POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

DeepMetric: Enhancing Synthetic Speech Detection through Support Tracks Generation

Tesi di Laurea Magistrale in
Music and Acoustic Engineering

Alessandro Orsatti, 10680665

Abstract: In modern days, new technological advances in speech generation have opened up the possibilities of creating synthetic speech with low effort and using consumer-level devices. This phenomenon has opened the doors to new scenarios and exciting possibilities, such as the development of new vocal assistants, devices for people with pathological conditions, didactics, etc. However, when these technologies are used by malicious users, they can lead to illegal consequences and dangerous scenarios for the population. For instance, it is possible to clone one person's voice using just a few seconds of reference audio and make them say arbitrary utterances. This can lead to credibility issues, privacy concerns, threatening audio, identity theft, and other issues. To address the malicious use of synthetic speech and counterfeit audio data, the forensic community guides the research toward new methods for spoofing and detecting those newly generated fakes. In this work, we focus on the problem of synthetic speech detection, which consists of taking an audio signal as input and determining whether this is authentic or not. Modern detection architectures present encouraging results, but are far from being perfect due to the fast progress in deepfake generation techniques. In this thesis, we propose a pipeline to improve the detection performance of a synthetic speech detector by making use of support tracks. Given a speech audio signal, we generate a set of support tracks that contain the same textual and voice content as the track under analysis and use them to improve the detection accuracy of the system. The method is based on the computation of the distances between the reference and these support tracks, leading to more stable and reliable predictions. The support tracks are implemented using multiple existing Text-to-Speech and Voice Conversion methods. Results provided by the complete pipeline proposed in this work show an exciting new possibility for improving deepfake detection through a modular method, unbounded from specific models.

Advisor:
Prof. Paolo Bestagini

Co-advisors:
Davide Salvi
Daniele Ugo Leonzio

Academic year:
2022-2023

Key-words: Multimedia Forensics, Synthetic Speech, Deepfake, Voice Conversion, Text-to-Speech

1. Introduction

Nowadays the widespread of Artificial Intelligence (AI)-powered tools has led to an immense new world of expressive possibilities, not only for big tech societies but also for common citizens, due to their relative accessibility. Thanks to these new technologies it is now possible to generate synthetic media with very low effort. Starting from a textual input, we can mimic one person's voice or generate a piece of digital content that is incredibly realistic and sometimes difficult to differentiate from authentic media. These new tools, besides having a positive effect over multiple aspects such as dubbing, augmented reality, medical purposes, etc., can also have a harmful impact on public information [33] and privacy [23]. As a matter of fact, when such technologies are used in a malicious way, they can lead to threats and dangers. This is the case of deepfakes: synthetic content generated through Deep Learning (DL) techniques that appropriate an existing person's physical features (e.g., voice, physical aspect) and provide unseen or unspoken generated content. The problems that society faces caused by the phenomenon of fake news are now boosted by deepfakes.The appropriation and creation of fake content based on a true person's characteristics not only exploits the public credibility of the victim, to convey, often, false information but also exposes the person to the consequences of an action or statement attributed to him/her. This is clearly dangerous not only for the common people but, in a major way, for politicians [4], institutions, and more in general public figures [50]. The accessibility to deepfake generation led to a worrying increase in problems related to fake news and their exploitation, exceedingly aggravated by the new fast-feed format from which people decide to consume information nowadays.

All the challenges described above are faced by the MultiMedia Forensic (MMF) community, which guides the research towards detecting deepfakes for public and private security purposes. Despite the results and methods utilized for deepfake spoofing have led to a more comfortable situation concerning the specific field, several difficulties, such as predictability of the detection on unseen data, are still problems for the community. This is the inspiring factor behind the decision and development of this thesis's main topic: Deepfake Audio Detection [60]. Deepfake Audio Detection establishes its roots in signal processing techniques aimed at highlighting the artifacts left by synthetic speech generators, such as bispectral [1] or low-to-mid level feature [43] analysis, and Machine Learning (ML) classification methods. The base of these methods relies on techniques such as Logistic Regression [42], Gaussian Mixture Models (GMM) and Support Vector Machines (SVM) [7], and Quadratic Support Vector Machine (Q-SVM) [48]. However, the latest synthetic speech generators, able to perform more and more convincing voices, outperform older detectors based on features [6], all that favors the implementation of newly thought methods based on (DL) [2]. Modern detectors based on neural networks vary from simple architectures based only on Convolutional Neural Network (CNN)s to more complex architectures which can take as input images, such as a 2D spectrogram (ResNet) [3], or raw waveform (RawNet) [52]. The last cited, RawNet, lends itself well to a fast application by being optimized and skipping the pre-processing phase for the feature extraction. Although present and functioning, those architectures are far from being perfect, given the constant introduction of new synthetic speech generators. The detectors, indeed, tend to work well in recognizing a deepfake when they see data generated with its specific generative architecture during training struggle to detect deepfakes generated with an unseen architecture during the training phase. New synthetic speech generators become available at a fast pace and each one of them produces different artifacts in the final audio result. The requirements of consistency and robustness towards unseen synthetic speech models force us to experiment with different solutions to account for the unpredictability of the cases. The approach presented in this paper aims to be reliable and modular in incorporating information about new models in the detection phase, without the need to retrain the detector with new data. To address this problem, we propose in this thesis a system named DeepMetric. DeepMetric aims to make a detector reliable and consistent by utilizing speech deepfakes synthesized with new generative models, among which several Text-to-Speech (TTS) and Voice Conversion (VC), never seen during training. The main idea is that, instead of considering the output of the detector, the distance in the latent space between the original track and the generated support tracks is used as a discriminant to make detection and improve performance. Distance between feature vectors in the latent space is highly descriptive for finding similarities between the inputs of the model. By comparing the embedding derived from the original input track with those of the deepfakes, we can discern whether the input has been artificially generated. All the deepfake copies contain the same textual component

of the original track and the embeddings used to compute the distance are extracted for each track from the penultimate layer of the neural network and represent high-level, non-descriptive features of its input. Through this method we not only achieve a result in contrasting unseen architectures by directly comparing them to the original audio in the latent space, but we also take into account a modular setup by creating a flexible system that has the possibility to incorporate more architectures that have yet to come. We underline that the proposed method is applied only in the test phase, therefore makes it unbound to a specific detection or generative architecture and capable of performing with any DL-based detector and any TTS or VC model.
The proposed system is composed of different phases, the first one is the generation of the deepfakes from the input audio. Text is extracted from an original signal through a procedure called Automatic Speaker Recognition (ASR). Then the text is fed to different Text-to-Speech (TTS) models that generate an audio file with the same textual component as the original. These deepfakes pass through a process called (VC) which takes as input an audio file and a reference in order to give to the former the same timbre of the latter. Again, different VC engines are used. The second phase is the embedding extraction. A neural network-based detector is utilized to extract latent vectors from each track. The final phase is the metrics comparison. Different metric functions have been selected so as to have different interpretations of the reciprocal distance of the vectors in the latent space. Distance metrics are computed for each original-deepfake combination and the results averaged and thresholded to obtain the final score.
This thesis is organized as follows. Section 2 provides all the different ideas behind the main technologies considered in this work. Section 3, formalizes the problem we address in this thesis and outlines the complete system we propose to solve it. Section 4 presents the whole setup in which we conducted the experiments, besides introducing the datasets and the metrics selected for this work. Section 5 presents the results we achieved in all the considered experiments. Finally, Section 6, concludes the work and provides some ideas about possible improvements and applications for future developments.

2. State-of-the-Art

First, an introduction to the main techniques for generating audio deepfakes with the help of neural architectures. These are composed of multiple phases, which are: extracting a textual cell from an audio (ASR), generating the audio from text (TTS), and converting one person's voice into another (VC). All of this, is followed by a brief chapter dedicated to the detection of fakes: the main concepts and ideas are introduced in a restricted form in order to give the reader all the necessary tools to understand everything presented on the following pages.

2.1. Synthetic Speech Generation

This paragraph focuses on the three main methods utilized in order to generate and manipulate audio in the audio deepfake generation realm: ASR, TTS, and VC.

2.1.1 Automatic Speaker Recognition

ASR, as presented in [27], initially addressed the problem of transforming a speech into a textual, interpretable input for the machine we interact with in modern days.
Through various techniques, reported in the following lines, an ideal ASR should be able to "recognize" the spoken words and then subsequently use the recognized words as input to another machine so that some "actions" can be performed on it. Older techniques consisted of hidden Markov model (HMM) and n-gram models [16], but from the beginning of the 2000s Artificial Neural Network models started to appear. Nowadays, a type of Recurrent Neural Network (RNN) called long-short term memory (LSTM) is being used for speech recognition, in combination with different DL techniques. ASR models based on neural networks are incredibly dependent on training data, that is due to all the possible different aspects of human speech, for instance, accent, dialect, emotional content, pronunciation, etc. Fortunately, as for now, we have at our disposal numerous consistent datasets and pre-trained model weights over a large quantity of training data. ASR, in a synthetic generation

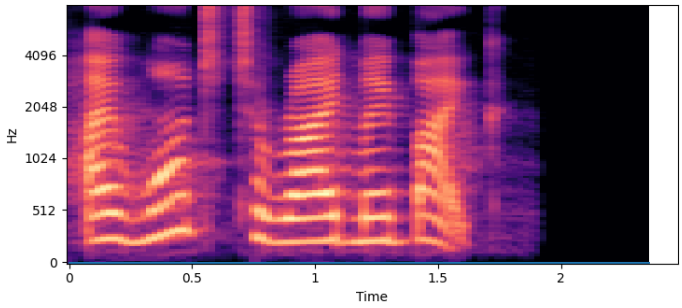
pipeline, takes the job of extracting the textual content of the original track to be fed to a Text-to-Speech model. The written text undergoes pre-processing to extract linguistic features essential for speech synthesis. This includes breaking down the text into linguistic units like phonemes (the smallest units of sound), extracting information about prosody (the rhythm, intonation, and stress of speech), and considering the linguistic context.

2.1.2 Text-to-Speech

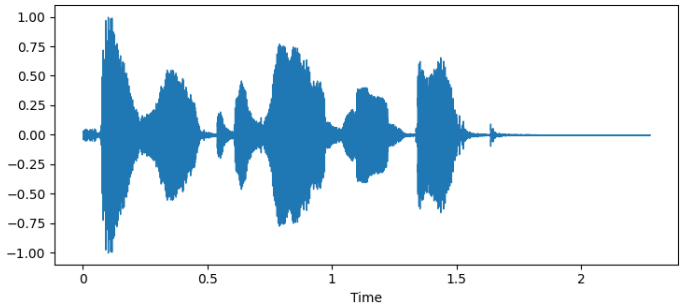
TTS transforms written text into audible speech. Utilizing neural networks, especially advanced DL models such as recurrent or transformer architectures, TTS systems generate human-like speech from textual inputs. This involves pre-processing the text, training neural networks on text-audio pairs, and synthesizing speech from new text inputs. TTS technology is applied in virtual assistants, accessibility solutions, and language learning platforms, facilitating seamless communication between users and computers through natural-sounding speech. It is undoubtedly a fundamental part of the modern interaction with digital systems needed by today's world society
The input text needs a pre-process in order to let the TTS model function properly. Text pre-processing may involve techniques such as tokenization (splitting the text into words or characters), part-of-speech tagging (identifying the grammatical category of each word), and syntactic parsing (analyzing the grammatical structure of sentences).
For the proper synthesis part, in older days we relied on Natural Language Processing (NLP) modules based on signal processing techniques [11], but now the almost whole entirety of systems present in online applications and standalone digital machines are based on DL.

A TTS DL model is trained on a dataset of text and corresponding speech samples. The goal of training is to teach the model to map textual inputs to their corresponding audio representations. Various architectures can be used for this task, including RNNs, CNNs, and transformer-based models. These models learn the complex patterns and relationships between text and speech through iterative optimization of their parameters. During training, the neural network adjusts its internal parameters (weights and biases) based on the difference between the predicted speech output and the actual speech samples in the training data using a loss function. The process is iterated several times to reach the desired accuracy. Once trained, the neural network can synthesize speech from new textual inputs. It processes the input text, generates a sequence of audio samples, and combines them to produce natural-sounding speech. Post-processing techniques, such as VC, may be applied to the synthesized speech to improve its quality and naturalness or to mimic the voice of a desired target reference. The structure of a TTS model can be summarized in two fundamental building blocks: Spectrogram Generator and Vocoder:

- **Spectrogram Generator:** The Spectrogram Generator is responsible for converting the input text into a mel-spectrogram representation. A spectrogram is a visual representation of the spectrum of frequencies of a signal with respect to time. In the case of TTS, the signal is the speech generated from the text. The textual input, as stated in the ASR paragraph, undergoes pre-processing in order to capture important linguistic features such as phonemes, prosody, and linguistic context. These features are then converted into a format suitable for the Spectrogram Generator. The generator typically utilizes neural network architectures, such as RNN or transformer-based models, to map the input text features to spectrogram representations. Encoder-decoder architectures are also used for this purpose, the encoder extracts high-level features from the input text and the decoder transforms them into a usable spectrogram. The output of the Spectrogram Generator is fed into the Vocoder for synthesis. In Figure 1a we can see a spectrogram representation, frequency on the y axis and time on the x axis, and the waveform (Figure 1b) from which it is derived.
- **Vocoder:** The Vocoder, short for voice encoder and decoder, is responsible for synthesizing speech from the spectrogram representation generated by the spectrogram generator. The Vocoder takes the spectrogram as input and reconstructs sample-by-sample the wave of the speech signal. This waveform represents the actual audio signal that can be played back as speech. Various Vocoder algorithms exist, ranging from traditional signal processing techniques



(a) Example of spectrogram derived from an audio waveform.



(b) Example of audio waveform.

to more sophisticated neural network-based models. One popular approach in modern TTS systems is to use neural network-based Vocoders, such as HiFiGan [22] or WaveGlow [36], which can generate high-quality and natural-sounding speech.

In summary, the Spectrogram Generator transforms textual input into a spectrogram representation, capturing the linguistic features of the speech signal, while the Vocoder synthesizes the speech waveform from the spectrogram, producing the final output of the system. Modern TTS systems utilize performative designs, such as Tacotron [56], that need a Vocoder or end-to-end models like fastspeech2 [40]. It's important to underline that, in the case of separate Spectrogram Generator and Vocoder, both need to be trained with the same data.

2.1.3 Voice Conversion

VC has applications in various domains, including dubbing and voice cloning, enabling the creation of customized speech content while maintaining naturalness and authenticity and, recently, utilized by the common people as an available resource for all kinds of purposes.
In deeper terms, VC alters aspects of the speech signal unique to each speaker, like spectral and prosodic elements, to change how the speaker's identity is perceived, while maintaining the linguistic content unchanged, which is independent of the speaker. The first VC models were based on spectrum mapping using parallel training data, where pairs of samples from both the source and target speaker with the same linguistic content were used during training. However, due to the difficulty of finding large-scale paired source and target speaker utterances in the real world, much more sophisticated models that do not impose this constraint have been developed. These are able to learn a mapping from source to target speaker without relying on parallel data and are therefore called non-parallel.
Figure 1 from [29] illustrates a typical VC system overview. During the training phase, the VC system is exposed to a collection of utterances recorded from both source and target speakers, known as training utterances. The speech waveform signal undergoes analysis and mapping feature computation steps, enabling the encoding of speech properties for modification. Alignment of speech segments from source and target speakers is conducted based on phonetic content similarity, facilitating the

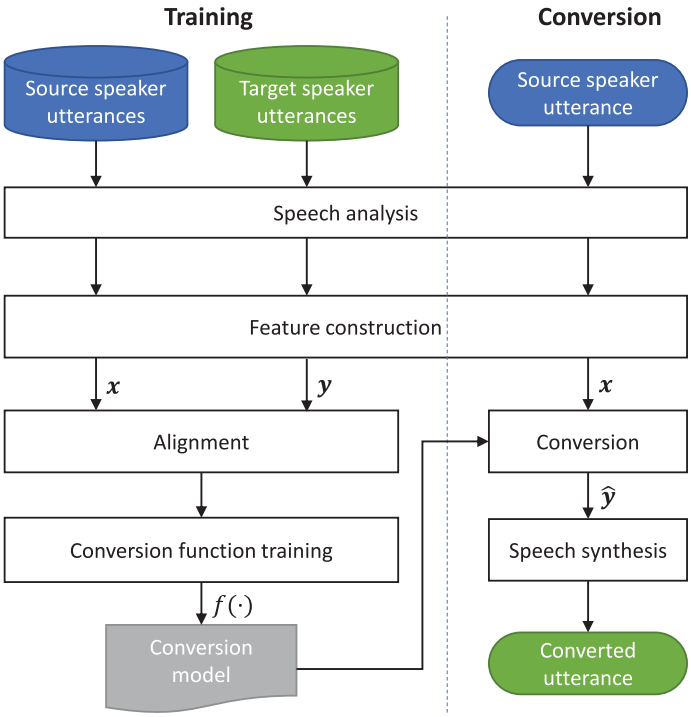


Figure 1: Structure of a basic Voice Conversion model.

association of corresponding segments in time. Subsequently, a mapping or conversion function is trained using these aligned mapping features. In the conversion phase, upon computing mapping features from a new source speaker utterance, these features undergo conversion using the trained conversion function. Speech features are derived from the converted features and utilized to synthesize the resulting waveform of the converted utterance.
Further advances led to new models that require no training starting to appear as usable to the public. These are called one-shot methods and implement an any-to-any mapping that doesn't require the input and the output speakers to be seen during training. A single audio for the desired voice to be cloned is needed, beyond the one from the original speaker to be converted. This allows any target voice to be modeled without needing a lot of data at hand and makes VC accessible to everyone, as there is no need to train the model on any identity.
A fine example is the one presented in [25] called FreeVC, which bases itself over the VICS [21] TTS end-to-end architecture, or YourTTS presented in [8].
FreeVC adopts the VITS framework for its exceptional reconstruction capabilities, yet it diverges by autonomously disentangling content information without relying on text annotations. The recent achievements in speech self-supervised learning have showcased its potential superiority over traditional acoustic features like mel-spectrograms in various downstream tasks, including speech recognition, speaker verification, and VC. To enable one-shot VC, a speaker encoder is integrated for extracting speaker information.

2.2. Synthetic Speech Detection

As technology advances, the capacity to distinguish between authentic and modified speech becomes increasingly consequential, affecting domains including cyber-security, privacy, establishment of trust in online interactions, etc. As a result, the MMF community has committed considerable resources to develop inventive detection techniques, tackling the challenge from multiple angles and investigating a variety of methodologies. The applications of audio deepfakes span a wide spectrum, this widespread use underscores the necessity for the development of new techniques to distinguish between authentic and manipulated audio signals. These techniques range from basic ML methods to

Layer	Input:**64000** samples	Output shape
Fixed Sinc filters	Conv(**129**,1,128) Maxpooling(3) BN & LeakyReLU	(**21290**,128)
Res block	{ BN & LeakyReLU Conv(3,1,128) BN & LeakyReLU Conv(3,1,128) Maxpooling(3) FMS } × 2	(**2365**,128)
Res block	{ BN & LeakyReLU Conv(3,1, **512**) BN & LeakyReLU Conv(3,1, **512**) Maxpooling(3) FMS } × 4	(**29**,512)
GRU	GRU(1024)	(1024)
FC	1024	(1024)
Output	**1024**	**2**

Figure 2: The RawNet2 architecture used for anti-spoofing. Modifications made to the original architecture are highlighted in boldface. BN refers to batch normalization.

advanced DL strategies, making notable contributions to the continuous quest for efficient detection of synthetic speech. Each method, whether utilizing conventional ML or harnessing DL, presents its own unique technical advantages in the effort to achieve reliable synthetic speech detection. Over the last four decades, there has been significant evolution in the models employed for speech and speaker-related spoofing applications. Previously, researchers frequently employed systems grounded in Discrete Vector Quantization (VQ) [49]. Subsequently, the cutting-edge transitioned towards solutions based on Gaussian Mixture Models (GMM) [41], and more recently, into factor analysis utilizing i-vector frameworks [28]. Currently, nearly every methodology founded on basic ML techniques has been supplanted by models based on DL. Synthetic Speech Detection, indeed, relies on specialized DL models designed to identify and detect manipulated or synthesized audio content, often referred to as audio deepfakes or synthetic speech. These detectors leverage advanced neural network architectures and signal processing techniques to analyze audio signals and distinguish between genuine and artificially generated speech. Neural detectors of audio deepfakes play a crucial role in maintaining trust and authenticity in audio communication channels by providing robust defenses against the proliferation of manipulated or synthesized audio content. Continued research and development in this field are essential to stay ahead of evolving audio deepfake techniques and ensure the reliability of audio-based information. Initially, regarding the domain of DL, a prevalent strategy involved employing advanced CNNs [26] to identify synthetic voice. While CNNs were initially developed for computer vision tasks, their adaptability and impressive capabilities have made them essential across various fields. Particularly in synthetic speech detection, architectures like ResNet (Residual Network) [3] have demonstrated remarkable effectiveness in processing audio signals and determining the authenticity of voice recordings. These sophisticated CNNs introduce a structural design incorporating residual connections or skip connections, which enhances their suitability for very deep network architectures. Essentially, ResNets represent a specialized form of CNNs that utilize residual connections to facilitate the training of deep neural networks. Besides ResNet, another commonly used Neural architectures for spoofing is RawNet [18]. The RawNet model was born as an enhancement of the CNN-LSTM model [19]. It is a neural network specifically crafted to directly process raw audio waveforms, enabling end-to-end speech analysis. This model has exhibited competitive performance in tasks related to synthetic speech detection and has been employed in various research endeavors to tackle the challenges posed by deepfake and synthetic speech technologies.
RawNet incorporates residual blocks [14], a Gated Recurrent Unit (GRU) layer [9], a fully-connected layer for speaker embedding extraction, and an output layer. In this architecture, input features undergo initial processing through the residual blocks to derive frame-level embeddings. These residual

blocks consist of convolutional layers with identity mapping, aiding in the training of deep architectures. Subsequently, a GRU is utilized to consolidate the frame-level features into a singular utterance-level embedding. This utterance-level embedding is then passed through a fully-connected layer, with the output serving as the speaker embedding connected to the output layer. The output layer's node count matches the number of speakers in the training set.
In the context of improving the state of the art and giving the possibility to small researchers to work on a consistent set of data, pivotal drivers such as the ASVspoof [58] (Automatic Speaker Verification and Spoofing Countermeasure) and ADD (Audio Deepfake Detection) Challenge have surfaced. These challenges serve as crucial platforms for researchers and practitioners to assess and introduce innovative detection methods and technologies, thereby advancing the state of the art.
A second version of the RawNet architecture originated from the ASVspoof2019 challenge [32].
Similar to its initial version, RawNet2 [52] functions directly on the raw speech waveform, significantly reducing reliance on manually engineered features. While its layer architecture undergoes slight modifications compared to the original RawNet, its fundamental features and operation remain consistent. In Figure 2 We can see displayed the complete structure of RawNet2. The initial adjustment pertains to the first layer of the architecture responsible for processing raw speech. In an effort to improve performance, developers opted not to employ layer normalization [21] for the input, as it led to subpar results. Additionally, the filter length, which determines the impulse response duration, has been optimized. This adjustment stems from the recognition that the cues utilized for spoofing detection may differ from those used in speaker recognition. While the original implementation utilized filter lengths of 251 samples, it was found that better results were achieved with a reduced length of only 129 samples.
Furthermore, while the configuration of the first residual block remains unaltered, a larger number of kernel filters (512) is employed in the second residual block. A GRU layer with 1024 hidden nodes is introduced to consolidate frame-level representations into a unified utterance-level representation. Unlike RawNet2, which generates embeddings, the output of the GRU layer is followed by an additional fully connected layer preceding the output layer. Subsequently, a softmax activation function is applied to facilitate two-class predictions: bonafide or spoof.

2.3. Latent Space and Embeddings

In the context of neural networks, particularly in the field of DL, the term "latent space" refers to a conceptual space where data is represented in a condensed or abstract form by the network's hidden layers. This term "latent" is used because these representations are not directly discernible in the input or output data; rather, they are deduced or acquired by the neural network through the training procedure. Every neural network layer is tasked with capturing input data in a hierarchical fashion. With each layer it traverses, the data undergoes a transformation into more abstract representations. This collection of abstract representations collectively defines what is known as the latent space. In many cases, the latent space serves as a compressed and distilled representation of the input data; it is often used, indeed, for tasks such as feature visualization and dimensionality reduction [34].
The vector representations of data in this space are called embeddings. In the contest of audio deepfakes detection, the embeddings extracted from audio recordings are compared to embeddings of authentic or known patterns by utilizing a metric function, their distance in the latent space gives useful information. Deviations or inconsistencies between these embeddings, and the lack of clustering if compared to embeddings derived from real audio, can indicate the presence of a deepfake. In Figure 3 there is represented a simple example of bi-dimensional latent space in which the embeddings are points in the space.

Embeddings play a critical role in audio deepfake detection by transforming raw audio signals into semantically meaningful representations, enabling DL models to identify anomalies or discrepancies indicative of manipulated or synthetic audio content.

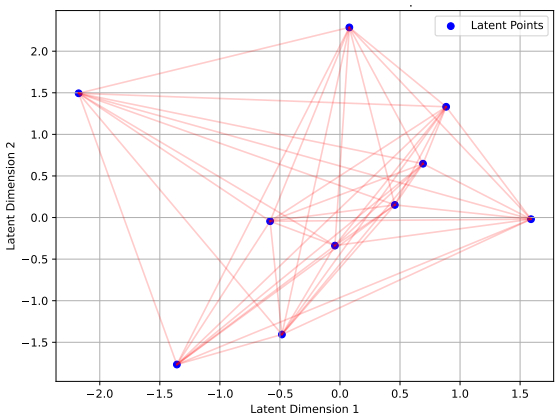


Figure 3: Latent space for bi-dimensional feature vectors, distances in the space are represented by the red lines.

3. Proposed method

This section formalizes the problem we address, namely speech deepfake detection. We present our proposed method defining the problem of speech deepfake detection, which is the main topic of the thesis, then providing details both about the system pipeline and its individual phases.

3.1. Problem formulation

The problem we address is formally defined as follows. Let us consider a discrete-time input speech signal $\mathbf{x}$ sampled with a sampling frequency f_s. The signal belongs to a class $y \in \{0, 1\}$, where 0 means that the signal is authentic while 1 indicates that has been generated with any speech synthesis technique.
Our goal is to develop a speech deepfake detector $\mathcal{D}$ trained to estimate the class of the signal $\mathbf{x}$ as $\hat{y} = \mathcal{D}(\mathbf{x})$, where $\hat{y} \in [0, 1]$ is a real number indicating the likelihood that the signal $\mathbf{x}$ is fake.
The performance of the detectors proposed in the state-of-the-art are excellent, especially when they are tested in controlled scenarios. However, when we consider more challenging conditions, they become less accurate and lead to less reliable predictions. This is a relevant aspect in the multimedia forensic field where we want our detectors to be able to generalize as much as possible over multiple domains.
In this thesis we propose a method to address this problem. Given a speech deepfake detector $\mathcal{D}$, during its inference phase, for each track $\mathbf{x}$ under analysis, we generate a set of *support tracks* $\mathbf{x}_i$ with $\mathbf{i} \in \{1, N\}$ that contain the same textual and speaker information of the original signal $\mathbf{x}$. This is achieved by considering a set of N synthetic speech generators that leverage TTS and VC techniques. The TTS component of the generator is used to synthesize a signal $\mathbf{x}_i$ with the same textual content as $\mathbf{x}$, while the TTS is used to modify the speaker of $\mathbf{x}_i$ and make it equal to that of $\mathbf{x}$. The N support tracks are used to make the detection process of $\mathcal{D}$ more robust.
To do so, we consider the detector $\mathcal{D}$ as embedding extractor and use it to compute a feature vector $\mathbf{e}$ for both the original track and the generated ones. Finally, we consider as detector score, no more the output $\hat{y}$ of the detector $\mathcal{D}$ but the thresholded mean of the distances $\mathbf{d}_i$ with $\mathbf{i} \in \{1, N\}$ between the embeddings $\mathbf{e}_i$ of the analyzed *support tracks* and $\mathbf{e}$ of the original signal.
The rationale behind this method is that the distance $\mathbf{d}_i$ in the latent space between the feature vectors of the original signal $\mathbf{x}$ and the *support tracks* $\mathbf{x}_i$ is more robust than the simple output $\hat{y}$ of the detector.
We highlight that the proposed system involves only the inference phase of the model and does not affect training. Furthermore, it is independent from both the synthetic speech generation and detection models utilized and the number N of *support tracks*. These aspects make the method applicable to any detector $\mathcal{D}$ and capable of being improved when new technologies appear on the market.

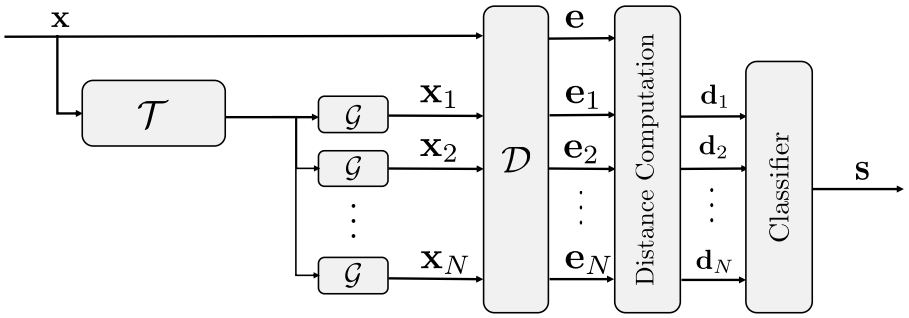


Figure 4: Pipeline of the proposed method.

In Figure 4 is presented the flow diagram of the system on which all the method is based. As stated in Figure 4, embedding extraction is performed once all N *support tracks* $\mathbf{x}_i$ are generated, soon followed by the metric computation based over them.
To summarize, here are the three main phases of the proposed methodology outlined in this thesis:

1. **Generation Phase:** This involves creating N deepfake copies (namely *support tracks* $\mathbf{x}_i$), with the same textual information, of the audio track $\mathbf{x}$ to be detected.
2. **Embeddings extraction Phase:** Embeddings $\mathbf{e}_i$ with $\mathbf{i} \in \{1, N\}$ for all *support tracks*, including $\mathbf{e}$ for the original $\mathbf{x}$, are extracted from the penultimate layer of a neural detector $\mathcal{D}$.
3. **Metric Comparison Phase:** Each embedding $\mathbf{e}_i$ from the *support tracks* is utilized to calculate its distance from the embedding $\mathbf{e}$ of the original track $\mathbf{x}$ using various metric functions. The resulting metric scores $\mathbf{d}_i$, corresponding in size to the number N of generated deepfakes, are then averaged and thresholded to generate the final detection score $\mathbf{s}$.

3.2. Considered system

3.2.1 Generation Phase

The goal of this procedure is to provide deepfakes copies as accurate as possible, with respect to the original track, to take advantage of the differences verifiable in the latent space and the previous knowledge of their falsity. The generation phase consists of the following steps:

- The original track $\mathbf{x}$ is taken as input from the ASR algorithm $\mathcal{T}$, which returns the textual component. The text undergoes pre-processing before the next step in order to eliminate special characters, given that the TTS models are not able to recognize them.
- The textual output from the ASR is passed as input to different models of TTS, each model is different and therefore provides different artifacts in the produced *support track* $\mathbf{x}_i$.
- The *support tracks* generated from the previous phase individually pass through different models of VC, which will give each a different output per track, to obtain a result which is similar to the original signal $\mathbf{x}$ in both semantic content and timbre. The original track $\mathbf{x}$ is passed as reference to the VC models.

The outgoing tracks coming from the TTS models, beyond passing through the VC phase, are passed directly as output to the system. The idea of retaining the *support tracks* that didn't pass through the VC phase is based over the fact that artifacts generated from the VC model could, in some ways,

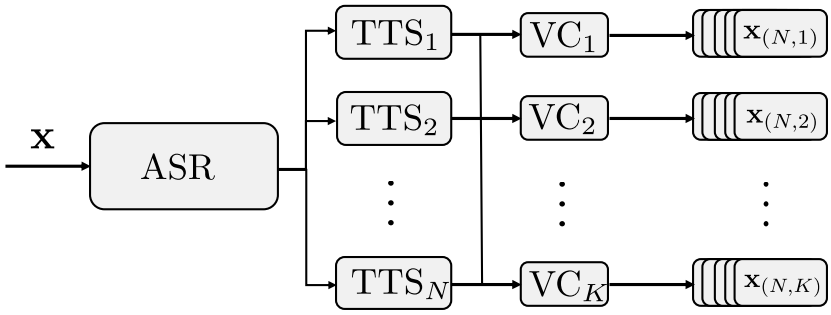


Figure 5: An ASR model extracts the text from the original audio input, the following TTS models receive the text and generate a track each, which is then fed to one or multiple VC models to obtain the *support tracks*.

mask the ones introduced by the TTS models alone. So, to obtain a better variety, all the tracks are utilized. This gives as a result several *support tracks* $\mathbf{x}_i$, some of which resemble the original signal $\mathbf{x}$ only for the textual component, while others are similar in both semantic content and timbre.
The process is, once again, summarized in Figure 5.

3.2.2 Embeddings extraction Phase

In our considered system, the detector $\mathcal{D}$ is used as a feature extractor. That is to say that we do not take in consideration the score provided by the last layer and only focus on saving the information retrieved by the network in its penultimate dense layer, storing them as embeddings $\mathbf{e}$ to be utilized in the following steps. Embeddings extracted from a neural network are highly informative in regard to high level features, and are commonly used in DL application. we hypothesize them to be functional in a synthetic speech detection task. By constructing a latent space with them, we are able to compare different inputs in a more abstract fashion to retrieve implicit information about the content and hidden similarities.

3.2.3 Metric Comparison Phase

Metric Computation, utilizes the embedding extracted in the previous phase from all the different *support tracks* $\mathbf{x}_i$ and the original signal $\mathbf{x}$ and computes the distance in the latent space between each of the original-deepfake couples. To ensure variety, different metrics were tested, among which:

- Cosine Distance
- Euclidean Distance
- Manhattan Distance

As stated before, all the metrics above are computed for each original-deepfake couple to obtain an input feature vector (with dimension N equal to the number of couples) to be averaged and thresholded to obtain the final score.
As an experiment we implemented a second step which utilizes a ML Classifier in order to make the most of all the features extracted from the distances between the support tracks and the original signal.
In ML, a classification problem involves categorizing input data into one of two or more classes or categories. Mathematically, let's denote the input features as $\mathbf{x}$ and the corresponding class labels as y. A classification algorithm learns a function f that maps input features $\mathbf{x}$ to class labels y.
In binary classification, there are only two possible classes: positive ($y = 1$) and negative ($y = 0$), which, in our case, corresponds respectively to fake and real. The function f predicts the probability that $\mathbf{x}$ belongs to the positive class.

Various methods have been explored to optimize at best the results. The ML Classifier utilized in our proposed method works with an input size N and gives as output a single value $\mathbf{s} \in [0, 1]$, representing the final score for the original signal $\mathbf{x}$.

4. Experimental setup

In this section, we outline the experimental setup, detailing the technical decisions that influenced the final outcomes we attained. Additionally, we present the different ASR, TTS and VC models used for developing the pipeline, along with the dataset utilized for conducting the experiments, the detector, the metrics employed for distance evaluation and the ML classifier utilized to extract the final score in the added experiment.

4.1. Synthetic speech generation pipeline

4.1.1 Automatic Speaker Recognition

For what concerns the ASR model, or rather, the model assigned to text extraction from the original track, we selected Whisper. Whisper [37] is a robust speech recognition model trained on an extensive collection of varied audio data and is also a multitasking model capable of executing tasks such as multilingual speech recognition, speech translation, and language identification. Several models for as many languages are available, more specifically for the English language four models were available. In Table 1 the complete collection with the relative speed and accuracy trade off:

	Parameters	**Relative Speed**
tiny	39M	16x
base	74M	8x
small	244M	3x
medium	769M	x1

Table 1: ASR models dictionary.

Finally, we chose the base model as it guaranteed the perfect balance between speed and accuracy.

4.1.2 Text-to-Speech

As stated in Section 2, a complete TTS models can be either composed of a combination of Spectrogram Generator and Vocoder trained on the same specific set of data, or an end-to-end model. All the models utilized in the generation pipeline were taken from the coqui.ai API and are all trained on a single language (in our case we took the ones trained on English data). In Table 2 the complete collection of model with the respective composition.

Model 1, 3, 4 and 5 were all trained on **ljspeech**, which is a public domain speech dataset consisting of 13,100 short audio clips of a single speaker reading passages from 7 non-fiction books. Model 2 is trained on the jenny dataset, a high-quality, varied 30hr voice dataset suitable for training a TTS model.

	Spectrogram Generator	Vocoder
TTS$_1$	Tacotron2 [12]	Univnet [17]
TTS$_2$	VITS [21]	-
TTS$_3$	Fast-Pitch [24]	HiFiGan [22]
TTS$_4$	Glow-TTS [20]	Multiband MelGan [59]
TTS$_5$	Tacotron2 [12]	Multiband MelGan [59]

Table 2: TTS models dictionary.

4.1.3 Voice Conversion

In this work we decided to opt for two different VC models. The first model **VC**$_1$ is provided, as for the TTS models, from the coqui.ai API in the form of a pre-trained model called FreeVC [25], based on the end-to-end framework of VITS [21]. FreeVC is a one-shot model which only needs one reference track in order to perform Voice Conversion. It is trained on the **VCTK** Corpus, which includes speech data uttered by 110 English speakers with various accents. Each speaker reads out about 400 sentences, that were selected from a newspaper, the rainbow passage and an elicitation paragraph used for the speech accent archive.
The second model **VC**$_2$ takes a different approach over VC and is called kNN-VC [5]. kNN-VC is an any-to-any VC model which simply utilizes a *k-Nearest Neighbors* regression. Any-to-any VC aims to transform source speech into a target voice with just a few examples of the target speaker as a reference. First, it uses a self-supervised speech representation model to extract feature sequences for both the source and reference utterances. Next, it converts to the target speaker by replacing each frame of the source representation with its nearest neighbor in the reference. The idea is that the matched target frames would have the same content as the source due to the fact that certain self-supervised representations capture phonetic similarity. Finally, the converted features pass through a neural Vocoder to obtain the converted speech.
By taking advantage of the composition of ASVspoof2019 presented in 7, when converting a track with kNN-VC from a certain speaker, all the utterances contained in the dataset from that same person were passed as a reference in order to optimize the process and obtain as a results tracks as similar as possible to the original.

	Name	Type
VC$_1$	FreeVC [25]	One-shot
VC$_2$	kNN-VC [5]	Any-to-any

Table 3: VC models dictionary.

4.1.4 Flow Diagram

As a summary, the whole generation pipeline utilized for the experiments in this thesis work is the one depicted in Figure 6. In the experiments carried out for this thesis work we implemented a pipeline composed of an ASR model extracts the text from the original audio, 5 TTS models that receive the text as an input and generate 5, and 2 VC models which each receive all TTS output tracks. The 5 tracks from the TTS output are also retained in order to have, as a result, 15 *support tracks*.

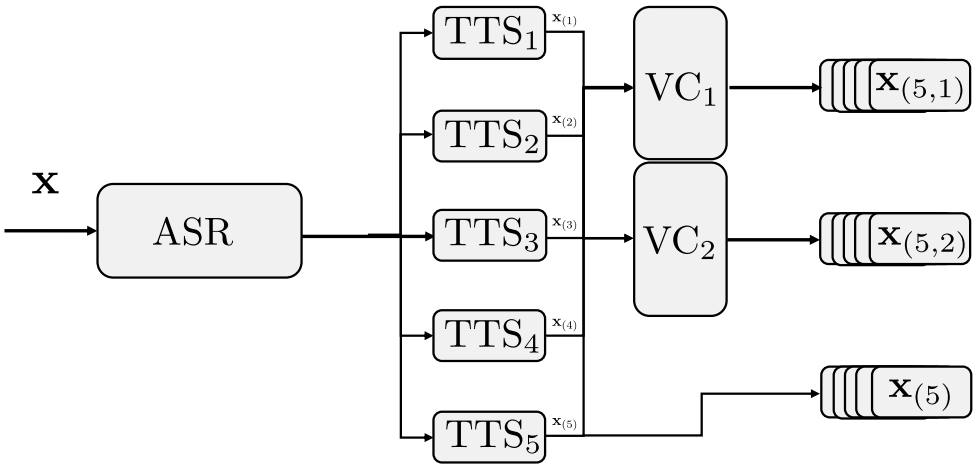


Figure 6: Generation pipeline utilized for the experiments conducted in this thesis work

4.2. Dataset

The dataset chosen to develop and test the idea presented in this thesis is ASVspoof2019 [55]. It bases itself as an access to the broad community to a complete dataset with the purpose of bettering the state of the art in the matter of Automatic Speaker Verification (Automatic Speaker Verification (ASV)). ASV stands out as one of the most intuitive and user-friendly methods for biometric person recognition. However, akin to other biometric systems, ASV faces vulnerabilities to spoofing, commonly known as "presentation attacks." These weaknesses are generally deemed unacceptable, necessitating the implementation of countermeasures or "presentation attack detection" systems. ASV systems are susceptible to replay, speech synthesis, and voice conversion attacks. The ASVspoof 2019 edition marks a significant milestone by addressing all three types of spoofing attacks within a single challenge [58]. Despite originating from the same source database and underlying protocol, these attacks are examined within two distinct use case scenarios.
The composition of ASVspoof2019 in presented in Figure 7.

	#speakers		#utterances			
Subset	Male	Female	Logical Access		Physical Access	
			Bona fide	Spoof	Bona fide	Spoof
Training	8	12	2,580	22,800	5,400	22,800
Development	4	6	2,548	22,296	5,400	24,300
Evaluation	21	27	7,355	63,882	18,090	116,640

Figure 7: ASVspoof2019 composition.

Spoofing attacks within a logical access (LA) scenario leverage cutting-edge speech synthesis and voice conversion technologies, including state-of-the-art neural acoustic and waveform model techniques. On the other hand, replay spoofing attacks within a physical access (PA) scenario are meticulously generated through controlled simulations, enabling more comprehensive analysis than previously possible.
For the development of this thesis, we chose to use only the logical access (LA) section of the dataset, its composition is presented in Table 4, where the rows are the different generation algorithms (Real for

	ASVspoof2019		
	Train Set	**Dev Set**	**Eval Set**
Real	2580	2548	7355
A01	3800	3716	-
A02	3800	3716	-
A03	3800	3716	-
A04-A16	3800	3716	4914
A05	3800	3716	-
A06-A19	3800	3716	4914
A07	-	-	4914
A08	-	-	4914
A09	-	-	4914
A10	-	-	4914
A11	-	-	4914
A12	-	-	4914
A13	-	-	4914
A14	-	-	4914
A15	-	-	4914
A17	-	-	4914
A18	-	-	4914
Total	25380	24844	71237
Num. Speakers	20	20	67

Table 4: Composition of the ASVspoof2019 dataset.

the bonafide tracks) and the columns are the partitions of the dataset. ASVspoof2019 has been used to train and validate the RawNet2 detector with its *train* and *dev* partition. The *eval* partition has been used to test the model by passing each track as input to the generation pipeline and completing the whole procedure proposed as idea in this thesis.

A second dataset, namely FoR [39], has been used for a further experiment with the purpose of testing the proposed method in an open-set scenario.

4.3. Detector

In this work we employed a single DL architecture for the detector utilized to extract embeddings and predictions for the proposed techniques. RawNet2 [52] processes raw audio and possesses the capability to learn cues that conventional countermeasure solutions may not detect. The second version of RawNet was selected with respect to its first iteration due to the bigger dimension of the penultimate layer (128 vs. 1024), used for embedding extraction, and the use of SincNet [38]. In short, the standard RawNet2 is composed as follows:

- **SincNet:** Processes the raw speech waveform to generate filterbanks using the Mel scale and

applies filtering to the input signal. This approach allows for the direct learning of low and high cutoff frequencies from the data, resulting in a more meaningful filter bank structure and outputs. In the original architecture it has size 64000.
- **Six Residual Blocks:** These blocks are stacked on each other and include batch normalization, leaky ReLU activation, and convolutional layers. They facilitate the propagation of information through deeper networks, enhancing the network's capability.
- **Attention Mechanisms:** Applied to the outputs of the residual blocks to derive more discriminative representations, enhancing the model's ability to focus on relevant features.
- **GRU (Gated Recurrent Unit):** A structure that utilizes a gating mechanism to control the flow of information. GRUs address the issue of vanishing gradients during backpropagation, which makes it challenging for the network to learn meaningful knowledge.
- **Fully Connected Layer:** The fully connected layer, with size 1024, takes the output of the GRU layer as input. This layer is utilized to extract embeddings.
- **Logoftmax:** The final layer is a logsoftmax layer with classes equals to the number of speakers if it is an attribution task or two if its a detection task.

In this thesis we tailored the network a bit to meet our needs. The input size has changed from 64000 to 32000, that is because we maintained a consistent sampling rate of 16kHz across all experiments and datasets and chose a window size of 2 seconds. When training the considered model, indeed, we always performed a windowing on the audio recordings, meaning that we considered only shorter portions of fixed length of them as input to the detector. To select the optimal length (in seconds) for the window, we conducted several detection experiment to evaluate our model performances on different window lengths. When needed, we padded signals too short to reach the minimum length (a single window length) by extracting a portion of the same signal from the beginning, as long as the number of samples needed, and then appending it at the end of the signal.
It is important to state that the window cut from the audio to be taken as input from the detector is sampled randomly along the usable audio length. This because, as presented in [31], different TTS models tend to have biases about the silence, in a generated deepfake, before the first word. This would make too easy for the detector to spot deepfakes, due to the repeating pattern of silence intrinsic in the different generative architectures.

4.4. Training procedure

The slightly modified version of RawNet2 utilized in this thesis has been trained with the ASVspoof2019 *train* partition and validated with the *dev* partition. As stated in the introduction of section 4, each audio input has been cut with a window of 2 seconds in a random position along the usable length. In the event of an audio shorter than 2 seconds the input is padded to the correct input length for the model. To maintain the training procedure unbiased, every batch uses the same amount of bonafide (real) and spoof (fake) tracks. The parameters of the training are the following:
- **Learning Rate:** 0.0001
- **Epochs:** 50
- **Batch Size:** 128
- **Window Length:** 2s/32000 samples at 16kHz
- **Loss Function:** Cross-Entropy
- **Optimizer:** Adam

The Cross-Entropy loss function is given by:

$$H(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{1}$$

where:

- y is the true distribution (usually a one-hot encoded vector),
- $\hat{y}$ is the predicted distribution (output of the model),
- n is the number of classes,

Metric Function	Formula
Cosine Distance	$1 - \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}$
Euclidean Distance	$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
Manhattan Distance	$\|x_2 - x_1\| + \|y_2 - y_1\|$

Table 5: Metric Functions and Formulas

- y_i and $\hat{y}_i$ are the i-th elements of y and $\hat{y}$ respectively.

It is important to underline that for the Cross-Entropy loss function, has been taken the possibility to take into account uncertainties in the dataset through the use of label-smoothing. Label smoothing is a regularization technique that addresses issues related to both overfitting and overconfidence. Label smoothing replaces one-hot encoded label vector $\hat{y}$ with a mixture of $\hat{y}$ and the uniform distribution:

$$y_{\text{ls}} = (1 - \alpha) \cdot y_{\text{hot}} + \frac{\alpha}{K} \tag{2}$$

where:

- y_{ls} represents the output vector,
- α is a scalar between 0 and 1,
- y_{hot} is the one-hot encoded vector, and
- K is the number of classes.

If $\alpha = 1$, we get the uniform distribution. In our case the α coefficient has been set equal to 0.3

4.5. Metric operations

The embeddings, or feature vectors, $\mathbf{e}_i$ with $\mathbf{i} \in \{1, 15\}$ relative to the 15 generated *support tracks* and the one for original track $\mathbf{e}$ are extracted from the penultimate layer of the RawNet2, a dense fully-connected layer composed of 1024 neurons. We then need to assess the distance $\mathbf{d}_i$ with $\mathbf{i} \in \{1, 15\}$ in the latent space between each of the *support tracks* embeddings $\mathbf{e}_i$ and those of the original track $\mathbf{e}$. This distance is evaluated using 3 different metric functions, which are detailed below in Table 5, along with their respective formulas. The first is the cosine distance, which is generally used as a metric for measuring distance when the magnitude of the vectors does not matter and represents how dissimilar two vectors are. The second is the Euclidean Distance, basically the length of the line segment connecting two points in the space. The last one is Manhattan distance, an unusual metric function to use in the latent space. It is the distance between two points measured along axes at right angles. Manhattan distance works very well for high-dimensional datasets.

After calculating the metrics, we receive 3 vectors (one for each type of metric function), each consisting of 15 values $\mathbf{d}_i$ with $\mathbf{i} \in \{1, 15\}$. As an added experiment the resulting feature vectors will be separately fed into the ML classifier to predict the final score.

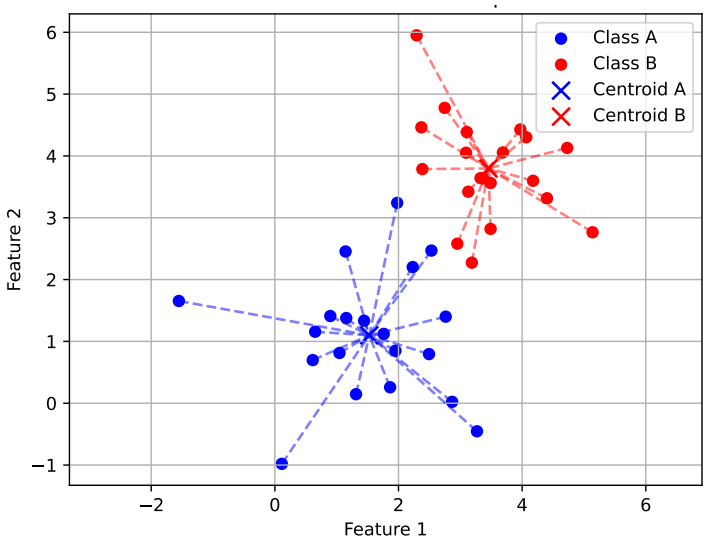


Figure 8: *Nearest Centroid* work principle.

4.6. Nearest Centroid Classifier

After testing several ML algorithms for classification, we decided to go with *Nearest Centroid* thanks to its optimal performances on balance accuracy and easy of use. *Nearest Centroid* is, indeed, one of the most rudimentary algorithms for classification. It works by representing data in the latent space and assigning to the input the label of the class of training samples whose mean (centroid) is closest to it, as depicted in Figure 8. In our experimental scenario, the *Nearest Centroid* algorithm takes as input a feature vector composed of the 15 distances extracted, always in the same order with respect to the generation models.

5. Results

In this section we present the results achieved by our proposed method. For simplicity this section is divided into four main subsections:

- **Support Tracks:** Here we evaluate the performances of our generation pipeline by simply test the *support tracks* $\mathbf{x}_i$ with our detector $\mathcal{D}$.
- **Distance metrics:** Here we perform an analysis over our complete method by utilizing directly the thresholded averaged distances $\mathbf{d}_i$ as our final score $\mathbf{s}$.
- **Classifier:** Here we perform an added experiment to evaluate the model utilizing the fifteen distances $\mathbf{d}_i$ as an input feature vector for a *Nearest Centroid* Classifier.
- **New dataset:** Here we test the proposed method on a completely different dataset, FoR, to extend the credibility of the results to different situations. The same generation pipeline and detector $\mathcal{D}$ are employed.

5.1. Support Tracks

As a preliminary experiment, it is interesting to state how our generated *support tracks* behave in a detection scenario with our detector $\mathcal{D}$ trained on ASVspoof2019. We took as score the output of the last layer of our implementation of the RawNet2. The last layer is a logsoftmax, so, in order to have the prediction in linear scale, an exponential function is applied to the results. In Figure 9 there is an histogram plot representing the scores of the *support tracks*. As we can clearly see, the majority of the tracks are classified correctly, 1 being fake and 0 real, although several tracks were classified as real, tricking the network.

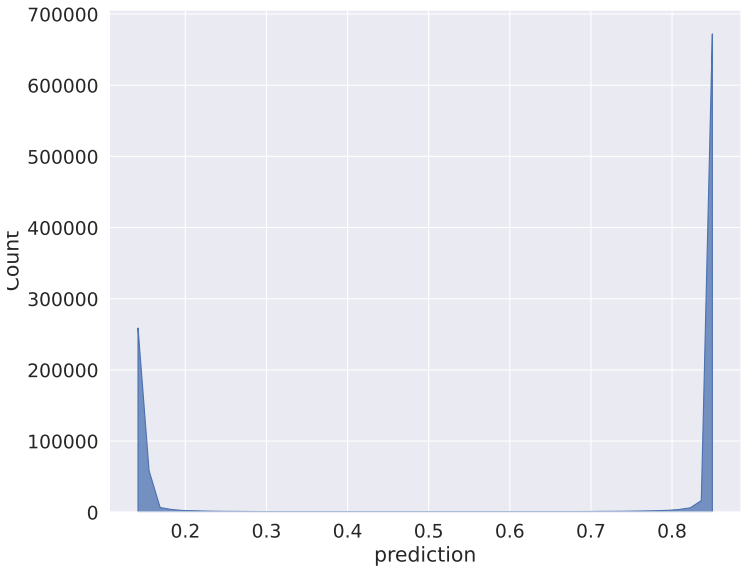


Figure 9: Histogram representing the score distribution for the *support tracks* tested with our detector $\mathcal{D}$, with 1 corresponding to the track being detected as fake and 0 to the track being detected as real.

As a reference to evaluate the performance of the detector with the *support tracks*, we can see in Figure 10 the predictions distribution for the original tracks of the ASVspoof2019 *eval* partition. For clarity, on the x axis 1 corresponds to the track being detected as fake and 0 to the track being detected as real. The grouping is based on the original label of the signal with the same logic of assignment.

Furthermore, in Figure 11, 12 and 13 there are histogram plots representing the scores of the *support tracks*, respectively for the ones generated with the first VC model, with the second VC model and the ones which skipped the VC step. In Figure 11 and 13 we can underlain a similar behaviour of the network. Almost all the tracks from these two subsets are classified correctly. It is important, however, to give variety to the latent space we want to construct by inserting wrongly predicted tracks, such as the ones in output of the second VC model depicted in Figure 12. This way of acting has shown to permit to have a more robust interpretation of the latent space in our experiments.

5.2. Distance Metrics

Here we report our result on the synthetic speech detection task based on the outputs of the metric computation phase. We proceeded in the following way: we took all the tracks from the ASVspoof2019 *eval* partition (71237 original tracks), we tested them with our detector to extract the predictions, generated the *support tracks* and extracted the distances, with each metric function, between them and the original in the latent space. Then we computed the average between the 15 values of the 3 resulting feature vectors, thresholded them and assigned the pseudo-score for the individual metric function, which represents the final result of our pipeline. The goal of this experiment is to understand how well the distances can describe the similarities between the *support tracks* and the original signal. As evaluation metric we selected the Area Under Curve (AUC). AUC is referred to the Receiver Operating Characteristic (ROC) curve, an evaluation metric for binary classification problems. It is a probability curve that shows the performance of a classification model at all classification thresholds. The AUC is the measure of the ability of a binary classifier to distinguish between classes and is used as a summary of the ROC curve. Figure 14 shows the results we achieved in terms of AUC,

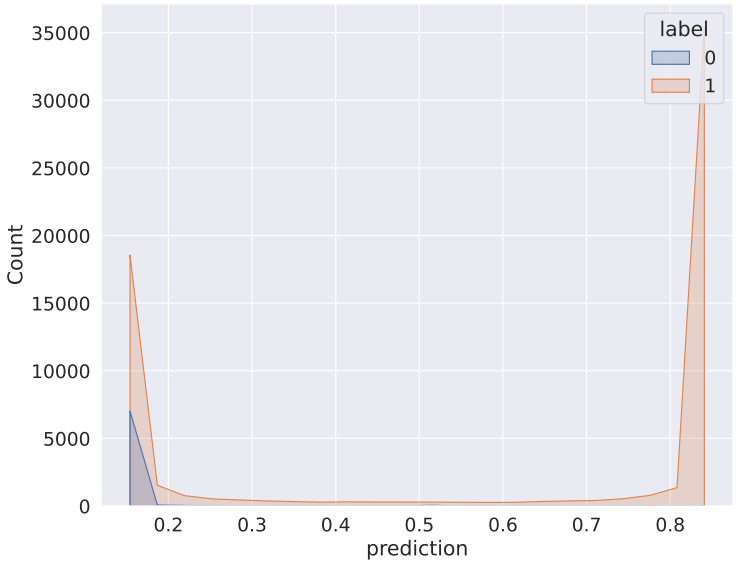


Figure 10: Histogram representing the score distribution for the original tracks, divided by original label, of the ASVspoof2019 *eval* partition tested with our detector $\mathcal{D}$

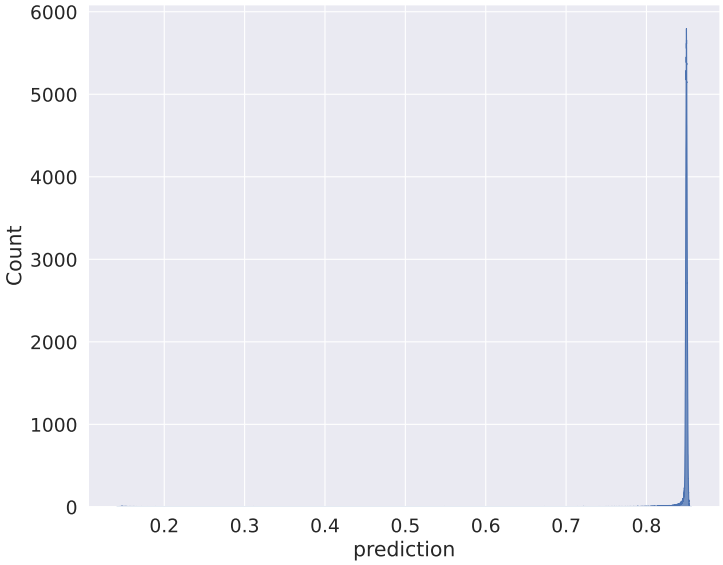


Figure 11: Histogram representing the score distribution for just the *support tracks* generated with the first VC model, tested with our detector $\mathcal{D}$.

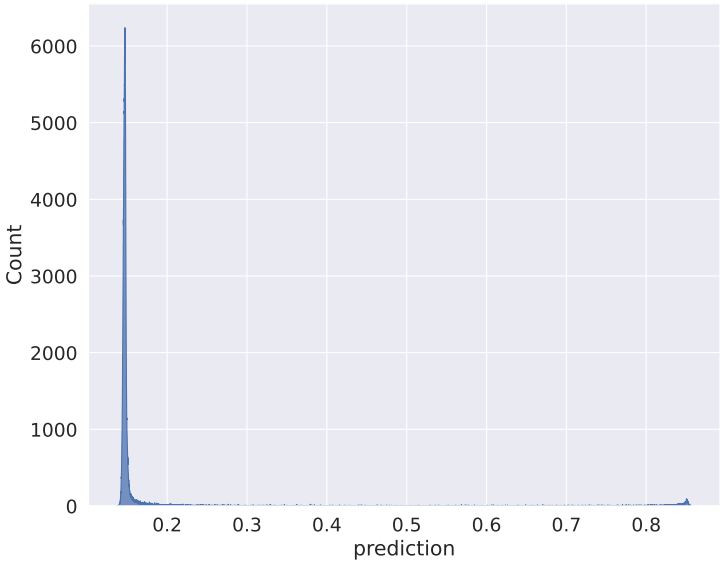


Figure 12: Histogram representing the score distribution for just the *support tracks* generated with the second VC model, tested with our detector $\mathcal{D}$.

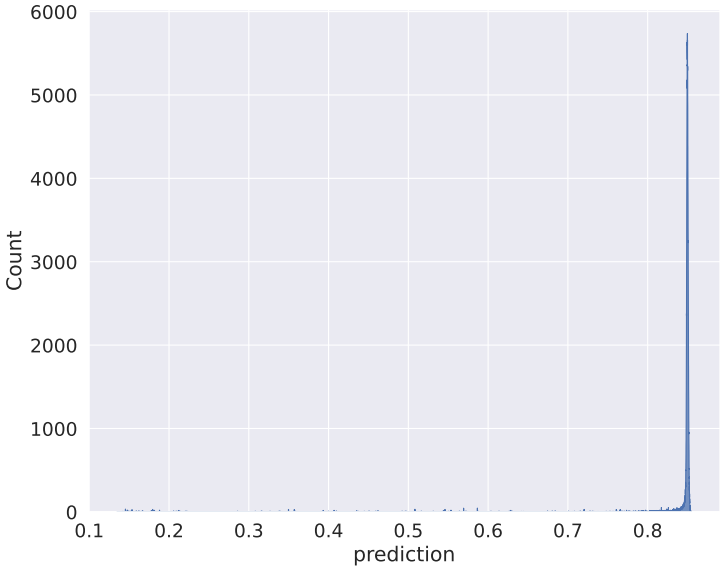


Figure 13: Histogram representing the score distribution for just the *support tracks* generated without the VC model step, tested with our detector $\mathcal{D}$.

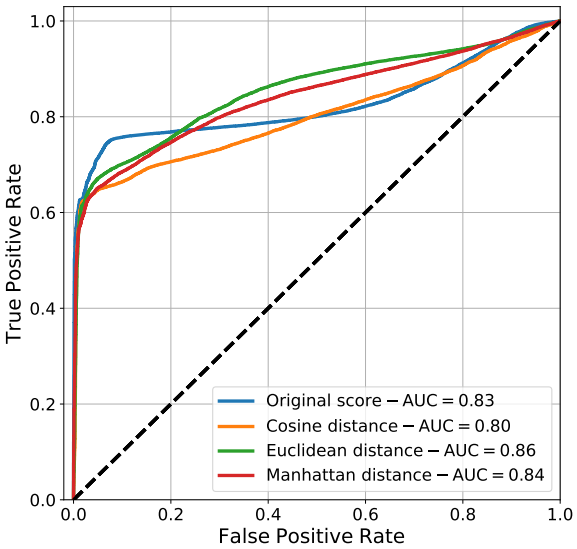


Figure 14: ROC AUC for the experiment conducted on the ASVspoof2019 dataset.

	AUC	**Equal Error Rate (EER)**
Prediction	0.827	0.228
Cosine Distance	0.802	0.275
Euclidean Distance	**0.856**	**0.226**
Manhattan Distance	0.842	0.234

Table 6: Evaluation metrics for the experiment conducted on the ASVspoof2019 dataset. Results in bold underline the best performance of the metrics with regard to the predictions.

where prediction is the score from the last layer of the network, while the different metrics are the pseudo-scores previously described.

As we can see, the graph shows promising results in terms of performance improvements of the proposed method compared to a simple detector. The score that performs the best is the Euclidean Distance, soon followed by the Manhattan Distance. Cosine Distance, in this form of experiment, performs slightly worst than the predictions of the detector. In Table 6 there are the results of the AUC for the experiment, along with the EER. The EER is the second evaluation metric we decided to use and is the point where the system is just as prone to incorrectly accepting a non-matching individual as it is to erroneously rejecting a matching individual. A perfect evaluation score for model would yield an EER of zero, so a lower EER value indicates a better performance. For the EER, we can see in 6 that the only metric function that outperforms the predictions of the detector is the Euclidean Distance, while Cosine and Manhattan Distance are slightly higher.

5.3. Classifier

For the added experiment of our evaluation of the proposed method we tested the performances of the *Nearest Centroid* algorithm in classifying the original signals. The input for the ML algorithm is

	AUC	EER	Balanced Accuracy
Prediction	0.827	0.228	0.805
Cosine Distance	0.805	0.240	0.805
Euclidean Distance	0.810	0.280	**0.810**
Manhattan Distance	0.809	0.270	0.809

Table 7: Evaluation metrics for the added experiment conducted on ASVspoof2019 with the ML Classifier. Results in bold underline the best performance of the metrics with regard to the predictions.

	AUC	EER
Prediction	0.489	0.511
Cosine Distance	**0.540**	**0.449**
Euclidean Distance	0.495	0.502
Manhattan Distance	0.463	0.528

Table 8: Evaluation metrics for the added experiment conducted on a different dataset. Results in bold underline the best performance of the metrics with regard to the predictions.

a feature vector of size 15 composed of the distances $\mathbf{d}_i$ per original track. The output is directly the predicted class. As for previous experiments, 1 corresponds to the track being detected as fake and 0 to the track being detected as real. Half of the input feature vectors relative to the original tracks were used for fitting the model, the other half to test it. In Table **??** are presented the evaluation metrics for each metric function, along with the one for the simple predictions. We decided to add the Balanced Accuracy due to the fact that we had to use half of the tracks to fit the model.
The results underline a different behavior with respect to the previous experiments. Here we can see that the AUC evaluation metric in Table 7 is not as high as the one in Table 6 and is similar across all metric functions.

5.4. Generalization test

To give more credibility to the results in a more generalized scenario we employed a different dataset (FoR) to test, once again, our proposed method. We tested it both in its original form and with the added classification step. In Table 8 and 9 are presented the results for respectively the test performed with the original pipeline and with the *Nearest Centroid* Classifier. It is interesting to underline that the trend with the results is opposite to the one of the previous experiment on the ASVspoof2019 dataset. Here, indeed, we found out that, besides performing better than the original predictions in any scenario, the method relying on the ML classifier performs better than the one that does not employ it. Furthermore, in both experiments we can see an improvement of the method proposed in almost all results, with respect to the prediction of the classifier. This encourages experiments with different classifiers in order to get the best out of the distance feature vectors.

6. Conclusions

Recent advancements in DL have led to the emergence of a new synthesized media content known as deepfake, which carries the potential to adversely affect individuals' lives. This risk underscores the urgency of deploying systems capable of automatically detecting such falsifications with accuracy.

	AUC	EER
Prediction	0.489	0.511
Cosine Distance	0.543	**0.173**
Euclidean Distance	0.541	0.211
Manhattan Distance	**0.563**	0.257

Table 9: Evaluation metrics for the added experiment conducted on a different dataset with the ML Classifier. Results in bold underline the best performance of the metrics with regard to the predictions.

The MMF community guides the research towards new methodology to counteract the problem and finding solutions for the problem of Synthesized Audio Detection. In this work we presented a new audio forensics approach suitable for audio deepfake detection, named DeepMetric, capable of detecting signals synthesized with the speech generation techniques of TTS and VC. The three main phases of the proposed method are: Generation Phase, Embedding extraction Phase and Metric Comparison Phase. In the generation phase we synthesize *support tracks* by taking advantage of existing TTS and VC models. During the second phase a detector is utilized as a feature extractor by taking as input the *support tracks* with the original signal and giving as output the embeddings from its penultimate layer. For the last phase, the distances between each generated track and the original signal is computed in the latent space constructed with the aforementioned embeddings, the distances are employed to compute a final score for the prediction. The main benefit of our method is that it does not rely on specific detection or generation models, but it is, instead, modular and capable of being easily upgradable with time. To perform a complete evaluation, we have tested the system with both the dataset used to train the detector, ASVspoof2019, and a different dataset, FoR. The results show a promising first step in the direction of boosting the performances of an Audio Deepfake Detector through a modular method, by utilizing distances in the latent space as a measure of similarity we can exploit the artifacts present in our generated *support tracks* to retrieve more robust predictions.
We can still apply many improvements to the proposed methodology. The first one focuses on improving the Embedding extraction Phase by employing different architectures for the detector, together with different pre-processing for the original input signal in order to make it more reliable in open-set scenarios. The second one requires to find an optimal classifier to exploit completely the distance feature vector obtained through embedding comparison with each *support track*. The third one is to find a strategy to optimize the TTS and VC model selection. *Support tracks* generation takes up the most amount of time relative to the whole pipeline proposed, therefore it is important to avoid implementing unnecessary models that would slow down the process.

References

[1] Ehab A AlBadawy, Siwei Lyu, and Hany Farid. Detecting AI-Synthesized Speech Using Bispectral Analysis. In *IEEE workshop on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] Zaynab Almutairi and Hebah Elgibreen. A review of modern audio deepfake detection methods: challenges and future directions. *Algorithms*, 15(5):155, 2022.

[3] Moustafa Alzantot, Ziqi Wang, and Mani B Srivastava. Deep residual neural networks for audio spoofing detection. *arXiv preprint arXiv:1907.00501*, 2019.

[4] Markus Appel and Fabian Prietzel. The detection of political deepfakes. *Journal of Computer-Mediated Communication*, 27(4):zmac008, 2022.

[5] Matthew Baas, Benjamin van Niekerk, and Herman Kamper. Voice conversion with just nearest neighbors. *arXiv preprint arXiv:2305.18975*, 2023.

[6] BT Balamurali, Kinwah Edward Lin, Simon Lui, Jer-Ming Chen, and Dorien Herremans. Toward robust audio spoofing detection: A detailed comparison of traditional and learned features. *IEEE Access*, 7:84229–84241, 2019.

[7] 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), 2021.

[8] Edresson Casanova, Julian Weber, Christopher D Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A Ponti. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International Conference on Machine Learning*, 2022.

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[10] Davide Cozzolino, Alessandro Pianese, Matthias Nießner, and Luisa Verdoliva. Audio-visual person-of-interest deepfake detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[11] Thierry Dutoit. High-quality text-to-speech synthesis: An overview. *Journal Of Electrical And Electronics Engineering Australia*, 17(1):25–36, 1997.

[12] Isaac Elias, Heiga Zen, Jonathan Shen, Yu Zhang, Ye Jia, RJ Skerry-Ryan, and Yonghui Wu. Parallel Tacotron 2: A non-autoregressive neural TTS model with differentiable duration modeling. *arXiv preprint arXiv:2103.14574*, 2021.

[13] Ameer Hamza, Abdul Rehman Rehman Javed, Farkhund Iqbal, Natalia Kryvinska, Ahmad S Almadhor, Zunera Jalil, and Rouba Borghol. Deepfake audio detection via MFCC features using machine learning. *IEEE Access*, 10:134018–134028, 2022.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] Brian Hosler, Davide Salvi, Anthony Murray, Fabio Antonacci, Paolo Bestagini, Stefano Tubaro, and Matthew C Stamm. Do deepfakes feel emotions? a semantic approach to detecting deepfakes via emotional inconsistencies. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[16] Xuedong Huang, James Baker, and Raj Reddy. A historical perspective of speech recognition. *Communications of the ACM*, 57(1):94–103, 2014.

[17] Won Jang, Dan Lim, Jaesam Yoon, Bongwan Kim, and Juntae Kim. Univnet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation. *arXiv preprint arXiv:2106.07889*, 2021.

[18] Jee-weon Jung, Hee-Soo Heo, Ju-ho Kim, Hye-jin Shim, and Ha-Jin Yu. Rawnet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification. *arXiv preprint arXiv:1904.08104*, 2019.

[19] Jee-Weon Jung, Hee-Soo Heo, Il-Ho Yang, Hye-Jin Shim, and Ha-Jin Yu. A complete end-to-end speaker verification system using deep neural networks: From raw signals to verification result. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[20] Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077, 2020.

[21] Jaehyeon Kim, Jungil Kong, and Juhee Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, 2021.

[22] Kong, Jungil and Kim, Jaehyeon and Bae, Jaekyoung. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

[23] Matthew B Kugler and Carly Pace. Deepfake privacy: Attitudes and regulation. *Nw. UL Rev.*, 116:611, 2021.

[24] Adrian Łańcucki. Fastpitch: Parallel text-to-speech with pitch prediction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[25] Jingyi Li, Weiping Tu, and Li Xiao. Freevc: Towards High-Quality Text-Free One-Shot Voice Conversion. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[26] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[27] Mishaim Malik, Muhammad Kamran Malik, Khawar Mehmood, and Imran Makhdoom. Automatic speech recognition: a survey. *Multimedia Tools and Applications*, 80:9411–9457, 2021.

[28] Driss Matrouf, Nicolas Scheffer, Benoit GB Fauve, and Jean-François Bonastre. A straightforward and efficient implementation of the factor analysis model for speaker verification. In *Interspeech*, 2007.

[29] Seyed Hamidreza Mohammadi and Alexander Kain. An overview of voice conversion systems. *Speech Communication*, 88:65–82, 2017.

[30] Nicolas M Müller, Pavel Czempin, Franziska Dieckmann, Adam Froghyar, and Konstantin Böttinger. Does audio deepfake detection generalize? *arXiv preprint arXiv:2203.16263*, 2022.

[31] Nicolas M Müller, Franziska Dieckmann, Pavel Czempin, Roman Canals, Konstantin Böttinger, and Jennifer Williams. Speech is silver, silence is golden: What do ASVspoof-trained models really learn? *arXiv preprint arXiv:2106.12914*, 2021.

[32] Andreas Nautsch, Xin Wang, Nicholas Evans, Tomi H Kinnunen, Ville Vestman, Massimiliano Todisco, Héctor Delgado, Md Sahidullah, Junichi Yamagishi, and Kong Aik Lee. ASVspoof 2019: spoofing countermeasures for the detection of synthesized, converted and replayed speech. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 3(2):252–265, 2021.

[33] Simon Parkin. The rise of the deepfake and the threat to democracy. *The Guardian*, 22:2019, 2019.

[34] Nikolaos Passalis and Anastasios Tefas. Dimensionality reduction using similarity-induced embeddings. *IEEE transactions on neural networks and learning systems*, 29(8):3429–3441, 2017.

[35] Alessandro Pianese, Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Deepfake audio detection by speaker verification. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2022.

[36] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[37] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, 2023.

[38] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *IEEE Spoken Language Technology Workshop (SLT)*, 2018.

[39] Ricardo Reimao and Vassilios Tzerpos. For: A dataset for synthetic speech detection. In *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, 2019.

[40] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*, 2020.

[41] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted Gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.

[42] Yohanna Rodríguez-Ortega, Dora María Ballesteros, and Diego Renza. A machine learning model to detect fake voice. In *International Conference on Applied Informatics (ICAI)*, 2020.

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

[44] Davide Salvi, Paolo Bestagini, and Stefano Tubaro. Reliability Estimation for Synthetic Speech Detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[45] Davide Salvi, Paolo Bestagini, and Stefano Tubaro. Towards Frequency Band Explainability in Synthetic Speech Detection. In *31st European Signal Processing Conference (EUSIPCO)*, 2023.

[46] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[47] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36, 2017.

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

[49] Frank K Soong, Aaron E Rosenberg, Bling-Hwang Juang, and Lawrence R Rabiner. Report: A vector quantization approach to speaker recognition. *AT&T technical journal*, 66(2):14–26, 1987.

[50] Catherine Stupp. Fraudsters used ai to mimic ceo's voice in unusual cybercrime case. *The Wall Street Journal*, 30(08), 2019.

[51] Nishant Subramani and Delip Rao. Learning efficient representations for fake speech detection. In *AAAI Conference on Artificial Intelligence*, 2020.

[52] Hemlata Tak, Jose Patino, Massimiliano Todisco, Andreas Nautsch, Nicholas Evans, and Anthony Larcher. End-to-end anti-spoofing with rawnet2. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[53] Luisa Verdoliva. Media forensics and deepfakes: an overview. *IEEE Journal of Selected Topics in Signal Processing*, 14(5):910–932, 2020.

[54] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *science*, 359(6380):1146–1151, 2018.

[55] Xin Wang, Junichi Yamagishi, Massimiliano Todisco, Héctor Delgado, Andreas Nautsch, Nicholas Evans, Md Sahidullah, Ville Vestman, Tomi Kinnunen, Kong Aik Lee, et al. Asvspoof 2019: A large-scale public database of synthesized, converted and replayed speech. *Computer Speech & Language*, 64:101114, 2020.

[56] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[57] Ke Wu, DM Hicks, and B Yegnanarayana. Voice conversion. *Speech Communication*, 8(2):147–158, 1989.

[58] Zhizheng Wu, Junichi Yamagishi, Tomi Kinnunen, Cemal Hanilçi, Mohammed Sahidullah, Aleksandr Sizov, Nicholas Evans, Massimiliano Todisco, and Hector Delgado. Asvspoof: the automatic speaker verification spoofing and countermeasures challenge. *IEEE Journal of Selected Topics in Signal Processing*, 11(4):588–604, 2017.

[59] Geng Yang, Shan Yang, Kai Liu, Peng Fang, Wei Chen, and Lei Xie. Multi-band melgan: Faster waveform generation for high-quality text-to-speech. In *IEEE Spoken Language Technology Workshop (SLT)*, 2021.

[60] Jiangyan Yi, Chenglong Wang, Jianhua Tao, Xiaohui Zhang, Chu Yuan Zhang, and Yan Zhao. Audio Deepfake Detection: A Survey. *arXiv preprint arXiv:2308.14970*, 2023.

Glossary

AI Artificial Intelligence. 2
ASR Automatic Speaker Recognition. 3, 10–13
ASV Automatic Speaker Verification. 14
AUC Area Under Curve. 19, 22–24

CNN Convolutional Neural Network. 2, 4, 7

DL Deep Learning. 2–4, 7, 8, 11, 15, 23

EER Equal Error Rate. 22–24

GRU Gated Recurrent Unit. 7

ML Machine Learning. 2, 6, 7, 11, 12, 17, 18, 22–24
MMF MultiMedia Forensic. 2, 6, 24

RNN Recurrent Neural Network. 3, 4
ROC Receiver Operating Characteristic. 19, 22

TTS Text-to-Speech. 2–6, 9–13, 16, 24

VC Voice Conversion. 2–6, 9–13, 19–21, 24

Abstract in lingua italiana

Al giorno d'oggi, i nuovi progressi tecnologici nella generazione del parlato hanno portato alla possibilità di creare un parlato sintetico con relativa facilità utilizzando dispositivi di livello consumer. Questo fenomeno ha aperto le porte a nuovi scenari e possibilità interessanti, come lo sviluppo di nuovi assistenti vocali, dispositivi per persone con condizioni patologiche, didattica, ecc. Tuttavia, quando queste tecnologie vengono utilizzate da utenti malintenzionati, possono portare a conseguenze illegali ed a scenari pericolosi per la popolazione. Ad esempio, è possibile clonare la voce di una persona utilizzando solo pochi secondi di audio di riferimento e farle pronunciare frasi arbitrarie. Questi nuovi falsi generati vengono chiamati deepfake. Questo può portare a problemi di credibilità, problemi di privacy, audio minacciosi, furto di identità e altri pericoli. Per affrontare l'uso malevolo del parlato sintetico e dei dati audio contraffatti, la comunità forense orienta la ricerca verso nuovi metodi per lo spoofing e il rilevamento di questi nuovi falsi generati. In questo lavoro ci concentriamo sul problema del rilevamento del parlato sintetico, che consiste nel prendere in input un segnale audio e determinarne l'autenticità. Le moderne architetture di rilevamento presentano risultati incoraggianti, ma sono ben lontane dall'essere perfette a causa dei rapidi progressi delle tecniche di generazione di deepfake. In questa tesi, proponiamo un sistema per migliorare le prestazioni di un rilevatore di parlato sintetico facendo uso di tracce di supporto. Dato un segnale audio vocale, generiamo un insieme di tracce di supporto che contengono lo stesso contenuto testuale e vocale della traccia in analisi e le utilizziamo per migliorare l'accuratezza del rilevamento del sistema. Il metodo si basa sul calcolo delle distanze tra la traccia di riferimento e queste tracce di supporto, che portano a previsioni più stabili e affidabili. Le tracce di supporto sono implementate utilizzando diversi metodi di conversione vocale e Text-to-Speech esistenti. I risultati forniti dalla pipeline completa proposta in questo lavoro mostrano una nuova ed entusiasmante possibilità di migliorare il rilevamento dei deepfake attraverso un metodo modulare, svincolato da modelli specifici.

Parole chiave: Multimedia Forense, Parlato Sintetico, Deepfake, Conversione Vocale, Conversione Testo-Parlato