University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Cybersecurity

Audio Deepfake Detection based on Implicit Neural Representations

Supervisor
Prof. Simone Milani
University of Padova

Advisors
dott. Daniele Mari
dott. Saverio Cavasin

Master Candidate
Niccolò Simonato

Student ID
2093050

Academic Year
2023-2024

”Your ~~eyes~~ ears can deceive you, don’t trust them.”
- Obi-Wan Kenobi

Abstract

The proliferation of audio deepfakes, synthetic audio generated using advanced machine learning techniques, poses significant risks to modern society, ranging from misinformation to security threats. As deepfake technology continues to advance, the need for robust detection mechanisms becomes critical.

This thesis introduces a novel approach for detecting audio deepfakes utilizing implicit neural representations. By leveraging these representations, we aim to capture and analyze the unique patterns inherent in audio, particularly focusing on its silenced parts. The model's performance is close to state-of-the-art, proving the effectiveness of this methodology.

To assess the differences in detection accuracy between humans and algorithms, perceptive tests were performed in a controlled environment, and compared against a baseline model, proving that generated samples are becoming harder to detect for humans. The results show that human testers perform poorly, in comparison to the baseline model, supporting the fact that generated samples are becoming harder to detect for humans. These insights might be helpful for future research and highlight the importance of this type of analysis.

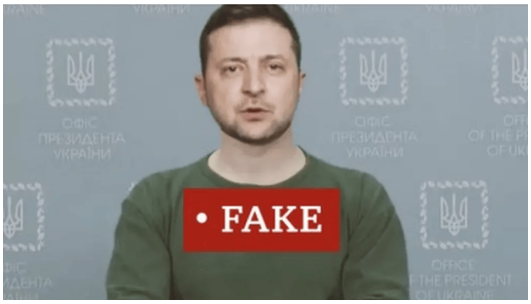
Contents

1 Introduction

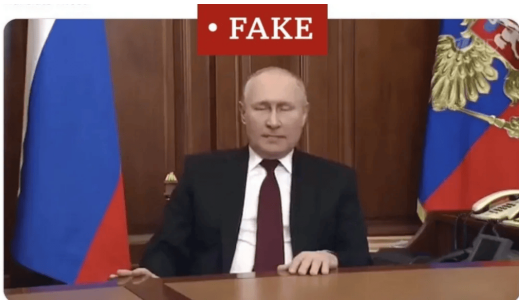
1.1 The need for Audio Deepfake detection

The last years were characterized by the groundbreaking introduction to the public of Generative AI models.
These models proved to be effective in generating multimedia content of every kind, such as photos, videos, audio, and textual content as well[1].
The motivation at the foundation of this study is the need for reliable means of detection of AI generated content, needed nowadays more than ever.
Social media aid the viral diffusion of Deep Fakes, whose content can be used to vehiculate disinformation. An example of this could be seen at the beginning of the Russian invasion of Ukraine in February 2022, when unconvincing Deep Fake videos that depicted president Volodymyr Zelens'kyj declaring peace went viral on X, formerly known as Twitter[2] (Figure 1.1a). A similar episode involved the Russian president, Vladimir Putin, as shown by Figure 1.1b.

Further instances of these episodes were recorded on X in September 2023, where several Deep Fake audios were used by the supporters of Muhammad Qasim to amplify the narrative that depicted him as the real Imam Mahdi[3]. Most of these audios are overlaid on short videos, that depict common Islamic cultural elements such as mosques and people in traditional clothing, and feature popular Islamic scholars, such as Mufti Menk (See Figure 1.2a) or Nouman Ali Khan (See Figure 1.2b), who endorses Muhammad Quasim as the "redeemer of Islam". In the context of

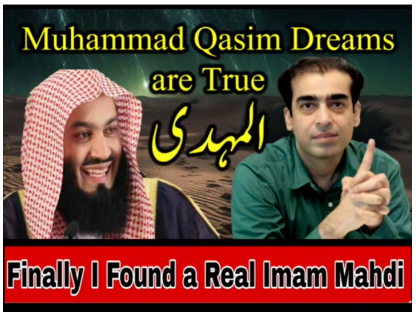


(a) A screenshot from the Deep Fake video picturing the Ukrainian President, Volodymyr Zelens'kyj.



(b) A screenshot from the Deep Fake video picturing the Russian President, Vladimir Putin.

Islamic culture, Imam Mahdi, also referred to as al-Mahdi is a figure that brings changes and renewal, and it's supposed to appear at the end of times as a savior, even though there are different interpretations among different Islamic groups[4]. The diffusion of these videos is an example of how Deep Fakes can be used as a vehicle of extremist propaganda, and pose a challenge for the moderation of media platforms.



(a) A screenshot from the video depicting Mufti Menk endorsing Muhammad Qasim.

(b) A screenshot from the video depicting Nouman Ali Khan endorsing Muhammad Qasim.

Episodes like this shake the foundation of trust in social media content, so much that social media platforms started implementing policies on this matter[5].
The existence of Deep Fakes is not a matter that only concerns social media, but also involves several aspects of everyday life. The following sections will delve into how Deep Fakes affect security systems, in particular Automatic Speaker Verification, and the evidence that is brought into trials.

1.2 Automatic Speaker Verification (ASV) systems

ASV systems are biometric verification systems based on voice, meaning that they leverage the voice patterns of the user to verify its identity[6]. This technology can be used effectively as a tool for multi-factor authentication, as the voice has two characteristics that make it ideal for this use case:

- It's easy to use.
- It's hard to forge.

The functioning of an ASV system involves two steps: enrollment and verification. During the first part, one or more voice samples of the user are recorded, and the system extracts the features that can be used to identify the user, such as the speaker's vocal tract, pitch, and speaking style[6]. These features are then encoded in a voiceprint, called a template, and associated with the identity of the user.
The verification phase involves collecting a single voice sample from the user and their claimed identity. The system then fetches from its voiceprint collection the one associated with the claimed identity. It compares the features from the template to the ones of the new sample by using a similarity metric. If the sample is close to a certain tolerance bound to the template, the user is authenticated, otherwise, it's rejected.
The introduction of Deep Fake (DF) techniques poses new challenges for Automatic Speaker Verification systems, as DFs can generate convincing voice samples with extreme ease. Numerous online tools are available, making it easier for malicious actors to perform spoofing attacks[7][8][9][10].

1.3 A brief history of the Audio Deep Fake Detection task

Since the use of Audio Deep Fakes became widespread, the scientific community didn't have to wait much to have some means of detection.
In 2021, Muller et al. [11] showed that one effective way to detect audio DF is to analyze silences, leveraging the anomalous distribution of silences' durations. In the same direction was the paper by Mari et al., where it was shown how first digit statistics, also known as Benford's Law, can be leveraged to distinguish fake audio samples [12].
The analysis of the silences is not the only approach that can be used: Short Term Long Term features and bicoherence features seem to be effective as well, as shown by Mari et al. [13].
Numerous Deep Learning architectures proved to be effective classifiers. It is worth mentioning the work of Cuccovillo et al.[14], who proposed an interpretable, transformer-based architecture for synthetic speech detection.
Given this background, it would seem like there are plenty of architectures that can properly detect fake audio tracks. The reality could not be more different: often, the performances of the models are tested on the very same dataset on which the models are trained, so the estimates of the performances tend to be overestimated. This was shown by Muller et al. [15], who tested several models for synthetic speech detection on an in-the-wild dataset. This is expected, as it is very hard to replicate a real-case scenario, due to the number of variables that should be taken into account: the encoding of the audio file, the compression rate, the quality of the microphone, etc.

1.4 Executive Summary

In this research, the detection of AI generated content will be addressed, and in particular, the detection of audio Deep Fakes. The aim of this study is the development of a Deep Learning model that can distinguish (to some extent) AI generated audio tracks from genuine ones, proving how Implicit Neural Representations can be used to generate significant features for Audio Deep Fake detection.
Finally, the results of a baseline model will be compared with the ones of human testers, so to assess how accurate are humans in distinguishing fake audio tracks from real ones.
This thesis is divided into the following chapters:

1. Introduction.
2. Related Works.
3. Implicit Neural Representations, this chapter illustrates their definition and their applications.
4. Implicit Neural Representation (INR)-based Deep Fake Detection, this chapter describes the proposed technique to perform Deep Fake Detection on audio tracks.
5. Perceptive Tests and Results Comparison, this section compares the performances of a baseline model with the ones of human testers.
6. In the final chapter, the conclusions of this study are drawn.

2 Related Works

2.1 Background on Artificial Intelligence

This section analyzes the topics most relevant to this study, providing an overview of the most crucial points.

2.1.1 Neural Networks' Architecture

Feed Forward Neural Network (FFNN)

Feed Forward Neural Networks are the most basic kind of Neural Network, based on the supervised learning paradigm, meaning that they need a dataset of labeled data to be trained.
One of the first examples of Neural Network was Rosenblatt's Perceptron, which is a linear binary classification model [16], meaning that it can classify the inputs in two classes, by employing a decision border in a linear space.
The design can be described as follows: a feature vector (x) is multiplied against a weight vector (w) and summed to a bias vector (b). The result is then compared to a threshold value (τ): if the result is greater or equal to τ, then the model returns $+1$, otherwise returns -1 (See Eq. 2.1)[16].

$$f_{w,\tau}(x) = \begin{cases} +1 & \text{if } w \cdot x \geq \tau \\ -1 & \text{otherwise.} \end{cases} \tag{2.1}$$

The value of w is obtained by performing the so-called *training*. This procedure is essentially an optimization problem, whose goal is to minimize the error of the model, measured by a *loss function* ($L(t, f_w(x))$), whose caveat, in general, is to be zero when the output of the model is the intended output (also known as *target*, t), and to grow in value as t and $f_w(x)$ grow in difference. In the Perceptron, the goal of the training phase is to minimize the loss function expressed in Eq. 2.2, which is the sum of the quantity $w_n^T \cdot x_n \cdot t_n$ among the subset of misclassified samples M[16].

$$E(w) = -\sum_{n \in M} w_n^T \cdot x_n \cdot t_n \tag{2.2}$$

The optimization problem is often solved by using a Stochastic Gradient Descent approach, which consists of iteratively changing the value of the vector w, according to the learning rule defined in Equation 2.3, where θ is the iteration number and η is the learning rate value [16].

$$w^{(\theta+1)} = w^{(\theta)} - \eta \cdot \nabla E(w^{(\theta)}) \tag{2.3}$$

The limitation of the Perceptron model is its inability to learn non-linear functions, such as the XOR function. This limitation is overcome by employing non-linear activation functions (preferably differentiable) on the output of the soft-decision function (Eq. 2.4), such as the sigmoid function (σ, Eq. 2.5) or the Rectified Linear Unit function (ReLU, Eq. 2.6). From now on, this variation of the Perceptron will be referred to as a *unit*.

$$soft(x) = w \cdot x + b \tag{2.4}$$

$$\sigma(x) = \frac{1}{1 + \exp -x} \tag{2.5}$$

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

In essence, Feed Forward Neural Network are made up of layers of these units, where ones in one layer transfer their outputs to the next one after receiving inputs from the previous layer. Figure 2.1 illustrates the general architecture of a simple FFNN.

The training of this architecture follows the same principles of Perceptron's learning rule, with some adaptations that are applied for efficiency reasons. This approach takes the name of *Back-propagation*, and it's essentially an extension of the Stochastic Gradient Descent algorithm with

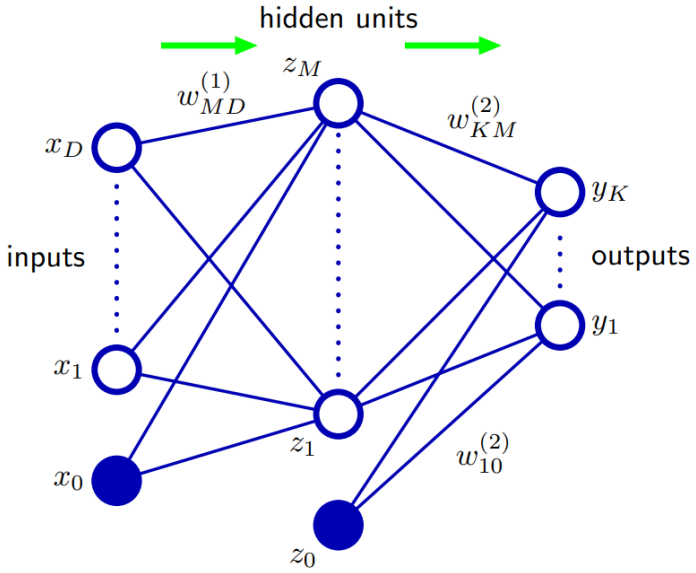


Figure 2.1: Architecture of a FFNN, adapted from [16].

the application of the Dynamic Programming technique. The procedure can be summarized as follows[16]:

1. The input value of x is propagated forward through the network and the activation values of the hidden units are computed.

2. Starting from the output units, the gradients of the value of the loss function with respect to the weights are computed, and a learning rule similar to one of the Perceptron is applied.

3. For each hidden layer, going backward, the last step is repeated with the values of the next layer.

As it will be reviewed in the following sections, this algorithm can be extended to other architectures, and it's still widely used to train more networks such as the Convolutional Neural Network or Transformers architectures[17][18].

Convolutional Neural Network (CNN)

Convolutional Neural Networks are a special kind of Neural Network, designed to reduce the number of parameters used by fully connected networks. This is achieved by using the convolution operation (See Eq. 2.7) which allows to exploit local correlation and weight sharing. In

particular, convolution allows one to learn one feature with a single set of weights, which is not possible for fully connected networks, thus achieving similar performances with fewer parameters.

$$C(i) = (L_1 * L_2)(i) = \sum_n L_1(n) \cdot L_2(i - n) \tag{2.7}$$

This concept is often applied to image-related tasks since the inductive biases introduced by convolutions are particularly well suited for this type of data. Additionally, working with fully connected networks at such a high resolution would lead to high computational complexity. Nevertheless, convolution is also often applied to one-dimensional data, such as time series or audio signals, since they also present locally correlated values that allow to fully exploit the properties of CNNs allowing to easily capture curves' patterns.
In classification tasks, this design allows the network to capture local patterns more efficiently than they would do by using fully connected layers: in fact, dense layers connect each neuron of the input layer to each neuron of the next layer. By contrast, convolutional layers connect each neuron of the input layer to the closest ones in the next layer. Figure 2.2 pictures the structure of the typical convolutional layer, while pictures an example of the MaxPooling operation.

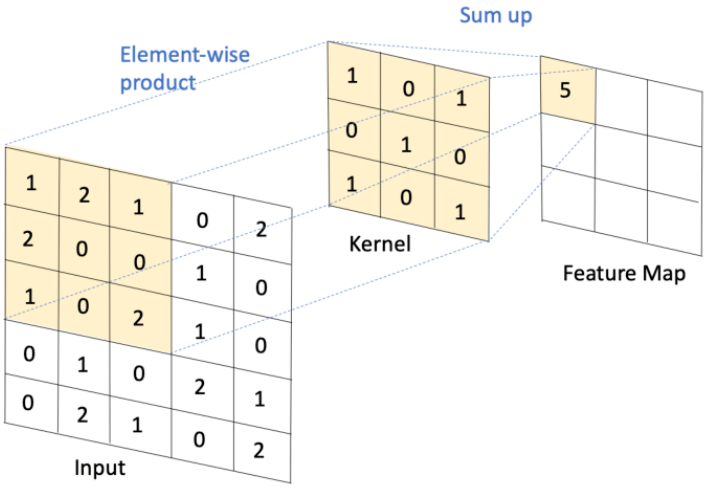


Figure 2.2: An example of the convolution operation, from [19].

Moreover, convolutional designs implicitly enforce some sort of regularization to the network:

- The kernel uses the same parameters across all the input layers, therefore applying **param-**

eter sharing. This reduces significantly the number of parameters in the network, thus reducing its complexity.

- For the same reason, a technique called **parameter tying** is applied. In the training phase, the parameters are changed according to the activation values of all the inputs. This implies that the parameters of the kernel are *tied*, i.e., the same parameter is changed accordingly to its corresponding values across the whole input.

Often, convolutional layers are paired with **pooling** layers. These layers output a summary statistic of the nearby outputs, most commonly the average or the maximum value, further reducing the number of parameters used in the network, as Figure 2.3 shows.

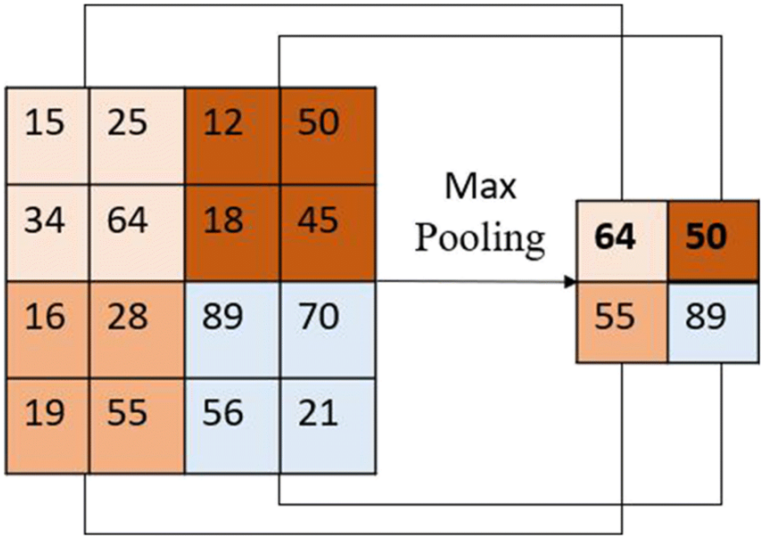


Figure 2.3: An example of the MaxPooling operation, from [20].

2.1.2 Hyperparameter tuning

When it comes to developing AI models, hyperparameter tuning involves optimizing the settings of the chosen model to achieve the best performance.
A common approach, named Grid Search, consists in defining a candidate space $\mathcal{H}_i$ for each i-th hyperparameter $h_i \in \mathcal{H}$ that needs to be optimized, and then training a model on each combination of $\rho_j \in \mathcal{H}_1, \times \mathcal{H}_2, \ldots, \mathcal{H}_N$. Then, the model that minimizes the target function, usually the loss function, is chosen.

This approach, while being straightforward to implement, is computationally demanding, as it requires that the training on each ρ_j is executed before ending the execution.
One possible optimization is the Random Search approach: instead of training the candidate models by selecting hyperparameters by a fixed order, candidate hyperparameters are uniformly randomized, and a limit on the number of iterations is set.
In the following section, a more advanced hyperparameter tuning method will be analyzed, and used in this study.

Bayesian Optimization

Bayesian Optimization is a class of machine-learning-based optimization methods focused on finding the global maximum of a function [21].
This algorithm consists of two main components:

- a Bayesian statistical model for modeling the objective function,
- and an acquisition function for deciding where to sample next.

The pseudo-code is reported in Algorithm 2.1, as defined by Frazier [21].
The objective function is initially evaluated on a collection of points chosen uniformly at random, that are used to update the posterior probability at the next iteration.

Algorithm 2.1 Pseudocode

Place a Gaussian process prior to f.
Observe f at n_0 points according to an initial space-filling experimental design. Set $n = n_0$.
while $n \leq N$
  Update the posterior probability distribution on f using all available data.
  Let x_n be a maximizer of the acquisition function over x, where the acquisition function is computed using the current posterior distribution.
  Observe $y_n = f(x_n)$.
  Increment n.
end while
return The point evaluated with the largest $f(x)$, or the point with the largest posterior mean.

2.2 Audio signals representations and analysis

Before diving into how Deep Fakes are generated and then detected, the audio signals and their representations will be introduced.

In general, from a mathematical point of view, audio signals are represented as sinusoidal basis waves, whose base function is formalized as shown in Eq. 2.8[22].

$$x(t) = A \cdot e^{j \cdot \frac{2\pi}{F} t} \tag{2.8}$$

In Eq. 2.8, the value of the wave x is a continuous function of time, where:

- A is the amplitude of the wave.
- F is the frequency of the wave.
- j is the imaginary constant.

Sampling When dealing with continuous analogical audio signals, it's impossible to memorize all the values, so usually the most common method is to perform uniform sampling, which consists of sampling the value of the signal $x(t)$ at equidistant time intervals, obtaining a series of values x_n[22].

Quantization As often happens, the number of bits that can be dedicated to save each value is finite, while the sampled value is a real number. To solve this issue, the real-valued samples are mapped into finite values in a process called quantization, and in particular, uniform quantization[22]. Uniform quantization truncates values that fall outside a given interval $[-\alpha, \alpha]$, and rescales them so that they fit inside a β bit number. The real interval $[-\alpha, \alpha]$ is therefore divided in $2^{\beta-1}$ bins, each one representing a discrete value, of size $\Delta = \frac{\alpha}{2^\beta}$.
The quantized values are obtained by using Eq. 2.9.

$$Q(x_n) = \Delta \lfloor \frac{x_n}{\Delta} + 0.5 \rfloor \tag{2.9}$$

2.2.1 Fourier Transforms

Once the quantized signal is obtained, the next step is usually to extract its properties. This is often performed by extracting its frequency components $X(\omega)$, called a spectrum, by performing a Fourier Transform, that returns the original signal as the sum of N sinusoids at different frequencies $n\omega$, as defined in Eq. 2.10[22].
It's important to underline that this transform is invertible, and therefore can be used to reconstruct a signal given a set of sinusoidal waves.

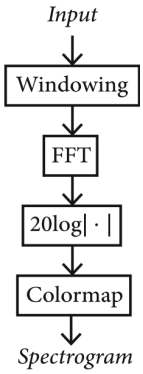


Figure 2.4: Process of computing the spectrogram of a signal, adapted from [22]

This transform can also be used to deal with discrete signal, giving the Discrete Fourier Transform (DFT), as defined in Eq. 2.11. The DFT is computationally expensive, so it's often implemented with a divide-and-conquer approach, taking the name of Fast Fourier Transform (FFT), that has a $O(N \log_2 N)$ complexity.

$$X(\omega_k) = \sum_{n=0}^{N-1} x_n \cdot e^{-j\omega_k n} \text{ where } \omega_k = \frac{2\pi k}{N} \text{ for } k = 0, \ldots, N-1 \tag{2.10}$$

$$X(\omega_k) = \sum_{n=0}^{N-1} x_n \cdot e^{-j\omega_k n}, \text{ where } \omega_k = \frac{2\pi k}{N}, \text{ for } k = 0, \ldots, N-1 \tag{2.11}$$

2.2.2 Spectrograms

Spectrograms are 2D representations of audio signals, that are often used to visualize them[22]. They represent the time dimension on the x-axis and the frequency spectrum on the y-axis; the power of each frequency in each time step is rendered with a color coding. Figure 2.4 pictures the process of computing the spectrogram of a signal, which can be resumed as follows:

1. The signal is divided into overlapping windows (windowing);
2. The Fast Fourier Transform is applied to the windows;
3. The values of the power of each frequency in decibels (dB) are computed ($20 \log_{10} | \cdot |$);

A variation of spectrograms, called Mel-spectrogram, uses the Mel power scale, that transforms the frequencies on a logarithmic scale. This spectrogram is often used in speaker recognition

systems[23], because it better resembles the human perception of sound frequencies. The transformation from the frequency expressed in Hz (f) to the one expressed in Mel (m) is described in Eq. 2.12[24].

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.12}$$

2.2.3 Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients (MFCC) are low-level spectral features that are often used in audio recognition features due to their capability of modeling the pitch and the frequency content of audio signals[25].
The MFCC are computed by applying a triangular band pass filter bank to the FFT power coefficients, as described in Eq. 2.13, where S_k is the output of the filter bank, that usually consists of 12 filters, and N is the total number of samples[25].

$$C_n = \sqrt{\frac{2}{k}} \sum_{k=1}^{K} (\log S_k) \cos n \frac{(k - 0.5)\pi}{k}, n = 1, 2, \ldots, N \tag{2.13}$$

MFCCs proved to be effective features when it comes to both speaker recognition and synthetic speech detection tasks.
For the first task, Sahidullah et al.[23] used a block transformation on MFCCs, that proved to be efficient and robust. For the latter, Mari et al.[26] used MFCCs to extract the first digit statistic (also known as Benford's Law), which was later used to detect synthetic speech tracks.

2.3 Deep Fakes, Audio Deep Fakes & Audio Deep Fake Detection

2.3.1 Definition of Deep Fakes

A Deep Fake (DF) is, in general, multimedia content generated by generative AI models, such as Generative Adversarial Network (GAN), whose structure will be analyzed later, or Autoencoders. Zahra et al. [27] identify 4 macro-categories of DFs, based on the kind of multimedia that is manipulated:

- Audio Deep Fakes;

- Video Deep Fakes;
- Image Deep Fakes;
- Text Deep Fakes.

All these categories aim to manipulate or generate from scratch some kind of content, which should resemble a genuine one.
The number of fields of application of this technology has been increasing in the last few years. Some examples are reported in the following:

- **Dataset Augmentation**: GANs can be used to increase the number of samples in a dataset, by generating synthetic data when the collection of samples is difficult. This approach is described by Goodfellow et al. in the original GANs paper [28].
- **Image Augmentation**: GANs can be used to increase the quality of images, aiding tasks such as the analysis of Magnetic Resonance Imaging (MRI) or Computer Tomography (CT) [29].
- **Text-To-Image generation**: Another application of generative models is the generation of images from a textual prompt, as proposed by Scott Reed, et al. in their 2016 paper [30].

As one might have already inferred, the uses of generative AI systems are not limited to good-faith applications, as a matter of fact, just as soon as the aforementioned examples were developed, someone applied these technologies with malicious intent.
An example of these malicious uses is the creation of Deep Fake pornography videos, that are non-consensually spread on the Internet. In 2019, the DeepNude application was released, which was able to remove clothing from images of women by using GAN models[31]. This application was later removed, although both free and paid versions were made available[32]. The dangerous potential of this kind of application was shown in the case of Rana Ayyub, an Indian investigative journalist, targeted with an online hate campaign, where AI-generated pornographic videos of her were used[33].

2.3.2 Audio Deep Fakes

The definition of Audio Deep Fakes is self-explanatory, so in this section, their classification and generation techniques will be explored.
The techniques to generate Audio Deep Fakes can be divided into two main categories:

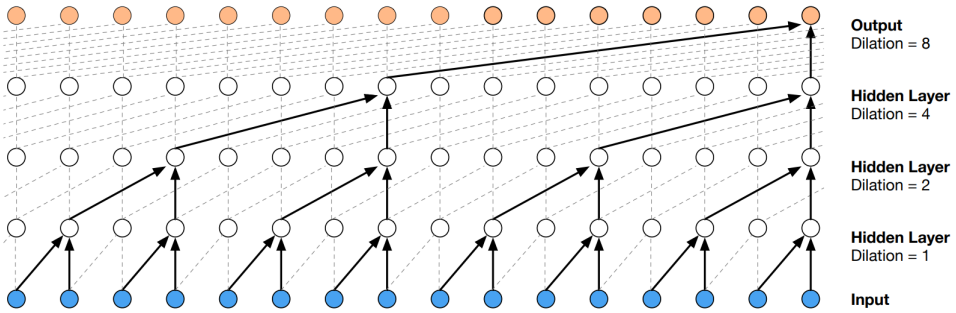


Figure 2.5: Dilated Causal Convolution, as pictured in the original paper by Van den Oord et al. [34]

- Text-To-Speech (TTS) techniques;
- Voice Conversion (VC) techniques.

Text-To-Speech techniques

The first category comprises all the techniques that convert a textual input into an audio track. These techniques are mostly used to read text without the help of a human reader, but can also be used in AI assistants. The following paragraphs will provide an overview of the most relevant TTS techniques.

WaveNet An example of a TTS technique is the WaveNet architecture, an autoregressive Deep Neural Network (DNN) that is able to generate raw audio signals [34]. The audio samples are modeled as a series of random variables $x = \{x_1, x_2, \ldots, x_n\}$, factorized as a product of conditional probabilities, each sample x_t is conditioned on the previous one x_{t-1}, as shown in Equation 2.14.

$$p(x) = \Pi_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1}) \tag{2.14}$$

The model that approximates this distribution is a deep Convolutional Neural Network, that employs dilated causal convolutional layers to extract the information from the inputs while respecting their order (Figure 2.5). The dilation of the convolutional filters allows the model to have a bigger receptive field therefore increasing the model's capacity. The advantage of this model is its computational efficiency: convolutional architectures require fewer parameters to train, and most importantly, the parameters of the same layer are shared. Recurrent architectures are typically more expensive to train, as they require "unrolling" the network along each time step, making it potentially time-consuming for long sequences [35].

Char2Wav Another example of an end-to-end TTS model is Char2Wav[36]. This speech synthesizer uses two components:

- A **reader**, which is an encoder-decoder model which employs Recurrent Neural Network (RNN). This component encodes the textual input letter-by-letter with a bidirectional RNN into a latent space and decodes it with an RNN that employs an attention mechanism. The output of this component is a sequence of vocoder features, which will be used by the second component to generate the audio signal.
- The **Neural Vocoder** uses the vocoder features generated by the reader to output the audio samples. This component is implemented as a SampleRNN, as defined by Mehri et al.[37].

The two components are first trained separately, and then the model is tuned as a whole.

Voice Conversion techniques

Voice Conversion techniques aim to convert the speech signal of the speaker into another speech signal, designed to sound like a second speaker.
These techniques are ideal to conduct impersonation attacks, as they may be used to bypass automated speech verification systems.
In the following paragraphs, some VC techniques will be analyzed. In particular, the works of Kain et al. [38], based on the Gaussian Mixture Model (GMM), and Yang et al.[39], based on a GAN.

Voice Conversion based on Gaussian Mixture Model This technique consists of mapping the extracted spectral features of the original speaker (source speaker) to the ones of the target speaker, by employing a Gaussian Mixture Model.
GMM is a soft-clustering algorithm, that approximates the distribution of a dataset of instances z as the sum of Q multivariate Gaussian distributions, as shown in Eq. 2.15.

$$p(z) = \sum_{i=1}^{Q} \alpha_i N(z; \mu_i, \Sigma_i), \sum_{i=1}^{Q} \alpha_i = 1, \forall i : \alpha_i \geq 0 \tag{2.15}$$

This method uses a GMM model to approximate the joint probability distribution $p(z) = p(x, y)$, where x is the feature vector of the source speaker, and y is the feature vector of the target speaker. The parameters of the model are used to find a linear conversion function $F(x) = \mathbb{E}[y|x]$ that maps the values of x to y, by finding the function that minimizes the MSE between y and $F(x)$.

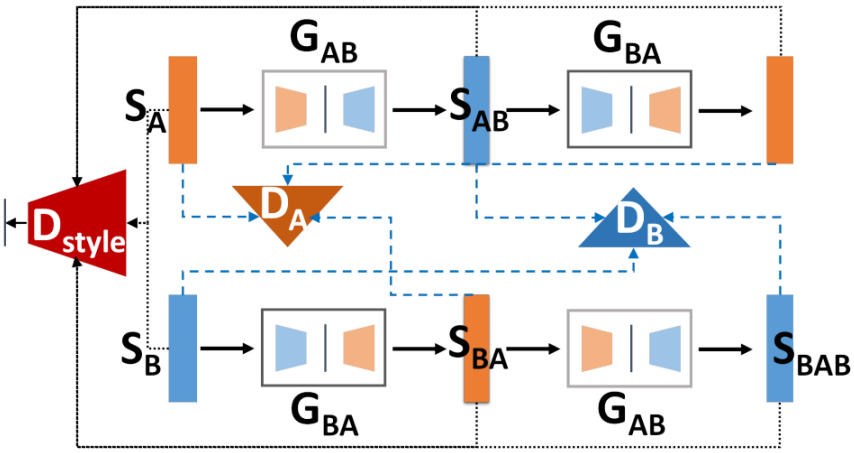


Figure 2.6: The architecture of VoiceGAN, from the original paper [39]

This method was tested on a database of diphone utterance vectors, from which the spectral features were extracted. In particular, the 16^{th} order Line Spectral Features (LSF) was used and converted to the Bark scale. This frequency scale transforms the spectral frequencies expressed in Hertz in a new scale in which equal distances correspond with perceptually equal distances, similar to the Mel scale. The authors chose such a configuration since it allows to weight of the MSE according to the human sensitivity of each frequency.

VoiceGAN This method consists of a modified version of a GAN, that operates on the spectrograms of the audio signals. The distinctive feature of this model is the use of Deep Convolutional Autoencoders, which are used to learn the transformation of the spectrograms from the source speaker to the target one, and vice versa. Figure 2.6 pictures the structure of VoiceGAN.

2.3.3 Taxonomy of the Audio Deep Fake Detection task

The previous section briefly introduced some techniques for fake audio detection. Here the topic will be presented in a more detailed manner.
Similarly to how Yi et al.[40] presented, the detection of Audio Deep Fakes can be grouped into three main families, based on the kind of features used for the classification:

- Spectral features.
- Prosodic features.
- Deep features.

Detection based on Spectral Features

The term "spectral features" refers to the use of the frequency spectrum of the audio track to perform the classification task. A further distinction can be made, between Short-Term and Long-Term features: the former ones use short time frames, often overlapping, which can capture anomalous artifacts; the latter can capture long-time correlations.
Mari et al. [26] used both of these features, together with bicoherence features, to develop a fusion model, which used encoding of all the aforementioned features as input of a fully connected Neural Network. This model also proved to be noticeably resilient to anti-forensic attacks, namely, MP3 compression and noise injection.

Detection based on Prosodic Features

Prosody refers to non-segmental aspects of speech, including for instance syllable stress, intonation patterns, speaking rate, and rhythm[41]. The fundamental frequency (F_0), also referred to as "pitch", is often used in speech recognition tasks, often combined with spectral features for increased prediction robustness[41].
In his work, Shreiber et al. [42] use prosodic features extracted from the estimated syllables to generate N-grams, which will be counted and used as a feature space that is classified with a Support Vector Machine (SVM). Duration of the syllable, pitch, and energy were the primary features used in this study, which allowed the model to perform comparably with the current state of the art of the time.

Detection based on Deep Features

Just like with other tasks, Deep Learning methods can offer an alternative to the two previous groups of features. It seems like this last family of features can be used to fill the gaps that the other ones cannot.
An example of this approach is the work of Chakravarty et al.[43], which uses a pre-trained CNN, a ResNet50 model, as a feature extractor. The input of the network is the Mel-spectrogram, whose definition will be provided later. The output is a tensor, which undergoes a Linear Discriminant Analysis (LDA) process, which reduces its dimensionality. This result is then provided to Machine Learning models which are used for the final DF recognition task.

2.4 Perceptual Audio Experiments

Muller et al. [44] experimented on the perception of audio Deep Fakes by human beings.
The test was conducted on a web application, designed as an online game in which the other opponent was an AI model. The task of the challenge at hand was listening to an audio file, sampled from the ASVspoof 2019 dataset and provided by the application, and deciding if the track was a Deep Fake or an authentic one. The decision of the AI model and the true label were not provided until the user made its own decision.
The results showed that the performances of the model were not particularly better than the humans' ones.
The most relevant limitation of this experiment is the lack of control over the execution of the test, in particular, the application didn't monitor the environment of the test and the equipment was employed by the user. For this reason, a similar experiment was conducted in an attempt to verify the claims of Muller et al. The details of that experiment can be found in Chapter 5.

3

Implicit Neural Representations

3.1 Implicit Neural Representations

3.1.1 Definition of Implicit Neural Representations

Implicit Neural Representation (INR) is a novel field of study that aims to represent discrete signals implicitly in the weights of a neural network that then models a continuous function[45]. This can be achieved by training a Neural Network (NN) on a single signal, on a coordinates-to-value task. The result is that the model learns an inner representation of the signal as the parameters of a continuous function: it's possible to sample from that function with arbitrary precision, with the only limitation being the overall complexity in the reconstruction of the content [45].
Recently, different architectures are being tested with interesting results[46], such as 3D shape generation [45], data compression [47] and most importantly 3D shape representations [48].
The following section will review the basic idea of INRs, and build up to the most innovative applications of this technology.

3.1.2 Intuition & Applications

Implicit Neural Representations are based on the concept of implicit representation of a function. In general, implicit functions are functions that represent a signal employing implicit equations, of the form $F(x_1, \ldots, x_n) = 0$. This is an elegant way to represent complex surfaces by composing

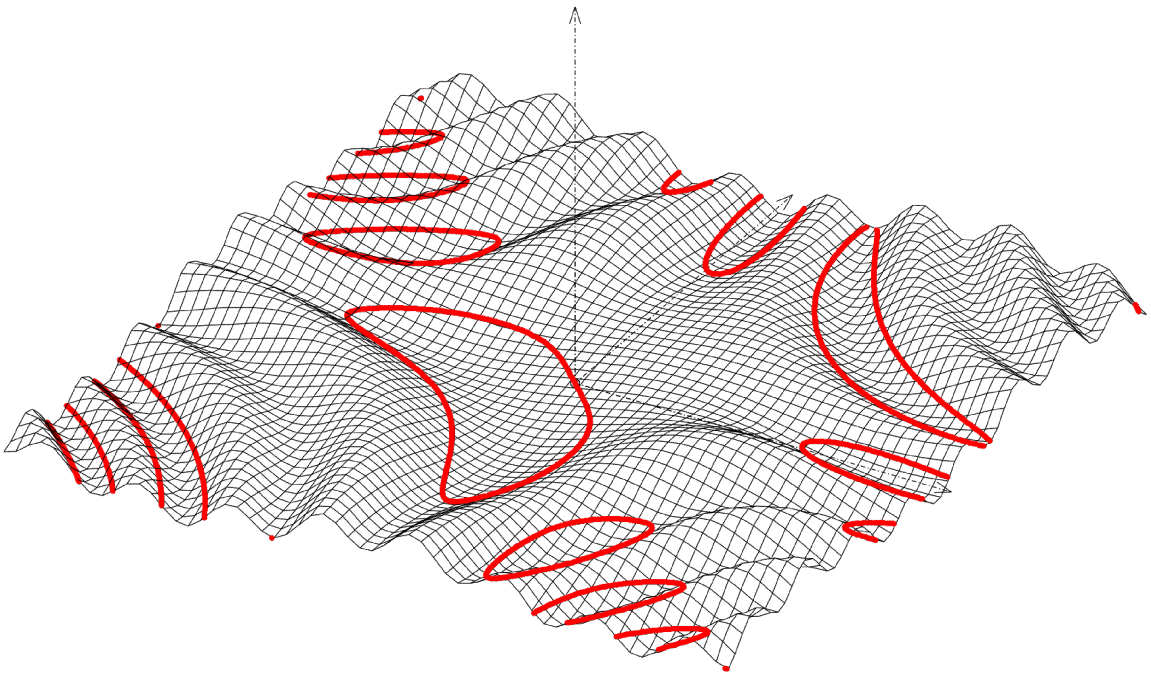


Figure 3.1: Visual representation of the implicit function $sin(x+y) - cos(xy) + 1 = 0$, as a level curve. Adapted from of [50].

a few elementary functions, such as shown in Figure 3.1.
INRs extend this concept by using Neural Networks as the implicit functions, whose goal is to parameterize the signal with arbitrary precision while being memory-efficient[49].

Neural networks are universal function approximations, making them the ideal tool to address this task. Neural implicit modeling is achieved by training the model on data, sampled from the same distribution that we want to approximate. The result is a set of organized weights, that all together contribute to the computation of the output values. These weights are the parameters that constitute the representation of the signal itself.

3.1.3 Use Cases

This section will present two use cases of Implicit Neural Representations.

NeRF Neural Radiance Field (NeRF) is a technique that uses INRs to synthesize novel views of complex scenes, by optimizing a radiance field in a 5D space, made of 3 spatial dimensions and 2 angular parameters that represent the viewing direction[51]. Figure 3.2 pictures a sample from

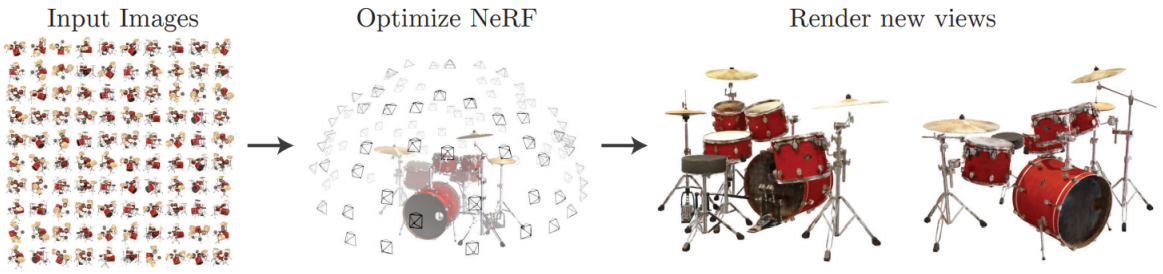


Figure 3.2: Visual representation of two novel renderings of an image learned using a NeRF INR. Adapted from [51].

the original paper by Mildenhall et al..
The INR in this case is a deep, fully connected, multi-layer perceptron, whose learning task is to convert the aforementioned 5D input, encoded with positional encoding, into a tuple, made of a 3-dimensional vector which represents an Red-Green-Blue (RGB) values of a given pixel, and a volume density value.
The INR, in this case, is just part of a bigger processing pipeline (Figure 3.3), which starts with a set of images and ends with the novel renderings. The overall procedure can be summarized as follows:

a) The first step consists of sampling the set of 5D points from the set of images.

b) The INR model is used to generate a representation of the scene.

c) The values generated in the previous step are used to generate the images of novel views, using volume rendering techniques.

d) The INR model is trained by computing the rendering loss between the generated scene and the ground truth images; since this function is differentiable, the INR can be trained by using common Gradient Descent approaches.

This method outperforms other state-of-the-art works and allows to generate fine-detailed renderings of small objects[51]. On the other hand, NeRFs are computationally demanding, as the training for a single scene can take to 2 days on a single NVIDIA V100 GPU[51].

HyperSound Similarly to the previous use case, Hypersound is an INR model that aims at approximating sound waves by means of a neural network[52].
This approach employs an Hypernetwork architecture[52], which is composed as follows:

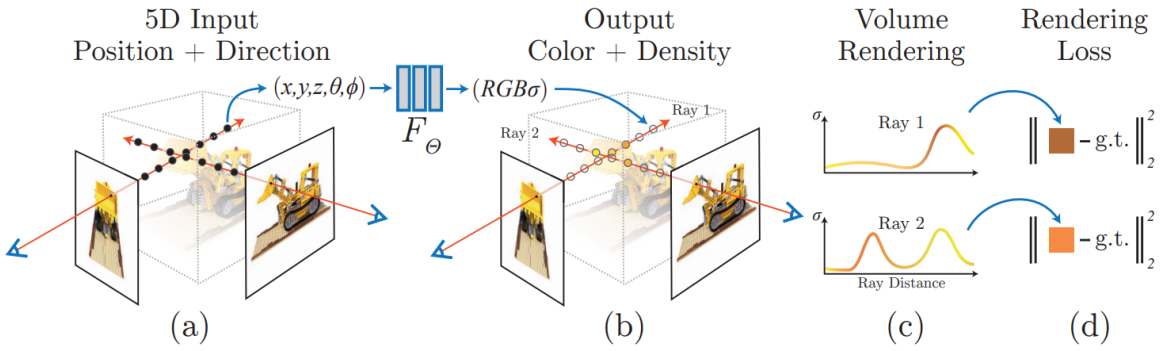


Figure 3.3: Pipeline of the generation and training of a NeRF model. Adapted from [51].

- A convolutional encoder, that maps the input into a lower dimensional space, acting as a feature extractor.
- The second component is a fully connected network with ELU activation, that outputs a flattened vector of the weights of the target neural network.
- This last step generates the weights for the INR model, which is essentially a one-input-one-output multi-layer perceptron made of 4 fully connected layers, each one made of 256 ReLU activated neurons, whose task is to return the amplitude value of the audio signal, given its coordinates.

The training of the hypernetwork employs a customized loss function that balances the reconstruction error in both the time and frequency domains (Equation 3.1).
The results of this work are comparable to the state-of-the-art, although the authors note the presence of noise and "robotic artifacts" that might compromise the perceptual scores[52].

$$L(x,\hat{x}) = \lambda_{SL1} \cdot L_{SL1}(x,\hat{x}) + \lambda_{STFT} \cdot L_{STFT}(x,\hat{x}) \tag{3.1}$$

3.1: Loss function of the Hypersound model, where: x is the original soundwave vector, $\hat{x}$ is the reconstructed soundwave, L_{SL1} is the smoothed L1 loss (penalizes the reconstruction error in the time domain), L_{STFT} is the Short-Time Fourier Transform loss (penalizes the reconstruction error in the frequency domain), and $\lambda_{SL1}, \lambda_{STFT}$ are the balancing parameters.

4

INR-based Deep Fake Detection

In this chapter the model that was used to classify Deep Fake tracks from genuine ones will be described in detail.

4.1 Experimental Setup

This phase of the study was conducted with a remote machine inside the LabLTTM of the University of Padua. Table 4.1 resumes the machine's specifics.

Table 4.1: Resuming table of the machine's specifics.

	Specifics		
OS	Ubuntu		
CPU	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz		
Cache Memory	L1d	128 KiB	4 instances
	L1i	128 KiB	4 instances
	L2	1 MiB	4 instances
	L3	8 MiB	1 instance
GPU	NVIDIA GeForce GTX 1070		1920 CUDA cores
GPU Memory	8192 MiB		
RAM storage	30 GiB		

4.2 Datasets

For the purposes of this study, the ASVspoof2019[53] and ASVspoof2021[54] datasets were used. These datasets were conceived by the ASVspoof initiative, which periodically issues challenges in the field of Automatic Speaker Verification (ASV) technologies. The goal of this initiative is to stimulate the development of countermeasures to spoofing attacks against ASV systems.
ASV systems are vulnerable to 4 main means of attacks[54]:

- Impersonation;
- Replay;
- Voice Conversion (VC);
- Text-To-Speech (TTS).

These last two received the most attention, as numerous open-source toolkits allow to conduct spoofing attacks[7].
The files of the audio samples were distributed as .flac files, sampled at 16 kHz and quantized at 16 bits [54].

4.2.1 ASVspoof 2019

This dataset contains bonafide and spoofed speech samples, which include synthetic speech and converted voice signals generated with the very latest, state-of-the-art technologies[53].
Two spoofing scenarios are considered:

- Logical Access (LA) implies a scenario where a remote user seeks access to a system or service protected by ASV.
- Physical Access (PA) implies the use of ASV to protect access to a sensitive or secure physical space or facility. In this scenario, the microphone is controlled by the authentication system designer, not by the user.

Considering that only the LA subset was used, the description of the algorithms will only focus on that part.

LA subset

The spoofed speech samples are generated from 19 algorithms, both of type Text-To-Speech and Voice Conversion.
Algorithms $A01 - A06$ are used in the training and evaluation dataset, while the remaining $A07 - A19$ compose the development dataset. With few exceptions, all the algorithms use methods based on Neural Networks to spoof the victim's speech. Table 4.2 summarizes the main characteristics of the LA subset.
The bonafide samples of the ASVspoof 2019 database are extracted from the Voice Cloning Toolkit (VCTK) corpus [55], a multi-speaker English speech database. The samples were recorded in a controlled environment, and they are utterances from 107 speakers (46 male, 61 female)[53]. The set of 107 speakers is partitioned into three speaker-disjoint sets for training, development, and evaluation. It's important to underline that the samples were recorded with 96 kHz frequency, and were all downsampled to 16 Hz [53].
This subset is partitioned into 3 datasets:

- Training subset, made of 5161 samples, used only for the training;
- Evaluation subset, made of 5089 samples, used to evaluate the performances of the model during the training on samples that weren't experienced before;
- Development subset, made of 14691 samples, used the evaluate the performances of the model at the end of the training, on samples never seen before.

The training and the evaluation partitions are made of bonafide and spoofed samples, in equal parts. The spoofed samples of these partitions are equally divided among the spoofing algorithms from A01 to A06.
Similarly, the development subset is equally divided between bonafide and spoofed samples, but the spoofed samples are equally divided among the algorithms from A07 to A19.

4.2.2 ASVspoof2021

The structure of the 2021 edition of the ASVspoof is similar to the previous one. The challenge is divided into three scenarios:

- Logical Access,
- Physical Access,

Table 4.2: Summary of the characteristics of the LA subset of ASVspoof2019. * indicates that the audio generation pipeline *doesn't* use neural networks.

	Type (TTS/VC)	Dataset partition
A01	TTS	Training/Evaluation
A02	TTS	Training/Evaluation
A03	TTS	Training/Evaluation
A04	TTS*	Training/Evaluation
A05	VC	Training/Evaluation
A06	VC*	Training/Evaluation
A07	TTS	Development
A08	TTS	Development
A09	TTS	Development
A10	TTS	Development
A11	TTS	Development
A12	TTS	Development
A13	VC	Development
A14	VC	Development
A15	VC	Development
A16	TTS*	Development
A17	VC	Development
A18	VC*	Development
A19	VC*	Development

Table 4.3: Summary of the conditions applied to the LA subset of ASVspoof2021.

	Description	Bitrate
C1	Identical to ASVspoof2019	16 kHz
C2	Transmitted through VoIP network	8 kHz
C3	Transmitted through PSTN an VoIP network	8 kHz
C4	Transmitted through VoIP network	16 kHz
C5	Transmitted through VoIP network	8 kHz
C6	Transmitted through VoIP network	8 kHz
C7	Transmitted through VoIP network	16 kHz

- Deep Fake.

Some modifications were introduced because most traces were deemed too clean to be used as a reliable authentication scenario [54].
As only the LA and DF subsets were used in this study, the description of the PA subset won't be covered.

LA subset

For this reason, the LA scenario was divided into 7 conditions ($C1 - C7$), summarized in Table 4.3. The goal of these modifications was to reduce the gap between ideal laboratory conditions and those to be expected in the wild.
This was achieved by transmitting the samples through a voice-over-internet-protocol (VoIP) system and a public switched telephone network (PSTN), at different bitrates and with different codecs. This introduced in the samples several different artifacts that increased the difficulty level of the detection task.

DF subset

This subset was conceived to improve the generalization capabilities of Deep Fake detection systems. It involves speech utterances that are encoded and decoded using various lossy codecs, introducing distortions based on codec type and settings. The DF subset extends the ASVspoof 2019 LA set by adding two additional sources, namely the 2018 and 2020 Voice Conversion Challenge (VCC) databases.

4.3 Problem setup

The classification problem at hand is a binary classification one.

- The input data is the training curve of the first 200 epochs of small neural networks, each one trained only on one track.
- The output data is the set $\{0, 1\}$, where 0 stands for a bonafide sample and 1 for a Deep Fake one.

The main intuition behind this approach is that given that fake samples are generally generated by neural networks when fitting INRs on them the learning behavior might be different from the one obtained with bonafide samples.
A classifier might then be able to see these different patterns in learning behavior and to use them to properly distinguish generated from bonafide samples.

4.4 Pre-processing

As anticipated in the previous section, the INRs were used as a means of feature extraction. The feature at hand for each audio sample is the learning curve of the INR.
In this section, the process that led from the raw audio sample to the learning curve will be reviewed. What follows is a detailed description of the pre-processing steps, through which each audio track went. Figure 4.1 resumes graphically the pre-processing steps.

Extracting the silences The first step of the pre-processing is the extraction of the silenced parts of the audio track. This is performed by filtering the samples inside the track with sound intensity lower than 40 dB.

Implicit Neural Representations The second step consists of extracting the learning curve of the Implicit Neural Representation. Each track is used to train a single Feed Forward Neural Network on a regression task, which consists of predicting the MSE value of a given sample number. To facilitate the learning task, the sample number is encoded with positional encoding (notated with the function f). Table 4.4 resumes the hyperparameters on which the INRs were trained, while Eq. 4.1 defines the positional encoding function.

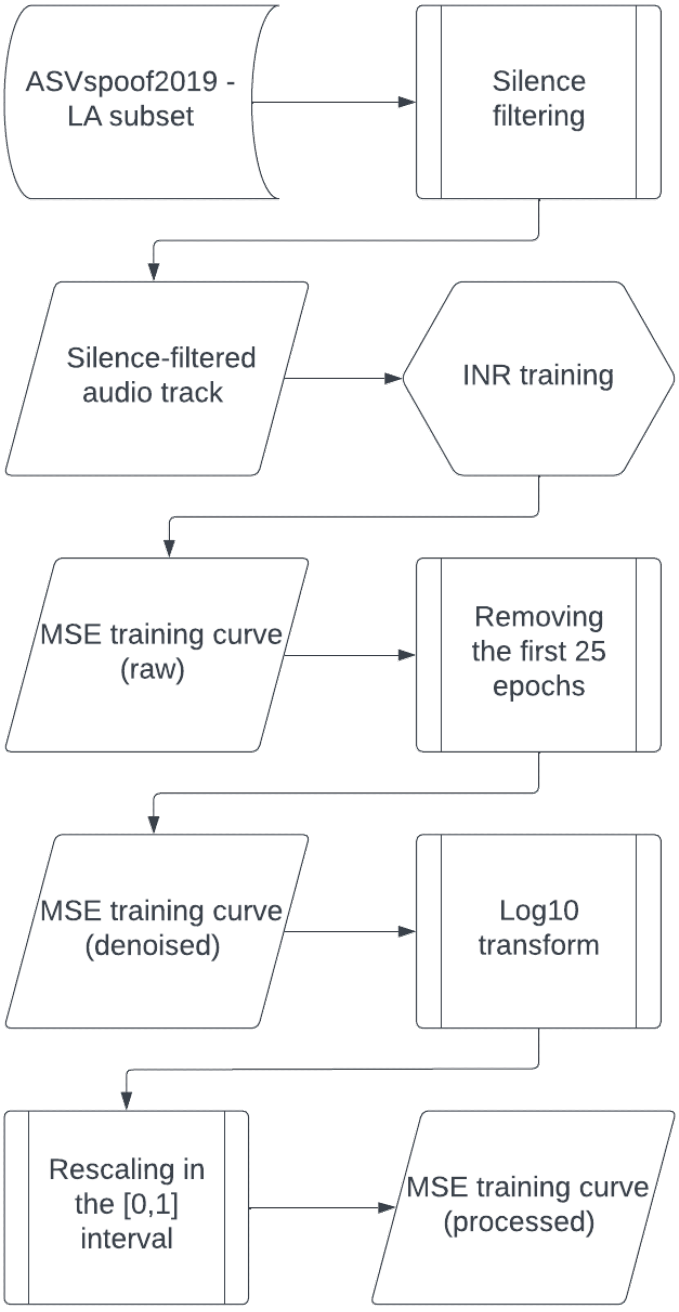


Figure 4.1: Flowchart of the main pre-processing steps, from the raw sample to the final training data for the model.

Table 4.4: Hyperparameters used by the INRs.

Hyperparameter	*Value*
Learning rate	0.001
Number of training epochs	200
Loss function	Mean Squared Error (MSE)

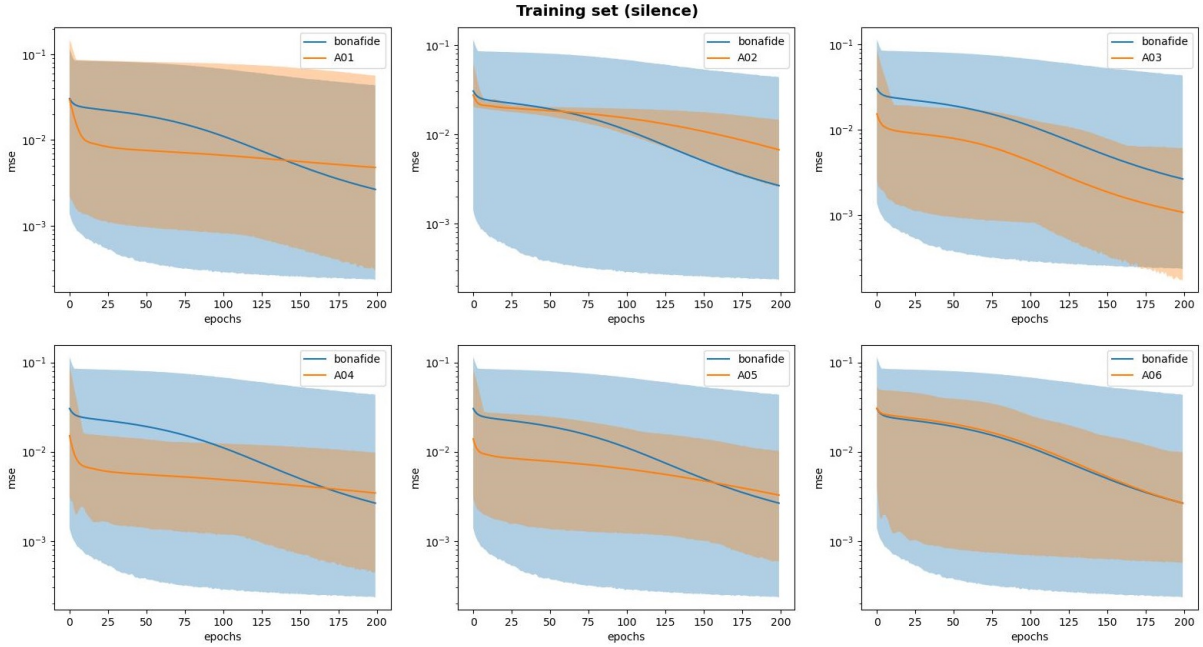


Figure 4.2: The mean MSE training curves, for each of the 6 algorithms of the LA subset of ASVspoof2019 dataset, compared with the one of the bonafide samples.

RESCALING Figure 4.2 shows the comparison of the mean MSE training curves, for each of the 6 algorithms of the LA subset of ASVspoof2019 dataset, compared with one of the bonafide samples. The values of the MSE span over three orders of magnitude, so, for each track, the logarithm function was applied to the values of the MSE, and then they were rescaled to the $[0, 1]$ interval.

CLEANING To reduce the noise caused by the initial part of the INRs' training, the first 25 epochs of each audio sample were removed.

$$f_i(x) = \begin{cases} \sin\ (2^i \cdot \pi \cdot x) & \text{if } i \text{ is even.} \\ \cos\ (2^i \cdot \pi \cdot x) & \text{otherwise.} \end{cases} \tag{4.1}$$

Table 4.5: Training parameters used for the training of the CNN model.

Training setting	*Value*
Learning rate	0.0001
Number of training epochs	50
Loss function	Binary Cross Entropy
Optimizer	Adam
Batch size	128
Early stopping - patience	15 epochs
Early stopping - delta	0

4.5 Model selection

The neural network architecture chosen for the classification task was the Convolutional Neural Network (CNN). The rationale for this decision is that CNNs can detect patterns in data while using fewer parameters than a Feed Forward Neural Network.

4.5.1 Training settings

The model was trained on the training partition of the ASVspoof2019 dataset, in particular, the LA subset.
To introduce some regularization, an Early Stopping callback was implemented, which stops the training procedure when the performance of the model has not improved after a given number of epochs. For this reason, this is often also referred to as the "patience" parameter. The performance measure used for this purpose was the value of the loss function, computed on the validation dataset.
Table 4.5 resumes the training settings.

4.5.2 Hyper parameters' selection

The selection of the hyperparameters was performed by using the Bayesian Optimization algorithm.
Figure 4.3 pictures the structure of the CNN used for this study, while Figure 4.4 summarizes the structure of its layers.

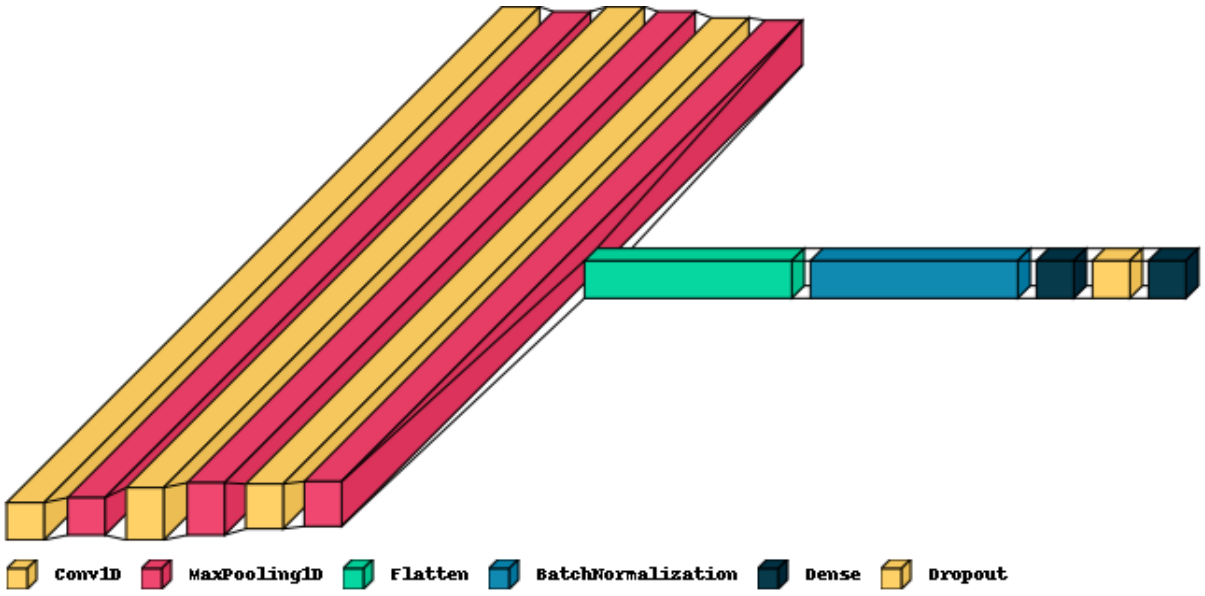


Figure 4.3: Representation of the Convolutional Neural Network model used for the classification task.

4.6 Model evaluation

4.6.1 Performances

In this section, the performances of the model will be evaluated by comparing its confusion matrices to the ones of the model developed by Mari et al. [12].
The performances of the models will be shown at first with a global focus, then they will be compared by algorithm, to evaluate the performances of the models on the single spoofing methods.
The performances of the models were also compared in Table 4.6 (evaluation dataset) and in Table 4.7 (development dataset) by using some common metrics:

- Accuracy, as defined in Eq. 4.2.

$$Accuracy = \frac{TN + TP}{TP + TN + FP + FN} \tag{4.2}$$

- Precision, as defined in Eq. 4.3.

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

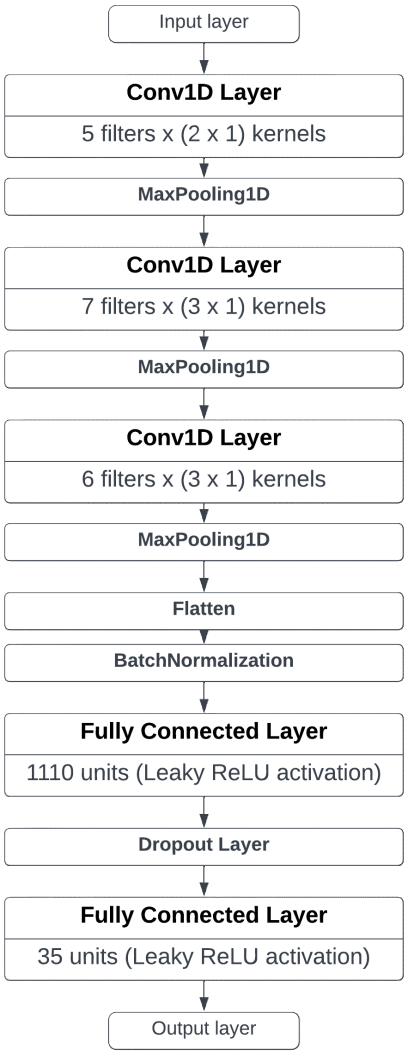


Figure 4.4: Representation of the Convolutional Neural Network model used for the classification task.

- Recall, as defined in Eq. 4.4.

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

- F1-Score, as defined in Eq. 4.5.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.5}$$

In these formulas, the following notation is used:

- *TP* is the number of True positives;
- *FP* is the number of False positives;
- *TN* is the number of True negatives;
- *FN* is the number of False negatives.

Table 4.6: In this table the most common metrics for comparing the models on the evaluation set are reported.

	Accuracy		Precision		Recall		F1 score	
Algorithm	Proposed	SoS	Proposed	SoS	Proposed	SoS	Proposed	SoS
Overall	0.7750	0.9247	0.7913	0.9326	0.7441	0.9148	0.7670	0.9236
bonafide	0.8057	0.9345	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
A01	1.0000	0.9896	1.0000	1.0000	1.0000	0.9896	1.0000	0.9948
A02	0.8303	1.0000	1.0000	1.0000	0.8303	1.0000	0.9073	1.0000
A03	0.6235	1.0000	1.0000	1.0000	0.6235	1.0000	0.7681	1.0000
A04	0.9950	0.6891	1.0000	1.0000	0.9950	0.6891	0.9975	0.8159
A05	0.9272	0.9951	1.0000	1.0000	0.9272	0.9951	0.9622	0.9976
A06	0.1349	0.8313	1.0000	1.0000	0.1349	0.8313	0.2378	0.9079

4.6.2 Comment on the results

The overall results clearly show that the proposed model is sufficiently accurate to be considered close to the state of the art, but at the same time is not preferable to the one presented by Mari et al.. Both the results from the Development dataset (Fig. 4.5) and the Evaluation dataset (Fig. 4.6) show that the performances of the proposed model are inferior.

Table 4.7: In this table the most common metrics for comparing the models on the development set are reported.

Algorithm	Accuracy		Precision		Recall		F1 score	
	Proposed	SoS	Proposed	SoS	Proposed	SoS	Proposed	SoS
Overall	0.7835	0.8533	0.7431	0.9157	0.8035	0.7473	0.7722	0.8230
bonafide	0.7668	0.9422	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
A07	0.9681	0.9585	1.0000	1.0000	0.9681	0.9585	0.9838	0.9788
A08	0.8518	0.9875	1.0000	1.0000	0.8518	0.9875	0.9200	0.9937
A09	0.9648	0.9863	1.0000	1.0000	0.9648	0.9863	0.9821	0.9931
A10	0.9300	0.8867	1.0000	1.0000	0.9300	0.8867	0.9637	0.9399
A11	0.9680	0.7849	1.0000	1.0000	0.9680	0.7849	0.9838	0.8795
A12	0.9602	0.7952	1.0000	1.0000	0.9602	0.7952	0.9797	0.8859
A13	0.9455	1.0000	1.0000	1.0000	0.9455	1.0000	0.9720	1.0000
A14	0.9538	0.9750	1.0000	1.0000	0.9538	0.9750	0.9763	0.9873
A15	0.9591	0.4182	1.0000	1.0000	0.9591	0.4182	0.9791	0.5898
A16	0.9223	0.8958	1.0000	1.0000	0.9223	0.8958	0.9596	0.9451
A17	0.5356	0.1554	1.0000	1.0000	0.5356	0.1554	0.6976	0.2690
A18	0.5496	0.5993	1.0000	1.0000	0.5496	0.5993	0.7094	0.7494
A19	0.2537	0.6581	1.0000	1.0000	0.2537	0.6581	0.4047	0.7938

In the following paragraph, the performances of the model on the two datasets will be analyzed in detail.

Evaluation dataset On the evaluation dataset, the proposed model is outperformed in most of the cases, but on algorithms A01 and A04, which are TTS algorithms, it seems to perform better. In particular, on A04 the proposed model appears to have a significant advantage (99,50% vs. 68,91% on the accuracy, 99,75% vs. 81,59% on the F1-score).

Development dataset On the development dataset, the situation is similar to the one in the evaluation dataset, but the gap in the performances appears to be closer, as Fig. 4.7a shows.
The proposed model seems to perform better on algorithms A07, A10, A11, A12, A15 and A16, but only in a few cases with significant advantage:

- A10, with 93,00% vs. 88,67%,
- A11, with 96,80% vs. 78,49%,

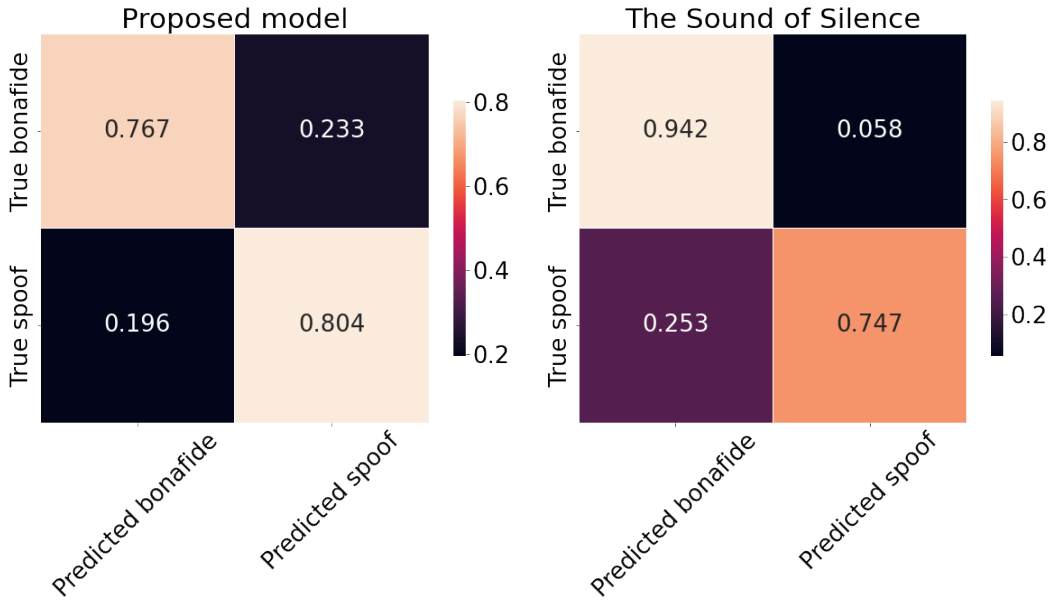


Figure 4.5: Global confusion matrix computed on the development dataset, compared with the one from Mari et al.[12].

- A12, with 96,02% vs. 79,52%,
- A15, with 95,91% vs. 41,82%.

Even though the accuracy of the proposed model on A17 is higher (53,56% vs. 15,54%), the other model still presents an advantage: this is because the task at hand is a binary classification problem, therefore, knowing that the SoS model has such a low accuracy, we could take the opposite of its output and obtain an accuracy of 84,46%.
With the only exception of A15, all the aforementioned algorithms are of the TTS kind.

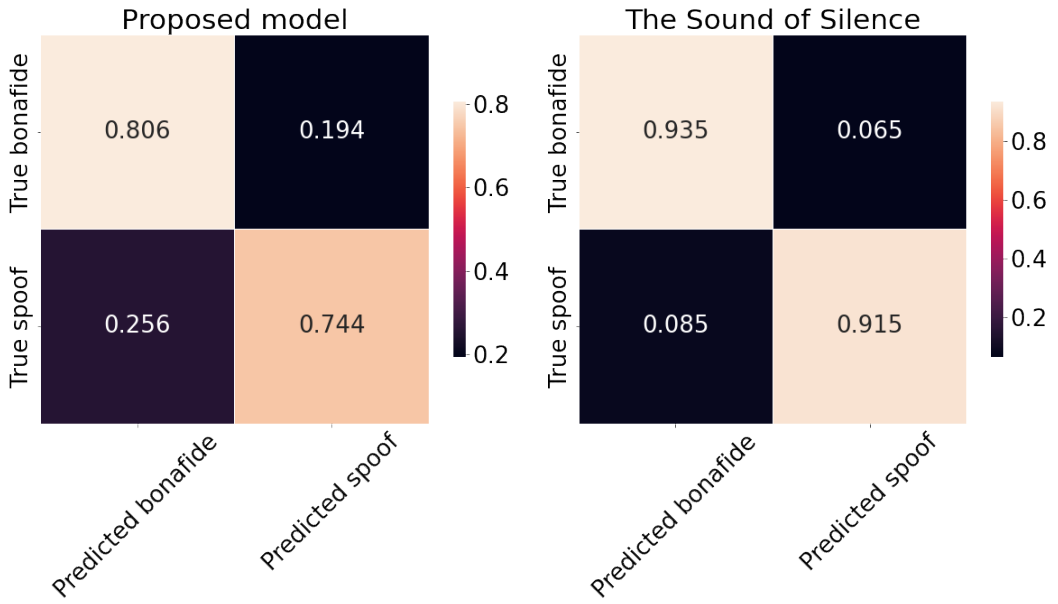
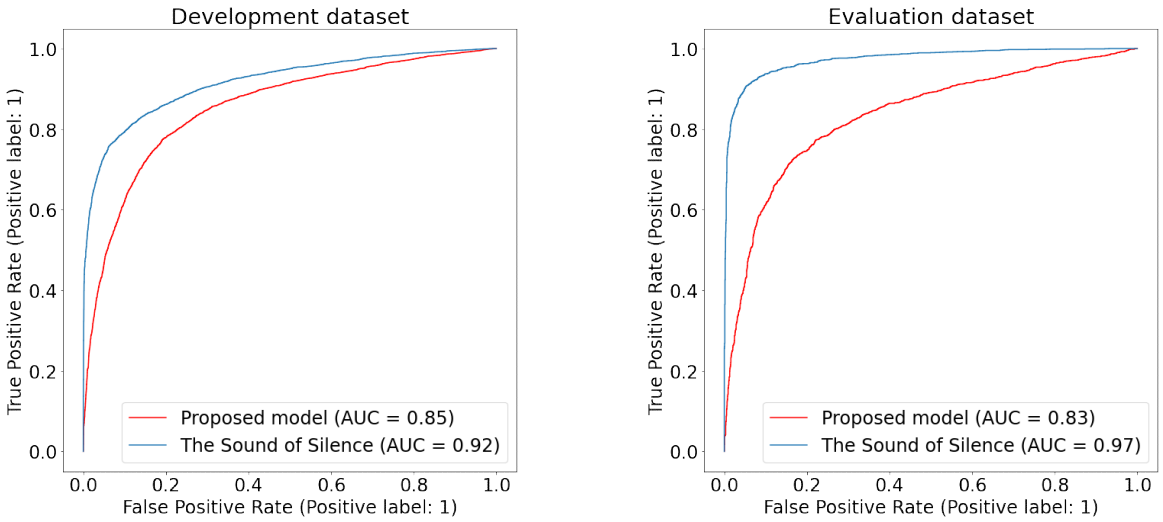


Figure 4.6: Global confusion matrix computed on the evaluation dataset, compared with the one from Mari et al.[12].



(a) Development dataset.

(b) Evaluation dataset.

Figure 4.7: Receiver Operator Characteristic curve of the proposed model, compared with the one from Mari et al.[12].

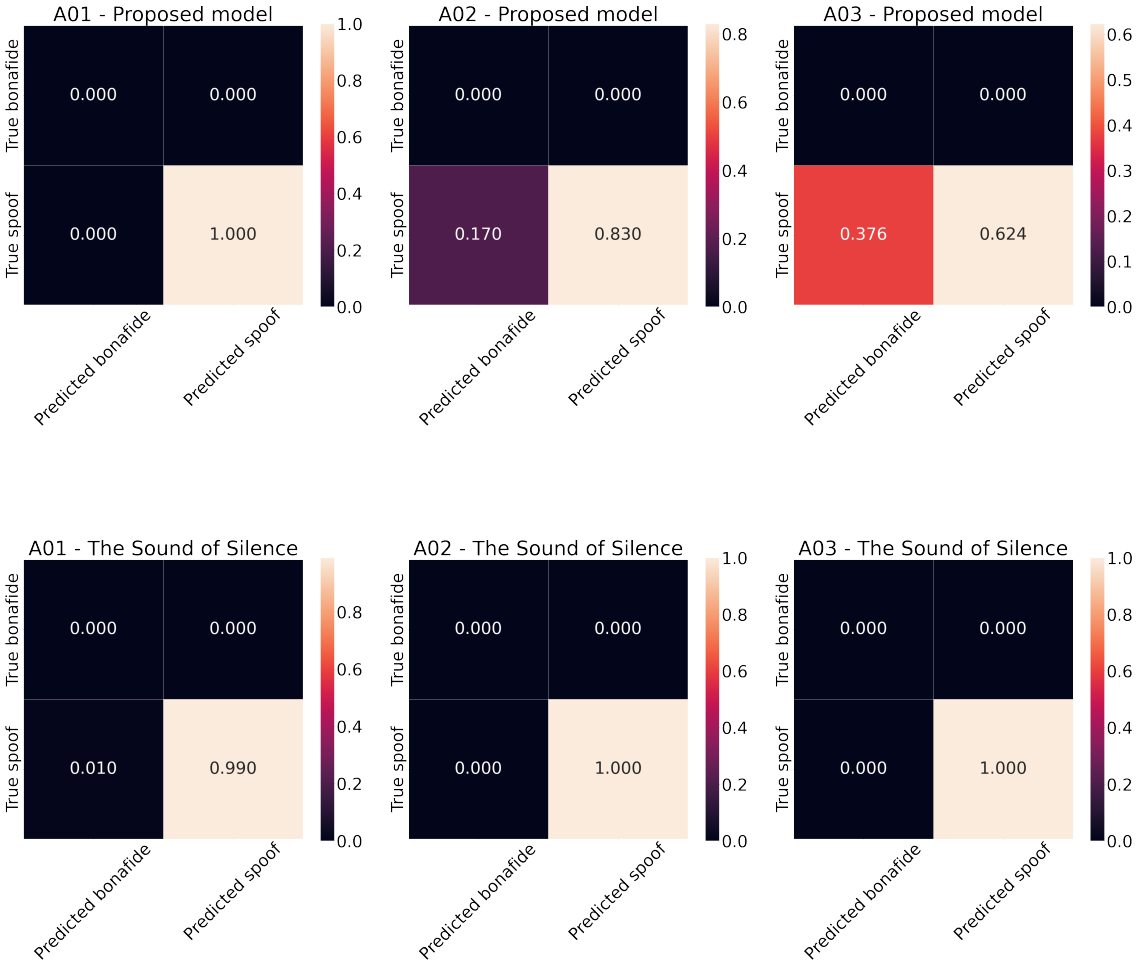


Figure 4.8: Comparison of the heatmaps divided by algorithm, algorithms A01-A03.

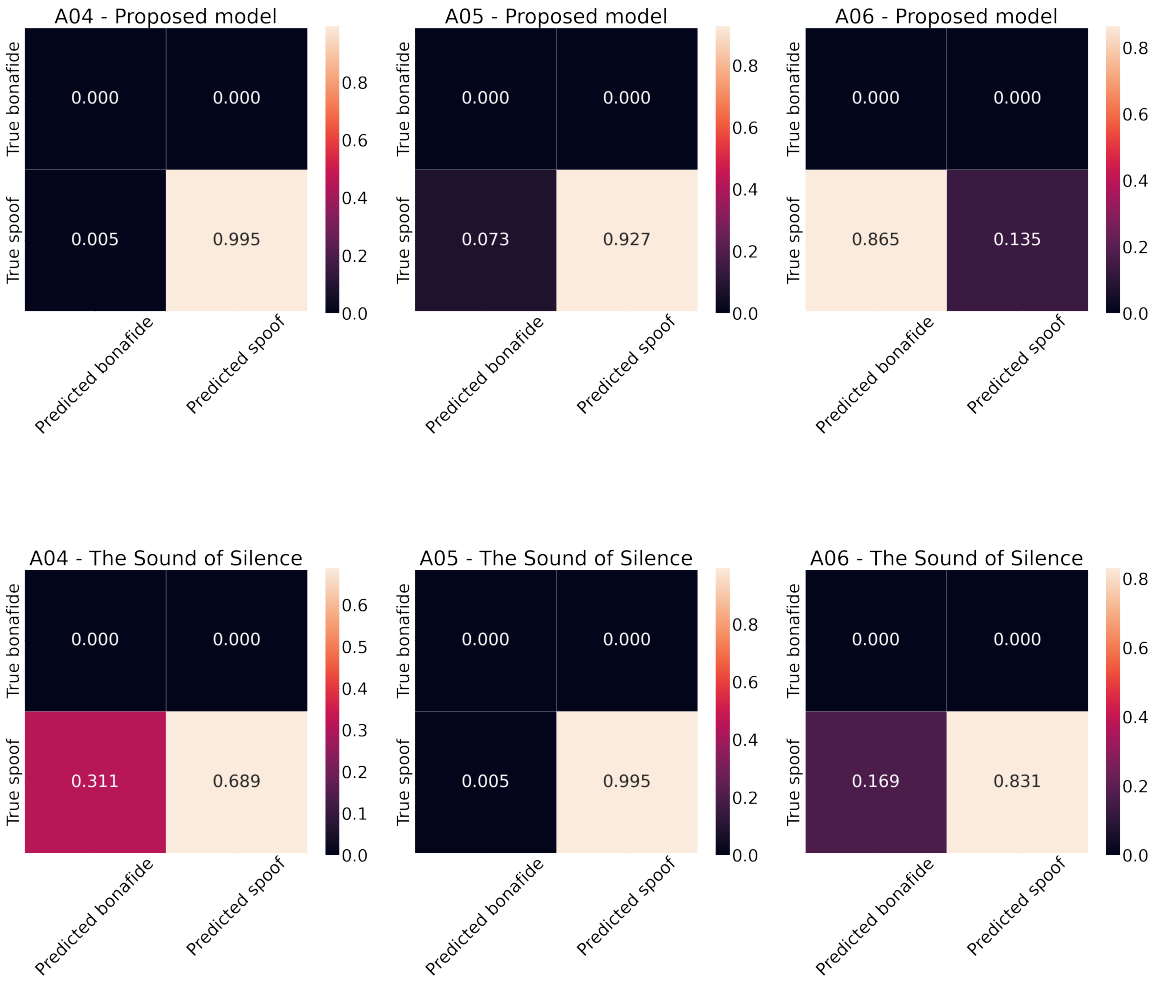


Figure 4.9: Comparison of the heatmaps divided by algorithm, algorithms A04-A06.

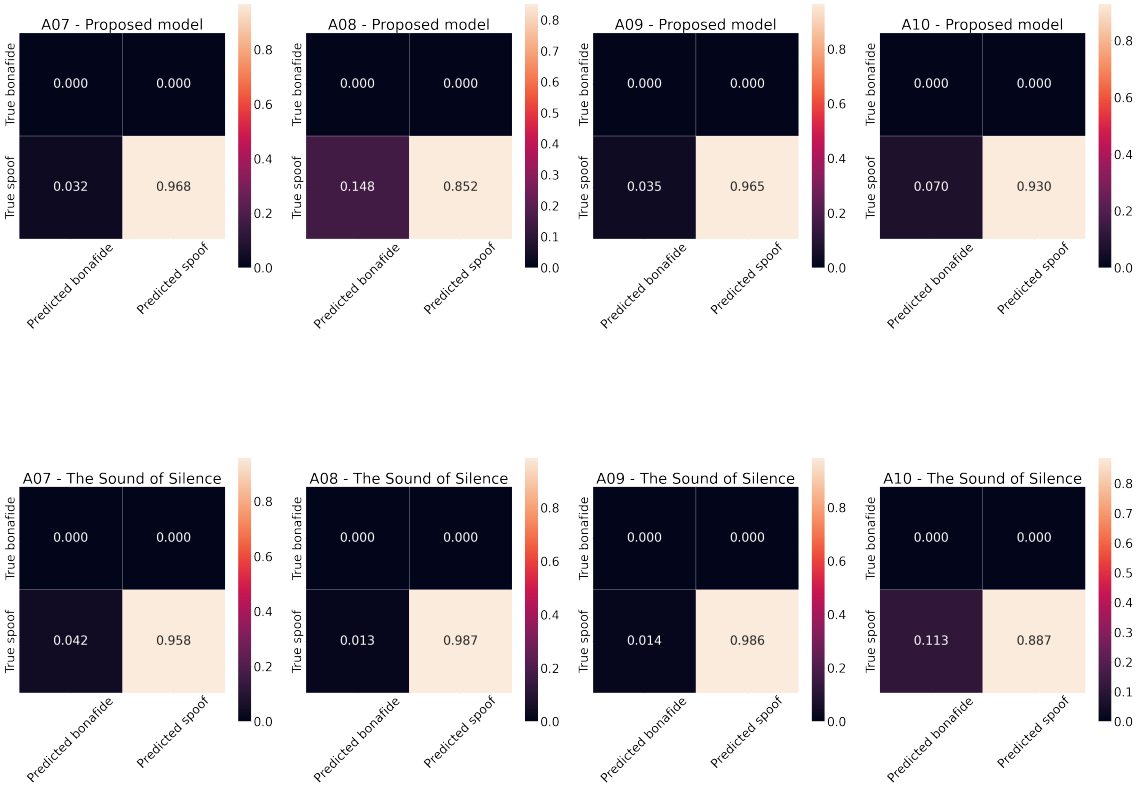


Figure 4.10: Comparison of the heatmaps divided by algorithm, algorithms A07-A10.

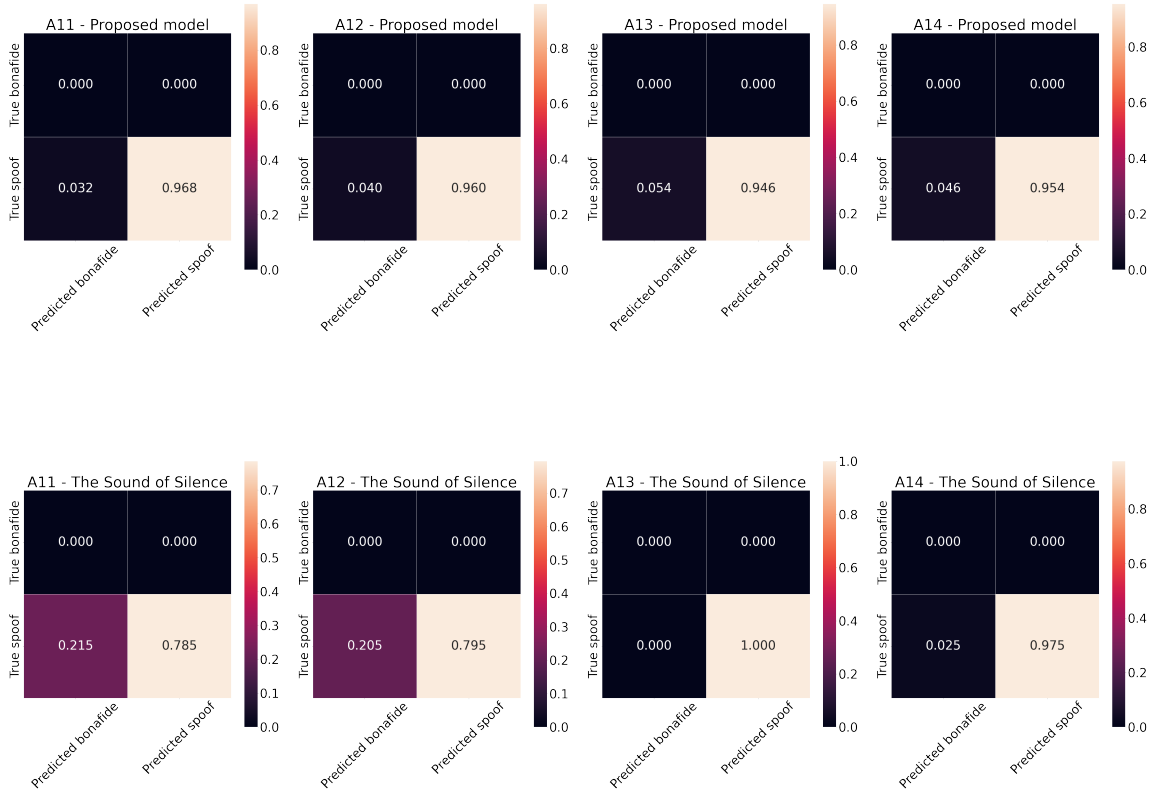


Figure 4.11: Comparison of the heatmaps divided by algorithm, algorithms A11-A14.

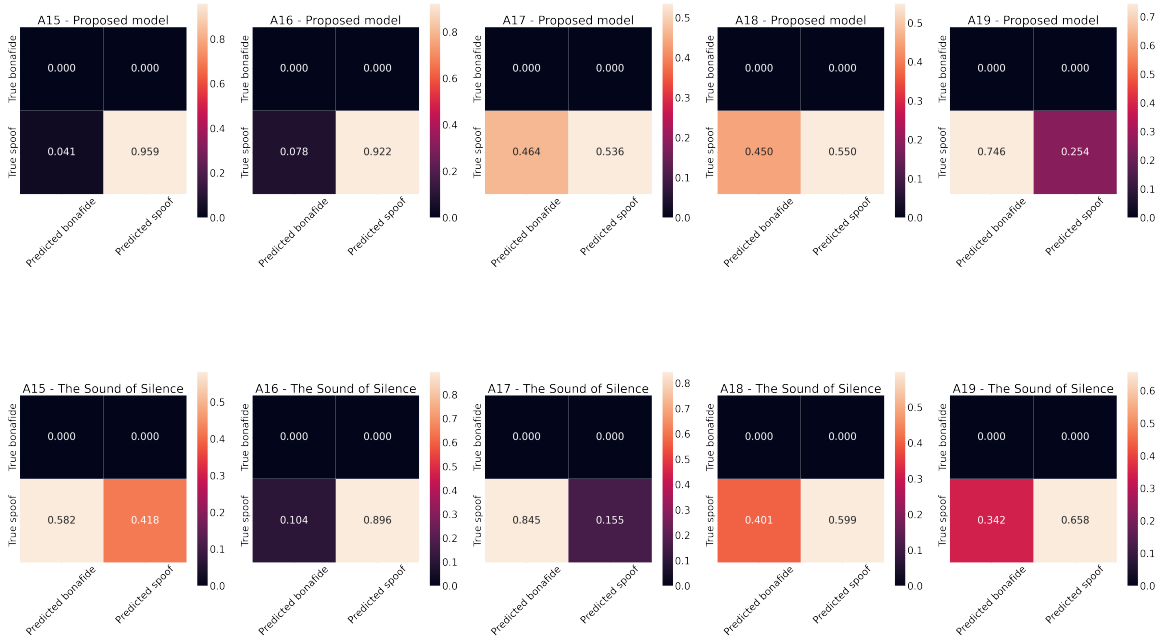


Figure 4.12: Comparison of the heatmaps divided by algorithm, algorithms A15-A19.

5

Perceptive Tests and Results Comparison

5.1 Goal of the Audio Perceptive Tests

The goal of this experiment is to assess with precision the ability of human testers to distinguish between Deep Fake tracks and genuine ones.
The inspiration for this study was originated by Müller et al. [44]. The goal is to repeat the test with tracks from a different dataset, in particular from ASVSpoof2021, but with an enhanced focus on the uniformity of the perception for the users.
In the following sections, the conduction of the perceptual tests is presented in detail.

5.2 Experimental setup

5.2.1 Equipment

The following equipment was used for the experiment:

- Microsoft Surface tablet, provided with a wireless mouse and a keyboard.
- Soundproof headphones.
- A MATLAB application was deployed on the tablet for test hosting.

5.2.2 Tests' location

The following spaces were used for the experiments, based on what was available at the university's buildings on the dates of the tests:

- DEI/A building:
  - Lab LTTM.
  - Meeting room DEI/A 201.
  - Meeting room DEI/D.
- Torre archimede:
  - 3rd floor, 3CD3 (ex InfLab).
  - 6th floor, 7A2 (ex 732).

5.3 Data collection

5.3.1 Human samples selection

The selection of the human sample for the experiment didn't follow any particular criterion. A public call for the experiment was advertised through social media pages managed by the University of Padova students (Fig. 5.1), as a "human feedback on audio Deep Fake" study.
The advertisement provided a brief description of the experiment and a link to a Doodle.com page (Fig. 5.2), on which the subject could book a time slot (Fig. 5.3).
It's important to highlight that a significant number of participants were Master's degree students and PhD students from the Department of Information Engineering and the Math department. It is thus important to highlight that the considered sample is biased towards educated people.

5.3.2 Experiment description

The experiments were conducted in a quiet room, where the subjects were provided with sound-proof headphones. This setting guarantees a good level of isolation from external interferences, which could tamper the test's results.
The subject is initially described in the experiment before being asked to complete the forms labeled "Age," "Sex," and "Hearing" (Figure 5.4a, the final term stands for "Hearing Impairments").

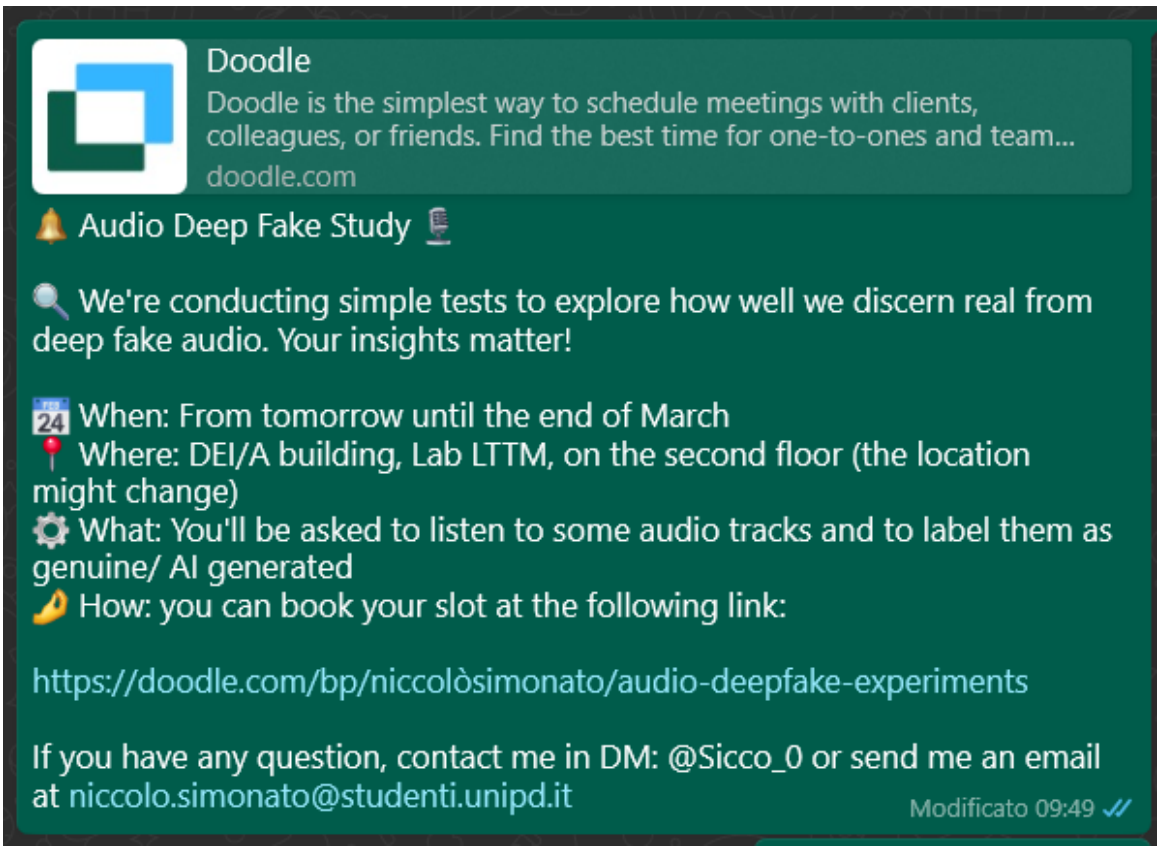


Figure 5.1: The advertisement that was published on the social media channels of the University of Padova.

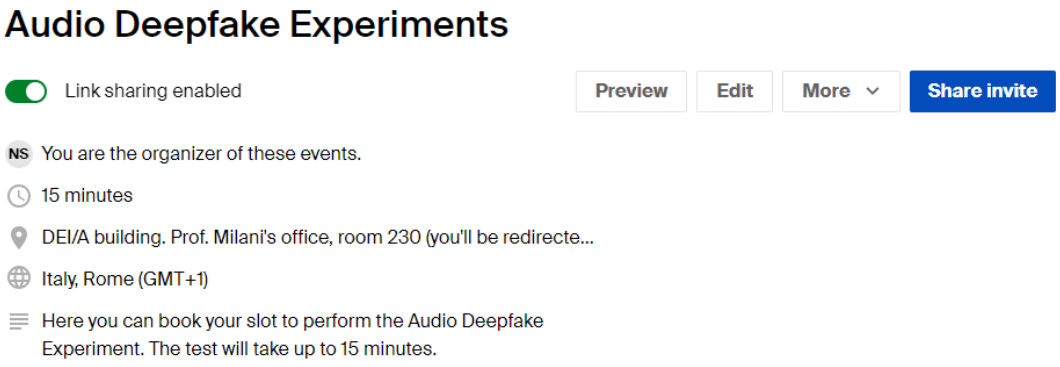


Figure 5.2: The homepage of the booking page of the experiment.

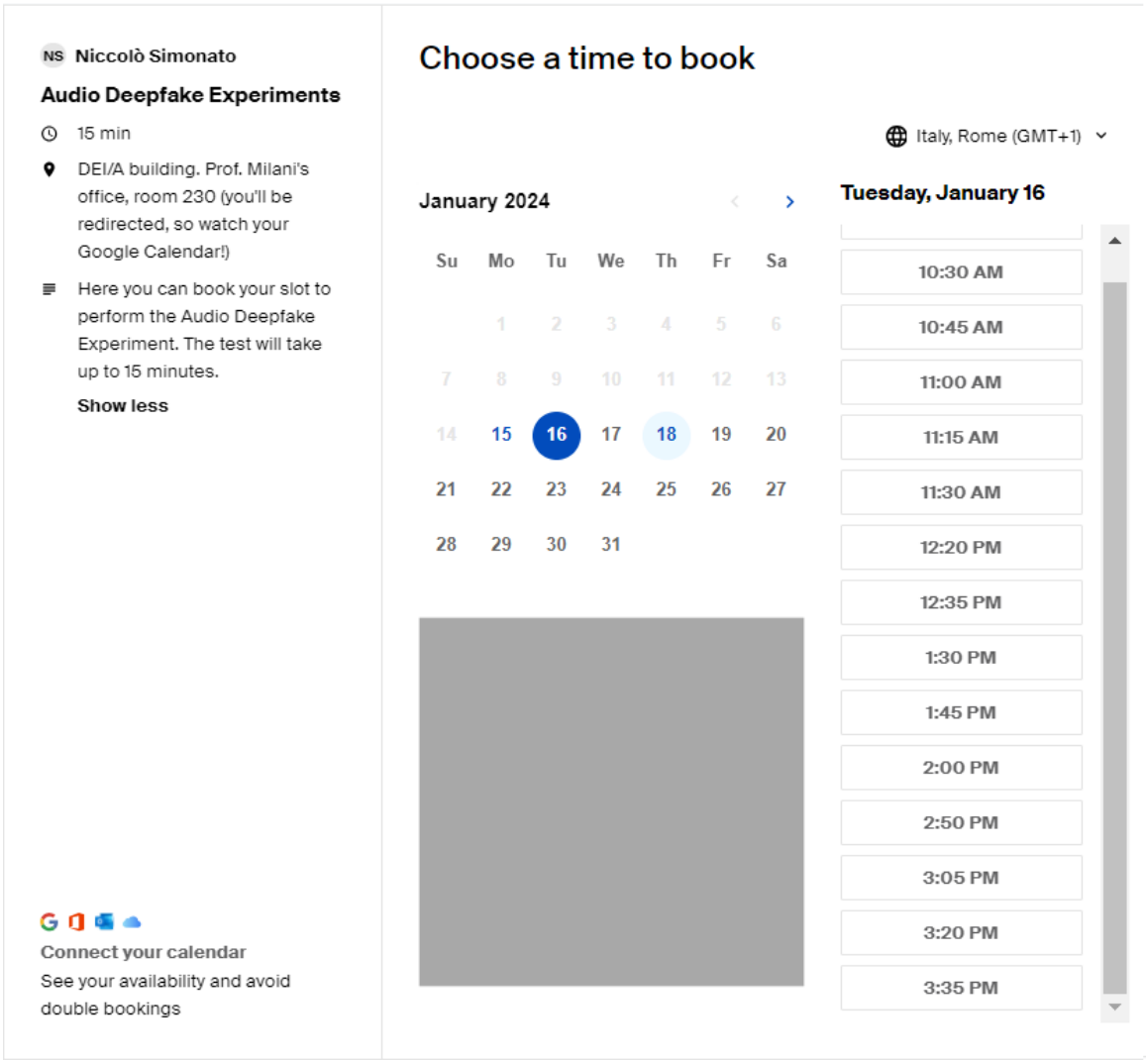


Figure 5.3: The booking page of the experiment.

In particular, the subject is told that the tracks are a mix between Deep Fake tracks and ”bonafide” tracks.
Once the fields are filled, the subject presses the “Start” button, and so the test begins. The test is composed of two sections: the training phase, during which 4 tracks are played, and the actual test, consisting of the listening of 20 more tracks. The rationale behind the training phase is the need for the user to familiarize with both the device and the audio tracks. For this reason, the data collected during the training phase are discarded.
Both the training phase and the actual test follow the same structure: an audio track is played exactly one time, and the user is asked to select one option in the form. Figure 5.4 shows the appearance of the interface.
Once the answer is selected, the user presses the button “Next track” to listen to the next audio file (Figure 5.4c). When the training phase is complete, the user is asked to click in the middle of the screen to proceed to the next phase of the test. At the end of the test, the subject is greeted and the application shows the homepage (Figure 5.4d).
The total time needed for the procedure was approximately 10 minutes.

5.4 Results

Figures 5.7, 5.10, 5.13, 5.16 show the violinplots of the responses of the users, per each track.
Violinplots are boxplots in their essence, with the addition of a kernel density estimation plot on the side[56], that can be useful to visualize the distribution of the samples.
Each track is identified by a unique code of the form *DF_E_XXXXXXX*, where *XXXXXXX* is a 7-digit number. Moreover, each ”violin” is colored with a shade of blue that is dependent on the median of the distribution; the darker the shade of blue, the closer the median of the responses is to 5.
In the following paragraphs, each batch of the experiment will be described in detail, and more information will be provided concerning the response times and the composition of the sample of users. The most common descriptive statistics will be used to analyze the responses, such as the average, the standard deviation, the median, and the 1st and 3rd quartiles; also, the median responses will be reported.

5.4.1 Batch 1

Batch 1 is characterized by a prominent presence of male testers, as shown by Figure 5.5. Moreover, the age of most of the testers is between 20 and 30 years old.

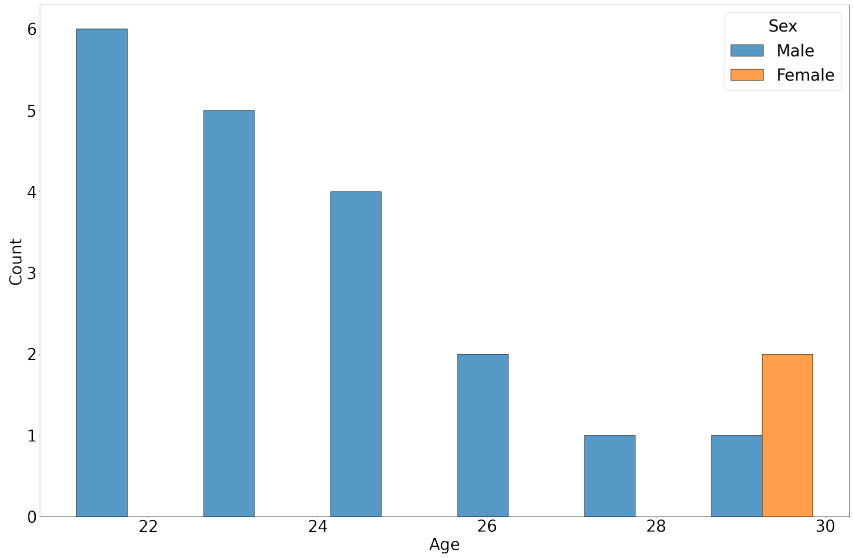


Figure 5.5: Batch 1. Distribution of the age of the human testers.

The average response time tends to decrease towards the end of the test, as does its standard deviation (Fig. 5.6); the specific reason was not investigated, as it's outside the scope of this research, but it seems reasonable to assume that toward the end of the test, the user might feel more confident in its ability, or might just want to end the test as soon as possible.

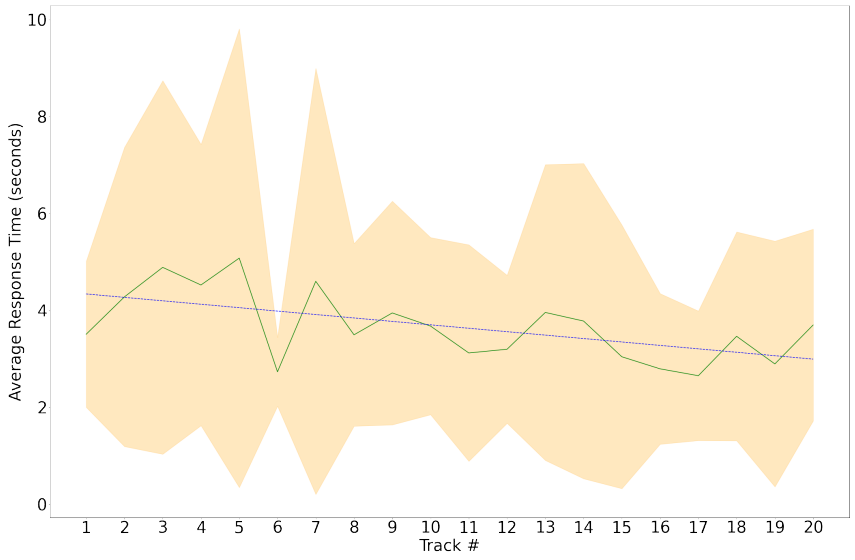


Figure 5.6: Batch 1. Distribution of response times of the human testers.

This batch's median responses show that the users perceived 11 tracks out of 20 as Deep Fakes (Table 5.1). Moreover, it looks like for most of the tracks the decisions made by the users tend to concentrate along the extremes of the spectrum, except the tracks DF_E_2065997, DF_E_2278809, DF_E_2310497 and DF_E_2104978, where the responses were almost evenly distributed (Fig. 5.7).

Table 5.1: Table of descriptive statistics for the results of Batch 1

	mean	std	1st quart.	2nd quart.	3rd quart.	Median response
DF_E_2019069	4.318	1.211	4.000	5.000	5.000	Deep Fake
DF_E_2593240	4.136	1.283	4.000	5.000	5.000	Deep Fake
DF_E_2582062	2.045	1.463	1.000	1.000	2.750	Bonafide
DF_E_2161802	2.227	1.193	1.000	2.000	3.000	Bonafide
DF_E_2265675	4.091	1.269	4.000	4.500	5.000	Deep Fake
DF_E_2656915	1.318	0.894	1.000	1.000	1.000	Bonafide
DF_E_2065997	3.409	1.501	2.000	4.000	5.000	Deep Fake
DF_E_2399273	1.545	0.912	1.000	1.000	2.000	Bonafide
DF_E_2101107	4.091	1.269	4.000	5.000	5.000	Deep Fake
DF_E_2410773	2.182	1.181	1.000	2.000	2.750	Bonafide
DF_E_2111216	4.409	0.908	4.000	5.000	5.000	Deep Fake
DF_E_2501864	1.864	1.283	1.000	1.000	2.000	Bonafide
DF_E_2278809	3.500	1.406	2.000	4.000	5.000	Deep Fake
DF_E_2250206	1.955	0.999	1.000	2.000	2.750	Bonafide
DF_E_2139003	4.409	0.796	4.000	5.000	5.000	Deep Fake
DF_E_2310497	3.591	1.681	2.000	4.000	5.000	Deep Fake
DF_E_2447144	2.091	1.342	1.000	2.000	2.000	Bonafide
DF_E_2104978	2.955	1.495	2.000	3.000	4.000	Bonafide
DF_E_2512621	4.545	0.800	4.000	5.000	5.000	Deep Fake
DF_E_2079980	3.864	0.990	3.250	4.000	4.750	Deep Fake

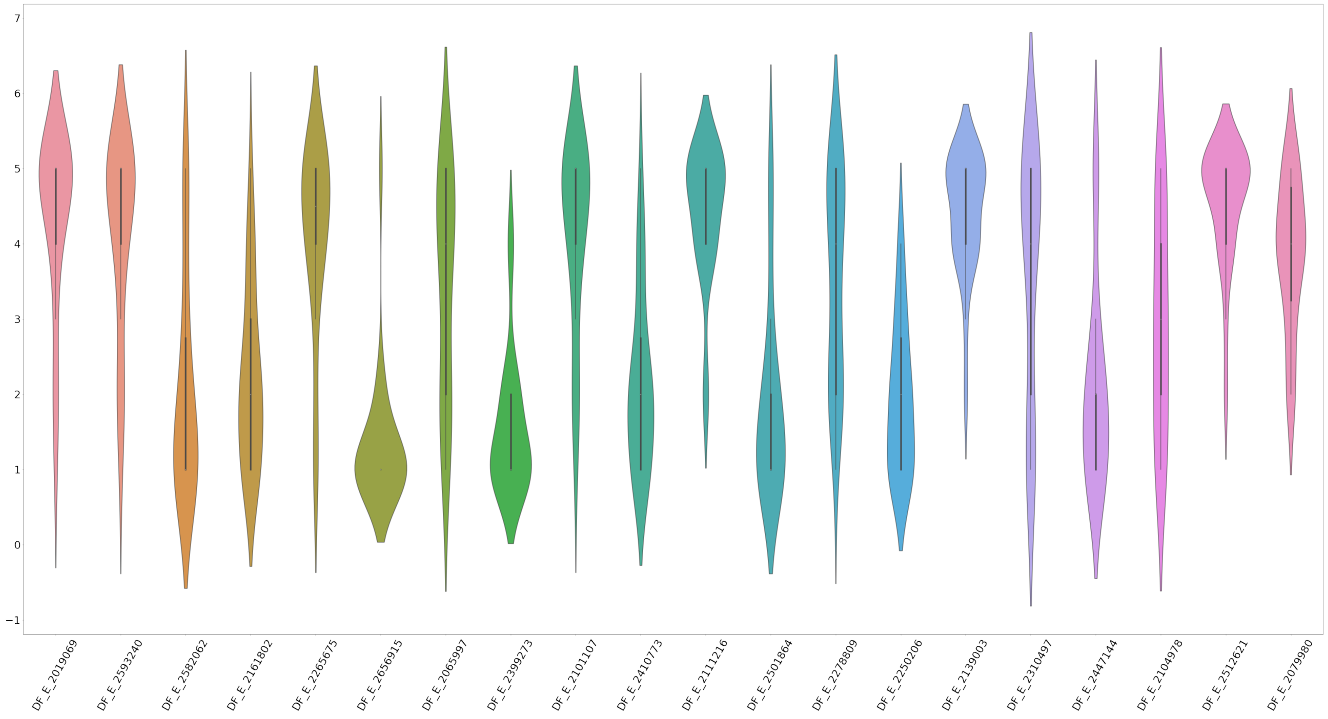


Figure 5.7: Batch 1. Distribution of the responses given by the human testers.

5.4.2 Batch 2

This batch resembles the previous one, but it shows a greater presence of female testers (Fig. 5.8). The prominent age range is also very similar.

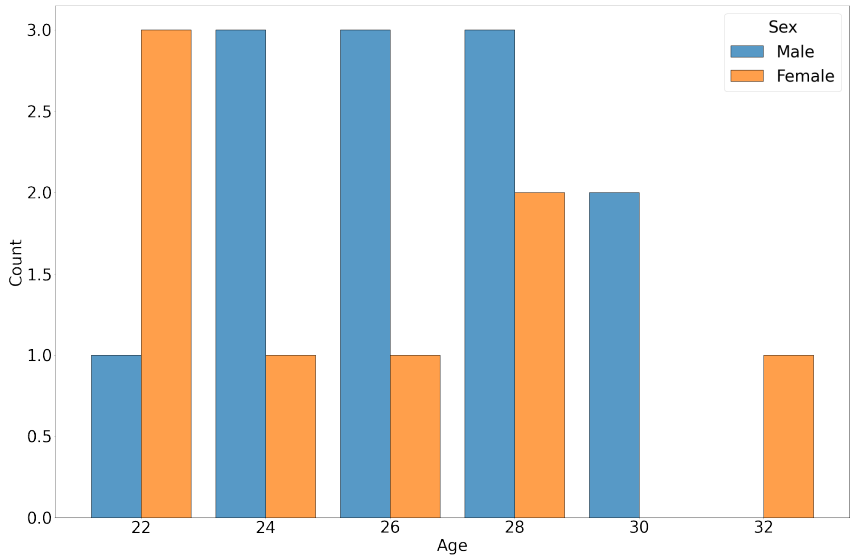


Figure 5.8: Batch 2. Distribution of the age of the human testers.

The distribution of the response time as well is not different from the previous batch, as shown by Figure 5.9.

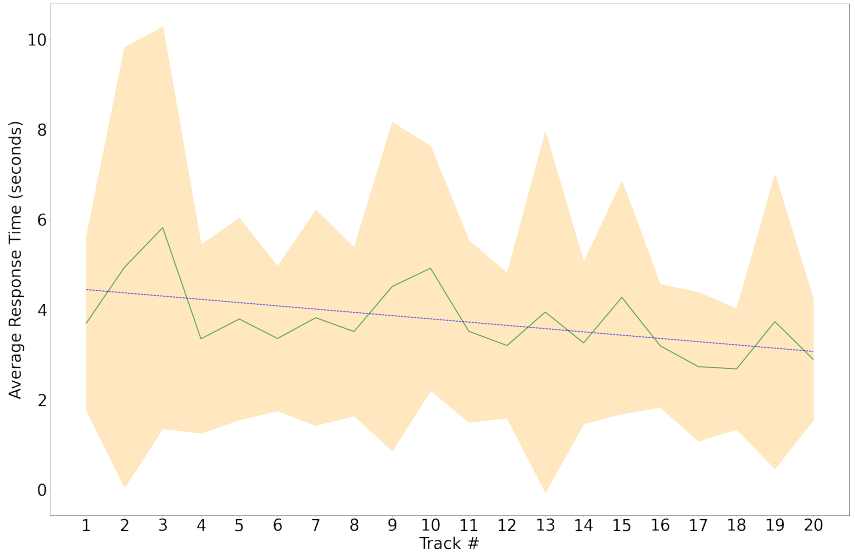


Figure 5.9: Batch 2. Distribution of response times of the human testers.

The responses given by the testers are, on the other hand, very different. Most of the tracks were, on the median, perceived as bonafide samples (15 out of 20, see Table 5.2). Just like the previous

batch, most of the responses seem to be focused on the extremes of the distribution, except for the tracks DF_E_2640797, DF_E_2696221, DF_E_2734713, DF_E_2533867 and DF_E_3034767 (Fig. 5.10).

Table 5.2: Table of descriptive statistics for the results of Batch 2

	mean	std	1st quart.	2nd quart.	3rd quart.	Median response
DF_E_2798755	1.318	0.716	1.000	1.000	1.000	Bonafide
DF_E_2640797	2.273	1.316	1.000	2.000	3.750	Bonafide
DF_E_2696221	3.318	1.427	2.000	4.000	4.750	Deep Fake
DF_E_3223452	4.591	0.734	4.000	5.000	5.000	Deep Fake
DF_E_2826871	1.682	0.894	1.000	1.500	2.000	Bonafide
DF_E_2734713	4.136	1.356	2.500	5.000	5.000	Deep Fake
DF_E_2890769	4.136	1.283	4.000	5.000	5.000	Deep Fake
DF_E_3102836	2.045	1.397	1.000	1.500	2.000	Bonafide
DF_E_2533867	2.318	1.427	1.000	2.000	3.750	Bonafide
DF_E_3174677	2.364	1.364	1.250	2.000	2.750	Bonafide
DF_E_3048575	1.409	0.796	1.000	1.000	1.750	Bonafide
DF_E_3034767	2.364	1.465	1.000	2.000	3.750	Bonafide
DF_E_2844669	1.455	0.912	1.000	1.000	1.750	Bonafide
DF_E_2892729	1.909	1.269	1.000	1.000	2.000	Bonafide
DF_E_2888267	1.636	0.790	1.000	1.500	2.000	Bonafide
DF_E_2798765	1.955	1.046	1.000	2.000	2.000	Bonafide
DF_E_2501739	4.773	0.528	5.000	5.000	5.000	Deep Fake
DF_E_3111710	1.909	1.151	1.000	2.000	2.000	Bonafide
DF_E_3134280	2.045	1.327	1.000	2.000	2.000	Bonafide
DF_E_2953912	1.955	1.214	1.000	1.500	2.000	Bonafide

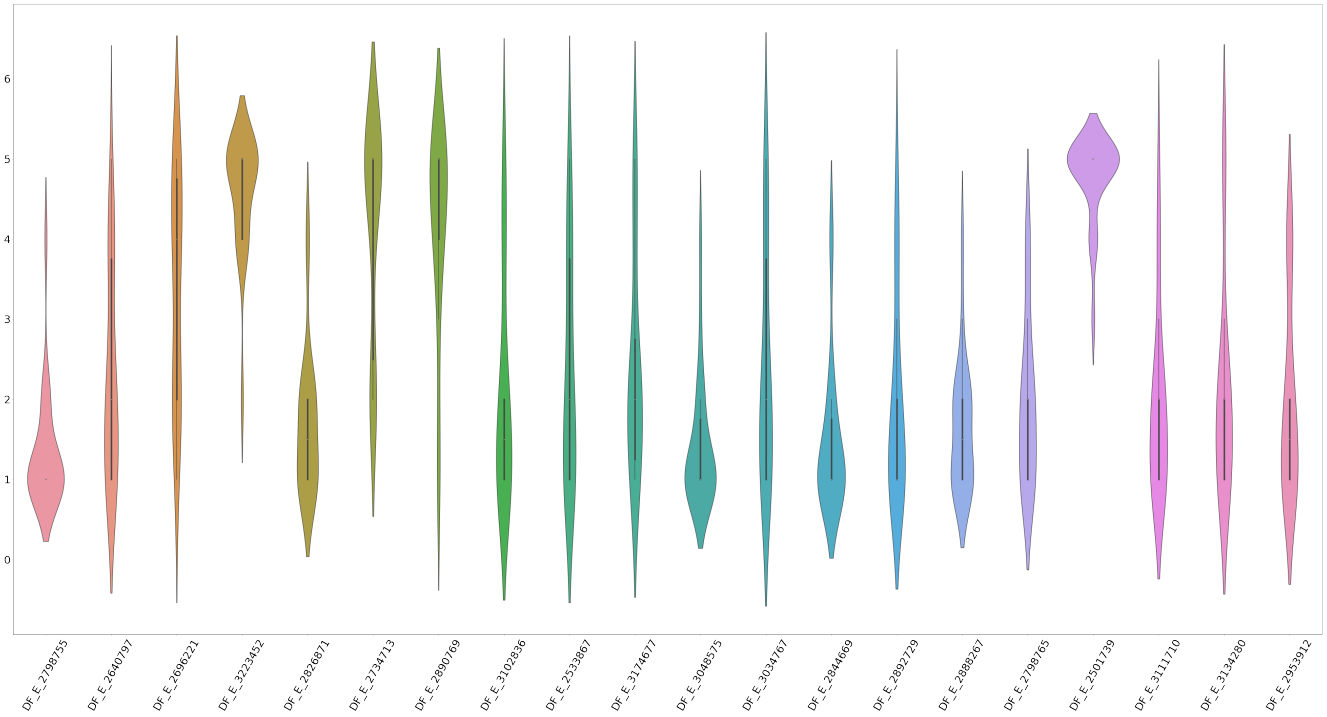


Figure 5.10: Batch 2. Distribution of the responses given by the human testers.

5.4.3 Batch 3

The distribution of the age of the testers resembles the one of the first batch, as the male presence is more prominent than the female one. The only remarkable difference is the age range of the female testers, which is lower by 5 years approximately (Fig. 5.11).

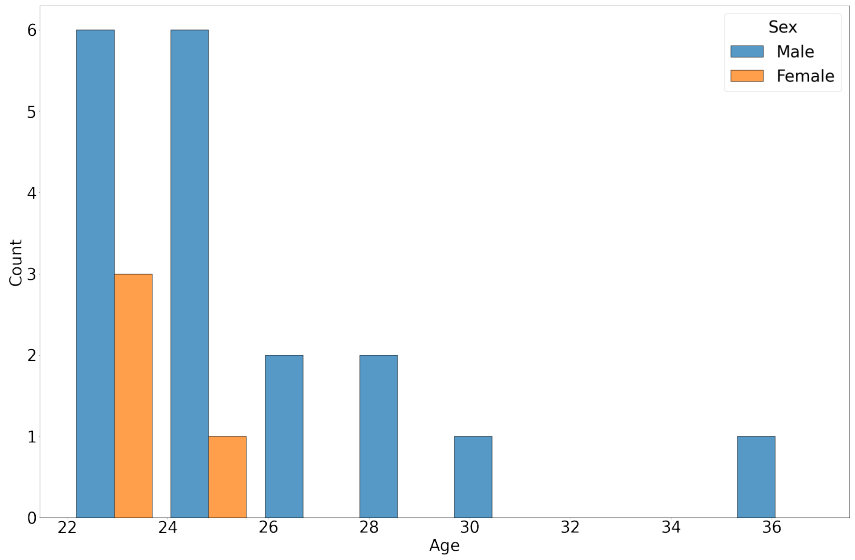


Figure 5.11: Batch 3. Distribution of the age of the human testers.

The distribution of the response times is also similar to the previous batches, except for the twelfth track (*DF_E_3034767*), which appears to have a significantly higher response time than expected.

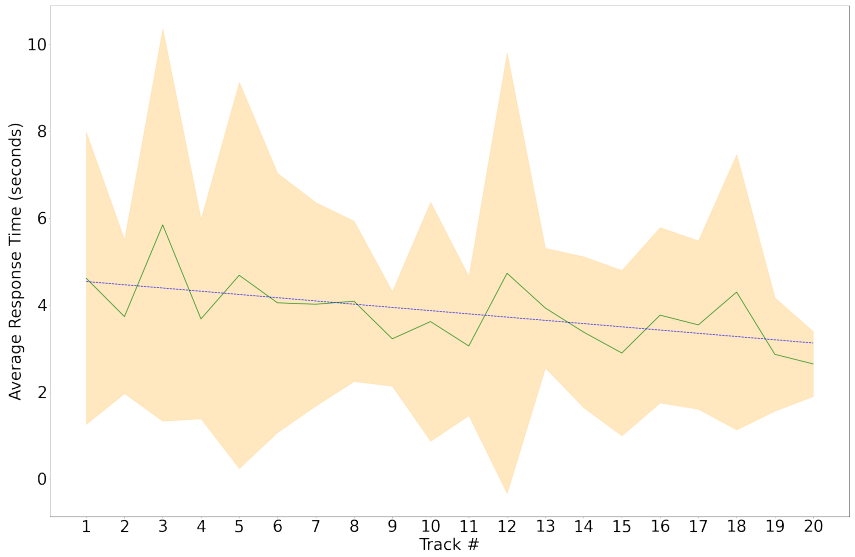


Figure 5.12: Batch 3. Distribution of response times of the human testers.

The responses recorded for this batch are in distribution different from the ones of the previous

batch, that is, they are almost evenly distributed in the $[1, 5]$ range and most of them are confident that the tracks are bonafide (12 out of 20, see 5.4). The list of the tracks on which the responses are less focused on the extremes is reported in Table 5.3.

Table 5.3: List of tracks on which the users responded in a non-uniform way (Batch 3).

Track code
DF_E_3249083
DF_E_3458366
DF_E_3406928
DF_E_3147542
DF_E_3145913
DF_E_3541778
DF_E_3011154
DF_E_3565230
DF_E_3535166
DF_E_3260913
DF_E_3270518

Table 5.4: Table of descriptive statistics for the results of Batch 3

	mean	std	1st quart.	2nd quart.	3rd quart.	Median response
DF_E_3249083	3.955	1.046	3.000	4.000	5.000	Deep Fake
DF_E_3458366	3.318	1.615	2.000	4.000	5.000	Deep Fake
DF_E_3406928	2.500	1.263	2.000	2.000	4.000	Bonafide
DF_E_3147542	3.455	1.595	2.000	4.000	5.000	Deep Fake
DF_E_3533115	2.318	1.359	1.000	2.000	3.000	Bonafide
DF_E_3065323	2.045	1.430	1.000	1.500	2.000	Bonafide
DF_E_3145913	3.227	1.232	2.000	3.000	4.000	Bonafide
DF_E_3516804	1.727	1.279	1.000	1.000	2.000	Bonafide
DF_E_3541778	3.000	1.662	1.000	4.000	4.000	Deep Fake
DF_E_3011154	3.455	1.535	2.000	4.000	5.000	Deep Fake
DF_E_3565230	2.455	1.438	1.000	2.000	4.000	Bonafide
DF_E_3298078	1.864	1.390	1.000	1.000	2.000	Bonafide
DF_E_3535166	2.273	1.352	1.000	2.000	3.750	Bonafide
DF_E_3056228	3.818	1.097	3.250	4.000	4.750	Deep Fake
DF_E_3260913	3.364	1.620	2.000	4.000	5.000	Deep Fake
DF_E_3238722	3.682	1.323	3.000	4.000	5.000	Deep Fake
DF_E_3725877	1.409	0.796	1.000	1.000	1.750	Bonafide
DF_E_3203878	1.455	0.739	1.000	1.000	2.000	Bonafide
DF_E_3594417	1.545	0.912	1.000	1.000	2.000	Bonafide
DF_E_3270518	2.091	1.306	1.000	2.000	2.750	Bonafide

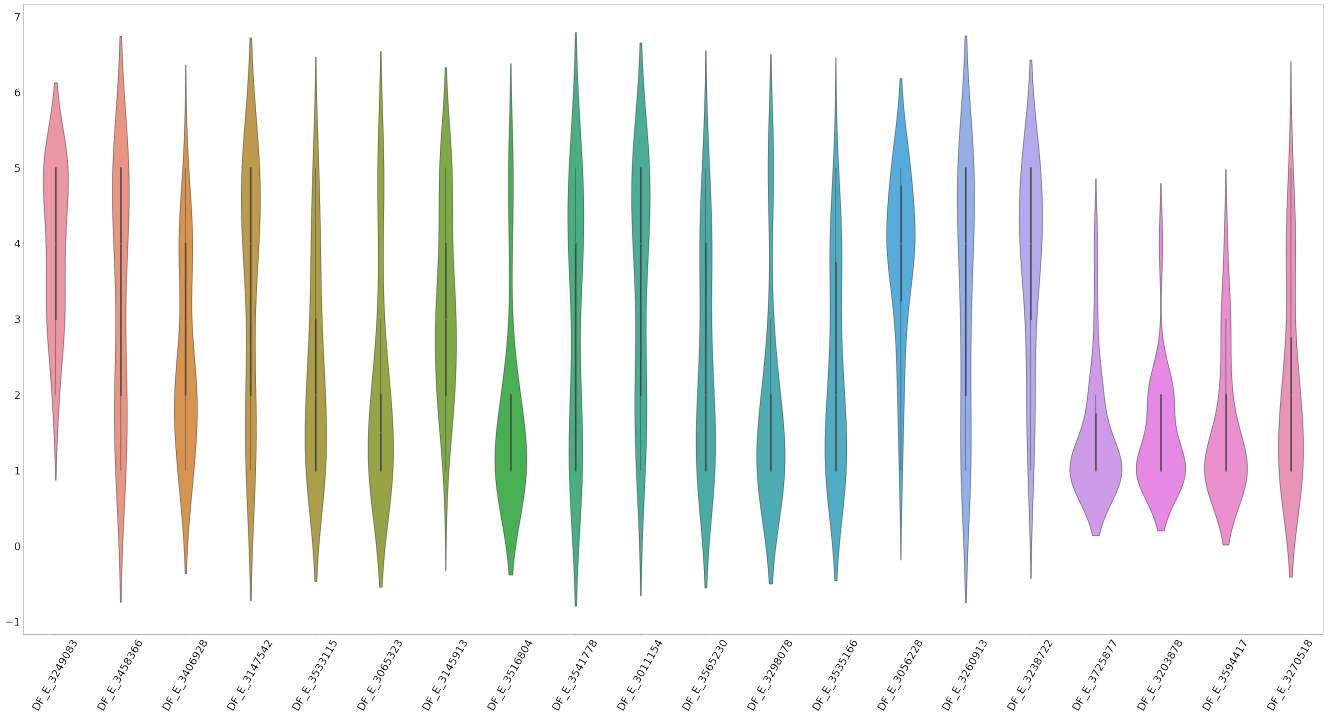


Figure 5.13: Batch 3. Distribution of the responses given by the human testers.

5.4.4 Batch 4

This batch resembles the second one, for the similar presence of female testers (Fig. 5.14), and for a similar age range.

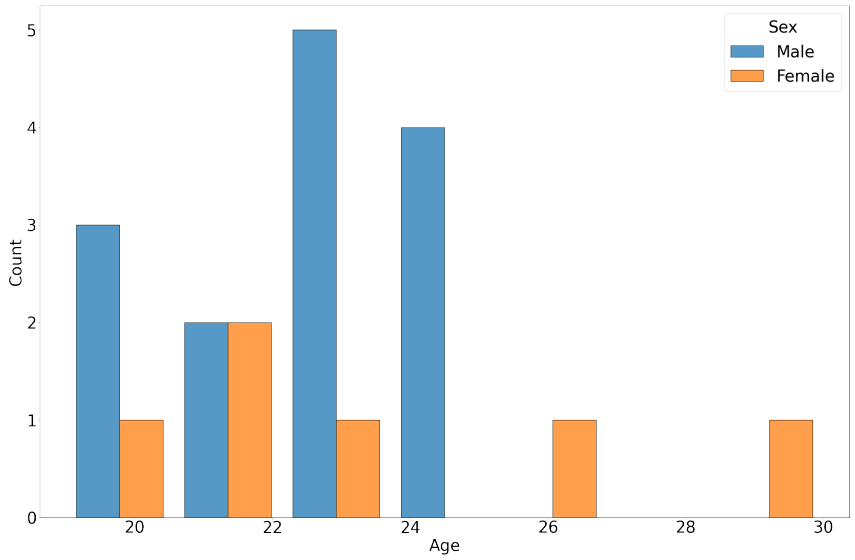


Figure 5.14: Batch 4. Distribution of the age of the human testers.

The distribution of the response time as well is not different from the other batches, but it seems that the standard deviation of the responses is slightly higher (Fig. 5.15).

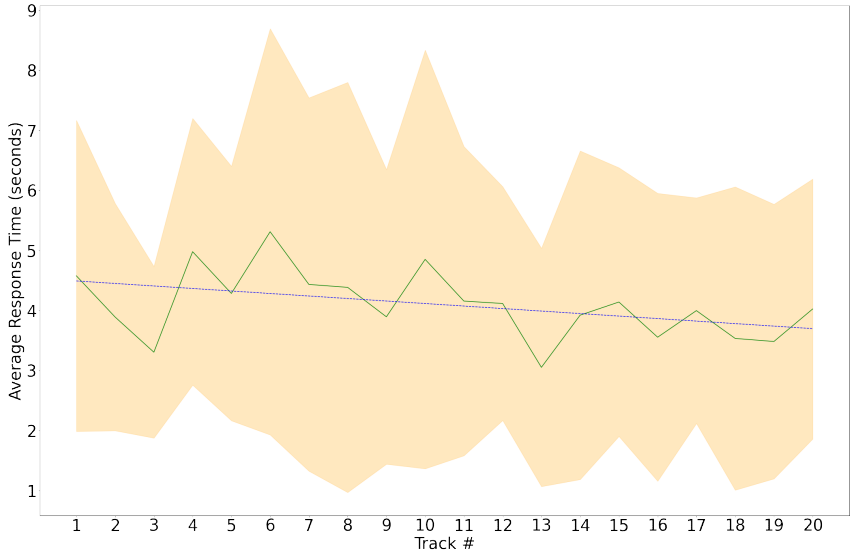


Figure 5.15: Batch 4. Distribution of response times of the human testers.

This batch's median responses show that the users perceived 10 tracks out of 20 as Deep Fakes (Table 5.6). Just like the first batch, the responses of the users tend to concentrate along the ex-

tremes of the spectrum, except for the tracks reported in table 5.5, where the responses are more evenly distributed (Fig. 5.16).

Table 5.5: List of tracks on which the users responded in a non-uniform way (Batch 4).

Track code
DF_E_4254160
DF_E_3175374
DF_E_3630644
DF_E_3835736
DF_E_3730698
DF_E_3217288
DF_E_4121342
DF_E_4054820

Table 5.6: Table of descriptive statistics for the results of Batch 4

	mean	std	1st quart.	2nd quart.	3rd quart.	Median response
DF_E_4181319	1.950	1.395	1.000	1.000	2.000	Bonafide
DF_E_3915900	4.700	0.470	4.000	5.000	5.000	Deep Fake
DF_E_4254160	3.400	1.569	2.000	4.000	5.000	Deep Fake
DF_E_3680574	2.400	1.603	1.000	2.000	2.750	Bonafide
DF_E_3175374	3.700	1.490	2.000	4.000	5.000	Deep Fake
DF_E_3954854	2.200	1.361	1.000	2.000	3.000	Bonafide
DF_E_3629407	3.900	1.553	3.500	5.000	5.000	Deep Fake
DF_E_3630644	3.750	1.517	2.000	4.500	5.000	Deep Fake
DF_E_3598963	1.900	1.410	1.000	1.000	2.000	Bonafide
DF_E_4235595	2.100	1.447	1.000	1.500	2.500	Bonafide
DF_E_3691064	2.150	1.461	1.000	2.000	2.250	Bonafide
DF_E_3835736	2.900	1.483	2.000	2.500	4.000	Bonafide
DF_E_3148563	4.250	1.209	4.000	5.000	5.000	Deep Fake
DF_E_3876831	3.950	1.317	3.000	4.500	5.000	Deep Fake
DF_E_3730698	3.250	1.293	2.000	3.000	4.250	Bonafide
DF_E_3217288	3.550	1.669	1.750	4.000	5.000	Deep Fake
DF_E_3745827	3.400	1.429	2.750	4.000	4.000	Deep Fake
DF_E_3225549	4.100	1.373	3.750	5.000	5.000	Deep Fake
DF_E_4121342	2.750	1.446	2.000	2.000	4.000	Bonafide
DF_E_4054820	2.842	1.463	2.000	2.000	4.000	Bonafide

Table 5.7: Results of the perceptive tests, divided per batch.

Batch	*Correctly classified tracks*	*Accuracy*
Batch 1	11/20	55%
Batch 2	5/20	25%
Batch 3	8/20	40%
Batch 4	10/20	50%

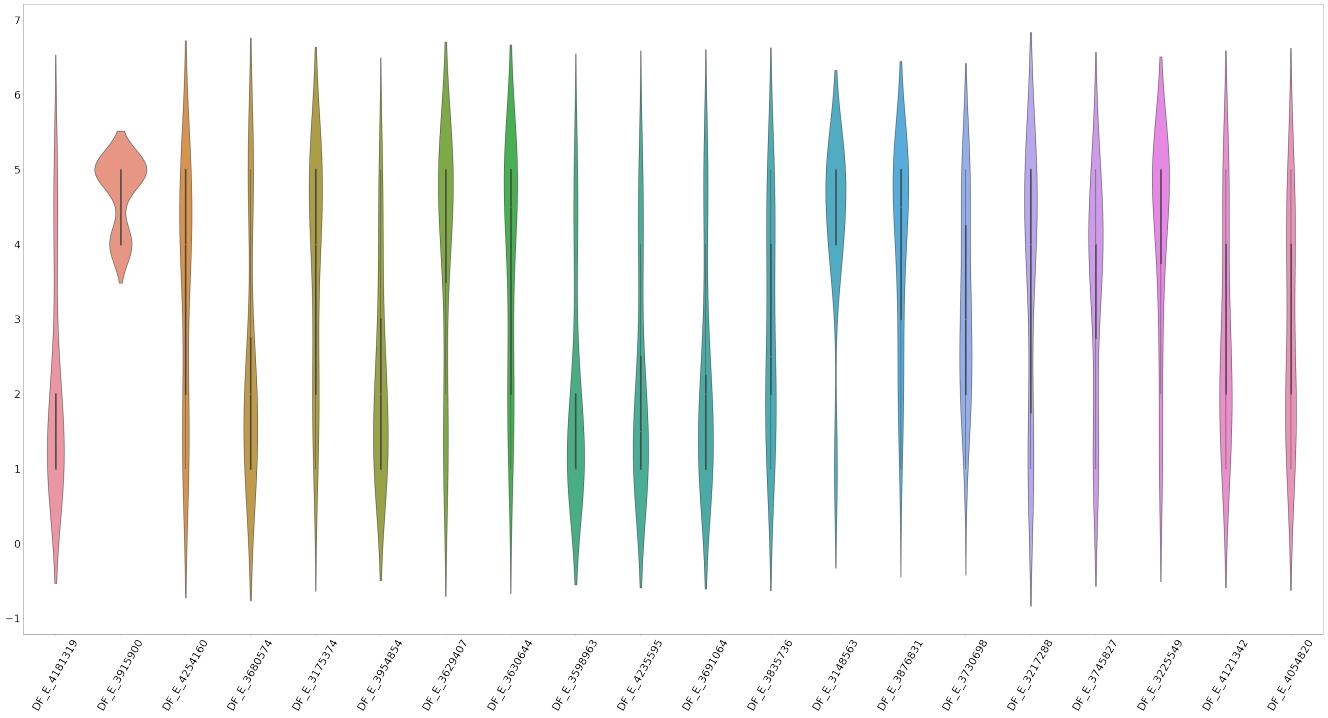
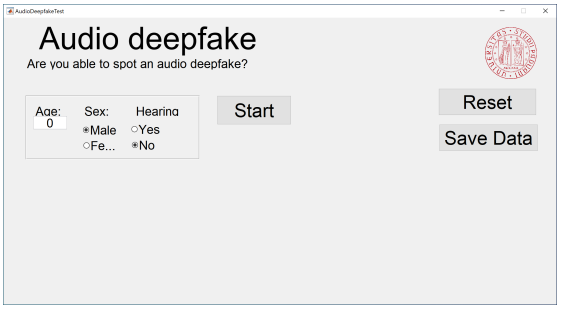


Figure 5.16: Batch 4. Distribution of the responses given by the human testers.

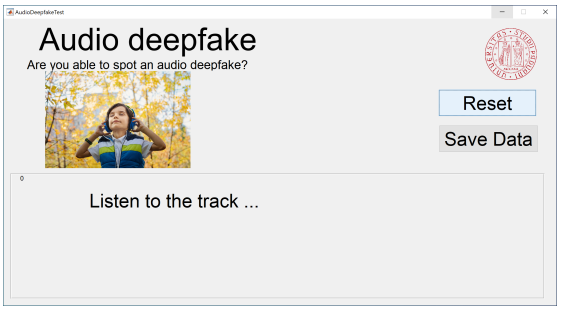
5.5 Comments on the results

An extensive summary of the results is provided in Table 5.7, which is a review of the median responses for each track. These results suggest that humans have a limited ability to recognize Deep Fake audio recordings, as about 50% of the audio samples in each batch were correctly identified. The batch with significantly lower results is Batch 2, where only 1 in 4 tracks were correctly classified.

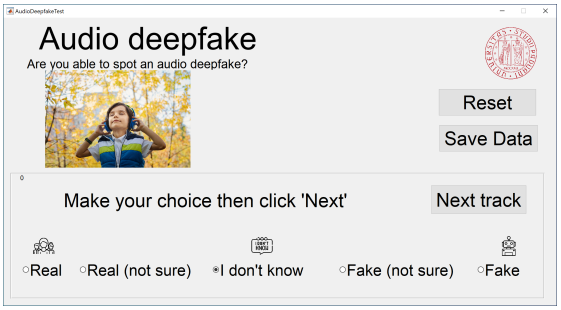
Figure 5.4: Screenshots from the application used to conduct the perceptual tests.



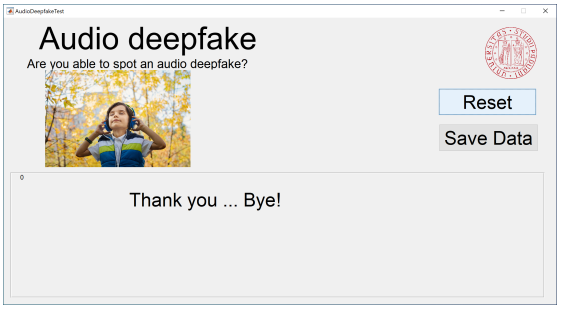
(a) Homepage of the application.



(b) How the interface appears while the track is playing.



(c) Interface of the form, on which the user can select the option.



(d) Appearance of the application at the end of the test.

6 Conclusion

This work illustrates an innovative methodology to perform the Deep Fake detection task, by leveraging the features extracted from the INR of the original track. Even though the performances of the model are inferior to the ones of other state-of-the-art models, the relative simplicity of the process makes it an interesting, alternative solution.
Moreover, this research confirms the low reliability of the human ear in detecting the presence of synthetic speech. This underlines the need for more accurate methods to conduct the aforementioned task, and the need to implement them in all those contexts where it's crucial to use the voice as a means of authentication.

6.1 Future directions

The natural prosecution of this research is to find a way to close the performance gap between the other state-of-the-art methods and the proposed ones.
What follows is a brief sequence of suggestions that may provide interesting insights to future research in this field.

6.1.1 Exploring different features for INRs

The use of the MSE as a feature for the Deep Fake detection task showed limited effectiveness, as the resulting accuracy was below the state-of-the-art. Different features derived from the INR

model might be more significant and therefore yield better results.
The activation values of the hidden neurons of the Neural Network are already being used for improving the explainability of both Machine Learning and Deep Learning models[57], and therefore might provide useful insights that could improve the presented method. In particular, this method uses a game-theory-based approach that allows to compute the "contribution" value of each feature, allowing to ease the interpretability of the prediction. Building on this, the analysis of the differences in the contribution values of the hidden layers of the INRs between Deep Fake and bonafide samples might lead to interesting discoveries in this field.

6.1.2 Combining different approaches

The successful use of the feature fusion approach adopted by Mari et al.[13] suggests that this direction could provide interesting results, as features of different natures are leveraged to obtain the result.
A new attempt to accomplish the same task could employ features extracted from Implicit Neural Representation models, that would just be added to the ones that are already being used.

6.1.3 Assessing the change in the perception of Deep Fake audios

The constantly changing landscape of the Deep Fake generation methods pushes the need to monitor how the improvements in this field affect the perception of synthetic speech.
Future studies may assess how the results shown in the previous chapter 5 may differ from similar tests, conducted with state-of-the-art methods.
In particular, the focus could be the effectiveness of anti-forensic techniques, such as the injection of noise and the increase of the compression strength[13], that may be adopted to prevent the detection.

Listing of figures

Listing of tables

Listing of acronyms

AI Artificial Intelligence. 3, 7, 13, 17, 23

ASV Automatic Speaker Verification. ix, 5, 30

CNN Convolutional Neural Network. 11, 12, 19, 22, 37–39, 73, 75

CT Computer Tomography. 18

DF Deep Fake. i, ix, 3–7, 14, 17, 18, 21–23, 29, 33, 34, 49, 50, 53, 55, 56, 59, 63, 65, 67, 68, 71, 72

DFT Discrete Fourier Transform. 16

DL Deep Learning. 72

DNN Deep Neural Network. 19

FFNN Feed Forward Neural Network. 9–11, 34, 73

FFT Fast Fourier Transform. 16, 17

GAN Generative Adversarial Network. 17, 18, 20, 21

GMM Gaussian Mixture Model. 20

INR Implicit Neural Representation. i, 7, 25–28, 34, 71–73

LA Logical Access. 30–33, 36, 37, 73, 75

LDA Linear Discriminant Analysis. 22

LSF Line Spectral Features. 21

MFCC Mel Frequency Cepstral Coefficients. 17

ML Machine Learning. 22, 72

MRI Magnetic Resonance Imaging. 18

MSE Mean Squared Error. 20, 21, 36, 71, 73

NeRF Neural Radiance Field. 26, 27

NN Neural Network. 9, 11, 22, 25, 72

PA Physical Access. 30, 31, 33

RGB Red-Green-Blue. 27

RNN Recurrent Neural Network. 20

ROC Receiver Operator Characteristic. 43, 73

SVM Support Vector Machine. 22

TTS Text-To-Speech. 19, 20, 30, 31, 41, 42

VC Voice Conversion. 19, 20, 30, 31

VCC Voice Conversion Challenge. 33

VCTK Voice Cloning Toolkit. 31

References

[1] R. Glotzbach, "Generative ai: An overview," in *Proceedings of Society for Information Technology & Teacher Education International Conference 2024*, J. Cohen and G. Solano, Eds. Las Vegas, Nevada, United States: Association for the Advancement of Computing in Education (AACE), March 2024, pp. 754–756. [Online]. Available: https://www.learntechlib.org/p/224036

[2] J. Wakefield, "Deepfake presidents used in Russia-Ukraine war," https://www.bbc.com/news/technology-60780142, 2022, [Online; accessed 21-May-2024].

[3] "'Deepfake Doomsday': The Role of Artificial Intelligence in Amplifying Apocalyptic Islamist Propaganda - GNET — gnet-research.org," https://gnet-research.org/2023/08/29/deepfake-doomsday-the-role-of-artificial-intelligence-in-amplifying-apocalyptic-islamist-propaganda/, [Accessed 21-10-2024].

[4] F. M. Nur, "Imam mahdi in sunni tradition: Differences in beliefs and interpretations," *Abrahamic Religions: Jurnal Studi Agama-Agama*, vol. 3, no. 2, pp. 313–325, 2023.

[5] BBC, "Meta requires political advertisers to mark when deepfakes used," https://www.bbc.com/news/technology-67366311, 2023, [Online; accessed 21-May-2024].

[6] K. V. VS and S. Naveed, "A review of automatic speaker verification systems with feature extractions and spoofing attacks," in *2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2024, pp. 1999–2005.

[7] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, "Spoofing and countermeasures for speaker verification: A survey," *Speech Communication*, vol. 66, pp. 130–153, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167639314000788

[8] "Speechify: Free AI Voice Generator | Lifelike Voices & Voice Cloning — speechify.com," https://speechify.com/ai-voice-generator/, [Accessed 10-11-2024].

[9] "ElevenLabs: Free Text to Speech & AI Voice Generator | ElevenLabs — elevenlabs.io," https://elevenlabs.io/, [Accessed 10-11-2024].

[10] "Free AI Voice Generator: Versatile Text to Speech Software | Murf AI — murf.ai," https://murf.ai/, [Accessed 10-11-2024].

[11] N. M. Müller, F. Dieckmann, P. Czempin, R. Canals, K. Böttinger, and J. Williams, "Speech is silver, silence is golden: What do asvspoof-trained models really learn?" *arXiv preprint arXiv:2106.12914*, 2021.

[12] D. Mari, F. Latora, and S. Milani, "The sound of silence: Efficiency of first digit features in synthetic audio detection," in *2022 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2022, pp. 1–6.

[13] D. Mari, D. Salvi, P. Bestagini, and S. Milani, "All-for-one and one-for-all: Deep learning-based feature fusion for synthetic speech detection," *arXiv preprint arXiv:2307.15555*, 2023.

[14] L. Cuccovillo, M. Gerhardt, and P. Aichroth, "Audio spectrogram transformer for synthetic speech detection via speech formant analysis," in *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2023, pp. 1–6.

[15] N. M. Müller, P. Czempin, F. Dieckmann, A. Froghyar, and K. Böttinger, "Does audio deepfake detection generalize?" 2022.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[17] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *CoRR*, vol. abs/1511.08458, 2015. [Online]. Available: http://arxiv.org/abs/1511.08458

[18] A. Ivanov, N. Dryden, T. Ben-Nun, S. Li, and T. Hoefler, "Data movement is all you need: A case study on optimizing transformers," *CoRR*, vol. abs/2007.00072, 2020. [Online]. Available: https://arxiv.org/abs/2007.00072

[19] L.-C. Chen, J.-T. Sheu, Y.-J. Chuang, and Y. Tsao, "Predicting the travel distance of patients to access healthcare using deep neural networks," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 10, pp. 1–11, 2022.

[20] P. Gupta and M. Dixit, "Image-based crack detection approaches: a comprehensive survey," *Multimedia Tools and Applications*, vol. 81, pp. 1–49, 05 2022.

[21] P. I. Frazier, "A tutorial on bayesian optimization," 2018. [Online]. Available: https://arxiv.org/abs/1807.02811

[22] M. G. Christensen, *Introduction to audio processing*. Springer, 2019.

[23] M. Sahidullah and G. Saha, "Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition," *Speech Communication*, vol. 54, no. 4, pp. 543–565, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167639311001622

[24] S. DO, "Speech communication human and machine," 1987.

[25] M. Xu, L.-Y. Duan, J. Cai, L.-T. Chia, C. Xu, and Q. Tian, "Hmm-based audio keyword generation," in *Pacific-Rim Conference on Multimedia*. Springer, 2004, pp. 566–574.

[26] D. Mari, D. Salvi, P. Bestagini, and S. Milani, "All-for-one and one-for-all: Deep learning-based feature fusion for synthetic speech detection," 2023. [Online]. Available: https://arxiv.org/abs/2307.15555

[27] Z. Khanjani, G. Watson, and V. P. Janeja, "How deep are the fakes? focusing on audio deepfake: A survey," *CoRR*, vol. abs/2111.14203, 2021. [Online]. Available: https://arxiv.org/abs/2111.14203

[28] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[29] "MedSRGAN: medical images super-resolution using generative adversarial networks - Multimedia Tools and Applications — link.springer.com," https://link.springer.com/article/10.1007/s11042-020-08980-w, [Accessed 27-07-2024].

[30] S. E. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *CoRR*, vol. abs/1605.05396, 2016. [Online]. Available: http://arxiv.org/abs/1605.05396

[31] S. Cole, "Deepnude: The Horrifying App Undressing Women — vice.com," https://www.vice.com/en/article/kzm59x/deepnude-app-creates-fake-nudes-of-any-woman, [Accessed 27-07-2024].

[32] J. Vincent, "Copies of AI deepfake app DeepNude are easily accessible online — and always will be — theverge.com," https://www.theverge.com/2019/7/3/20680708/deepnude-ai-deepfake-app-copies-easily-accessible-available-online, [Accessed 27-07-2024].

[33] S. Maddocks, "'a deepfake porn plot intended to silence me': exploring continuities between pornographic and 'political' deep fakes," *Porn Studies*, vol. 7, no. 4, pp. 415–423, 2020. [Online]. Available: https://doi.org/10.1080/23268743.2020.1757499

[34] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[35] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018. [Online]. Available: http://arxiv.org/abs/1808.03314

[36] J. M. R. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. C. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:30919574

[37] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," *CoRR*, vol. abs/1612.07837, 2016. [Online]. Available: http://arxiv.org/abs/1612.07837

[38] A. Kain and M. Macon, "Spectral voice conversion for text-to-speech synthesis," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, vol. 1, 1998, pp. 285–288 vol.1.

[39] Y. Gao, R. Singh, and B. Raj, "Voice impersonation using generative adversarial networks," *CoRR*, vol. abs/1802.06840, 2018. [Online]. Available: http://arxiv.org/abs/1802.06840

[40] J. Yi, C. Wang, J. Tao, X. Zhang, C. Y. Zhang, and Y. Zhao, "Audio deepfake detection: A survey," *arXiv preprint arXiv:2308.14970*, 2023.

[41] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: from features to supervectors," *Speech Communication*, vol. 52, pp. 12–40, 01 2010.

[42] E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke, "Modeling prosodic feature sequences for speaker recognition," *Speech Communication*, vol. 46, no. 3, pp. 455–472, 2005, quantitative Prosody Modelling for Natural Speech Description and Generation. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167639305001020

[43] D. M. Chakravarty N., "A lightweight feature extraction technique for deepfake audio detection - Multimedia Tools and Applications — link.springer.com," https://link.springer.com/article/10.1007/s11042-024-18217-9, [Accessed 16-10-2024].

[44] N. M. Müller, K. Markert, and K. Böttinger, "Human perception of audio deepfakes," *CoRR*, vol. abs/2107.09667, 2021. [Online]. Available: https://arxiv.org/abs/2107.09667

[45] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," 2019.

[46] D. Xu, P. Wang, Y. Jiang, Z. Fan, and Z. Wang, "Signal processing for implicit neural representations," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 13 404–13 418. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/575c450013d0e99e4b0ecf82bd1afaa4-Paper-Conference.pdf

[47] E. Dupont, A. Goliński, M. Alizadeh, Y. W. Teh, and A. Doucet, "Coin: Compression with implicit neural representations," 2021.

[48] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser, "Learning shape templates with structured implicit functions," 2019.

[49] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *CoRR*, vol. abs/2006.09661, 2020. [Online]. Available: https://arxiv.org/abs/2006.09661

[50] "File:Fl-sin-cos-nivk-s.svg - Wikimedia Commons — commons.wikimedia.org," https://commons.wikimedia.org/wiki/File:Fl-sin-cos-nivk-s.svg, [Accessed 06-08-2024].

[51] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *CoRR*, vol. abs/2003.08934, 2020. [Online]. Available: https://arxiv.org/abs/2003.08934

[52] F. Szatkowski, K. J. Piczak, P. Spurek, J. Tabor, and T. Trzciński, "Hypersound: Generating implicit neural representations of audio signals with hypernetworks," 2024. [Online]. Available: https://arxiv.org/abs/2211.01839

[53] X. Wang, J. Yamagishi, M. Todisco, H. Delgado, A. Nautsch, 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," 2020. [Online]. Available: https://arxiv.org/abs/1911.01601

[54] X. Liu, X. Wang, M. Sahidullah, J. Patino, H. Delgado, T. Kinnunen, M. Todisco, J. Yamagishi, N. Evans, A. Nautsch, and K. A. Lee, "Asvspoof 2021: Towards spoofed and deepfake speech detection in the wild," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 2507–2522, 2023.

[55] "CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92) — doi.org," https://doi.org/10.7488/ds/2645, [Accessed 08-07-2024].

[56] "Violin Plot — itl.nist.gov," https://www.itl.nist.gov/div898/software/dataplot/refman1/auxillar/violplot.htm, [Accessed 01-11-2024].

[57] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowledge and Information Systems*, vol. 41, pp. 647–665, 12 2013.

Acknowledgments