0.592
Silence Δ=1-2	0.953	0.953	0.965	0.954	0.956	0.959	0.960	0.939	**0.888**	0.861	0.598	0.626	0.597
Silence Δ=1-3	**0.955**	**0.957**	**0.968**	**0.956**	**0.957**	**0.962**	**0.962**	0.941	**0.888**	0.864	0.600	0.629	**0.599**
Silence Δ=1-4	0.951	0.955	0.965	0.952	0.956	0.960	0.959	**0.942**	0.887	0.864	**0.601**	0.625	0.598
Silence b=10	0.925	0.933	0.944	0.927	0.929	0.936	0.928	0.912	0.860	0.846	0.598	**0.642**	**0.599**
Silence b=20	0.919	0.897	0.929	0.924	0.924	0.903	0.945	0.889	0.842	0.820	0.590	0.579	0.593
Full Δ=1-4	0.941	0.942	0.952	0.939	0.940	0.915	0.951	0.896	0.853	**0.866**	0.597	0.581	0.596
Voiced Δ=1-3	0.656	0.796	0.798	0.629	0.648	0.628	0.687	0.720	0.709	0.640	0.526	0.533	0.580

Table 3.2: Off-set results and ablation studies for the proposed algorithm

Moreover, A16 and A04 approaches show an higher misclassification probability. These algorithms are based on waveform concatenation, i.e., signals are obtained by concatenating real samples from big databases of real diphones. Such composite nature makes the synthetic speech FD statistics closer to that of bonafide samples leading to a lower detection accuracy.

On the other hand, an high accuracy is reached by the algorithms A13, A14, and A15, in which voice converted samples are generated starting from TTS outputs. This could happen because VC algorithms do not change drastically silenced parts as they are not relevant in characterizing speaker ID and present a completely different statistics w.r.t. voiced parts. In these cases, the statistics of the original silenced intervals are not altered leading to a higher misclassification probability. However, this result is not verified whenever VC is applied after TTS since in that case also the generated nature of silence leads to non-conventional FD statistics, which lead to higher divergences.

3.4 Final Remarks

In this chapter, we focused on the effect of voiced and silenced parts in synthetic speech detection. After some analysis on silences, we pointed out that most of the discriminative information in a speech signal is contained in the silenced parts. Consequently, we proposed a method for synthetic audio detection based on first digit statistics. Despite having very low computational complexity, this approach obtained good detection performance against different types of algorithms. Numerical results showed that most algorithms are able to produce statistically meaningful voice signals but often fail at creating realistic silences, especially in the case of algorithms based on neural networks.

4 Implicit Neural Representations

In this chapter a synthetic speech detection method that uses neural networks for implicit neural representations of speech signals is presented. The results obtained on the ASVSpoof 2019 dataset with the proposed solution are reported.

* * *

The proposed strategy relies on Implicit Neural Representations (INRs), which is a novel way to parameterize signals of all kinds using neural networks [73]. It has emerged as a powerful paradigm, offering many possible benefits over conventional representations. Indeed in recent years, the introduction of neural representations has completely changed the generation and representation of images in deep learning [67]. Although much attention has been attracted for neural representation in images, few studies have been done in the audio domain.

The suggested method implements a neural network for INRs of speech signals to address the problem of synthetic speech detection. More specifically, each signal is parametrised using a neural network that is trained to map 1D temporal coordinates of the specific input speech signal to its corresponding amplitude values. This procedure is repeated for all the available audio recordings, so that a way that each network is overfitted to its specific input and is able to reconstruct this signal only. This operation is performed also on silent and voiced parts of the signals, independently. By comparing the trends of the training losses in terms of epoch vs reconstruction MSE, in most of the cases a different behaviour between fake and bonafide samples is observed. However, the noise presents in silent sequences proves to be useful and almost necessary for an efficient synthetic speech detection.

In this chapter the theoretical background related to the INRs technique is introduced. Then, it is explained how this representation of speech signals can be used efficiently for detecting synthetically generated audio. In the final part of this chapter, this strategy is applied to the ASVSpoof 2019 dataset [83] and the obtained results are discussed.

4.1 Background

Implicit Neural Representations (INRs), also known as coordinate-based neural representations, are a class of techniques to parametrise signals using neural networks [38, 53]. This paradigm has recently attracted the attention of the machine learning community for their capability of representing complex signals, especially in computer vision, 3D rendering, and image synthesis applications [44, 46].

Even if the world around us is not discrete, real-world signals such as images or sound are usually represented in a discrete manner. For example, images are represented as a grids of pixels, 3D shapes as grids of voxels, point clouds, or meshes, and audio signals as discrete samples of amplitudes (Fig. 4.1). However, discrete representations come with a significant drawback: they only contain a discrete amount of information regarding the signal. For example, the amount of information we have for an image is bound by the size of the pixel grid.

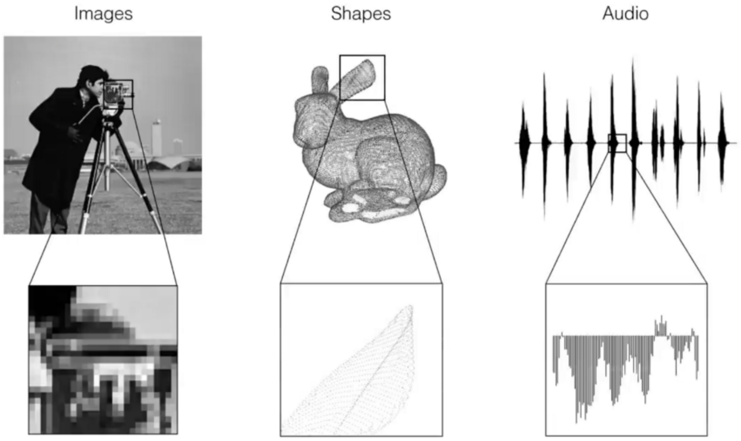


Figure 4.1: Discrete representations of various signals.

In contrast, INRs parameterize a signal as a continuous function that maps the domain of the signal to whatever is at that coordinate. In the case of images, this function will take as input the 2D pixel coordinates and outputs the correct RGB value for that specific pixel. In this way, we could sample pixels at any resolution, without the constraints of the pixel grid. Regarding the speech signals instead, this function will parameterize the signal as a mathematical formula such that it will take as input the 1D temporal coordinate and outputs the correct amplitude value at that specific time. As V. Sitzmann from MIT's Scene Representation Group writes, such functions are too complex to simply *"write them down"* [73] and thus this method is introduced. Indeed, neural implicit representations use neural networks that learn how to estimate a function which mapping the domain of the input signal to whatever is at that coordinate (Fig. 4.2).

4.2 INR for Synthetic Speech Detection

In this section, we demonstrate that the performances of the neural network in terms of MSE are different between bonafide and fake audio recordings, thus these can be used for

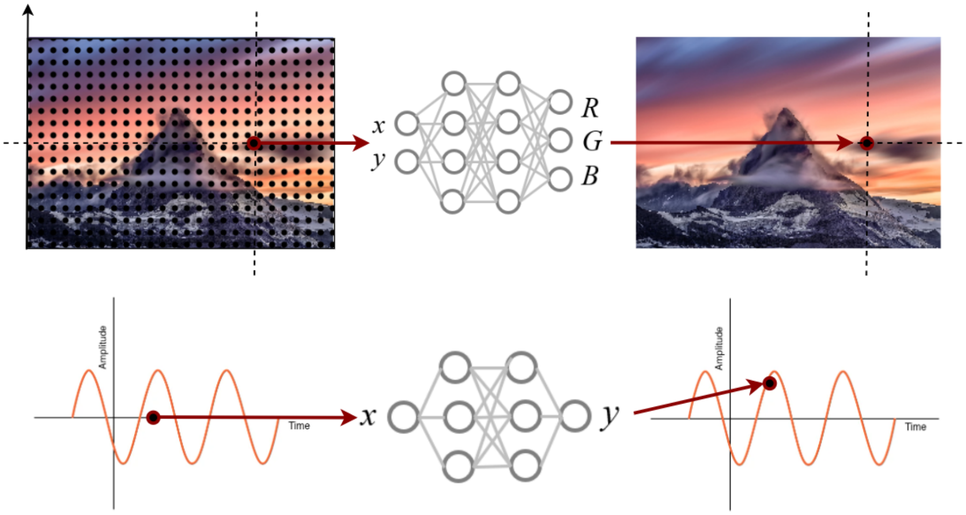


Figure 4.2: Implicit neural reprentations for images and speech signals.

the SSD task. In particular, in the first part we explain the rationale behind the proposed method. After that, we provide a formal definition of the SSD problem and report all the technical details about the detection strategy we propose.

4.2.1 Proposed Method

The application of INR to time-series data has been relatively underdeveloped. Representation of time varying 3D geometry has been explored [51], but they do not investigate time-series data. Although SIREN [74] showed the capability to represent audio, its focus was limited to the high-quality representation of the input signals. More recently in 2022, [26] uses INR to solve the problem of time-series anomaly detection.

In this thesis we propose the use of INRs of speech signals to detect synthetically generated audio. More in details, let denote a speech signal as

$$X = \{(t_1, y_1), (t_2, y_2), (t_3, y_3), \dots, (t_N, y_N)\} \tag{4.1}$$

where t_i denotes the 1D temporal coordinate, y_i is the corresponding amplitude value, and N indicates the length of the sequence. After preprocessing the time coordinate input via an encoding function γ (see Section 4.2.1.1), our aim is to learn a function $f_\theta : \mathbb{R} \mapsto \mathbb{R}$ with parameters θ that maps the encoded time $\gamma(t_i)$ to its corresponding value y_i of the data, i.e.,

$$f_\theta[\gamma(t_i)] = y_i \qquad i = 1, \dots, N. \tag{4.2}$$

This function is different for each specific audio and it can be learned by a neural network, which is overfitted to that particular sample. Indeed after training, the estimated function would be implicitly encoded in the neural network, hence the name "Neural Implicit Representation".

To represent a given speech signal, we build a fully connected neural network, resulting in a simple yet powerful model capable of representing audio (see Section 4.2.1.2).

4.2.1.1 Positional Encoding

Recent works have highlighted that INRs benefit from computing sinusoidal transformations of the coordinates with a positional encodings (PEs) [46, 96], which enables the neural network to represent higher frequency functions.

Indeed, a network f_θ that directly operates on the input t_i performs poorly at representing high-frequency variation in amplitude. This is consistent with recent work by [61] which shows that deep networks are biased toward learning lower frequency functions. They additionally show that mapping the inputs to a higher dimensional space using high-frequency functions before passing them to the network enables better fitting of data that contains high-frequency variation. We leverage these findings in our context, and we reformulate f_θ as a composition of two functions:

$$f_\theta = f'_\theta \circ \gamma. \tag{4.3}$$

Here γ is fixed and it is a mapping from $\mathbb{R}$ into a higher dimensional space $\mathbb{R}^{2L}$, and f'_θ is learned and it is still simply a regular neural network. Formally, the encoding function we use is:

$$\gamma(p) = \Big(\sin(2^0\pi p), \cos(2^0\pi p), \cdots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)\Big). \tag{4.4}$$

This function $\gamma(\cdot)$ is applied separately to each coordinate values t_i, which are previously normalized to lie in [0, 1].

4.2.1.2 Architecture

We implement a fully connected neural network without any convolutional layers, often referred to as a ReLU-based Multi-Layer Perceptron (MLP).

Since the network is trained on only one sample the time, there is no need to build a very deep neural network. Indeed, a deep network has many layers and consequently many parameters, which require an huge amount of time to be trained without improving performance too much given the reduced dataset size. For this reason, a fully connected feed-forward network with only three hidden layers is defined to perform this specific task. As activation function for the hidden layers, the ReLU function is chosen. However, other activation functions (like tanh, sigmoid,. . .) have also been tried, but the ReLU function was preferred since it is less prone then the others to the vanishing gradient problem, which could impair the networks capability to learn. Instead, a sigmoid activation function is used for the output layer since it produces a vector where each element lies in the range [0, 1] (the inputs were previously normalized in this range). Before the output layer, dropout is introduced. This technique aims to contain overfitting effects, since during the training phase it randomly ignores some neurons according to a certain probability. The entire architecture is summarized in Figure 4.3.

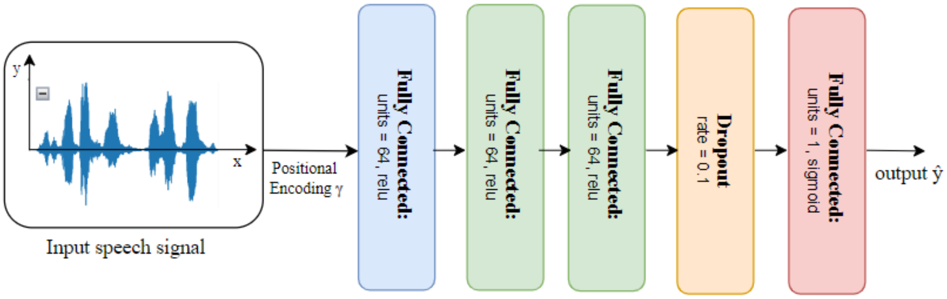

y
x
Input speech signal
Positional Encoding γ
Fully Connected: units = 64, relu
Fully Connected: units = 64, relu
Fully Connected: units = 64, relu
Dropout rate = 0.1
Fully Connected: units = 1, sigmoid
output $\hat{y}$

Figure 4.3: MLP architecture

To accelerate the SGD procedure we choose Adam (Adaptive Moment Estimation) as optimization algorithm, being it the most recent and reliable among the others (e.g. Adadelta, RMSprop,. . .) with two parameters: the learning rate and the L2 regularization (weight decay) to avoid network weights taking too large values.

The selected loss function is the Mean Square Error (MSE) which computes the squared L2 norm between true and predicted values. In other words, we compare the predicted value at each time i to its ground-truth value y_i, resulting in the following optimization problem:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left\|y_i - f_\theta[\gamma(t_i)]\right\|_2^2. \tag{4.5}$$

where the sum is over all temporal values and $\|\cdot\|$ indicates the L2 norm.

4.2.2 Problem Formulation

Given a speech signal, our goal is to identify whether it is bonafide or a fake (assigning a label to it). This is the same binary classification problem described in Section 3.2.2 of the previous chapter.

We will try two different strategies. The first will be based on the computation of Δ_{MSE}, i.e., the difference between the MSE of the last and of the first epoch (see Section 4.2.3.1). In the second method, we will perform a linear and a polynomial fit on the MSE curves (see Section 4.2.3.2). In both approaches, we will put a discriminative threshold respectively on Δ_{MSE} and on fitting coefficients to distinguish bonafide from spoof signals.

4.2.3 Detection Method

In the following, we explore two different strategies to address the problem of synthetic speech detection using INRs. We develop these methods starting from the comparison of the trends of the losses in terms of epoch vs reconstruction MSE. In particular, we produce some interesting plots for both voiced and silent parts of the signals, as the ones shown in Figure 4.4. In these plots, the solid lines represent the trend of the mean of the MSE of all bonafide and spoof training samples, while the shaded regions fill the space between the min and max values of the MSE. The same plots for development and evaluation sets are reported in Figures A.2, A.3 of Appendix A.

Also in this case, we can see that most of the information useful for the discrimination between bonafide and spoof signals is contained in the silenced parts. Indeed, the loss curves of voiced parts show an equal behaviour for both bonafide and spoof samples. For this reason, in the rest of the following analysis, we will focus only on the *Silence* version of the datasets (defined in Section 3.3.1). Indeed for all the three *Silence* parts of the datasets, a different trend between fake and bonafide losses is observed. On the basis of a first visual analysis, it seems that bonafide losses decrease down faster than the spoof ones. Starting from this observation, we develop two different detection methods to verify this hypothesis.

4.2.3.1 Thresholding of Δ_{MSE}

At this point, we want to verify that bonafide MSE trends decrease more rapidly than the spoof ones, as the number of epochs increase. To do so, we compute for each sample the Δ_{MSE} that we define as

$$\Delta_{MSE} = MSE_{n_{epochs}} - MSE_{10_{epochs}} \tag{4.6}$$

where $MSE_{n_{epochs}}$ and $MSE_{10_{epochs}}$ are the MSE values at the last epoch and at epoch number 10 respectively. Initially the difference was computed between the last and the first epoch, but worst results were obtained. Since the the MSE shows an unstable and oscillating behavior in the first epochs, we decided to exclude these values from the computation of Δ_{MSE}.

Our goal is to find a discriminative threshold between bonafide and spoof Δ_{MSE}, i.e., we would like to find a threshold such that

$$\begin{cases} \Delta_{MSEbonafide} > threshold \\ \Delta_{MSEspoof} < threshold. \end{cases} \tag{4.7}$$

If these inequalities are respected by the majority of the samples, we will have demonstrated that in the neural networks trained on bonafide samples, the MSE values decrease more than the ones of spoof samples (considering the same number of epochs).

In other words, we want to find a threshold such that the inequalities defined in Equation 4.7 are verified. To do this, we compute the Receiver Operating Characteristic (ROC)

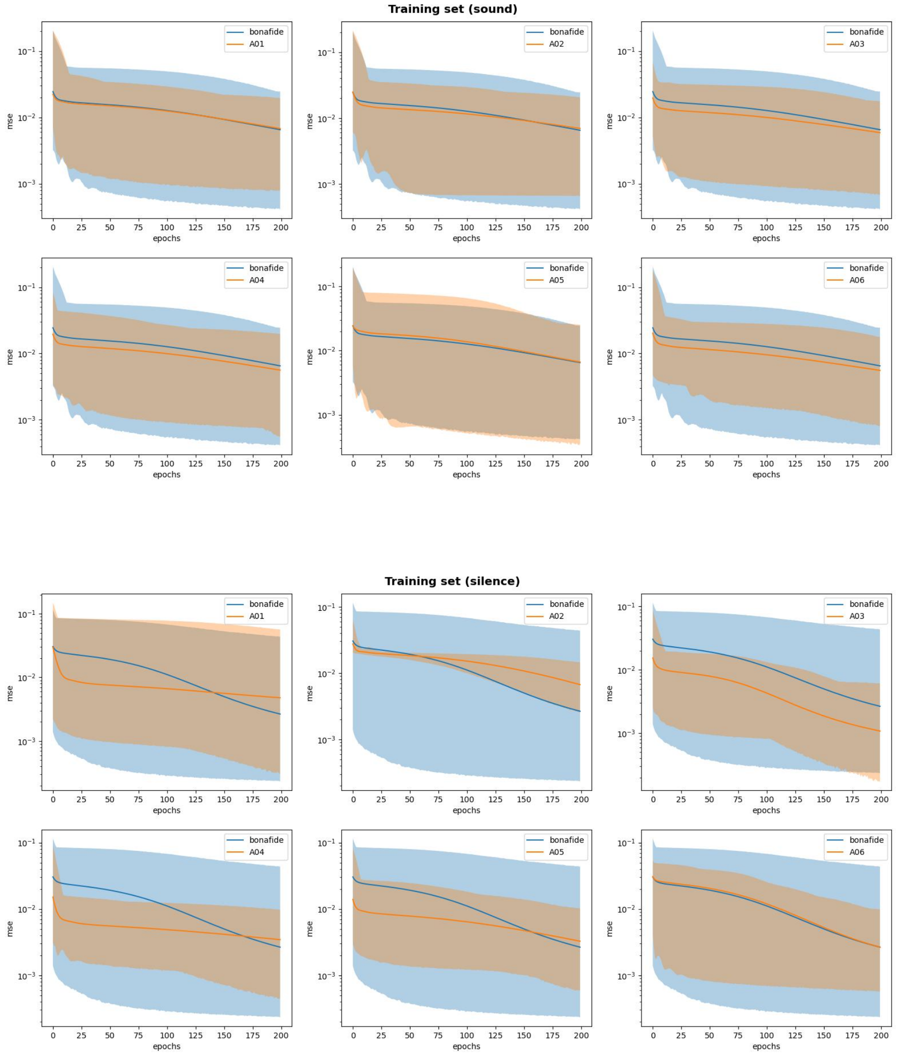
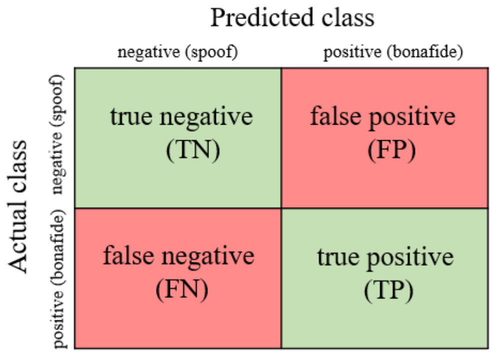


Figure 4.4: Trends of the MSE means (*solid lines*) and ranges between min and max MSE values (*shaded regions*) for both bonafide and spoof signals for *Voiced* and *Silence* version of the training set.

curves. An ROC curve is a graph showing the performance of a classification model at all classification thresholds.

In binary classification, there are four possible outcomes for a test prediction: true positive, false positive, true negative, and false negative (Fig. 4.5). In our case, we define as positive class (label=0) the bonafide category.



		Predicted class	
		negative (spoof)	positive (bonafide)
Actual class	negative (spoof)	true negative (TN)	false positive (FP)
	positive (bonafide)	false negative (FN)	true positive (TP)

Figure 4.5: Confusion matrix structure for binary classification problems

The ROC curve is produced by calculating and plotting the true positive rate against the false positive rate for a single classifier at a variety of thresholds. One advantage presented by ROC curves is that they aid us in finding a classification threshold that suits our specific problem.

In our case, the threshold would be the predicted probability of a Δ_{MSE} values belonging to the positive class. Normally, if a Δ_{MSE} value is predicted to be positive at > 0.5, it is labeled as positive. However, we could really choose any threshold between 0 and 1 (0.1, 0.3, 0.6, 0.99, etc.) and ROC curves help us visualize how these choices affect classifier performance.

The true positive rate (TPR) can be represented as:

$$TPR = \frac{TP}{TP + FN} \tag{4.8}$$

where TP is the number of true positives and FN is the number of false negatives. The true positive rate is a measure of the probability that an actual positive instance will be classified as positive.

The false positive rate (FPR) can be written as:

$$FPR = \frac{FP}{FP + TN} \tag{4.9}$$

where FP is the number of false positives and TN is the number of true negatives. The false positive rate is essentially a measure of how often an actual negative instance will be classified as positive.

The threshold that guarantees the inequalities defined in equation 4.7 is choosed as the one that maximizes the difference between true positive rate and false positive rate, i.e.,

$$threshold = \arg \max_{thresholds} (TPR - FPR) \tag{4.10}$$

4.2.3.2 *Thresholding of MSE fitting coefficients*

Now we develop a more detailed analysis of the loss curves. In a first step we perform a linear curve fitting, i.e., we fit our curves shown in Figure 4.4 to a straight line. Given the equation of a generic straight line, defined as

$$f(x) = mx + q \tag{4.11}$$

where m is the slope and q is the y-intercept of the line, we want to find the coefficients m and q that fit the data in a least-squares sense.

To discriminate between bonafide and spoof samples, we would like to find a threshold on the slope coefficients such that

$$\begin{cases} m_{bonafide} < threshold \\ m_{spoof} > threshold. \end{cases} \tag{4.12}$$

The threshold is found using ROC curves with the method described in the previous section. If these inequalities are verified by the majority of the samples, we will have demonstrated that bonafide losses decrease down faster than the spoof ones.

To obtain a better fit, we also perform a polynomial curve fitting, i.e., we fit our curves with a second-degree polynomial defined as

$$f(x) = ax^2 + bx + c. \tag{4.13}$$

We want to discover the coefficients a, b and c that better fit the curves.

Since in this case we want to find a threshold on two coefficients (a and b), we will use a Support Vector Machine (SVM) classifier to discriminate between bonafide and spoof signals.

4.3 Application

In this section we describe the dataset preprocessing and the technical details related to our proposed methods. Then, we present and discuss the results obtained on the ASVSpoof 2019 dataset.

4.3.1 Dataset Preparation

In order to test the proposed methods, we adopt the ASVSpoof 2019 dataset [83] that contains the three datasets $\mathcal{D}_{\text{ASV tr}}$, $\mathcal{D}_{\text{ASV dev}}$ and $\mathcal{D}_{\text{ASV eval}}$. As it was already said in Section 4.2.3, we will focus our attention only on the *Silence* version of the datasets.

However, all the three datasets include speech signals with different lengths. When we train our neural network on these samples individually, this might be a problem and can compromise the final results. Consequently, we cut all the signals to the length of the shortest audio of the specific dataset. In this way, the *Silence* versions of $\mathcal{D}_{\text{ASV tr}}$, $\mathcal{D}_{\text{ASV dev}}$ and $\mathcal{D}_{\text{ASV eval}}$ datasets will contain only signals with the same length.

4.3.2 Experimental Setup

Our neural network is built as described in Section 4.2.1.2 and the following parameter values were selected after some optimizations:

- In the function $\gamma : \mathbb{R} \rightarrow \mathbb{R}^{2L}$ of the positional encoding, we set $L = 16$.
- In the neural network, we use three fully connected layers with $n_{units} = 64$ neurons each.
- The probability of the dropout layer is set to $p = 0.1$.
- Each network is trained for a maximum of $n_{epochs} = 200$ epochs.
- The learning rate adopted by the adam optimizer is set to $\lambda = 0.001$.

In this experiment we consider only a portion of the three dataset $\mathcal{D}_{\text{ASV tr}}$, $\mathcal{D}_{\text{ASV dev}}$ and $\mathcal{D}_{\text{ASV eval}}$: we prefer a balanced dataset. For this reason, we extract the same number of real and fake audio. In particular, for the $\mathcal{D}_{\text{ASV tr}}$ dataset we select the 2580 natural audio and only 2580 audio generated by all the algorithms (specifically, 430 recordings for each algorithm). Instead for the $\mathcal{D}_{\text{ASV dev}}$ dataset, we use 2544 bonafide audio and 2544 spoof audio (424 signals for each algorithm). Finally, for the $\mathcal{D}_{\text{ASV eval}}$ dataset we consider only 7345 natural audio and 7345 generated audio (565 samples for algorithms). In this way, we guarantee that the total number of bonafide and spoof audio is balanced and that there is the same amount of generated audio for each synthetic algorithm.

4.3.3 Results

Here, we report the results obtained by the two methods described before to detect whether a speech signal is real or fake.

4.3.3.1 *Thresholding of* Δ_{MSE}

In Table 4.1 are reported the results obtained with a discriminative threshold on Δ_{MSE} values. In particular, we show the accuracy in detecting fake speech signals based on the used synthetic algorithm.

The lower performance are obtained for the algorithms A06 and A19, which perform a voice conversion task starting from real audio samples as input and converting them into voiced samples for a desired speaker.

On the other hand, an high accuracy is reached for the algorithms A04 and A16. These algorithms are based on waveform concatenation, i.e., signals are obtained by concatenating real samples from big databases of real diphones.

	Dataset		
Algorithm	**Training**	**Development**	**Evaluation**
A01	0.888	0.912	
A02	0.862	0.900	
A03	0.850	0.903	
A04	0.929	0.961	
A05	0.880	0.925	
A06	0.504	0.544	
A07			0.685
A08			0.799
A09			0.864
A10			0.674
A11			0.717
A12			0.862
A13			0.589
A14			0.874
A15			0.887
A16			0.881
A17			0.704
A18			0.827
A19			0.514

Table 4.1: Results for the Δ_{MSE} method

In Figure 4.6 are reported the statistics of Δ_{MSE} values of the training dataset, i.e., frequency histograms that present on the horizontal axis the Δ_{MSE} values and on the vertical axis the frequencies with which each different value occurs. Moreover, the thresholds obtained with the ROC curves and used for the classification bonafide vs spoof are shown. The same plots for development and evaluation datasets are reported in Figure A.4 in Appendix A.

As we can see, the adopted thresholds separate quite well bonafide statistics from spoof ones for the majority of the algorithms. The statistics of bonafide and spoof overlap only for algorithms A06 and A19. In these cases, the threshold is not able to separate bonafide histogram from spoof ones.

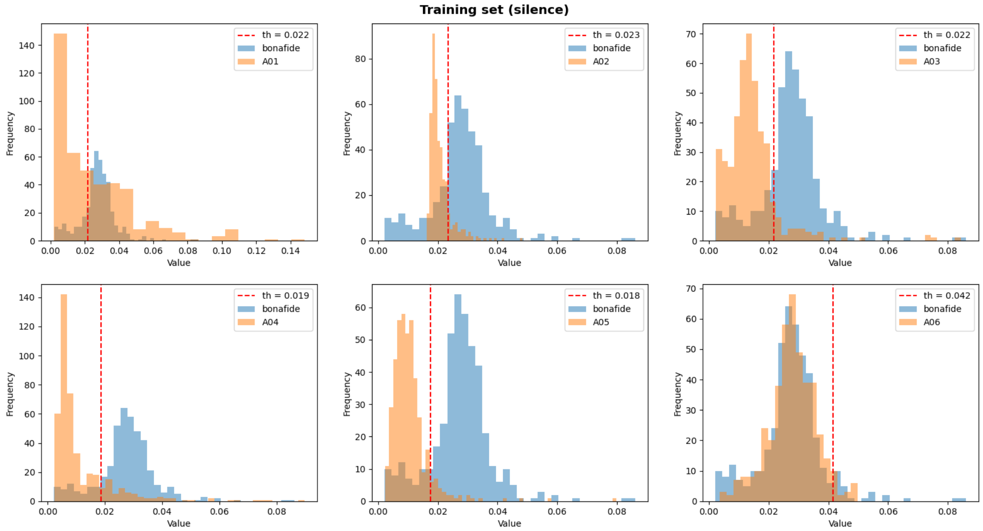


Figure 4.6: Δ_{MSE} values vs frequency of occurrence for bonafide (*blue*) and spoof (*orange*) training samples. The discriminative threshold is highlighted (*dashed red line*).

4.3.3.2 *Thresholding of MSE fitting coefficients*

The results obtained with the linear curve fitting are not as good as we expected. In Figure A.5 in Appendix A the MSE curves are fitted to a straight line and the slope coefficients are reported.

The classification accuracies reached by placing a threshold on the slope coefficient of the fitting line are shown in Table A.1 in Appendix A. As we can see, this approach is not able to discriminate between bonafide and spoof samples.

For this reason, we try to fit our curves with a second-degree polynomial (the results are shown in Figure A.6 in Appendix A for development and evaluation datasets).

In Table 4.2 are reported the classification accuracies reached by placing a threshold on the polynomial fitting coefficients. With this approach, slightly better results are obtained. The lower accuracies are obtained by voice conversion algorithms A06, A17, A18, A19. Instead, better results are obtained by algorithms based on CNNs approaches, such as A11 and A13.

	Dataset		
Algorithm	**Training**	**Development**	**Evaluation**
A01	0.529	0.500	
A02	0.821	0.781	
A03	0.585	0.530	
A04	0.715	0.711	
A05	0.777	0.739	
A06	0.522	0.481	
A07			0.733
A08			0.658
A09			0.789
A10			0.754
A11			0.809
A12			0.702
A13			0.800
A14			0.784
A15			0.667
A16			0.668
A17			0.585
A18			0.597
A19			0.519

Table 4.2: Results for the polynomial curve fitting method.

4.4 Final Remarks

In this chapter, we proposed two methods for synthetic audio detection based Implicit Neural Representations. In particular, we focus on studying the trends of the neural network losses of bonafide and spoof samples. We find out that the neural networks trained on bonafide samples have a loss that decrease faster than the spoof ones. This observation is used to discriminate bonafide from spoof sample by looking only at the trend of the losses. This approach obtained discrete detection performance against different types of algorithms.

5 Conclusions

In this chapter the overall work is summarized while final conclusions and remarks are drawn. Finally, a perspective towards future works is presented.

* * *

In this thesis we presented two different research perspectives for audio forensics analysis and we proposed solutions that use classic signal processing techniques or recent ML methodologies. We mainly focused on Synthetic Speech Detection (SSD) task, which is the task of estimating whether a speech signal under analysis has been synthetically generated or it is real. This research theme is not lacking of challenges, given the number and the variety of methods for creating synthetic speech.

In Chapter 2 we described the state of the art algorithms for synthetic speech generation and synthetic speech detection to better understand the challenges that lie behind the SSD problem. Moreover, we provided a detailed description of the dataset that we used to tested our methods. We chose the ASVSpoof 2019 dataset, a large dataset containing speech signals that are real or that are created by several different synthesis algorithms.

In Chapter 3 we presented a first approach to address the SSD problem. This method uses the First Digit (FD) statistics computed on MFCC coefficients to detect peculiar characteristics of fake audio signals.

In Chapter 4 we introduced a second method that instead adopts Implicit Neural Representations (INRs) of speech signals, which are obtained with neural networks overfitted on each signal, to distinguish fake samples from bonafide ones.

In both methods, we analyzed the impact of voiced and silent parts in synthetic speech detection. After some preliminary analysis and comparisons, we pointed out that silent parts within the speech contain most of the discriminative information, thus they have a fundamental role in synthetic speech detection. Indeed, we tested our methods separately on voiced and silent parts of the speech signals. Empirical results showed that most audio forging algorithms are able to produce statistically meaningful voice signals but often fail at creating realistic silences.

Indeed on silence parts, both methods achieves good detection performance against a variety of algorithms. Instead, lower performances are obtained in both cases on voice conversion algorithms, which start from real audio samples as input and convert them into voiced samples for a desired speaker. A possible explanation for this evidence is that VC algorithms do not change significantly silenced sections as they are not relevant in characterizing speaker ID. In these cases, the original silenced parts are not altered and this leads to a higher misclassification probability.

Comparing the numerical results, the first procedure based on FD statistics reaches better resutls. Indeed, it is computationally-lightweight and effective on many different algorithms since it does not rely on large neural detection architecture and obtains an accuracy above 90% in most of the classes of the ASVSpoof dataset.

5.0.1 Future Works

Future works should try to tackle the problem of detection in a voice conversion scenario (possibly by integrating this with other well working state of the art approaches) since the transformation of a naturally acquired signal could retain most of the statistics for the silenced parts thus leading to a higher misclassification probability.

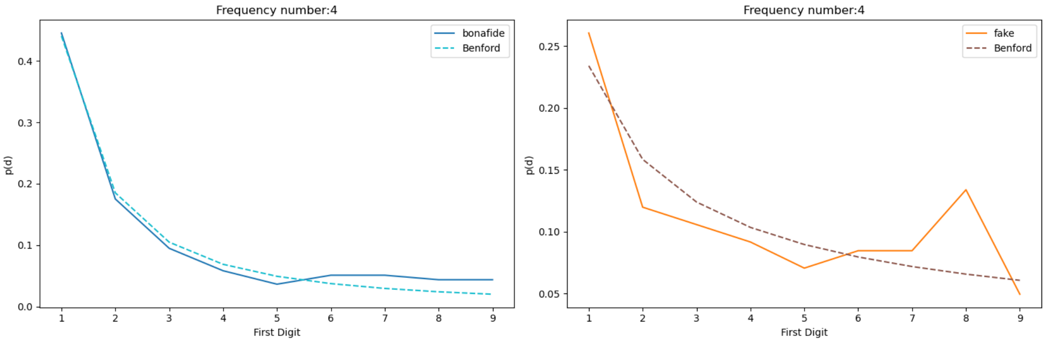
Moreover, the FD statistics method may be applied on different types of audio features (e.g., FFT, LPC, ..) or may be combined with a similar state of-the-art algorithm (i.e. with separate feature extraction and classification steps). For example with the work by Borrelli et. al. [9], which exploits short-term and long-term (STLT) cues and the bicoherence matrix to extract a discriminative representation between spoof and bonafide samples. For the INRs method, other more complex neural architectures may be investigated (e.g., more layers may be added or some parameters may be changed). Furthermore, other analysis may be done on the loss curves of the neural networks or additional useful information may be extracted from them. Also, both methods may be tested on other datasets due to the huge variety of synthetic speech generation methods that exist.

Despite the achieved promising results, several scenarios need further investigation. For example, we only considered the logical access synthetic speech detection problem, i.e., we analyze a clean recording of each speech. It is therefore part of our future studies to consider what happens if speech tracks get corrupted by noise, coding, or transmission errors. This scenario is particularly important if we consider that synthetic speech recordings may be shared through social platforms or used live during phone calls.

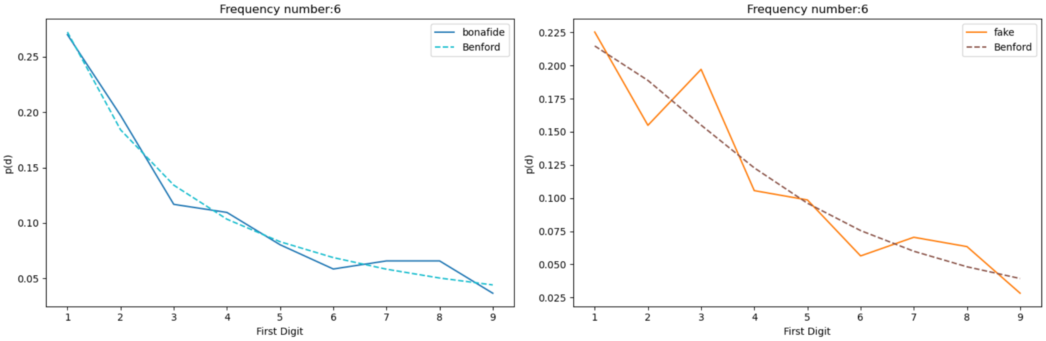
A Appendix

In this chapter additional data concerning the analyses led on the dataset are reported.

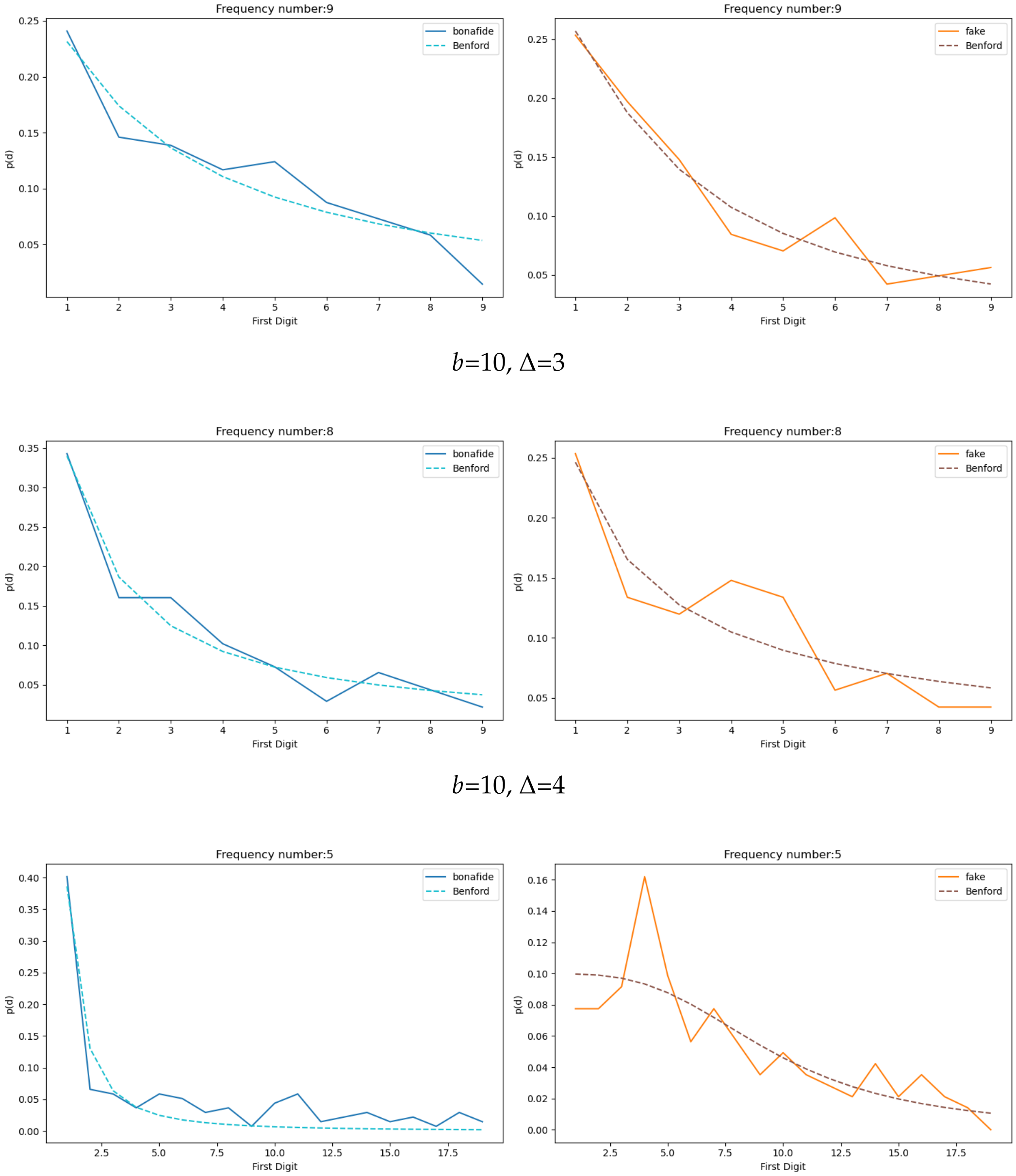
* * *



b=10, Δ=1



b=10, Δ=2



b=10, Δ=3

b=10, Δ=4

b=20, Δ=1

Figure A.1: Pmf $\hat{p}$ for bonafide (*blue*) and fake (*orange*) speech signal compared to the ideal Benford curve (*dashed light blue* and *red* respectively) for different frequency numbers, quantization steps Δ and bases b.

	Dataset		
Algorithm	**Training**	**Development**	**Evaluation**
A01	0.500	0.500	
A02	0.504	0.501	
A03	0.644	0.724	
A04	0.500	0.500	
A05	0.500	0.500	
A06	0.551	0.509	
A07			0.500
A08			0.509
A09			0.500
A10			0.500
A11			0.500
A12			0.500
A13			0.500
A14			0.500
A15			0.500
A16			0.500
A17			0.500
A18			0.502
A19			0.524

Table A.1: Results for the linear curve fitting method.

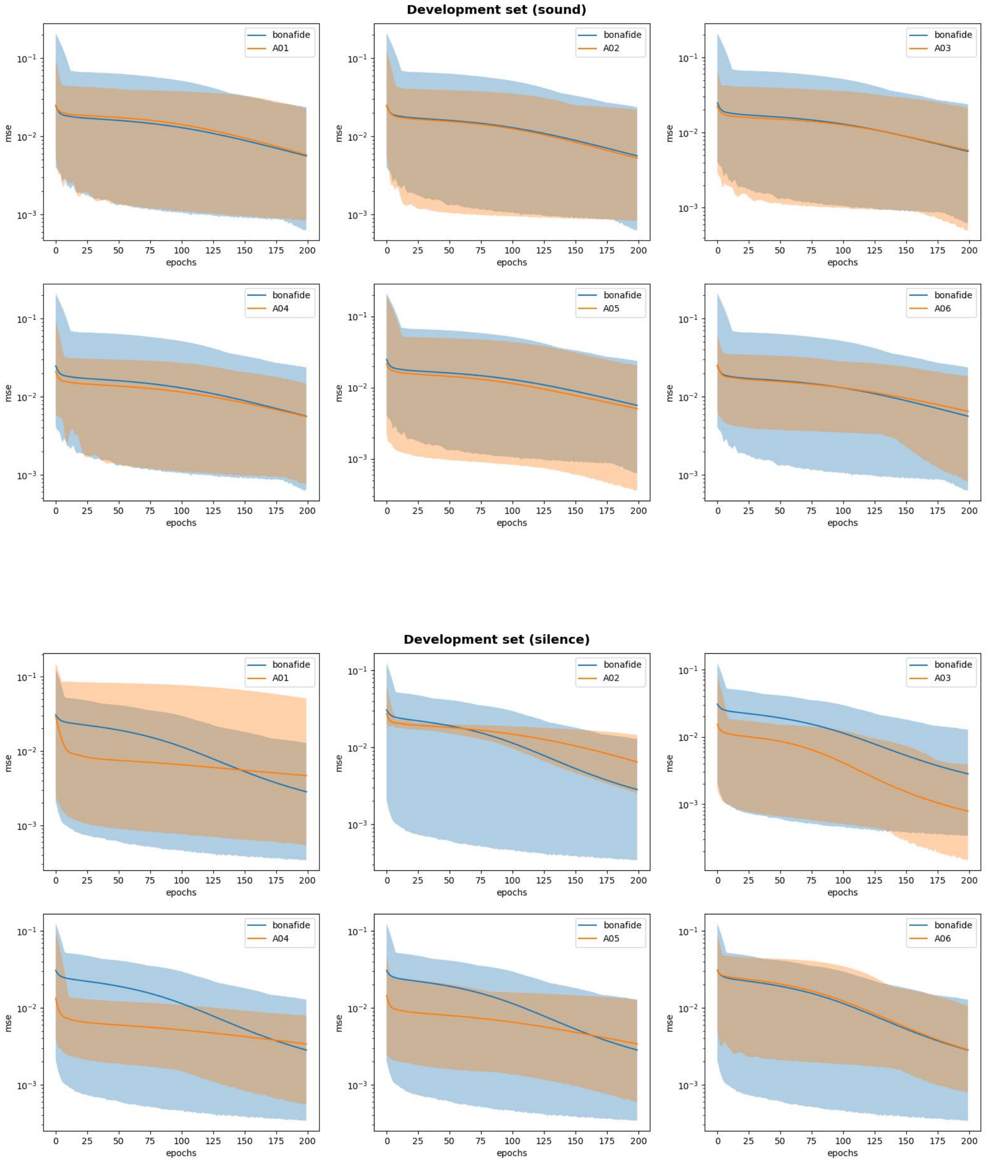


Figure A.2: Trends of the MSE means (*solid lines*) and ranges between min and max MSE values (*shaded regions*) for both bonafide and spoof signals for *Voiced* and *Silenced* version of the development set.

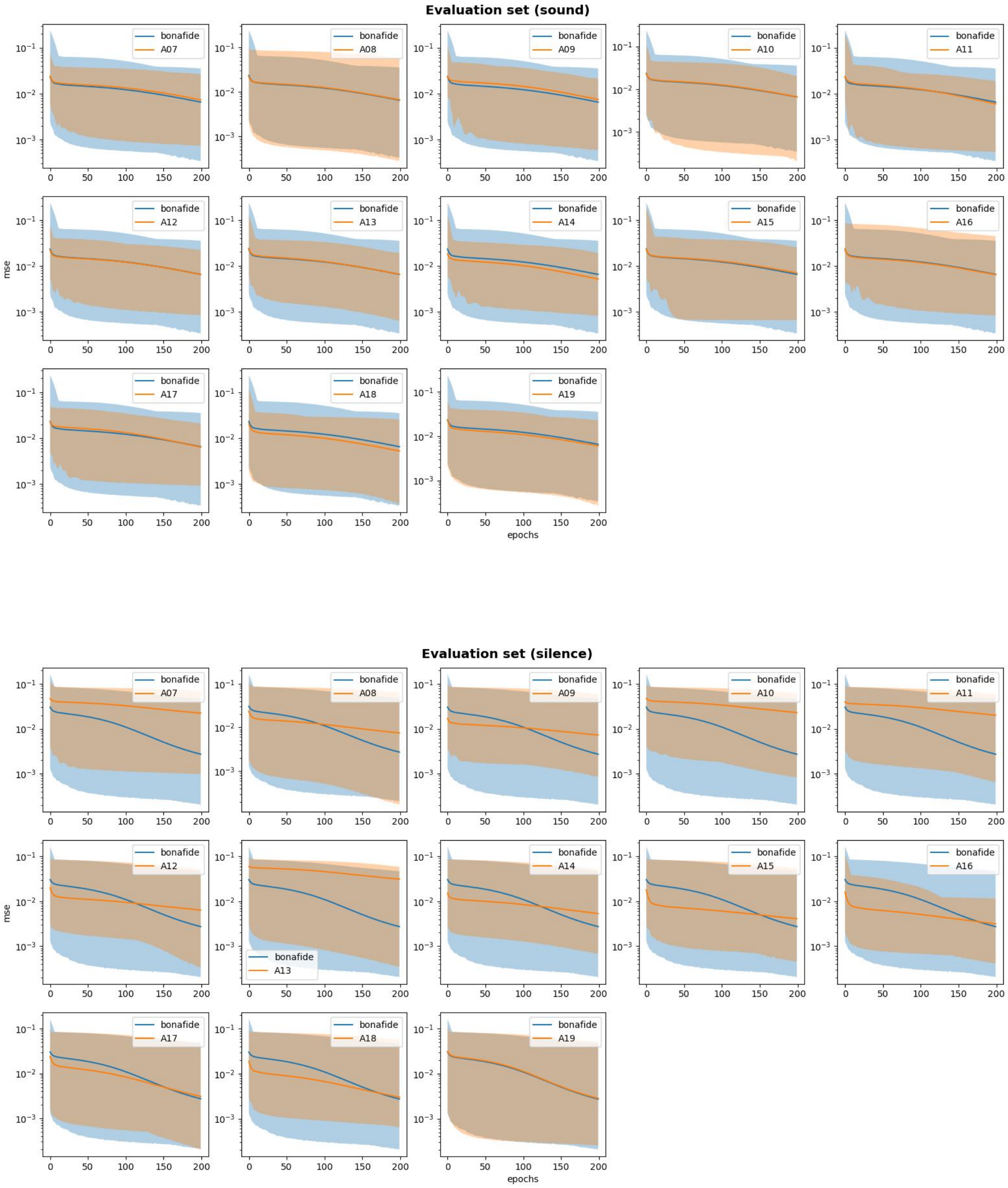


Figure A.3: Trends of the MSE means (*solid lines*) and ranges between min and max MSE values (*shaded regions*) for both bonafide and spoof signals for *Voiced* and *Silenced* version of the evaluation set.

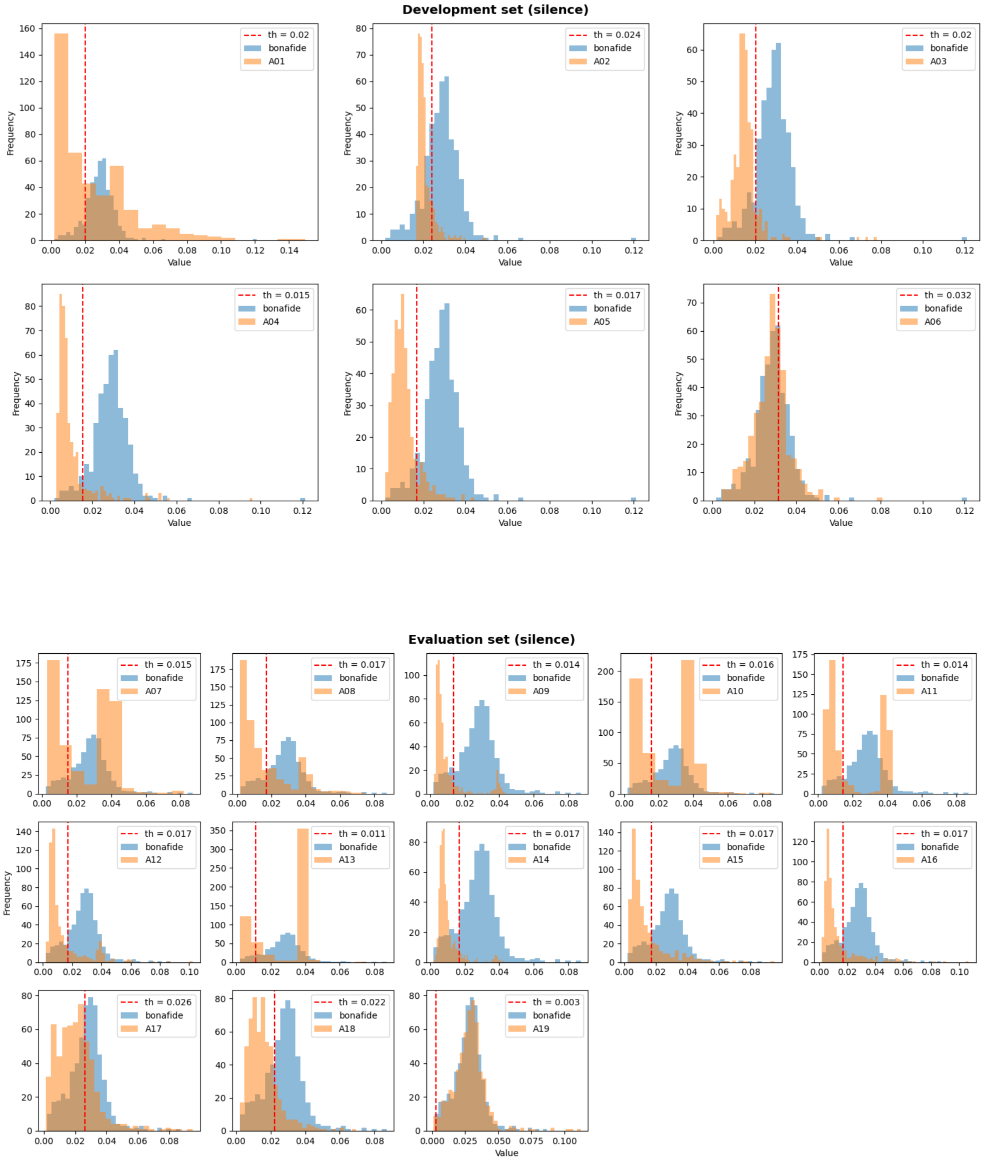
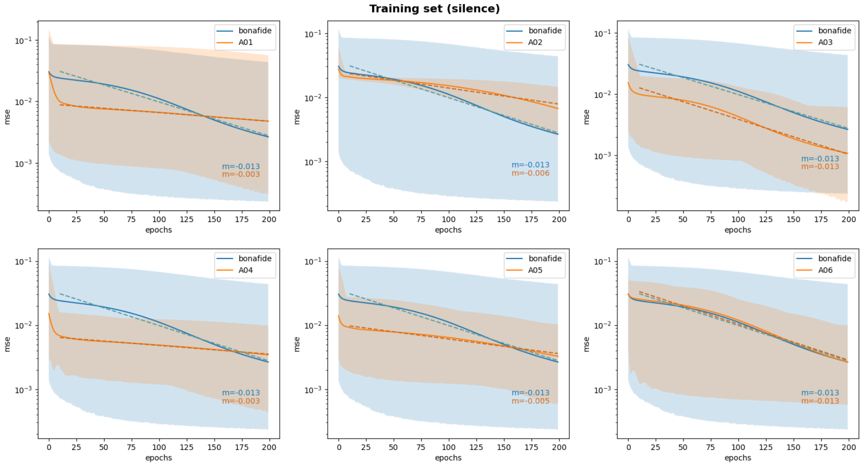
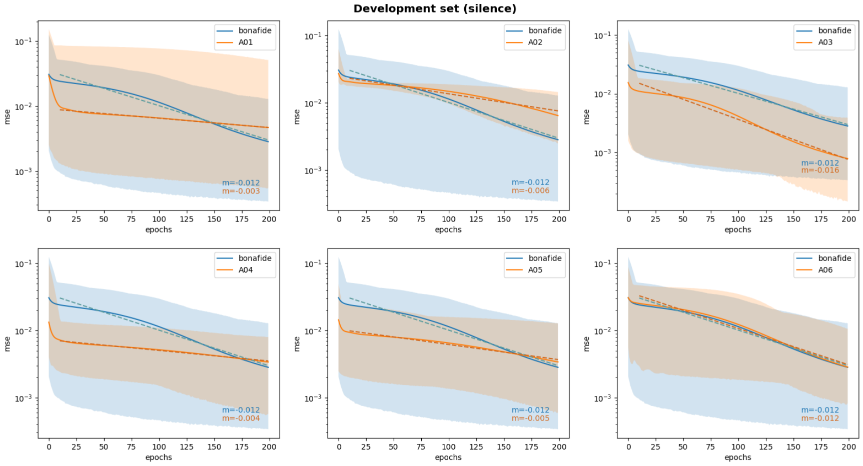


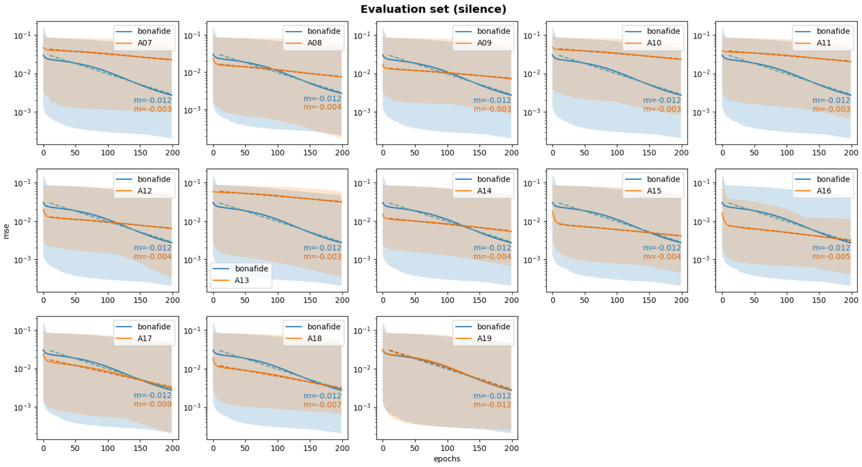
Figure A.4: Δ_{MSE} values vs frequency of occurrence for bonafide (*blue*) and spoof (*orange*) samples for both development and evaluation datasets. The discriminative threshold is highlighted (*dashed red line*).



Linear fitting - training set

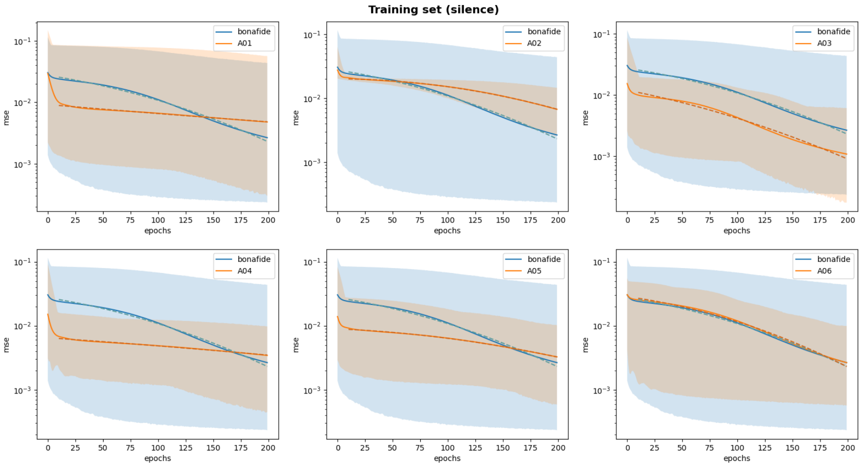


Linear fitting - development set

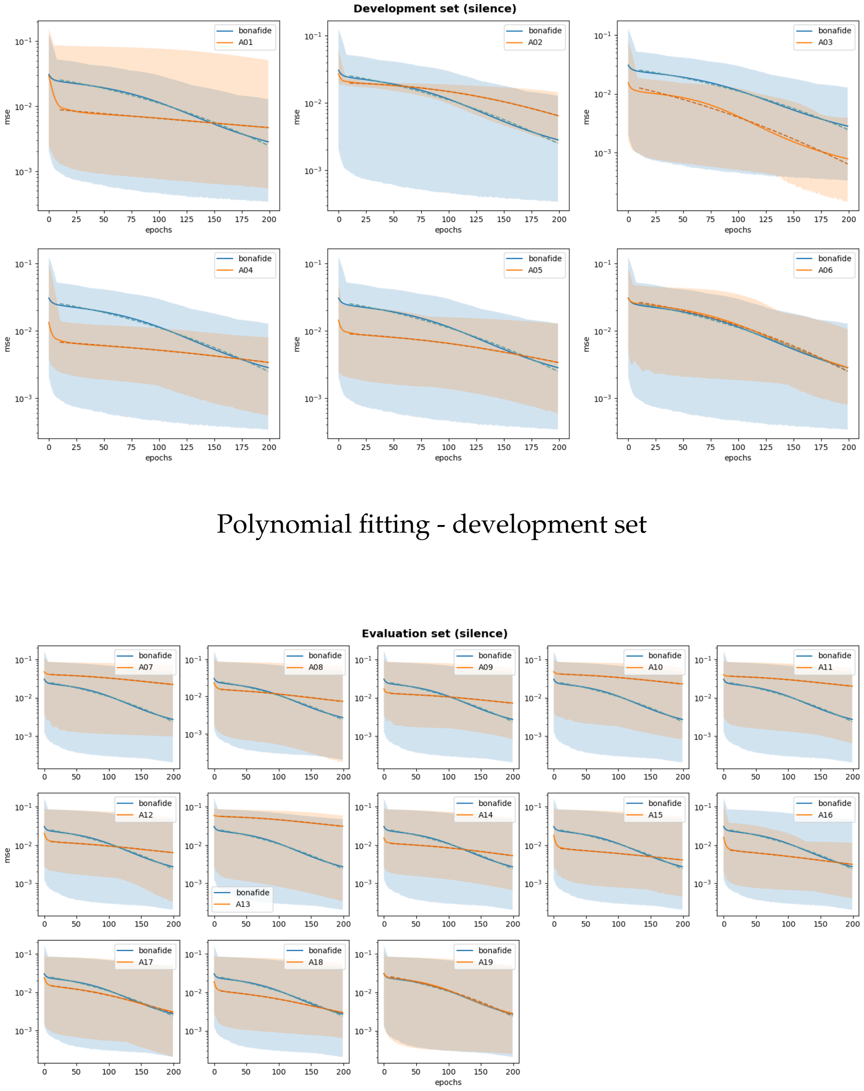


Linear fitting - evaluation set

Figure A.5: Linear fitting on MSE curves.



Polynomial fitting - training set



Polynomial fitting - development set

Polynomial fitting - evaluation set

Figure A.6: Polynomial fitting on MSE curves.

Acknowledgements

Vorrei ringraziare tutte le persone che sono state al mio fianco durante questo percorso, senza di voi non sarei mai arrivata fino a qui.

Ringrazio innanzitutto il Prof. Milani ed i dottorandi Daniele ed Elena del laboratorio LTTM, per avermi accompagnata con gentilezza e disponibilità in questo percorso di tesi stimolante ed istruttivo.

Il grazie più sentito va alla mia Famiglia, è merito vostro e di tutti i sacrifici che avete fatto se sono arrivata fino a qui. In particolare, a mamma Maria Grazia per avermi accompagnata durante tutta la mia carriera scolastica e per non avermi mai fatta sentire sola durante i lunghi pomeriggi di studio. A papà Vincenzo, per essere stato sempre disponibile ad aiutarmi e per essere sempre riuscito a risollevarmi il morale nei momenti di difficoltà. A mia sorella Francesca, per essere sempre stata un modello di rifermento e fonte di ispirazione nello studio e nella vita, crescere ed imparare insieme a te è stato fondamentale per arrivare fino a qui.

Un sentito grazie va anche a tutti i miei parenti, nonni, zii e cugini. In particolare a mia cugina Noemi, con cui ho da sempre un legame speciale. Sei sempre riuscita a capirmi, supportarmi e a darmi buoni consigli.

Ci tengo poi a ringraziare le mie amiche di una vita Alesia, Beatrice e Martina che sono al mio fianco praticamente da sempre. Senza di voi tutto il mio percorso di studi, dalle scuole elementari al liceo, non sarebbe stato lo stesso. Se oggi sono arrivata fin qui è anche grazie a voi e al vostro esserci sempre state.

Ringrazio poi la mia compagnia di amici Aeiouipsilon, per tutti i bei momenti che abbiamo vissuto insieme e per quelli che verranno.

Infine, ringrazio la me stessa del passato. Per aver avuto il coraggio di intraprendere questo percorso impegnativo, che inizialmente tanto la spaventava. Per non aver mai mollato nonostante le numerose difficoltà incontrate. Per avermi condotto fino a qui. E questo è soltanto l'inizio.

Bibliography

[1] Y. Agiomyrgiannakis. Vocaine the vocoder and applications in speech synthesis. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4230–4234. IEEE, 2015.

[2] Y. Agiomyrgiannakis. Vocaine the vocoder and applications in speech synthesis. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[3] E. A. AlBadawy, S. Lyu, and H. Farid. Detecting ai-synthesized speech using bispectral analysis. In *CVPR workshops*, pages 104–109, 2019.

[4] J. Allen, S. Hunnicutt, R. Carlson, and B. Granström. Mitalk-79: The 1979 mit text-to-speech system. *Journal of the Acoustical Society of America*, 65, 1979.

[5] Tiziano Bianchi, Alessia De Rosa, Marco Fontani, Giovanni Rocciolo, and Alessandro Piva. Detection and classification of double compressed mp3 audio tracks. In *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security*, page 159–164, 2013.

[6] A. Black and Nick N. Campbell. Optimising selection of units from speech databases for concatenative synthesis. 1, 07 1996.

[7] Nicolo Bonettini, Paolo Bestagini, Simone Milani, and Stefano Tubaro. On the use of benford's law to detect gan-generated images. In *2020 25th international conference on pattern recognition (ICPR)*, pages 5495–5502. IEEE, 2021.

[8] 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*, 2021(1):1–14, 2021.

[9] Clara Borrelli, Paolo Bestagini, Fabio Antonacci, Augusto Sarti, and Stefano Tubaro. Synthetic speech detection through short-term and long-term prediction traces. *EURASIP Journal on Information Security*, 2021(1):1–14, 2021.

[10] Davide Capoferri, Clara Borrelli, Paolo Bestagini, Fabio Antonacci, Augusto Sarti, and Stefano Tubaro. Speech audio splicing detection and localization exploiting reverberation cues. In *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2020.

[11] C. H. Coker. A model of articulatory dynamics and control. *Proceedings of the IEEE*, 64:452–460, 1976.

[12] G. Colombetti. From affect programs to dynamical discrete emotions. *Philosophical Psychology - PHILOS PSYCHOL*, 22:407–425, 08 2009.

[13] Emanuele Conti, Davide Salvi, Clara Borrelli, Brian Hosler, Paolo Bestagini, Fabio Antonacci, Augusto Sarti, Matthew C Stamm, and Stefano Tubaro. Deepfake speech detection through emotion recognition: A semantic approach. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8962–8966. IEEE, 2022.

[14] L. Cuccovillo, S. Mann, M. Tagliasacchi, and P. Aichroth. Audio tampering detection via microphone classification. pages 177–182, 09 2013.

[15] R. K. Das, J. Yang, and H. Li. Long range acoustic features for spoofed speech detection. In *Proc. Interspeech 2019*, pages 1058–1062, 2019.

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

[17] N. Diakopoulos and D. Johnson. Anticipating and addressing the ethical implications of deepfakes in the context of elections. volume 23, pages 2072–2098, 2020.

[18] A. Diekmann. Not the first digit! using benford's law to detect fraudulent scientific data. *Journal of Applied Statistics*, 34:321–329, 2007.

[19] H. Dudley and T. H. Tarnóczy. The speaking machine of wolfgang von kempelen. *Journal of the Acoustical Society of America*, 22:151–166, 1949.

[20] Euronews. Media forensics and deepfakes: an overview. 2019. `https://www.euronews.com/2019/10/09/french-charity-publishes-deepfake-of-trump-saying-aids-is-over`.

[21] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing (TASLP)*, 1984.

[22] D. W. Griffin and J. S. Lim. Signal estimation from modified short-time fourier transform. In *ICASSP*, 1983.

[23] Andrea Hauser. Deepfake - an introduction. *scip Labs*, October 2018.

[24] C. Hsu, H. Hwang, Y. Wu, Y. Tsao, and H. Wang. Voice conversion from non-parallel corpora using variational auto-encoder. *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2016.

[25] A.J. Hunt and A.W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 373–376, 1996.

[26] K. Jeong and Y. Shin. Time-series anomaly detection with implicit neural representation. 01 2022.

[27] YY. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, z. Chen, P. Nguyen, R. Pan, I. Lopez Moreno, and Y. Wu. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[28] The Wallstreet Journal. Fraudsters used ai to mimic ceo's voice in unusual cybercrime case. 2019. https://www.wsj.com/articles/fraudsters-use-ai-tomimic-ceos-voice-in-unusual-cybercrimecase.

[29] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu. Efficient neural audio synthesis. *International Conference on Machine Learning (ICML)*, 2018.

[30] Madhu R. Kamble, Hardik B. Sailor, Hemant A. Patil, , and Haizhou Li. Advances in anti-spoofing: from the perspective of asvspoof challenges. *APSIPA Transactions on Signal and Information Processing*, 9, 2020.

[31] H. Kawahara. Straight, exploitation of the other aspect of vocoder: Perceptually isomorphic decomposition of speech sounds. *Acoustical science and technology*, 27(6):349–353, 2006.

[32] H. Kawahara, I. Masuda-Katsuse, and A. De Cheveigne. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds. *Speech Communication*, 1999.

[33] T. Kinnunen, J. Lorenzo-Trueba, J. Yamagishi, T. Toda, D. Saito, F. Villavicencio, and Z. Ling. A spoofing benchmark for the 2018 voice conversion challenge: leveraging from spoofing countermeasures for speech artifact assessment. *The Speaker and Language Recognition Workshop*, 2018.

[34] K. Kobayashi, T. Toda, and S. Nakamura. Intra-gender statistical singing voice conversion with direct waveform modification using log-spectral differential. *Speech Communication*, 2018.

[35] G. Lavrentyeva, S. Novoselov, T. Andzhukaev, M. Volkova, A. Gorlanov, and A. Kozlov. Stc antispoofing systems for the asvspoof2019 challenge. pages 1033–1037, 09 2019.

[36] X. Li, N. Li, C. Weng, X. Liu, D. Su, D. Yu, and H. Meng. Replay and synthetic speech detection with res2net architecture. 10 2020.

[37] Alessandro Lieto, Daniele Moro, Francesco Devoti, Claudia Parera, Vincenzo Lipari, Paolo Bestagini, and Stefano Tubaro. " hello? who am i talking to?" a shallow cnn approach for human vs. bot speech classification. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2577–2581. IEEE, 2019.

[38] Y. Lipman. Phase transitions, distance functions, and implicit neural representations. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6702–6712. PMLR, 18–24 Jul 2021.

[39] M. T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[40] S. Lyu. Deepfake detection: Current challenges and next steps. In *2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–6, 2020.

[41] T. Masuko, K. Tokuda, , T. Kobayashi, and S. Imai. Speech synthesis using hmms with dynamic features. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 389–392, 1996.

[42] D. Matrouf, J. Bonastre, and C. Fredouille. Effect of speech transformation on impostor acceptance. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2006.

[43] D. M.Ballesteros, Y. Rodriguez-Ortega, D. Renza, and G. Arceb. Deep4snet: deep learning for fake speech classification. *Expert Systems with Applications*, 184:115465, 2021.

[44] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space, 2018.

[45] Simone Milani, Pier Francesco Piazza, Paolo Bestagini, and Stefano Tubaro. Audio tampering detection using multimodal features. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4563–4567, 2014.

[46] B. Mildenhall, P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[47] M. Morise, F. Yokomori, and K. Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE TRANSACTIONS on Information and Systems*, 99(7):1877–1884, 2016.

[48] M. Morise, F. Yokomori, and K. Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, 2016.

[49] E. Moulines and F. Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication*, 9(5):453–467, 1990.

[50] Nicolas Müller, Franziska Dieckmann, Pavel Czempin, Roman Canals, Jennifer Williams, and Konstantin Böttinger. Speech is silver, silence is golden: What do asvspoof-trained models really learn? In *2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge*, 06 2021.

[51] M. Niemeyer, L. M. Mescheder, M. Oechsle, and A. Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5378–5388, 2019.

[52] S. P. Panda and A. K. Nayak. A waveform concatenation technique for text-to-speech synthesis. *International Journal of Speech Technology*, 20(4):959–976, 2017.

[53] J. J. Park, P. Florence, J. Straub, R. Newcomb, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[54] Cecilia Pasquini, Giulia Boato, and Fernando Perez-Gonzalez. Multiple jpeg compression detection by means of benford-fourier coefficients. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 113–118, 2014.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research (JMLR)*, 2011.

[56] F. Perez-Gonzalez, C. Abdallah T. T. Quach, G. L. Heileman, and S. J. Miller. Application of benford's law to images. *Benford's Law: Theory and Applications*, 2015.

[57] Fernando Perez-Gonzalez, Greg L. Heileman, and Chaouki T. Abdallah. Benford's lawin image processing. In *2007 IEEE International Conference on Image Processing*, volume 1, pages I – 405–I – 408, 2007.

[58] Tomas Pevny and Jessica Fridrich. Detection of double-compression in jpeg images for applications in steganography. *IEEE Transactions on Information Forensics and Security*, 3(2):247–258, 2008.

[59] W. Ping, K. Peng, A. Gibiansky, S. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller. Deep voice 3: 2000-speaker neural text-to-speech. *CoRR*, abs/1710.07654, 2017.

[60] Y. Qian, N. Chen, and K. Yu. Deep features for automatic spoofing detection. *Speech Commun.*, 85(C):43–52, dec 2016.

[61] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. 2018.

[62] R. A. Raimi. The first digit problem. *The American Mathematical Monthly*, 83:521–538, 1976.

[63] M. Ravanelli and Y. Bengio. Speaker recognition from raw waveform with sincnet. pages 1021–1028, 12 2018.

[64] Y. Rodriguez-Ortega, D.M. Ballesteros, and D. Renza. A machine learning model to detect fake voice. In *Applied Informatics*, pages 3–13, 2020.

[65] Md Sahidullah, Tomi Kinnunen, and Cemal Hanilçi. A comparison of features for synthetic speech detection. 2015.

[66] M. Schröder, M. Charfuelan, S. Pammi, and I. Steiner. Open source voice creation toolkit for the mary tts platform. *Conference of the International Speech Communication Association (INTERSPEECH)*, 2011.

[67] N. Seonghyeon, M. A. Brubaker, and M. S. Brown. Neural image representations for multi-image fusion and layer separation, 2021.

[68] O. Sercan, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman, S. Sengupta, and M. Shoeybi. Deep voice: Real-time neural text-to-speech. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 195–204. PMLR, 06–11 Aug 2017.

[69] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, and R. Skerrv-Ryan. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[70] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. 2018.

[71] Arun Kumar Singh and Priyanka Singh. Detection of ai-synthesized speech using cepstral & bispectral statistics. In *2021 IEEE 4th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 412–417, 2021.

[72] B. Sisman, J. Yamagishi, S. King, and H. Li. An overview of voice conversion and its challenges: From statistical modeling to deep learning. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29:132–157, jan 2021.

[73] V. Sitzmann and C. M. Jiang. Awesome implicit neural representations. 2021.

[74] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020.

[75] Y. Stylianou, O. Cappe, and E. Moulines. Continuous probabilistic transform for voice conversion. *IEEE Transactions on Speech and Audio Processing*, 6(2):131–142, 1998.

[76] I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[77] H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and A. Larcher. End-to-end anti-spoofing with rawnet2. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373. IEEE, 2021.

[78] Tanaka, H. Kameoka, T. Kaneko, and N. Hojo. Wavecyclegan2: Time-domain neural post-filter for speech waveform generation.

[79] B. Thormundsson. Potential ai-enabled cyberattacks on companies worldwide 2021. *Statista - The Statistics Portal*, 2022.

[80] C. Tianxiang, K. Avrosh, N. Parav, S. Ganesh, and K. Elie. Generalization of Audio Deepfake Detection. In *Proc. The Speaker and Language Recognition Workshop (Odyssey 2020)*, pages 132–137, 2020.

[81] The New York Times. Pennsylvania woman accused of using deepfake technology to harass cheerleaders. 2021. https://www.nytimes.com/2021/03/14/us/raffaela-spone-victory-vipers-deepfake.html.

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

[83] M. Todisco, X. Wang, 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. *In Conference of the International Speech Communication Association (INTERSPEECH)*, 2019.

[84] K. H. Todter. Benford's law as an indicator of fraud in economics. *German Economic Review*, 10:339–351, 2009.

[85] X. Tong, L. Wang, X. Pan, and J. Wang. An overview of deepfake: The sword of damocles in ai. In *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pages 265–273, 2020.

[86] J. M. Valin and J. Skoglund. Lpcnet: Improving neural speech synthesis through linear prediction. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5891–5895, 2019.

[87] A. van den Oord, S. Dieleman, K. Simonyan H. Zen, O. Vinyals, N. Kalchbrenner A. Graves, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *ISCA Workshop on Speech Synthesis Workshop*, page 125.

[88] L. Verdoliva. Media forensics and deepfakes: an overview. *CoRR*, 2020.

[89] W. Wang, S. Xu, and B. Xu. First step towards end-to-end parametric tts synthesis: Generating spectral parameters with neural attention. In *INTERSPEECH*, 2016.

[90] X. Wang, S. Takaki, and J. Yamagishi. Neural source-filter-based waveform model for statistical parametric speech synthesis. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[91] X. Wang, J. Yamagishi, M. Todisco, A. Nautsch H. Delgado, N. Evans, M. Sahidullah, V. Vestman, T. Kinnunen, K. A. Lee, L. Juvela, P. Alku, Y.H. Peng, H. T. Hwang, Y. Tsao, H. M. Wang, S. L. Maguer, M. Becker, F. Henderson, R. Clark, Y. Zhang, Q. Wang, Y. Jia, K. Onuma, K. Mushika, T. Kaneda, Y. Jiang, L.J. Liu, Y.C. Wu, W. C. Huang, T. Toda, K. Tanaka, H. Kameoka, I. Steiner, D. Matrouf, J.F. Bonastre, A. Govender, S. Ronanki, J.X. Zhang, and Z. H. Ling. Asvspoof 2019: A large-scale public database of synthesized, converted and replayed speech. *Computer Speech Language*, 2020.

[92] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous. Tacotron: A fully end-to-end text-to-speech synthesis model. *CoRR*, abs/1703.10135, 2017.

[93] Wikipedia. Speech synthesis — wikipedia, the free encyclopedia. 2021. http://en.wikipedia.org/w/index.php?title=Speech%20synthesis&oldid=1020857981.

[94] Z. Wu, O. Watts, and S. King. Merlin. An open source neural network speech synthesis system. *Speech Synthesis Workshop (SSW)*, 2016.

[95] X. Xiao, X. Tian, S. Du, H. Xu, C. E. Siong, and H. Li. Spoofing speech detection using high dimensional magnitude and phase features: the ntu approach for asvspoof 2015 challenge. In *INTERSPEECH*, 2015.

[96] L. Yariv Y., Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2492–2502. Curran Associates, Inc., 2020.

[97] J. Yamagishi, C. Veaux, and K. MacDonald et al. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92). 2019.

[98] J. Yamagishi, X. Wang, M. Todisco, M. Sahidullah, J. Patino, A. Nautsch, X. Liu, K. A. Lee, T. Kinnunen, N. Evans, and H. Delgado. Asvspoof 2021: accelerating progress in spoofed and deepfake speech detection. In ISCA, editor, *ASVspoof 2021, Automatic Speaker Verification<br /> Spoofing And Countermeasures Challenge, 16 September 2021*, 2021.

[99] J. Yang, R. K. Das, and H. Li. Significance of subband features for synthetic speech detection. page 2160–2170, 2020.

[100] H. Zen, Y. Agiomyrgiannakis, N. Egberts, F. Henderson, and P. Szczepaniak. Fast, compact, and high quality lstm-rnn based statistical parametric speech synthesizers for mobile devices. *In Conference of the International Speech Communication Association (INTERSPEECH)*, 2016.

[101] H. Zen and H. Sak. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4470–4474, 2015.

[102] H. Zen, A. Senior, and M. Schuster. Statistical parametric speech synthesis using deep neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 7962–7966, 2013.

[103] Chunlei Zhang, Chengzhu Yu, and John HL Hansen. An investigation of deep-learning frameworks for speaker verification antispoofing. *IEEE Journal of Selected Topics in Signal Processing*, 11(4):684–694, 2017.

[104] J. Zhu, T. Park, P. Isola, and A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. pages 2242–2251, 10 2017.