VoiceRadar: Voice Deepfake Detection using Micro-Frequency and Compositional Analysis

Kavita Kumari†, Maryam Abbasihafshejani‡, Alessandro Pegoraro†, Phillip Rieger†, Kamyar Arshi†, Murtuza Jadliwala‡ and Ahmad-Reza Sadeghi†
†*Technical University of Darmstadt,* ‡*The University of Texas at San Antonio*

***Abstract*—Recent advancements in synthetic speech generation, including text-to-speech (TTS) and voice conversion (VC) models, allow the generation of convincing synthetic voices, often referred to as audio deepfakes. These deepfakes pose a growing threat as adversaries can use them to impersonate individuals, particularly prominent figures, on social media or bypass voice authentication systems, thus having a broad societal impact. The inability of state-of-the-art verification systems to detect voice deepfakes effectively is alarming.**

We propose a novel audio deepfake detection method, VoiceRadar, that augments machine learning with physical models to approximate frequency dynamics and oscillations in audio samples. This significantly enhances detection capabilities. VoiceRadar leverages two main physical models: (i) the Doppler effect to understand frequency changes in audio samples and (ii) drumhead vibrations to decompose complex audio signals into component frequencies. VoiceRadar identifies subtle variations, or micro-frequencies, in the audio signals by applying these models. These micro-frequencies are aggregated to compute the observed frequency, capturing the unique signature of the audio. This observed frequency is integrated into the machine learning algorithm's loss function, enabling the algorithm to recognize distinct patterns that differentiate human-produced audio from AI-generated audio.

We constructed a new diverse dataset to comprehensively evaluate VoiceRadar, featuring samples from leading TTS and VC models. Our results demonstrate that VoiceRadar outperforms existing methods in accurately identifying AI-generated audio samples, showcasing its potential as a robust tool for audio deepfake detection.

I. Introduction

Modern deep learning-based generative AI has demonstrated astonishing capabilities to produce compelling and authentic voices, often referred to as *audio deepfakes* [53]. Tools for generating such audio deepfakes are publicly available and relatively easy to deploy and execute on traditional computing devices without significant technical know-how. Thus, their usage, especially for harmful purposes such as disseminating misinformation and propaganda [19], and defamation and character assassination [59], [41], is rapidly growing. A recent incident involved fraudsters who used AI-driven deep-fake software to impersonate US President Joe Biden's voice in a robocall urging people not to vote [66]. The vast amount of voice recordings that are shared online daily left individuals, organizations, politicians, and governments vulnerable to targeted attacks by phony perpetrators [36]. Thus, developing efficient and practical techniques for verifying the authenticity of audio (voice) recordings to prevent such abuse by means of audio deepfakes has become a crucial challenge [8].

Consequently, numerous techniques have been proposed in the literature to distinguish audio deepfakes from authentic human voice, which can be broadly categorized into *machine learning (ML) based approaches* [63], [7], [70], [12], [78], [52] and *deep learning (DL) based approaches* [71], [78], [9], [43], [45], [23], [13], [87], [82]. However, these methods have significant shortcomings and are not effective in determining the authenticity of audio samples. One drawback of ML-based approaches is that they require manual feature extraction and extensive pre-processing for optimal performance. This time-intensive and inconsistent nature of ML-based approaches has led to the development of advanced DL methods for deepfake detection. Although DL-based approaches eliminate manual feature extraction and the need for lengthy training, they require specific audio data transformations for algorithmic compatibility. For example, input audio data is transformed to speech log probabilities or the input audio is transferred to scatter plot images of neighboring samples. However, one of the main limitations of all existing audio deepfake detectors is their limited adaptability and generalization to detect a wide variety of deepfake audios. Bhagtani *et al.* [11] compared several recently proposed detection tools on their new dataset and demonstrated that synthetic speech detectors show significant drops in accuracy compared to their performance on training datasets. Similarly, Muller *et al.* [57] evaluated the performance of various detection tools on in- and out-of-domain test data. Their findings confirm that these existing detection tools do not generalize well to unseen data. Moreover, existing audio deepfake detection approaches show insufficient performance, with an EER of at least 6.1%, as we demonstrate in our evaluation (see Sect. V). Therefore, further research is required to address these shortcomings and to develop effective and practical audio deepfake detection techniques.

Our goals and contributions: We propose *VoiceRadar*, a versatile and, to the best of our knowledge, the first deep-

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.243389
www.ndss-symposium.org

fake audio detection technique inspired by the physical phenomenon of wave propagation, precisely the Doppler effect and drumhead vibrations, which are used to capture tiny frequency changes (or micro-frequencies) caused by small, subtle movements within an object.

The rationale behind our approach is twofold: First, by augmenting the model training process with physical principles like the Doppler effect and drumhead vibration directly, we leverage well-established knowledge to guide learning, enabling the extraction of meaningful features that represent dynamic frequency shifts in human voice. Despite advances in state-of-the-art generative tools, AI-generated audio lacks subtle yet important features that exist in the rich human voice, such as temporal variability (e.g., timing, inconsistencies in rhythm), micro-intonations, and pitch fluctuation influenced by emotions, physical effort, or conversational dynamics, and emotional nuances and contextual adaptability. These features capture the continuous variability inherent in human-generated audio, which is difficult for AI-generated audio to replicate. Consequently, AI-generated audio vibrates at lower frequency values compared to human-produced audio or does not produce subtle frequency changes or properties observed in human-generated audio. Using physical processes (mentioned above) to model micro-frequencies of a given input audio, VoiceRadar uncovers these distinct subtle properties of audio waves, thus enhancing detection accuracy. Second, feeding these features into machine learning models makes the model learn the natural distribution of frequencies present in human-generated audio and distinguish them from the more uniform distribution in AI-generated audio.

In our approach, we simulate natural sound wave propagation from a source to an observer to distinguish between human- and AI-generated audio. Unique values in the audio embedding are treated as discrete radii in a concentric circle model, where the smallest values start at the source and the largest reach the observer. Using the physical model of the Doppler effect [58], we determine the frequency shift as sound moves from the source (human or AI) to the observer (VoiceRadar). We then use the physical model of drumhead vibrations [65], [26] to model different micro-frequencies of each radius (each unique value in the audio embedding) in a concentric circle model. This captures the audio's rhythmic beats, subtle variations, and nuances, which differ for human- and AI-generated audio. In essence, the source sends the audio waves modulated with various frequencies (translation, vibrational, and rotational) toward the observer, who perceives the obtained waves with a modified frequency.

This observed frequency information is integrated into our machine learning model as a regularization term, helping it to learn natural audio patterns and distinguish them from synthetic ones. By embedding physical knowledge into the training process, our approach effectively bridges the gap between human-like audio dynamics and AI-generated patterns. This approach is rooted in incorporating prior physical knowledge into machine learning models [46], [50], which is crucial for their effectiveness.

In very recent work, Kumari *et al.* [40] utilized the principles of the Doppler effect and drumhead vibration to design a ChatGPT detector, called DEMASQ tailored to distinguish between ChatGPT-generated text and human-written content. Their model cannot be employed in the audio domain, as audio is a continuous, multidimensional waveform involving time, frequency, and amplitude. Consequently, DEMASQ was limited to approximating the Doppler effect to capture only the translational biases in text generation, while our approach captures the translational, rotational, and vibrational biases. This difference in input complexity significantly influences the methodologies used in the two approaches. Thus, we take inspiration from this work and extend it substantially to utilize the physical models of the Doppler effect and drumhead vibration to integrate micro-frequencies which helps to classify the deepfake audio efficiently.

To evaluate our proposed VoiceRadar technique and to provide a benchmark dataset for future work, we created a dataset incorporating the latest Generative AI tools for speech synthesis. The dataset consists of more than 100 000 different audio samples, generated using 8 text-to-speech (TTS) models [89], [4], [49], [2], [37], [3], [10] and 4 Voice Conversion (VC) [16], [61], [44], [5] frameworks.

Our contributions in this paper can be summarized as follows:

- We present the design and implementation of VoiceRadar, a novel detection technique to accurately distinguish between audio deepfakes and human-generated audio.
- Our approach approximates the physical models of the Doppler effect and drumhead vibrations to capture inherent audio signal variations. By integrating the frequency signatures of audio waves' translation, vibration, and rotation frequencies, we model the observed frequency within the Doppler effect framework. This captures translational changes, rhythmic patterns, and subtle nuances within the audio signal. The observed frequency is then used in the loss function of a supervised learning algorithm to classify audio signals accurately, as shown in Sect. IV.
- We create a comprehensive benchmark dataset comprising of over 500 000 audio samples, generated using state-of-the-art TTS generators and VC modules, as outlined in Sect. V-B.
- We extensively evaluate VoiceRadar using this new benchmark dataset, showing the robustness and effectiveness of our approach in detecting audio deepfakes. We also compare VoiceRadar with existing state-of-the-art deepfake detection frameworks and demonstrate its superior detection performance, as shown in Section V.

II. Background and Intuition

This section provides an overview of the essential background knowledge needed to understand the design of VoiceRadar. We will explain the Doppler effect and drumhead vibrations and how their approximate physical models are used to analyze micro-frequencies in audio signals.

Drumhead vibrations give us a model for how to break down complex audio signals into their component micro-frequencies. These micro-frequencies are then aggregated to compute the observed frequency by employing the physical model of the Doppler effect. The Doppler effect helps us understand how the frequency of a wave changes relative to an observer when the source of the wave is moving.

Feeding the extracted observed frequency into the training of a machine-learning model helps the model to learn the natural distribution of frequencies present in human-generated audio and distinguish them from the more uniform distribution in AI-generated audio.

A. Doppler Effect

The Doppler Effect [58] is a fundamental phenomenon observed in wave physics, occurring when there is relative motion between a wave source and an observer. This effect applies to sound and light waves and to all types of waves. It manifests as a change in the wave frequency as perceived by the observer. When a wave source moves towards an observer, there is an apparent increase in frequency, leading to a higher perceived pitch in sound waves or a shift towards the blue end of the spectrum in light waves. Analogously, the frequency decreases when the source moves away from the observer, resulting in a lower perceived pitch or a shift toward the red end of the spectrum. Formally, the relationship between the observed frequency (f_o) and the emitted frequency (f_s) is described by the Doppler equation:

$$f_o = \left(\frac{c_v \pm v_o}{c_v \pm v_s} \right) f_s \tag{1}$$

Here, c_v represents the propagation speed of waves in the medium, v_o denotes the observer's speed relative to the medium, and v_s represents the speed of the source relative to the medium. Depending on the direction of motion, these speeds are added or subtracted accordingly.

As we will show later, we model the audio signal as a wave. The Doppler effect measures variations in the wave's frequencies due to the movements of the source. The source can be human or AI-generated audio, while the movement is caused by compositional biases between human and machine-generated audio. Consequently, in modeling the audio embeddings, these compositional biases will be reflected in each value of the audio embedding. Thus, to simulate the outgoing circular pattern of the sound wave propagation from a source to an observer, we treat the unique values (minimum value originating at the source and maximum value reaching the observer) in the audio embedding as discrete radii in a concentric circle model.

Alternatively, these biases can be interpreted as different rhythmic beats of the audio or subtle variations or nuances in the audio wave. When the audio waves reach the observer (VoiceRadar, in our case), the source's movement causes the waves' frequency to change. The frequency that VoiceRadar observes differs from the one originally emitted. This frequency has been altered by the Doppler Effect.

In other words, the observed frequency of the audio wave (at VoiceRadar) differs for a human-generated and an AI-generated audio wave because human and AI voices have distinct characteristics, leading to different modulations in the emitted audio waves. In the next section, we study these modulations using the Drumhead vibrations model, which breaks down complex audio signals into their component frequencies.

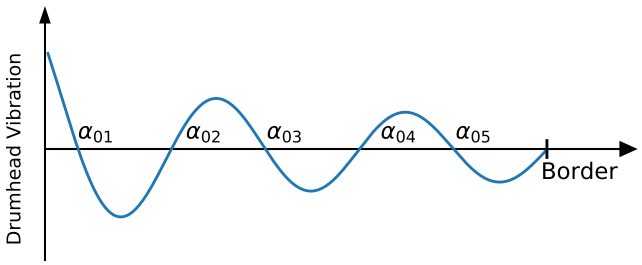
B. Drumhead vibrations to analyze audio signals

A drum skin, when hit, vibrates and creates sound. These vibrations happen in patterns across the drum skin, known as drumhead vibrations. The vibrations of an idealized drumhead can be comprehended by examining the behavior of a two-dimensional elastic membrane under tension [65], [26].

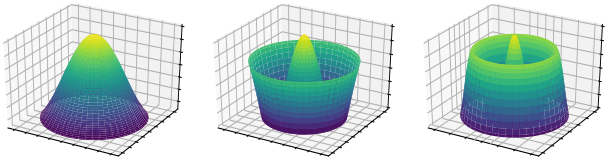
This membrane, conceptualized as a circular surface with uniform thickness connected to a rigid frame, is used to characterize the wave's motion. The membrane can hold and vibrate at certain frequencies, like a musical instrument playing specific notes. This is called resonance. The movement of the membrane's surface creates patterns called standing waves or normal modes, the specific patterns in which the membrane can vibrate. For example, the simplest and lowest frequency vibration pattern is known as the *fundamental mode*, like the basic sound to hear when hitting the middle of the drum.

The membrane possesses infinite normal modes (each characterized by distinct vibrational patterns) modeled using two quantum numbers: *circular mode* n and *angular mode* m. The n mode is used to characterize how the wave travels from the drum's center to its outer boundary in a circular fashion. For example, $n = 1$ mode represents a single wave at the drum's center going up and down. More precisely, it has a single crest (a point where the drumhead is at its maximum displacement in the positive direction, i.e., upward) or trough (a point where the drumhead is at its maximum displacement in the negative direction, i.e., downward). For $n > 1$, the number of crests or troughs increases as the waves travel to the drum's outer boundary. If $m = 1$, the membrane can be pictured as divided into two halves, with one crest or trough on each side. These halves move up and down alternately. For $m > 1$, the membrane keeps diving into segments; each segment has one crest or trough moving up and down alternately. The higher the circular/angular mode numbers, the more complex the vibration pattern becomes.

To understand and describe the vibrations of a drumhead, several mathematical concepts and methods are used: a two-dimensional wave equation, which is the solution to a partial differential equation that models how waves propagate across a surface over time, is represented by $u(x, y, t)$ where (x, y) denotes the position of the drumhead displacement at time t. Dirichlet Boundary Conditions constraints are applied to the edges of the drumhead to reflect that the edges are fixed and do not move, typically denoted by $u(x, y, t) = 0$, for all (x, y) on the boundary "a" of the drumhead. Thus, $u(x, y, t)$ describes the height of the membrane at a certain (x, y) at time t.



(a) Full Bessel Function



(b) n=1 (c) n=2 (d) n=3

Fig. 1: The impact of different modes of the zeroth Bessel function ($J_0(\lambda_{0n}r)$) affects the drumhead's vibration frequency. For example, we demonstrate the wave's shape when it travels in a drum with fixed $m = 0$ and varying $n = 1$, $n = 2$, $n = 3$.

In this context, $J_m(\lambda_{mn}r)$ are mathematical functions that describe the radial part of vibration patterns where m denotes the order of the Bessel function and r is the radial distance. This paper focuses on the asymmetric case (with $m = 0$ and n being any integer) in which the $zero^{th}$ order Bessel function $J_0(\lambda_{0n}r)$ describes the height of the drumhead as it vibrates, with respect to the stationary membrane of the drumhead. The values of $\lambda_{0n} \cdot a = \alpha_{0n}$, known as the zeros (critical points where the wave transitions to a still drum) of the Bessel function $J_0(\lambda_{0n}r)$, determine the shape of the drum, consequently influencing its vibrational frequencies, as shown in Figure 1. In the following work, we will denote the Bessel function $J_m(\lambda_{mn}r)$ as $J_m(m, n)$ throughout the rest of this paper or the $zero^{th}$ order Bessel function $J_0(\lambda_{0n}r)$ will be denoted as $J_0(0, n)$.

Analysis model for audio signals. As mentioned in Sect. I, we approximate the physical model of drumhead vibrations to model different micro-frequencies of each radius or each unique value in the audio embedding in a concentric circle model. This captures the audio's rhythmic beats, subtle variations, and nuances, which differ for human- and AI-generated audio. In essence, the source sends the audio waves modulated with various frequencies (translation, vibrational, and rotational) toward the observer, who perceives the obtained waves with a modified frequency.

Specifically, we approximate drumhead vibration modes to characterize the source wave's translational, rotational, and vibrational micro-frequencies. In Sect. IV, we will describe in detail how the embeddings are modeled in a concentric circle model and how different frequencies of each embedding are modeled to characterize their compositional biases. Next, we explain how we interpret a wave's translational, rotational, and vibrational micro-frequencies.

The *Translational Frequency* is moving up (crest) and down (trough) uniformly along the axis of travel, as shown in Figure 1. In this work, we have defined the axis of travel as the x-axis (straight line between the source and the observer). When the wave travels from the source to the observer in translational motion, the critical points are assumed to cut the x-axis; thus, the y-axis coordinate is zero in $u(x, y, t)$. Hence, the waves only travel (radially) straight along the x-axis. In this work, each unique value in the audio embedding will represent the critical points of the Bessel function.

Rotational Frequency reflects the angular motion of the different segments of the audio signal. Each critical point may have rotated by a certain angle from the axis of travel. Thus, when the wave travels from the source to the observer in rotational motion, the critical points are assumed to have both the x-axis and the y-axis coordinates in $u(x, y, t)$. Hence, the radial waves travel at an angle θ from the axis of travel, i.e., the x-axis.

It also represents changes in the audio signal as it rotates around the axis of travel, such as tone or pitch variations. This can be compared to the intonation in speech, like the rising and falling pitch when asking a question ("Where are you going?").

Vibrational Frequency in audio signals corresponds to the complex oscillations of the critical points on the axis of travel. Each critical point has a constant x-axis coordinate and variating y-axis coordinate in $u(x, y, t)$. Hence, this motion adds rapid, fine changes to the wave.

In audio signals, this is exemplified by speech with a tremor or background noise, where the rapid, fine changes resemble the complex vibrational behavior of the drumhead.

In Sect. IV, we show how translational, rotational, and vibrational frequencies can be realized by approximating the Bessel function's circular mode n for each frequency while keeping the angular mode m zero. More precisely, the Bessel function will be used to calculate the micro-frequencies as well as f_s in the Doppler model (cf. Sect. II-A), being crucial for calculating the observed frequency f_o in the Doppler effect (Equation 1). This process involves computing the zeros of the Bessel function $J_0(0, n)$, which then allows us to derive the aggregated source frequency (translational plus rotational plus vibrational), as shown in Sect. IV.

C. Deep Neural Networks

A DNN can be expressed as a mathematical function ($\mathcal{F} : \mathcal{R}^n \rightarrow \mathcal{R}$) denoted as $\mathcal{F}(X; W)$, with X representing the input data samples and W denoting the network's parameters (comprising weights and biases).

The process of training a neural network involves adjusting its parameters (W) to minimize a predefined loss (or cost) function $\mathcal{L}_R$, typically achieved through optimization algorithms such as gradient descent. These algorithms iteratively update the network's parameters in a direction that minimizes the loss function, aiming to improve the network's performance on unseen data. One popular approach for training

DNNs is the Stochastic Gradient Descent (SGD) [64], which solves the following equation by iteratively updating the parameters W:

$$W_{t+1} = W_t - \eta \nabla \mathcal{L}_R(W_t)$$

where W_t represents the parameter vector at iteration t, η is the learning rate which regularises the magnitude of the parameter updates in each iteration. $\mathcal{L}_R(W_t)$ denotes the regularized loss function, where:

$$\mathcal{L}_R(W_t) = \mathcal{L}(W_t) + \gamma g(W_t)$$

$\mathcal{L}(W_t)$ is the original loss function (e.g., cross-entropy loss, mean squared error) computed on the training data, γ is the regularization parameter, and $g(W_t)$ is the regularization term, which penalizes certain properties of the model parameters W_t.

In this work, we regularize $\mathcal{L}(W_t)$ of the model function $\mathcal{F}$ with f_o to penalize it when the actual label of $\mathcal{E}(\overrightarrow{x})$ differs from the model prediction $\mathcal{F}(\mathcal{E}(\overrightarrow{x}))$, where $\mathcal{E}(\overrightarrow{x})$ is the embedding of input audio sample $\overrightarrow{x}$. In this work, we handle the transformed embedding vector $\mathcal{E}(\overrightarrow{x})$ as the input to the model function $\mathcal{F}$.

More details are discussed in Sect. IV.

III. Adversary Model

In our model, the adversary $\mathcal{A}$ aims to generate an audio deepfake (or fake voice) that resembles the voice of a victim. There are different reasons for developing such deep audio fakes, including using them to spread fake news, e.g., for political campaigns or fraudulent actions such as phishing attacks. For generating the (fake) audio, $\mathcal{A}$ can use a synthetic voice generation tool trained with the (real) voice of the victim. The input to such a tool can either be written text (in the case text-to-speech or TTS tools) [89], [5], [4], [49], [37], [10], [3] or an audio sample of a natural person speaking some text (in case of voice conversion or VC tools) [16], [61], [44], [5].

$\mathcal{A}$'s goals are two-fold. First, it must ensure that the generated audio is consistent with the given text/audio input. Therefore, the (output) spoken text must be related to the given text. Secondly, the generated voice needs to sound as natural as possible and indistinguishable from the actual (real) voice of the victim. We assume that the adversary is aware of any possibly deployed defense. However, especially for trained detectors, such as DNN-based detectors, we assume the adversary only has access to the used algorithm but not the specific trained parameters of the DNN. To conduct the adaptive attacks, $\mathcal{A}$ can utilize the knowledge of VoiceRadar to compute the f_o (observed frequency of the radar) and map it to create adversarial audio samples. We consider an advanced adversary with the working knowledge of VoiceRadar. However, we assume that $\mathcal{A}$ does not know the trained parameters of VoiceRadar, as it does not have access to the used training data. Notably, $\mathcal{A}$ can use a separate dataset to train its model (shadow model), following the exact training algorithm of VoiceRadar to approximate the used detector (see Sect. V-I).

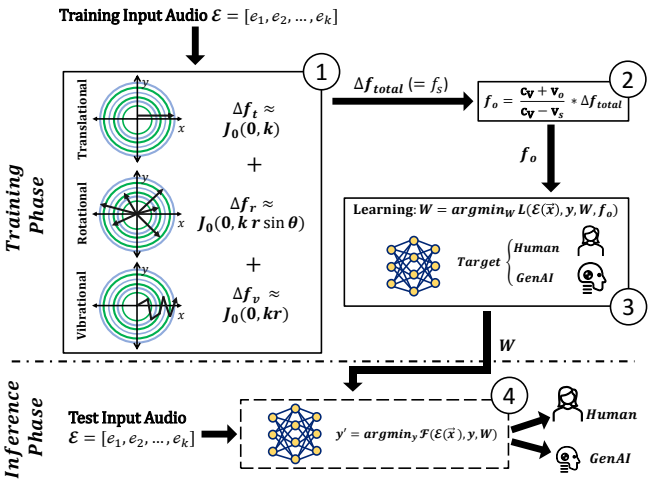


Fig. 2: Design of VoiceRadar.

On the other hand, we do not make any assumptions about specific methods used for generating the audio. In particular, we do not make any assumptions about the availability of additional knowledge about the victim to the detection tool, such as comparison data of the imitated voice.

IV. Approach

In this section, we provide a high-level description and the details of our approach, particularly on how we approximate the physical model of the Doppler effect and the drumhead vibrations to design an effective framework to detect deepfake audio. The system comprises four main modules: (1) Frequency Distribution Analysis, (2) Wave-Based Analysis, (3) Physics-Augmented ML Training, and (4) Inference Phase. Figure 2 shows the working of our scheme.

A. High-level Description

In our system model, VoiceRadar takes the role of an observer of emitted waves from a source that can be human or AI. After observing a wave, VoiceRadar should distinguish effectively whether the audio source is a human or AI. To realize this idea, we first conduct a *frequency distribution analysis* (Step 1) to determine the unique characteristics or signature of the audio waves. We deploy a physics-augmented approach to approximate the physical model of drumhead vibrations and the Doppler effect for audio waves (cf. Section II). Here, given the input audio sample $\overrightarrow{x}$, we handle the transformed embedding $\mathcal{E}(\overrightarrow{x}) := [e_1, e_2, \ldots, e_k, \ldots, e_l]$ vector of the audio sample as waves. The intuition is that the distinct values (we ignore repetitive values) in embedding $\mathcal{E}(\overrightarrow{x})$ are viewed as positions along the x-axis (assumed axis of travel) akin to a wave's movement. Alternatively, they are assumed as the zeroes (or critical points) (α_{0n}) of the Bessel function $J_0(0, n)$ (Section II-B) which can cut the x-axis (translation motion), rotate from the x-axis (rotational motion), or oscillate along x-axis (vibrational motion).

Fig. 3 shows the modeling of three types of motions by different interpretations of the unique values in $\mathcal{E}(\overrightarrow{x})$

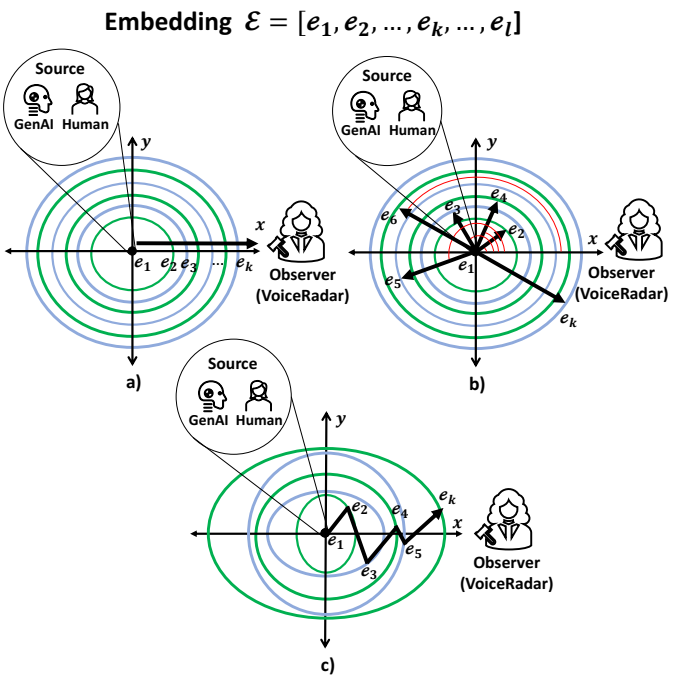


Fig. 3: The computation of the circular mode (n) to determine the frequency with which the drumhead vibrates. k represent the number of unique values in $\mathcal{E}$. The red lines in b) represent the angles drawn by the rotation.

in a concentric circle model. Thus, we have identified the following key micro-frequency components of the audio wave (cf. Sect. II): i) *Translational Frequency* is shown by the straight line path along the x-axis in Figure 3a), ii) *Rotational Frequency* based on the rotation of the values $e_i \in \mathcal{E}(\overrightarrow{x})$ from the x-axis in Figure 3b), where the red lines represent the rotation angels of the data points, and iii) *Vibrational Frequency* based on the oscillations of data points along x-axis.

Next, we conduct a *wave-based analysis* (Step 2) where we approximate the physical model of the Doppler effect to compute the frequency that the observer VoiceRadar measures, i.e., the frequency f_o in Equation 1, as described in Sect. II. We assume the velocity of the audio sample v_s is based on the variance of the embedding string, where we presume VoiceRadar being stationary $v_o = 0$. Thus, we utilize all the variables detailed above and aggregate the micro-frequencies of these waves to compute f_o. Consequently, this allows us to capture the unique "shape" of the audio signals.

Next, we run a *physics-augmented ML training* (Step 3) where we incorporate the observed frequency f_o into the classification model by constraining the training process to integrate this prior physics information in its loss function. This enhancement establishes a connection between the input audio signal and its output label (0 for human, 1 for AI-generated). We can effectively identify features that distinguish human audio from AI-generated audio by optimizing this loss function. Unlike existing methods, our approach adapts to the evolving nature of text-to-speech (TTS) and voice conversion (VC) technologies.

Algorithm 1 Drumhead Frequency Computation

1: **Input: Number of angular mode** (0)**, Number of circular mode** (k)
2: **Output: Drumhead Frequency**, f_s
3: Initialize $m \leftarrow 0$ ▷ Diametric nodes
4: Initialize $n \leftarrow k$ ▷ Circular nodes
5: Compute fundamental frequency $F_{J_0}(0,1)$
6: Compute $F_{J_0}(0,k)$
7: Calculate Drumhead Frequency: $f_s \leftarrow \frac{F_{J_0}(0,k)}{F_{J_0}(0,1)}$
8: **return** f_s

Lastly, we do *inference* (Step 4) on the trained model $\mathcal{F}$ to differentiate deepfakes from original audio samples effectively.

B. Design structure

In the following, we present the details of VoiceRadar's individual components.

Embedding Generation: To extract high-level speech representations from raw audio samples, we utilize the pre-trained HuBERT model [24]. HuBERT, which stands for Hidden-Unit BERT, is similar to the BERT model [18] used in Natural Language Processing (NLP). While BERT is based on the Transformer architecture [79], HuBERT is based on the Wav2Vec architecture [6] and is widely used for speech representation learning and processing [55], [69], [14], [42], [84], [15], [54], [17].

HuBERT's strength lies in its ability to learn from raw audio data without relying on manually engineered features like Mel-Frequency, Pitch, and Spectral Features. Instead, it uses convolutional layers and a self-supervised learning approach to extract distinctive features from the audio. This process enables HuBERT to learn rich and robust speech representations that are highly effective for various speech-processing tasks.

Frequency Distribution Analysis: Next, we develop a framework to configure the micro-motions of the input audio embedding $\mathcal{E}(\overrightarrow{x})$, differentiating human-produced audio from deepfakes. We interpret the source's translational frequency (Δf_t), the rotational frequency (Δf_r), and the vibrational frequency (Δf_v) utilizing the Bessel function $J_0(\lambda_{0n} r)$ (Section II-B) and it's value of n, as detailed below.

Translation motion: We determine the translational frequency Δf_t associated with the given audio embedding, $\mathcal{E}(\overrightarrow{x})$, when it is assumed that $\mathcal{E}(\overrightarrow{x})$ travels along the x-axis (Figure 3a). That is, the distinct (critical) points of $\mathcal{E}(\overrightarrow{x})$ map to different points on the x-axis, with no $e_i \in \mathcal{E}(\overrightarrow{x})$ having the y-axis coordinates. Note that $\mathcal{E}(\overrightarrow{x})$ can contain duplicate values. Hence, our initial step involves computing the count of only distinct values in $\mathcal{E}(\overrightarrow{x})$, as well as the value of n. The reason is that we do not want to consider multiple circles with the same radius. If we consider the duplicate values, it will result in redundant circles or information that needs to be transmitted to VoiceRadar.

The translational frequency is determined by calculating the circular mode n used in $J_0(0,n)$ as follows: We assume each value in $\mathcal{E}(\overrightarrow{x})$ as the radius of a concentric circle, with the center of each circle at the origin $(0,0)$ of the $x-y$ coordinate system. Next, we utilize the value of n to determine Δf_t of $\mathcal{E}(\overrightarrow{x})$. One critical factor to consider is that the frequency at mode $m=0,n$ is expressed in terms of the roots of the Bessel function $J_0(0,n)$, labeled as $F_{J_0}(0,n)$. As the specific values that are needed to compute these frequencies (c_d and a in [30]) are unknown, we normalize the calculation of f_s with the fundamental frequency at $J_0(0,1)$. This normalization is done by computing $f_s = \frac{F_{J_0}(0,n)}{F_{J_0}(0,1)}$. Consequently, we only need to measure n to determine different source frequencies (translational, rotational, or vibrational). Thus, we use $(F_{J_0}(0,n)/F_{J_0}(0,1))$ to compute the relative frequency for different zeroes of $J_0(0,n)$ with respect to fundamental mode $J_0(0,1)$ frequency, using the Algorithm 1.

First, we set the number of diametric nodes (m) to zero and the number of circular nodes (n) to the number of unique values in $\mathcal{E}(\overrightarrow{x})$. This is passed to $J_0(0,1)$ to compute the Bessel function frequency. Algorithm 1 starts by setting m to zero (line 3) and n to the number of unique values (k) in $\mathcal{E}(\overrightarrow{x})$ (line 4). Next, the drumhead's fundamental frequency ($F_{J_0}(0,1)$) is computed on line 5, and the frequency at mode $m=0, n=k$ ($F_{J_0}(0,n)$) is computed on line 6. Finally, the drumhead frequency is computed by dividing $F_{J_0}(0,n)$ by $F_{J_0}(0,1)$ (line 7), with the algorithm returning this frequency as the output (line 8).

Rotational motion: Here, our motivation is to determine the rotational frequency (Δf_r) associated with each audio embedding, $\mathcal{E}(\overrightarrow{x})$. The unique (critical) points of $\mathcal{E}(\overrightarrow{x})$ have corresponding x-axis and y-axis coordinates, and each value in $\mathcal{E}(\overrightarrow{x})$ may be rotated by an angle θ from the x-axis, as shown in Figure 3b.

We again achieve this by computing the circular mode (n) and using it in the Bessel function $J_0(0,n)$. When considering the Doppler shift due to rotation, it's essential to understand the spatial distribution of the rotating components relative to VoiceRadar's position. Here, the spatial distribution is the radial distance between the source and the observer (the length of unique values in $\mathcal{E}(\overrightarrow{x})$. Additionally, in the case of rotation, the Doppler shift is also influenced by the tangential velocity of the rotating components, which varies with the angle θ between the direction of motion and the line of sight. Hence, we consider the spatial distribution and the angular information to determine n and thus the rotational frequency Δf_r of $\mathcal{E}(\overrightarrow{x})$.

We utilize Algorithm 1 to determine Δf_r. Thus, by utilizing $(F_{J_0}(0,n)/F_{J_0}(0,1))$, we compute the relative frequency for different zeroes of $J_0(0,n)$ with respect to the fundamental mode $J_0(0,1)$ frequency. When approximating the value of n, we use $k \cdot r \cdot \sin(\theta)$, where k is the number of unique values in $\mathcal{E}(\overrightarrow{x})$, r is the radial distance from the axis of rotation (the last value in $\mathcal{E}(\overrightarrow{x})$ sorted in ascending order), and θ is the angle of rotation. By incorporating the radial distance (scaled by k) and the angle θ into the Bessel function J_0, we capture the spatial distribution of rotation of the values of $\mathcal{E}(\overrightarrow{x})$.

We use $\sin(\theta)$ because it represents the tangential component of the velocity, which is perpendicular to the line of sight and directly affects the Doppler shift due to rotation. This is crucial in capturing the rotational motion's effect on the frequency. In summary, by using $k \cdot r \cdot \sin(\theta)$ as the argument for the Bessel function J_0, we account for the tangential velocity component, which varies with the angle θ between the direction of motion and the line of sight.

Vibrational motion: To compute the Doppler shift due to the vibrational frequency Δf_v of the values in $\mathcal{E}(\overrightarrow{x})$, we use a similar methodology as for computing the rotational frequency, with the key difference being that we only focus on the spatial distribution of the vibrating components in $\mathcal{E}(\overrightarrow{x})$. Different values in $\mathcal{E}(\overrightarrow{x})$ oscillate along the x-axis, and incorporate it into the Bessel function $J_0(0,n)$. It is essential to factor in the spatial arrangement of the vibrating elements when evaluating the Doppler shift resulting from vibration. The following steps outline the process to determine Δf_v and n.

We compute Δf_v using the Drumhead vibrations, following a similar approach to Algorithm 1. For vibration, the radial distance r from the vibration center (which is the origin $(0,0)$ and the number of unique values k in $\mathcal{E}(\overrightarrow{x})$ is used as the argument for the Bessel function $J_0(0,n)$. Thus, n becomes $k \cdot r$, where k is the number of unique values and r represents the radial distance. This formulation accounts for the spatial distribution of vibrating components relative to VoiceRadar and how their velocities contribute to the Doppler shift observed by VoiceRadar.

Wave-Based Analysis: Next, we utilize the aggregated frequency $\Delta f_{total}(=f_s) = \Delta f_t + \Delta f_r + \Delta f_v$ as the audio embedding's frequency in the Doppler effect formulation (defined in Sect. II) to compute the observed frequency of VoiceRadar. To obtain the respective variables in Equation 1, we perform the following computations:

$$v_o = 0$$

$$v_s = y \cdot abs(var(\mathcal{E}(\overrightarrow{x})))$$

$$c_v = var(\mathcal{E}(\overrightarrow{x})) \cdot F_{J_0}(0,1)$$

The reasoning for determining the above values for the velocities of the audio embedding (v_s) and VoiceRadar (v_o) is as follows: Considering frequency indicates the rate of change of a specific value within a given timeframe. Therefore, to compute the audio embedding's velocity, we consider the variation of the input audio embedding multiplied by its label. Thus, for the source's audio embedding, the velocity (v_s) is computed as $y \cdot abs(var(\mathcal{E}(\overrightarrow{x})))$, where y is the classification label (0 or 1) assigned to input audio embedding. Since we modeled VoiceRadar as a stationary body, we assume it moves with a constant velocity of 0. Generally, in the Doppler effect, the speed of the medium, denoted as c_v, is constant (either the speed of sound or light). Hence, in this paper, we integrate a constant of fundamental frequency $F_{J_0}(0,1)$ by multiplying

it with the variance of $\mathcal{E}(\overrightarrow{x})$. This work assumes that the medium represents a space where deepfake tries to imitate content similar to human-produced audio.

Physics-Augmented ML Training: To implement the model, we employ a neural network architecture consisting of six fully connected layers with Rectified Linear Unit (ReLU) activations. The sizes of the layers are set to 512, 256, 128, 64, 32, and 1, respectively. For the training, we used the Adam optimizer with varying learning rates and training epochs to prevent the overfitting problem. A batch size of 64 is used.

In the following, we define and utilize the observed frequency of VoiceRadar to account for the relationship between $(\mathcal{E}(\overrightarrow{x}), y)$ using the principles of the Doppler effect. We define the loss of this model consisting of the Binary Cross Entropy (BCE) loss and the regularized term f_o as:

$$\text{loss} = \text{BCE} + 0.6 \cdot (f_o(\overrightarrow{x}, y) - (f_o(\overrightarrow{x}, \mathcal{F}(\mathcal{E}(\overrightarrow{x}))))) \quad (2)$$

where, $f_o(\overrightarrow{x}, y)$ represents the observed frequency for the true label y of the audio embedding $\mathcal{E}(\overrightarrow{x})$ and $f_o(\overrightarrow{x}, \mathcal{F}(\mathcal{E}(\overrightarrow{x})))$ represents the observed frequency for the output label $\mathcal{F}(\mathcal{E}(\overrightarrow{x}))$ of the audio embedding $\mathcal{E}(\overrightarrow{x})$, where $\mathcal{F}$ is the classification model (Sect. II-C). Thus, we constrain the loss function by the difference between the observed frequency of the true label $f_o(\overrightarrow{x}, y)$, and the observed frequency computed using the label output by the model, $f_o(\overrightarrow{x}, \mathcal{F}(\mathcal{E}(\overrightarrow{x})))$. The hyperparameter for the regularizing term was empirically determined to 0.6 to ensure convergence of the loss.

Inference Phase: Lastly, we do inference on the trained model $\mathcal{F}$ to differentiate deepfakes from original audio samples effectively.

V. Evaluation

In this section we empirically demonstrate the efficacy of VoiceRadar compared to the state-of-the-art detection mechanisms for each TTS and VC framework. Further, we show how the frequency values vary for the classification label of 1 (machine-generated audio) and the classification label of 0 (human-produced audio). Next, we showcase the generalizability of VoiceRadar across the TTS and VC. Lastly, we demonstrate the efficacy of our VoiceRadar against the adaptive attacks.

A. Experimental Setup

Our experiments are conducted using the PyTorch [1] framework on a server equipped with 4 NVIDIA RTX 8000 (each with 48GB memory), an AMD EPYC 7742, and 1024 GB of main memory.

Due to advancements in voice deepfake generation, existing datasets do not include voices generated with the latest state-of-the-art voice generators. To ensure a realistic and practical evaluation, we, therefore, leverage a new dataset that consists of more than 500 000 deepfake samples created by 12 of the most recent generation approaches [2], [3], [4], [5], [10], [16], [37], [44], [49], [61], [80], [89]. To ease the comparison with existing literature, we repeated the comparison using well-established datasets [86], [83], [56]. *ASVspoof2019* [86] and *ASVspoof2021* [83] are benchmark datasets widely used in existing work on deepfake detectors [22], [28], [48], [73], [76], [74], [76], [34], [72], [51], [25], [47], created for the Automatic Speaker Verification (ASV) spoofing challenge series. These datasets contain a wide range of spoofing attacks generated using various voice conversion and text-to-speech technologies, based on the VCTK dataset, which contains data from 107 different speakers, to test the robustness of ASV systems. Additionally, Müller *et al.* created and published an in-the-wild deepfake dataset consisting of 37.9 hours of audio clips, with 17.2 hours being fake and 20.7 hours real, to evaluate their models on realistic, unseen data [56].

TABLE I: Dataset statistics for TTS and VC approaches.

	Generation Approach	#of samples
TTS	VALL-E-X [89]	32000
	SpeechT5 TTS [5]	32000
	Bark [4]	32000
	StyleTTS2 [49]	32000
	Jenny [2]	32040
	Vits [37]	32040
	XTTS [3]	32040
	Tortoise [10]	32000
VC	DiffHierVC [16]	145465
	DiffVC [61]	145465
	HierSpeech++ [44]	145483
	SpeechT5 [5]	145483
	Ground Truth [80]	16943

B. Dataset Generation

Table I outlines the high-level statistics for our dataset generated using different text-to-speech (TTS) and voice conversion (VC) approaches and for the ground truth samples in the dataset. Compared to classical and established benchmark datasets that we also evaluate in Sect. V-E, our dataset is created using the most recent audio generation tools. Thus, it allows a more realistic evaluation on more challenging samples, as the comparison of existing detectors shows (see Sect. V-D). The ground truth samples within the dataset have been generated using the VCTK dataset [80], composed of 109 native speakers of English with various accents, each reading around 400 phrases. The selection of the speaker has been made based on the characteristics of the speakers, e.g., age, gender, accent, and region of provenance. Out of the 109 speakers, 20 speakers were selected for training and further 20 speakers were selected for testing, such that the distribution of gender, ages, and regions replicates the distribution of the whole dataset, i.e., resulting in 10 male and 10 female speakers for each dataset split. We generated the dataset's synthetic (or fake) voice samples using the latest TTS tools and VC frameworks [89], [4], [49], [2], [37], [3], [10], [16], [61], [44], [5], as outlined below.

1) Text-To-Speech: For generating synthetic (or fake) voice samples using the TTS approach, we employedstate-of-the-art models shown in Table I, all of which were proposed since 2021. The used VCTK dataset contains audio samples for 13K unique phrases, while various sentences are spoken by multiple speakers. For computational reasons, we focused on the first

800 texts. In other words, our dataset contained synthetic voice samples generated from 800 phrases using the above TTS tools.

2) Voice Conversion: Voice Conversion (VC) approaches convert the text that is spoken by a person into another voice of a given reference sample (target voice). The VC modules we utilized for synthetic speech generation in our dataset are outlined in Table I.

To make the generation more generalized, the audio input for the VC approaches was randomly selected. For each speaker from the training/test sets, we used one sample as a reference for the target voice and randomly selected input audio samples from other speakers, which are neither part of the training nor from the test set. Thus, to ensure a realistic and general evaluation, the audio-generation is performed using entirely new audio samples that were not used during any part of the training[1].

C. Evaluation Metrics

Aligned with existing work on deepfake detection [73], [28], [22], [34], we employed the following metrics to assess and compare the effectiveness of our approach:

- *True Positive Rate (TPR)* indicates the tool's sensitivity in detecting fake audio samples. Given the total number of correctly identified fake samples (True Positive, TP) and the number of incorrectly identified fake audio samples (False Negative, FN), the TPR is given as: $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$.
- *True Negative Rate (TNR)*, also called specificity, indicates the tool's ability to correctly recognize human-voice samples. Given the total number of correctly identified non-fake samples that are plain recordings of a human voice (True Negative, TN) and the non-fake samples that are incorrectly classified as fake (False Positive, FP), the FPR is given as: $\text{TNR} = \frac{\text{TN}}{\text{TN+FP}}$.
- *Equal Error Rate (EER)* is a combination of the False Positive Rate ($\text{FPR} = 1 - \text{TNR}$) as well as the False Negative Rate ($\text{FNR} = 1 - \text{TPR}$). Given the predicted scores of a detection tool, the predicted labels are obtained using a classification threshold. Therefore, FNR and FPR depend on the classification threshold choice. The EER threshold is determined where both FPR and FNR are equal. The EER indicates the FNR and FPR using this optimal threshold where both error rates (FPR and FNR) are equal. Thus, the system makes an equal number of errors in accepting fake audio and rejecting original human-produced audio.
- *F1-Score:* The F1-score considers both precision (the ratio of TP to the total number of positive predictions) and recall (the ratio of TP to the total number of actual positive instances) in its calculation, providing a balanced measure of a classifier's performance. It can be defined in terms of TP, TN, FP, and FN as follows: $\text{F1-Score} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP+FP+FN}}$. The F1-score reaches its best value at 1 and its worst at 0.

[1]To ensure a comprehensive evaluation, in Sect. V-I we evaluate VoiceRadar's ability to detect samples when the generator was trained on audio-samples for the target voice.

TABLE II: Comparison of VoiceRadar with existing deepfake detection approaches for text-to-speech (TTS) generated fake samples.

Approach	EER	TPR	TNR	F1-Score
RawGAT-ST [73]	41.7	58.4	58.2	0.727
AASIST [28]	51.9	48.2	48.0	0.638
Raw PC-DARTS [22]	44.2	52.6	58.9	0.680
wav2vec 2.0 [76]	6.1	93.9	93.9	0.965
Whisper Features [34]	37.2	89.1	36.6	0.94
VoiceRadar	**0.45**	**99.57**	**97.49**	**0.99**

TABLE III: Comparison of VoiceRadar with existing deepfake detection approaches for Voice Conversion (VC) generated fake samples.

Approach	EER	TPR	TNR	F1-Score
RawGAT-ST [73]	48.3	51.7	51.8	0.679
AASIST [28]	49.7	50.3	50.3	0.667
Raw PC-DARTS [22]	43.1	56.7	57.1	0.723
wav2vec 2.0 [76]	20.5	79.1	79.8	0.881
Whisper Features [34]	21.4	78.4	78.7	0.876
VoiceRadar	**1.6**	**99.9**	**91.8**	**0.99**

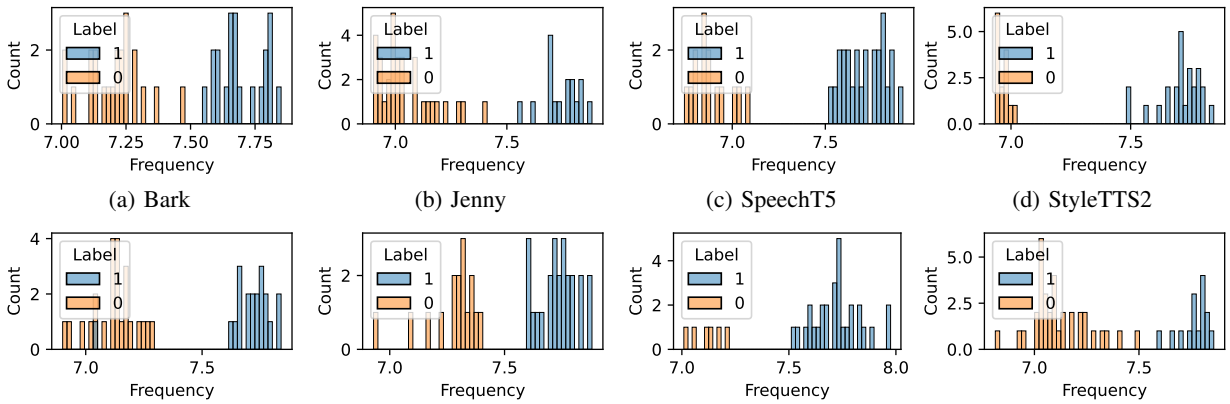
D. Baseline Evaluation

We compare our results with those of the following state-of-the-art and open-source detection tools, which have provided their checkpoints. We run these tools on our dataset (as described above in Section V-B).

- **RawGAT-ST-anti-spoofing** [73] is a spectro-temporal graph attention network (GAT) that fuses spectral and temporal sub-graphs, employing a graph pooling strategy to enhance accuracy.
- **AASIST** [28] builds upon RawGAT-ST and enhances it by integrating two heterogeneous graphs, spectral and temporal, using a novel layer called the heterogeneous stacking graph attention layer (HS-GAL). This system also employs max graph operation (MGO) and innovative readout techniques to improve its performance.
- **Raw PC-DARTS** [22] is an end-to-end differentiable architecture search (DAS) method is employed to automatically learn the network architecture for detecting speech deepfakes and spoofing.
- **Whisper Features** [34] has significantly improved detection rates by combining Whisper with well-established front-ends (LCNN, SpecRNet, and MesoNet).
- **Wav2vec 2.0** [76] could reduce equal error rates significantly by combining wav2vec 2.0 with a new self-attentive aggregation layer and data augmentation.

Tab. II and Tab. III show the results of existing deepfake detection approaches for TTS tools and VC modules, respectively. It can be observed that VoiceRadar outperforms all these existing detection frameworks in accurately identifying the deepfakes generated when testing the existing detectors and VoiceRadar against the combined dataset from TTS tools and VC modules.

For the TTS tools, VoiceRadar obtained an EER, TPR, TNR, and F1-score of 0.45, 99.57%, 97.49%, and 0.99, respectively,



(a) Bark (b) Jenny (c) SpeechT5 (d) StyleTTS2

(e) VALL-E-X (f) Tortoise (g) Vits (h) XTTS

Fig. 4: Illustration of the distribution of observed frequency f_o for different TTS generation approaches for deep-fake generated samples (label 1) and natural samples (label 0).

TABLE IV: Comparison for state-of-the-art detection tools of performance reported in the paper, the performance we reproduced for respective dataset and VoiceRadar on the dataset.

Detector	Dataset	Reported EER	Measured EER	VoiceRadar EER
RawGAT-ST [73]	ASVspoof 2019	1.06	1.05	0.10
AASIST [28]	ASVspoof 2019	0.83	0.83	0.10
Raw PC-DARTS [22]	ASVspoof 2019	2.10	2.04	0.10
wav2vec 2.0 [76]	ASVspoof 2021	0.82	0.82	0.06
Whisper Features [34]	DeepFake In-The-Wild	26.72	26.72	0.0
Channel Gated Res2Net [48]	ASVspoof 2019	1.78	1.78	0.10

compared to the lowest EER of 6.1, maximum TPR of 93.9%, maximum TNR of 93.9%, and the highest F1-Score of 0.965 obtained by the existing detection frameworks (all by wav2vec 2.0 in this case). We obtained similar results for the VC modules where VoiceRadar outperformed existing detection frameworks, as demonstrated in Tab. III. In summary, our results show that VoiceRadar performs much better compared to these existing detection systems.

E. Benchmark Comparison

The dataset that was introduced in Sect. V-B consists of audio samples generated using most recent deepfake tools. Notably, existing detector were mostly unable to detect these audio samples. To allow a more comprehensive comparison of VoiceRadar with existing detectors, we compare VoiceRadar with each detection approach using the dataset that was used in the respective publication. Tab. IV shows the performance of each evaluated detection approach, the performance that the authors of the detection approach reported, the performance that we reproduced, as well as VoiceRadar's performance on the respective dataset.

F. TTS Evaluation

In this section, we evaluate the efficacy of VoiceRadar utilizing the samples collected from the TTS tools, VC modules, and the ground truth dataset. First, we illustrate the distribution of the observed frequency values (f_o) by

TABLE V: Effectiveness of VoiceRadar for the individual text-to-speach (TTS) and voice conversion (VC) approaches.

	Generation Approach	EER	TPR	TNR	F1-Score
TTS	VALL-E-X [89]	0.0071	99.79%	99.12%	0.99
	SpeechT5 TTS [5]	0.0004	99.60%	98.51%	0.99
	Bark [4]	0.00096	99.56%	98.46%	0.99
	StyleTTS2 [49]	0.00	99.98%	98.55%	0.99
	Jenny [2]	0.0002	99.90%	98.60%	0.99
	Vits [37]	0.0002	99.96%	98.61%	0.99
	XTTS [3]	0.0008	99.06%	98.58%	0.99
	Tortoise [10]	0.0085	98.76%	97.57%	0.99
	Combined	0.0045	99.57%	97.49%	0.99
VC	DiffHierVC [16]	0.0072	99.78%	96.72%	0.99
	DiffVC [61]	0.014	99.77%	93.51%	0.99
	HierSpeech++ [44]	0.0056	99.96%	96.81%	0.9979
	SpeechT5 [5]	0.0	100%	98.69%	0.99
	Combined	0.016	99.88%	91.80%	0.99

VoiceRadar for each TTS tool and VC tool. Then, we detail the efficacy of VoiceRadar in detecting the fake samples from each TTS tool and the combined samples of all TTS tools in terms of the EER, TPR, TNR, and F1-Score metrics. Similarly, we do the same for the VC tools.

Illustration of Frequency values: Figure 4 illustrates the distribution of the observed frequency values (f_o) obtained by VoiceRadar. In the inference phase, we stored f_o for each dataset and each label. Then, it was discerned that for label 0, i.e., for the deepfake audio embeddings, the f_o values realized were below the threshold of 7.5, and for label 1, f_o values realized were above the threshold of 7.5, except some mis-classification. This trend in the frequency values demonstrates the discriminatory power of the micro-DE, which efficiently penalized the loss function (Equation 2) to differentiate the fake samples from the original ones. In Figure 4, we just demonstrated the distribution of frequency values for a batch of 64 samples. Thus, in each plot of Figure 4, number of 1's and 0's varies.

VoiceRadar statistics for TTS: To study the efficacy of the trained model $\mathcal{F}(\mathcal{E}(\overrightarrow{x})$ in classifying the deepfake samples and the ground truth samples accurately, we compute TPR, TNR, EER, and F1-score metrics. For each TTS tool, we

individually train our model (described in Section IV) and analyze the frequency spread for the ground truth audio and TTS-generated samples. Table V details TPR, TNR, EER, and F1-Score metrics statistics. As can be seen from Table V, VoiceRadar obtains the very low value of 0.0045 EER for the combined TTS dataset. Moreover, it obtains a TPR of 99.57% and a TNR of 97.49% on the dataset that consists of all VC-generated samples. For the F1-score metric, a value of 0.99 is achieved for all the datasets. These results show that VoiceRadar effectively discerns TTS-generated deepfakes from the original audio samples.

Additionally, since we train various models for individual datasets, we generate unique detectors for diverse fields using the same foundational model by just integrating the observed frequency of VoiceRadar (f_o). Thus, our work provides a generic detector that can be deployed in any field to detect whether the audio is machine or human-produced.

G. VC Evaluation

In this section, we evaluate the efficacy of VoiceRadar utilizing the samples collected from different VC algorithms and the ground truth dataset. First, we illustrate the distribution of the observed frequency values (f_o) by VoiceRadar. Then, we detail the efficacy of VoiceRadar in detecting the fake samples from each VC framework and the combination of the samples from each VC module in terms of the TPR, TNR, EER, and F1-Score metrics.

Illustration of Frequency values: Figure 5 illustrates the distribution of the observed frequency values (f_o) obtained by VoiceRadar. In the inference phase, we stored f_o for each dataset for each label. For VC also, it was discerned that for label 0, i.e., for the deepfake audio samples, the f_o values realized were below the threshold of 7.5, and for label 1, f_o values realized were above the threshold of 7.5, except few misclassifications. These results highlight the micro-DE discriminatory performance on different datasets, allowing for a comparative analysis of their ability to identify the fake samples from the original ones accurately. Hence, f_o values effectively efficiently penalized the loss function (Equation 2) to increase the classification accuracy (as shown in Table V). Figure 5 demonstrates the distribution of frequency values for a batch of 64 samples. Thus, in each plot of Figure 4, the number of 1's and 0's varies.

VoiceRadar statistics for VC: We individually train our model, as outlined in Section IV, for each VC framework. Utilizing the distribution of frequencies in both ground truth and VC-generated samples, we effectively penalized $\mathcal{F}(\mathcal{E}(\overrightarrow{x}))$ for incorrect predictions. As Table V shows, VoiceRadar obtains a very low value of 0.016 EER for the combined VC dataset. For TPR and TNR, it obtains 99.88% and 91.8%, respectively. Again, for the F1-score metric, a value of 0.99 is observed for all the datasets. Hence, VoiceRadar effectively discerns VC-generated deepfakes from the original audio samples.

TABLE VI: Cross-evaluation of VoiceRadar for TTS datasets.

Trained	Tested	EER	TPR	TNR	F1-Score
Bark, Jenny, SpeechT5, StyleTTS2	Tortoise, VALL-E-X, XTTS, Vits	0.0006	99.68%	98.29%	0.99
StyleTTS2, Tortoise, VALL-E-X, Vits	Bark, Jenny, SpeechT5, XTTS	0.0069	99.64%	96.89%	0.99

TABLE VII: Cross-evaluation of VoiceRadar for VC datasets.

Trained	Tested	EER	TPR	TNR	F1-Score
DiffVC, DiffHierVC	HierSpeech++, SpeechT5	0.019	99.80%	89.73%	0.99
HierSpeech++, SpeechT5	DiffVC, DiffHierVC	0.0043	99.97%	97.11%	0.99

H. Generalization Evaluation

This section demonstrates the generalizability capability of VoiceRadar across the TTS, VC, and the cross of TTS and VC samples.

VoiceRadar statistics for TTS: Table VI illustrates the generalization of VoiceRadar across the TTS dataset. In this experiment, we conducted two sets of experiments randomly dividing the TTS tools (thus, their dataset) into two categories, such that one category of tools was used for training and another for inference. In the first experiment, we used Bark, Jenny, SpeechT5, StyleTTS2, and ground truth datasets to train the model, and Tortoise, VALL-E-X, XTTS, and Vits were used for inference. As one can observe from Tab. VI, VoiceRadar achieved a low EER of 0.0006, a TPR of 99.68%, TNR of 98.29%, and an F1-score of 0.99. In the second experiment, we used StyleTTS2, Tortoise, VALL-E-X, Vits, and ground truth datasets for training, and Bark, Jenny, SpeechT5, XTTS for the inference. We got an EER, TPR, TNR, and F1-score of 0.0069, 99.64%, 96.89%, and 0.99, respectively. Thus, VoiceRadar efficiency is much better than that of the existing detectors (Table II) when comparing the generalizability capability of VoiceRadar across the TTS dataset.

VoiceRadar statistics for VC: Table VII illustrates the generalization of VoiceRadar across the VC dataset. Again, in this experiment, we conducted two sets of experiments randomly dividing the VC modules (thus, their dataset) into two categories: one category of tools was used for training and another for inference. In the first experiment, we used DiffVC, DiffHierVC, and ground truth datasets to train the model, and HierSpeech++ and SpeechT5 were used for inference. We got an EER, TPR, TNR, and F1-score of 0.019, 99.80%, 89.73%, and 0.99, respectively. In the second experiment, we used HierSpeech++, SpeechT5, and ground truth datasets model training, and DiffVC, and DiffHierVC for model inference. In this case, we got an EER, TPR, TNR, and F1-score of 0.0043, 99.97%, 97.11%, and 0.99, respectively. Again, VoiceRadar surpassed the existing detectors where they could only achieve a low EER of 0.21, a high TPR of 78.4%, a high TNR of 78.7%, and a maximum F1-score of 0.88.

As mentioned, we train various models for individual

datasets (and the combined), and each trained model behaves as a unique detector. Thus, these detectors can be deployed in diverse fields using the same foundational model by integrating the observed frequency of VoiceRadar (f_o) to detect whether the audio is machine or human-produced. Hence, the obtained results for the generalization demonstrate that, given the high accuracy of VoiceRadar, we can have two detectors for the TTS tools and VC frameworks to efficiently discern the deepfakes generated using either a TTS or a VC.

I. Adaptive Attacks

In this section, we evaluate the efficacy of VoiceRadar against the adaptive attacks that $\mathcal{A}$ can launch to compromise VoiceRadar. We conducted two types of adaptive attacks based on the knowledge of $\mathcal{A}$: First, a projected gradient descent (PGD) attack, in which $\mathcal{A}$ has the working knowledge of VoiceRadar. Thus, $\mathcal{A}$ can use this knowledge to generate audio-deepfake samples similar to the original ones. Second, an overfitting attack in which a model was used for the generation task, which is fine-tuned, focusing on the target voice to improve the naturalness of the imitated voice.

PGD attack: The PGD attack is an iterative method used to generate adversarial examples. It aims to find an adversarial example $\mathbf{x}^*$ close to the original input $\mathbf{x} \in X$ but causes the model to make an incorrect prediction. In this work, $\mathcal{A}$ trains a model on the combined datasets of Bark [4], Jenny [2], SpeechT5 [5], and StyleTTS2 [49] using the loss function defined in Sect. IV. Then, using this pre-trained model, $\mathcal{A}$ generates adversarial examples of the Vits [37] dataset by projecting them into the ϵ-ball around the original input. ϵ was set to 0.01 and the number of iterations to 50.

To evaluate the robustness of VoiceRadar against this attack, we tested the generated adversarial examples against our combined pre-trained model for the TTS samples. We achieved an efficacy of 100% TPR and 0.0% EER. That is, VoiceRadar was perfectly able to detect all the adversarial examples.

Overfitting attack: In the previous experiments, we used general models to create deepfakes. Neither target voice nor source text nor source voice was used for training the generator (see Sect. III). Especially considering the abuse of deepfakes for criminal purposes, it is impractical to train every victim in their model. For a comprehensive evaluation, we evaluated VoiceRadar also for deepfakes created using generators specifically fine-tuned for the target person. For these experiments, we utilized the DiffVC [61] approach. We loaded the already trained weights for the voice conversion task and continued training them with a ten times reduced learning rate for 40 epochs. The dataset consists of sample speeches from the target person extracted from different videos.

We utilized our combined pre-trained model for the TTS samples to evaluate the robustness of VoiceRadar against this attack. We achieved an efficacy of 97% TPR and 0.0% EER. That is, VoiceRadar was robust against such types of attacks as well.

VI. Security Analysis

In this section, we analyze the security of VoiceRadar in our adversary model (Section III), even in the face of an adaptive adversary with the working knowledge of VoiceRadar. To bypass VoiceRadar, the adversary $\mathcal{A}$ has to ensure that VoiceRadar cannot distinguish between deepfake audio and human-generated audio. The critical factor in VoiceRadar is the choice of the method to generate the embeddings ($\mathcal{E}(\overrightarrow{x})$). We analyzed different micro-motions (translation, rotational, and vibrational) to determine the "shape/signature" of $\mathcal{E}\overrightarrow{x}$. Hence, we assume the adversary's goal is to evade VoiceRadar by utilizing $\mathcal{E}(\overrightarrow{x})$. We identify the following four adversarial settings:

First, $\mathcal{A}$ can manipulate the translational motion of $\mathcal{E}(\overrightarrow{x})$, i.e., can modify the unique values k of $\mathcal{E}(\overrightarrow{x})$. Then, map this information to train a model by constraining its loss and creating adversarial examples. However, since k varies for each audio embedding, creating the same embeddings as we obtained using the pre-trained HuBERT model [24] (Section IV), to obtain the same model as VoiceRadar is unlikely. The reason is that $\mathcal{E}(\overrightarrow{x})$ is a large tensor filled with arbitrary values. It also depends on the type of the model used to create $\mathcal{E}(\overrightarrow{x})$. Thus, even though $\mathcal{A}$ has the working knowledge of our approach (Section III), there is less chance the adversarial examples created by $\mathcal{A}$ can prevent the detection from VoiceRadar, as demonstrated in Section V-I.

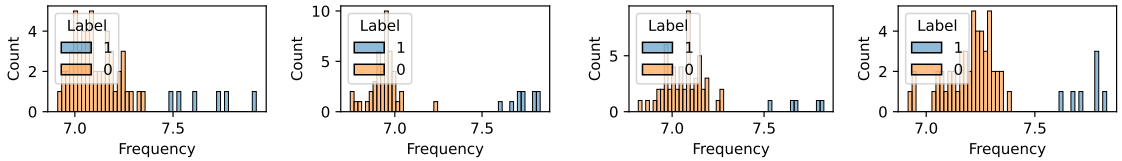
Second, $\mathcal{A}$ can manipulate the rotational motion of $\mathcal{E}(\overrightarrow{x})$, i.e., operate on the spatial (r), angular orientation (θ) of $\mathcal{E}(\overrightarrow{x})$, and again on the unique values k of $\mathcal{E}(\overrightarrow{x})$, to map this information generate adversarial examples. With the same reasoning as above, it is improbable to have the same values r and θ to obtain the same model as VoiceRadar to generate adversarial examples.

Third, $\mathcal{A}$ can manipulate the vibrational motion of $\mathcal{E}(\overrightarrow{x})$, i.e., operate only on the spatial (r) of $\mathcal{E}(\overrightarrow{x})$. Again, it's improbable to reflect the manipulation of all these variables to generate adversarial examples for the reasons detailed above.

Furthermore, an adversary may attempt to create an inconspicuous pattern that distracts VoiceRadar without being detectable by human ears. In this scenario, the adversary has two strategies. The first is a general attack, which involves fine-tuning the generator specifically for the target individual, thereby enhancing the authenticity of the generated fake samples against various detectors by improving the quality of the deep fake. The second strategy is a targeted attack, designed for the specific deployed model, which involves overlaying a pattern similar to an adversarial example to mislead VoiceRadar. Both attack strategies were implemented and evaluated as part of our extensive evaluation, showing that neither approach is capable of bypassing VoiceRadar.

VII. Related Work

This section reviews state-of-the-art deep learning-based voice synthesis tools and detection methods proposed to identify such deepfake voices.



(a) DiffVC (b) SpeechT5 (c) DiffHierVC (d) HierSpeech++

Fig. 5: Illustration of the distribution of frequency values for different VC deep-fake generated samples (label 1) and natural samples (label 0).

A. AI-Driven Voice Generation

Text-to-speech (TTS) and Voice Conversion (VC) are the two main (human) speech synthesis techniques found in the research literature. Concatenative Speech Synthesis and Statistical Parametric Speech Synthesis are two early TTS models. The former concatenates acoustic units to create synthetic speech, while the latter uses vocoders to extract and synthesize speech parameters from a training database [33]. Modern deep learning-based approaches, however, outperform these methods in terms of performance and naturalness. WaveNet[60] is designed for generating raw audio. It synthesizes audio by processing spectrograms and has been widely adopted in various tools.

End-to-end models like Tacotron2 [67], an autoregressive model, have architectures consisting of convolutional and LSTM layers. However, Tacotron2's autoregressive nature results in long output production times and thus suffers from slow inference speed. Additionally, generated speech is inaccurate. As a result, FastSpeech[62], a feed-forward network based on self-attention in the Transformer that is non-autoregressive and operates in parallel for TTS to reduce the inference speed. Meanwhile, MelGAN [39], a generative adversarial network (GAN) model, addresses the need for large datasets in voice synthesis, generating high-quality voices. Its generator uses a fully convolutional architecture with transposed convolutional layers and residual blocks. The discriminator employs a multi-scale architecture with three discriminators: one for raw audio and two for audio downsampled by factors of 2 and 4. As a result of introducing new and sophisticated deep learning and diffusion models, more powerful tools are introduced. VALL-E [81] is a zero-shot text-to-speech synthesizer with outstanding performance when faced with unseen voices without prior training. The NaturalSpeech [77] tools series also offers significant advancements in the quality of TTS tools. NaturalSpeech targets single-speaker synthesis using flow-based generative models, NaturalSpeech2[68] expands to zero-shot diverse multi-speaker scenarios and leverages latent diffusion models, and NaturalSpeech3[27] achieves human-level naturalness with multiple speakers using factorized diffusion models. The StyleTTS2[49] employs non-autoregressive modeling and adversarial training to achieve human-level TTS synthesis using large speech language models (SLMs) like WavLM.

In VC, an input speech waveform is transformed into the voice of a different speaker while maintaining the linguistic content. GANs, particularly the CycleGAN architecture, have shown proficiency in capturing the general characteristics of speech waveforms, making them highly suitable for voice conversion tasks[31], [32]. SpeechT5[5] was introduced as the first unified encoder-decoder framework for diverse spoken language processing tasks. It employs cross-modal vector quantization to effectively align acoustic and textual data. HierSpeech++[44] achieves human-level TTS and VC both on single and multi-speaker datasets, illustrating the benefits of style diffusion and adversarial training on large SLMs. Diff-HierVC[16] proposed a hierarchical voice conversion tool that utilizes diffusion-based technology to resolve the pitch inaccuracies that previous VC tools usually faced. The trend of creating human-like deepfake voice generation continues, with more powerful TTS and VC tools that surpass previous works in accuracy and performance.

B. DeepFake Voice Detection

As AI voice synthesis tools continue to improve, researchers have proposed new tools and approaches for such synthetic or deepfake voice detection. Tak et al. [74] proposed a Graph Attention model incorporating temporal attention (GAT-T) and spectral attention (GAT-S) networks for detecting speech spoofing attacks, achieving an Equal Error Rate (EER) of 4.71 and a minimum Detection Cost Function (DCF) of 4.48 for GAT-T, and an EER of approximately 0.0894 and a minimum DCF of 0.0914 for GAT-S. In another work, Tak et al. [75] developed an end-to-end electro-temporal Graph Attention Network [73] for deepfake detection, achieving a notable 1.06% EER and 0.0335 minimum t-DCF on the ASVspoof 2019 database by integrating spectral and temporal cues. Jung et al. [28] introduced AASIST, a deepfake detection tool utilizing an integrated spectro-temporal Graph Attention Network This method enabled them to achieve an EER of 0.83 and a minimum t-DCF of 0.0275. Ge et al.[22] proposed an end-to-end differentiable architecture search methodology for speech deepfake and spoofing detection, achieving a t-DCT score of 0.0517 on the ASVspoof 2019. Kawa et al.[35] introduced SpecRNet, a novel architecture inspired by RawNet2 [29] (a CNN-GRU speaker recognition system) based on spectrogram representation, which demonstrated improved EERs and performance onVoxCeleb1 dataset. Liu et al.[51] improved deepfake detection accuracy by identifying audio authenticity cues during the mono-to-stereo conversion. Sun et al. [72] analyzed vocoder artifacts and acoustic

anomalies added to voices due to voice synthesis and used a multi-task classifier to detect deepfake voices. Yadav et al. [85] introduced the Patched Spectrogram Synthetic Speech Detection Transformer (PS3DT), a synthetic speech detection system. This detector converts a time-domain speech signal into a mel-spectrogram and processes it in patches using a transformer neural network. PS3DT was evaluated using the ASVspoof2019 dataset and achieved an EER of 4.54%. As AI voice synthesis evolves, introducing new security risks, the development of additional online detection tools[38], [21], [20], [88] continues to address these challenges. Most previous works conducted their evaluations on ASVspoof2019 and ASVspoof2021. Muller *et al.* [57] performed a comparison study using various deep fake detectors. They showed that while increasing model capacity helps in-domain scenarios where the testing data is from the same dataset, it does not effectively address generalization challenges across unseen datasets. Bhagtani *et al.* [11] generated a new synthetic dataset using five diffusion model-based synthetic speech generators. They evaluated six top-performing synthetic speech detectors based on their error rates on the ASVspoof2019. Their findings indicate that these detection tools perform well on the training dataset but show poor performance and lack of generalization when tested on unseen speech samples. In this work, we provide a new dataset generated by recent Text-to-Speech and Voice Conversion tools and propose a more generalized detector.

In another work, Kumari *et al.* proposed DEMASQ, inspired by the Doppler effect and drumhead vibration principles to distinguish AI-generated from human-produced text [40]. However, the approach is restricted to detecting AI-generated text. Inspired by their approach, we significantly extend their solution by integrating micro-frequency compositional analysis to determine the deepfake audio samples.

VIII. Conclusion

This paper introduces VoiceRadar, a novel method to accurately differentiate between audio deepfakes and human-generated audio. Our approach addresses variations (micro-frequencies) in audio signals by approximating the Doppler effect and drumhead vibrations. By integrating these micro-Doppler signatures into the loss function of a supervised learning algorithm, VoiceRadar effectively classifies deepfakes. We curated a benchmark dataset with samples from the latest TTS tools and VC frameworks. Extensive evaluations demonstrate VoiceRadar's robustness and effectiveness in detecting audio deepfakes.

Acknowledgment

Research reported in this publication was partially supported by the National Science Foundation (NSF) under award number 1943351.

This research also received funding from the following organization: Deutsche Forschungsgemeinschaft (DFG) SFB-1119 CROSSING/236615297 S2, Horizon program of the European Union under grant agreements No. 101093126 (ACES) and No. 101070537 (CROSSCON) as well as the Federal Ministry of Education and Research of Germany (BMBF) within the IoTGuard project and Athene projects.

References

[1] Pytorch, 2022. https://pytorch.org.

[2] 48khz vits model of the jenny (dioco) tts dataset, 2023. https://www.kaggle.com/datasets/noml4u/tts-models--en--jenny-dioco--vits.

[3] Xtts, 2023. https://huggingface.co/spaces/coqui/xtts.

[4] Suno AI. bark: A github repository, 2023. Accessed: date-of-access.

[5] Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, et al. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. *arXiv preprint arXiv:2110.07205*, 2021.

[6] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[7] Dora M Ballesteros, Yohanna Rodriguez, and Diego Renza. A dataset of histograms of original and fake voice recordings (h-voice). *Data in brief*, 29, 2020.

[8] Dora M Ballesteros, Yohanna Rodriguez-Ortega, Diego Renza, and Gonzalo Arce. Deep4snet: deep learning for fake speech classification. *Expert Systems with Applications*, 184:115465, 2021.

[9] Emily R Bartusiak and Edward J Delp. Frequency domain-based detection of generated audio. *arXiv preprint arXiv:2205.01806*, 2022.

[10] James Betker. Better speech synthesis through scaling. *arXiv preprint arXiv:2305.07243*, 2023.

[11] Kratika Bhagtani, Amit Kumar Singh Yadav, Paolo Bestagini, and Edward J Delp. Are recent deepfake speech generators detectable? In *Proceedings of the 2024 ACM Workshop on Information Hiding and Multimedia Security*, pages 277–282, 2024.

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

[13] Steven Camacho, Dora Maria Ballesteros, and Diego Renza. Fake speech recognition using deep learning. In *Applied Computer Sciences in Engineering: 8th Workshop on Engineering Applications, WEA 2021, Medellín, Colombia, October 6–8, 2021, Proceedings 8*, pages 38–48. Springer, 2021.

[14] Xuankai Chang, Takashi Maekaku, Pengcheng Guo, Jing Shi, Yen-Ju Lu, Aswin Shanmugam Subramanian, Tianzi Wang, Shu-wen Yang, Yu Tsao, Hung-yi Lee, et al. An exploration of self-supervised pretrained representations for end-to-end speech recognition. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 228–235. IEEE, 2021.

[15] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022.

[16] Ha-Yeong Choi, Sang-Hoon Lee, and Seong-Whan Lee. Diff-hiervc: Diffusion-based hierarchical voice conversion with robust pitch generation and masked prior for zero-shot speaker adaptation. *International Speech Communication Association*, pages 2283–2287, 2023.

[17] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[19] Nicholas Diakopoulos and Deborah Johnson. Anticipating and addressing the ethical implications of deepfakes in the context of elections. *New media & society*, 23(7):2072–2098, 2021.

[20] Thien Phuc Doan, Kihun Hong, and Souhwan Jung. Gan discriminator based audio deepfake detection. In *Workshop on Security Implications of Deepfakes and Cheapfakes*, 2023.

[21] Thien-Phuc Doan, Long Nguyen-Vu, Souhwan Jung, and Kihun Hong. Bts-e: Audio deepfake detection using breathing-talking-silence encoder. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[22] Wanying Ge, Jose Patino, Massimiliano Todisco, and Nicholas Evans. Raw differentiable architecture search for speech deepfake and spoofing detection. *arXiv preprint arXiv:2107.12212*, 2021.

[23] Heinz Hofbauer and Andreas Uhl. Calculating a boundary for the significance from the equal-error rate. In *2016 International Conference on Biometrics (ICB)*, pages 1–4. IEEE, 2016.

[24] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units, 2021.

[25] Guang Hua, Andrew Beng Jin Teoh, and Haijian Zhang. Towards end-to-end synthetic speech detection. *IEEE Signal Processing Letters*, 28:1265–1269, 2021.

[26] Christopher HM Jenkins and Umesh A Korde. Membrane vibration experiments: An historical review and recent results. *Journal of Sound and Vibration*, 295(3-5):602–613, 2006.

[27] Zeqian Ju, Yuancheng Wang, Kai Shen, Xu Tan, Detai Xin, Dongchao Yang, Yanqing Liu, Yichong Leng, Kaitao Song, Siliang Tang, et al. Naturalspeech 3: Zero-shot speech synthesis with factorized codec and diffusion models. *arXiv preprint arXiv:2403.03100*, 2024.

[28] Jee-weon Jung, Hee-Soo Heo, Hemlata Tak, Hye-jin Shim, Joon Son Chung, Bong-Jin Lee, Ha-Jin Yu, and Nicholas Evans. Aasist: Audio anti-spoofing using integrated spectro-temporal graph attention networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[29] Jee-weon Jung, Seung-bin Kim, Hye-jin Shim, Ju-ho Kim, and Ha-Jin Yu. Improved rawnet with feature map scaling for text-independent speaker verification using raw waveforms. *arXiv preprint arXiv:2004.00526*, 2020.

[30] Mark Kac. Can one hear the shape of a drum? *The american mathematical monthly*, 73(4P2):1–23, 1966.

[31] Takuhiro Kaneko and Hirokazu Kameoka. Cyclegan-vc: Non-parallel voice conversion using cycle-consistent adversarial networks. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2100–2104. IEEE, 2018.

[32] Takuhiro Kaneko, Hirokazu Kameoka, Kou Tanaka, and Nobukatsu Hojo. Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[33] Navdeep Kaur and Parminder Singh. Conventional and contemporary approaches used in text to speech synthesis: A review. *Artificial Intelligence Review*, 56(7):5837–5880, 2023.

[34] Piotr Kawa, Marcin Plata, Michał Czuba, Piotr Szymański, and Piotr Syga. Improved DeepFake Detection Using Whisper Features. In *INTERSPEECH*, 2023.

[35] Piotr Kawa, Marcin Plata, and Piotr Syga. Specrnet: Towards faster and more accessible audio deepfake detection. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022.

[36] Suwajanakorn S Seitz SM Kemelmacher-Shlizerman et al. I synthesizing obama: learning lip sync from audio. *ACM Trans. Graph.(ToG)*, 36(4):95, 2017.

[37] Jaehyeon Kim, Jungil Kong, and Juhee Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech, 2021.

[38] Nikhil Valsan Kulangareth, Jaycee Kaufman, Jessica Oreskovic, and Yan Fossat. Investigation of deepfake voice detection using speech pause patterns: Algorithm development and validation. *JMIR Biomedical Engineering*, 9:e56245, 2024.

[39] Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre De Brebisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32, 2019.

[40] Kavita Kumari, Alessandro Pegoraro, Hossein Fereidooni, and Ahmad-Reza Sadeghi. Demasq: Unmasking the chatgpt wordsmith. *arXiv preprint arXiv:2311.05019*, 2023.

[41] Steven Kurutz. Welcome to scam world. https://www.nytimes.com/2024/04/21/style/scams-identity-theft.html, Apr 2024.

[42] Kushal Lakhotia, Eugene Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu-Anh Nguyen, Jade Copet, Alexei Baevski, Abdelrahman Mohamed, et al. On generative spoken language modeling from raw audio. *Transactions of the Association for Computational Linguistics*, 9:1336–1354, 2021.

[43] Mohammed Lataifeh, Ashraf Elnagar, Ismail Shahin, and Ali Bou Nassif. Arabic audio clips: Identification and discrimination of authentic cantillations from imitations. *Neurocomputing*, 418:162–177, 2020.

[44] Sang-Hoon Lee, Seung-Bin Kim, Ji-Hyun Lee, Eunwoo Song, Min-Jae Hwang, and Seong-Whan Lee. Hierspeech: Bridging the gap between text and speech by hierarchical variational inference using self-supervised representations for speech synthesis. *Advances in Neural Information Processing Systems*, 35:16624–16636, 2022.

[45] Zhenchun Lei, Yingen Yang, Changhong Liu, and Jihua Ye. Siamese convolutional neural network using gaussian probability feature for spoofing speech detection. In *Interspeech*, pages 1116–1120, 2020.

[46] Li Li, Stephan Hoyer, Ryan Pederson, Ruoxi Sun, Ekin D Cubuk, Patrick Riley, and Kieron Burke. Kohn-sham equations as regularizer: Building prior knowledge into machine-learned physics. *Physical review letters*, 126(3):036401, 2021.

[47] Xu Li, Na Li, Chao Weng, Xunying Liu, Dan Su, Dong Yu, and Helen Meng. Replay and synthetic speech detection with res2net architecture. In *ICASSP 2021-2021 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6354–6358. IEEE, 2021.

[48] Xu Li, Xixin Wu, Hui Lu, Xunying Liu, and Helen Meng. Channel-wise gated res2net: Towards robust detection of synthetic speech attacks. *arXiv preprint arXiv:2107.08803*, 2021.

[49] Yinghao Aaron Li, Cong Han, Vinay Raghavan, Gavin Mischler, and Nima Mesgarani. Styletts 2: Towards human-level text-to-speech through style diffusion and adversarial training with large speech language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[50] Frank Liu and Agniva Chowdhury. Deep learning with physics priors as generalized regularizers. *arXiv preprint arXiv:2312.08678*, 2023.

[51] Rui Liu, Jinhua Zhang, Guanglai Gao, and Haizhou Li. Betray oneself: A novel audio deepfake detection model via mono-to-stereo conversion. *arXiv preprint arXiv:2305.16353*, 2023.

[52] Tianyun Liu, Diqun Yan, Rangding Wang, Nan Yan, and Gang Chen. Identification of fake stereo audio using svm and cnn. *Information*, 12(7):263, 2021.

[53] Siwei Lyu. Deepfake detection: Current challenges and next steps. In *2020 IEEE international conference on multimedia & expo workshops (ICMEW)*, pages 1–6. IEEE, 2020.

[54] Ambuj Mehrish, Navonil Majumder, Rishabh Bharadwaj, Rada Mihalcea, and Soujanya Poria. A review of deep learning techniques for speech processing. *Information Fusion*, 99:101869, 2023.

[55] Abdelrahman Mohamed, Hung-yi Lee, Lasse Borgholt, Jakob D Havtorn, Joakim Edin, Christian Igel, Katrin Kirchhoff, Shang-Wen Li, Karen Livescu, Lars Maaløe, et al. Self-supervised speech representation learning: A review. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1179–1210, 2022.

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

[57] Nicolas M Müller, Nicholas Evans, Hemlata Tak, Philip Sperl, and Konstantin Böttinger. Harder or different? understanding generalization of audio deepfake detection. *arXiv preprint arXiv:2406.03512*, 2024.

[58] Cristian Neipp, A Hernández, JJ Rodes, Andrés Márquez, Tarsicio Beléndez, and Augusto Beléndez. An analysis of the classical doppler effect. *European journal of physics*, 24(5):497, 2003.

[59] Dawn Nici. Fcc outlaws use of ai-faked voices in robocalls. https://www.forbes.com/advisor/personal-finance/fcc-bans-ai-voices-in-robocalls, Feb 2024.

[60] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[61] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, Mikhail Kudinov, and Jiansheng Wei. Diffusion-based voice conversion with fast maximum likelihood sampling scheme, 2022.

[62] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32, 2019.

[63] Yohanna Rodríguez-Ortega, Dora María Ballesteros, and Diego Renza. A machine learning model to detect fake voice. In *International Conference on Applied Informatics*, pages 3–13. Springer, 2020.

[64] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[65] Bernard Sapoval, Th Gobron, and A Margolina. Vibrations of fractal drums. *Physical review letters*, 67(21):2974, 1991.

[66] Patrick Semansky. Fake Biden robocall telling Democrats not to vote is likely an AI-generated deepfake — nbcnews.com. https://www.nbcnews.com/tech/misinformation/joe-biden-new-hampshire-robocall-fake-voice-deep-ai-primary-rcna135120. [Accessed 27-03-2024].

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

[68] Kai Shen, Zeqian Ju, Xu Tan, Yanqing Liu, Yichong Leng, Lei He, Tao Qin, Sheng Zhao, and Jiang Bian. Naturalspeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers. *arXiv preprint arXiv:2304.09116*, 2023.

[69] Bowen Shi, Wei-Ning Hsu, Kushal Lakhotia, and Abdelrahman Mohamed. Learning audio-visual speech representation by masked multimodal cluster prediction. *arXiv preprint arXiv:2201.02184*, 2022.

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

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

[72] Chengzhe Sun, Shan Jia, Shuwei Hou, and Siwei Lyu. Ai-synthesized voice detection using neural vocoder artifacts. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[73] Hemlata Tak, Jee-weon Jung, Jose Patino, Madhu Kamble, Massimiliano Todisco, and Nicholas Evans. End-to-end spectro-temporal graph attention networks for speaker verification anti-spoofing and speech deepfake detection. *arXiv preprint arXiv:2107.12710*, 2021.

[74] Hemlata Tak, Jee-weon Jung, Jose Patino, Massimiliano Todisco, and Nicholas Evans. Graph attention networks for anti-spoofing. *arXiv preprint arXiv:2104.03654*, 2021.

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

[76] Hemlata Tak, Massimiliano Todisco, Xin Wang, Jee-weon Jung, Junichi Yamagishi, and Nicholas Evans. Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation. *arXiv preprint arXiv:2202.12233*, 2022.

[77] Xu Tan, Jiawei Chen, Haohe Liu, Jian Cong, Chen Zhang, Yanqing Liu, Xi Wang, Yichong Leng, Yuanhao Yi, Lei He, et al. Naturalspeech: End-to-end text-to-speech synthesis with human-level quality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[78] Massimiliano Todisco, Xin Wang, Ville Vestman, Md Sahidullah, Héctor Delgado, Andreas Nautsch, Junichi Yamagishi, Nicholas Evans, Tomi Kinnunen, and Kong Aik Lee. Asvspoof 2019: Future horizons in spoofed and fake audio detection. *arXiv preprint arXiv:1904.05441*, 2019.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[80] Christophe Veaux, Junichi Yamagishi, and Kirsten MacDonald. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. In *University of Edinburgh*, 2017.

[81] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023.

[82] Run Wang, Felix Juefei-Xu, Yihao Huang, Qing Guo, Xiaofei Xie, Lei Ma, and Yang Liu. Deepsonar: Towards effective and robust detection of ai-synthesized fake voices. In *ACM international conference on multimedia*, 2020.

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

[84] Yingzhi Wang, Abdelmoumene Boumadane, and Abdelwahab Heba. A fine-tuned wav2vec 2.0/hubert benchmark for speech emotion recognition, speaker verification and spoken language understanding. *arXiv preprint arXiv:2111.02735*, 2021.

[85] Amit Kumar Singh Yadav, Ziyue Xiang, Kratika Bhagtani, Paolo Bestagini, Stefano Tubaro, and Edward J Delp. Compression robust synthetic speech detection using patched spectrogram transformer. *arXiv preprint arXiv:2402.14205*, 2024.

[86] Junichi Yamagishi, Xin Wang, Massimiliano Todisco, Md Sahidullah, Jose Patino, Andreas Nautsch, Xuechen Liu, Kong Aik Lee, Tomi Kinnunen, Nicholas Evans, et al. Asvspoof 2021: accelerating progress in spoofed and deepfake speech detection. In *ASVspoof 2021 Workshop-Automatic Speaker Verification and Spoofing Coutermeasures Challenge*, 2021.

[87] Hong Yu, Zheng-Hua Tan, Zhanyu Ma, Rainer Martin, and Jun Guo. Spoofing detection in automatic speaker verification systems using dnn classifiers and dynamic acoustic features. *IEEE transactions on neural networks and learning systems*, 29(10):4633–4644, 2017.

[88] Yongyi Zang, You Zhang, Mojtaba Heydari, and Zhiyao Duan. Singfake: Singing voice deepfake detection. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12156–12160. IEEE, 2024.

[89] Ziqiang Zhang, Long Zhou, Chengyi Wang, Sanyuan Chen, Yu Wu, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. Speak foreign languages with your own voice: Cross-lingual neural codec language modeling. *arXiv preprint arXiv:2303.03926*, 2023.

APPENDIX

A. Detailed Detection Results of Whisper Features

The approach Whisper Features [34] uses a DNN for detecting fake audio samples and provides different options for the used DNN. While in Tab. III and Tab. II we provided, for the sake of clarity, only the results for the best approach, Tab. VIII shows the results for all approaches.

TABLE VIII: Effectiveness of different DNN models that are provided by the Whisper Features [34] approach.

Approach	EER	TPR	TNR	F1-Score
LCNN	43.6	56.4	56.4	0.72
LCNN (Finetuned)	43.3	56.7	56.7	0.72
LFCC LCNN	30.7	89.3	49.4	0.94
LFCC LCNN (Finetuned)	30.7	89.3	49.4	0.94
LFCC Mesonet	24.8	75.1	75.3	0.86
LFCC Mesonet (Finetuned)	23.1	76.2	77.6	0.86
LFCC Specrnet	36.6	63.0	63.7	0.77
LFCC Specrnet (Finetuned)	34.9	64.3	65.9	0.78
MFCC LCNN	35.8	69.7	58.8	0.82
MFCC LCNN (Finetuned)	37.2	89.1	36.6	0.94
MFCC Mesonet	29.1	71.1	70.7	0.83
MFCC Mesonet (Finetuned)	33.5	66.5	66.5	0.80
MFCC Specrnet	38.3	61.7	61.7	0.76
MFCC Specrnet (Finetuned)	38.0	62.1	62.0	0.76
Mesonet	44.6	55.5	55.3	0.71
Mesonet (Finetuned)	40.4	59.6	59.6	0.74
Specrnet	42.8	57.2	57.2	0.72
Specrnet (Finetuned)	40.8	59.2	59.2	0.74
VoiceRadar	**0.0**	**99.8**	**99.9**	**99.9**

B. Detailed Detection Results for Dataset

In Tab. III and Tab. II, we showed the effectiveness of existing detection approaches using all generated deep fake samples. Tab. VIII shows the results the different models here were provided by the Whisper Features [34] scheme. Tab. IX shows the results for each tool separated.

TABLE IX: Equal-Error-Rate (EER) of the individual state-of-the-art detection approaches for each utilized deep fake generation tool.

	Dataset	RawGAT-ST [73]	AASIST [28]	Raw PC-DARTS [22]	wav2vec 2.0 [76]	Whisper Features [34]
TTS	Jenny [2]	34.4	46.5	23.1	4.3	27.8
	VALL-E-X [89]	33.0	32.0	43.8	4.4	19.7
	SpeechT5 TTS [5]	35.8	47.6	35.9	1.8	4.6
	StyleTTS2 [49]	30.3	24.3	42.7	13.6	43.2
	Bark [4]	41.7	37.1	44.6	1.3	12.0
	Tortoise [10]	48.6	38.3	42.1	5.7	34.9
	Vits [37]	8.0	31.2	24.0	2.5	46.5
	XTTS [3]	21.5	41.2	29.8	1.4	27.9
VC	DiffHierVC [16]	19.6	24.6	48.3	13.5	23.6
	DiffVC [61]	21.1	26.0	49.5	5.5	15.0
	HierSpeech++ [44]	38.9	24.7	42.3	40.1	22.9
	SpeechT5 [5]	42.2	33.3	38.2	5.5	22.4