AudioMarkNet: Audio Watermarking for Deepfake Speech Detection

Wei Zong*
University of Wollongong, Australia

Yang-Wai Chow
University of Wollongong, Australia

Willy Susilo
University of Wollongong, Australia

Joonsang Baek
University of Wollongong, Australia

Seyit Camtepe
CSIRO Data61, Australia

Abstract

Deep generative models have improved significantly in recent years to the point where generated fake images or audio are now indistinguishable from genuine media. As a result, humans are unable to differentiate between real and deepfake content. While this presents a huge benefit to the creative sector, its exploitation to fool the general public has resulted in a real-world threat to society. To prevent generative models from being exploited by adversaries, researchers have devoted much effort towards developing methods for differentiating between real and generated data. To date, most existing techniques are designed to reactively detect artifacts introduced by generative models. In this work, we propose a watermarking technique, called AudioMarkNet, to embed watermarks in original speech. The purpose is to prevent speech from being used for speaker adaptation (i.e., fine-tuning text-to-speech (TTS)), which is commonly used for generating high-fidelity fake speech. Our method is orthogonal to existing reactive detection methods. Experimental results demonstrate the success of our method in detecting fake speech generated by open-source and commercial TTS models. Moreover, our watermarking technique achieves robustness against common non-adaptive attacks. We also demonstrate the effectiveness of our method against adaptive attacks. Examples of watermarked speech using our proposed method can be found on a website[1]. Our code and artifacts are also available online[2].

1 Introduction

The rapid advancement of deep generative models is extremely beneficial to the creative sector. In the audio domain, Text-to-Speech (TTS) allows natural speech to be generated from text, as is employed in various interactive applications [27]. However, this technological advancement raises concerns because it is now difficult, or even impossible, to distinguish between real and generated data. If an adversary has access to a victim's recorded speech, e.g., a video on social media, the adversary can use a TTS model to mimic the victim's voice for malicious purposes. As a real-world example, criminals have stolen millions of dollars using software to synthesize a chief executive's voice[3].

To prevent generative models from being exploited, a great deal of effort has been devoted towards developing methods for detecting generated data. One research direction relies on discovering artifacts introduced by deep generative models in a reactive manner [3, 19]. However, continuous advances in deep generative models result in fewer artifacts, which will eventually make generated data technically indistinguishable from real data. Another limitation of detecting artifacts is that most methods cannot explain the reason for their results. This severely limits the practicality of such methods in law enforcement because explainability is crucial for deep learning models to be used for legal purposes [9].

To overcome the limitations of reactive detection, another line of research is to proactively use watermarking techniques for detecting fake data. The goal is to embed predefined watermarks in generated data such that fake data can be detected reliably and the predictions can be explained. This line of research can be divided into two categories. The first focuses on responsible disclosure of generative models [23, 37]. Work in this category aims to insert watermarks into generative models before they are published. Although such work has demonstrated effectiveness in detecting fake data, it cannot work if an adversary trains their own generative models from scratch.

The other category of work focuses on detecting modifications by generative models [1, 10]. Predefined watermarks are embedded in the original data to protect it from malicious use. If an adversary modifies watermarked data using generative models, the embedded watermarks can still be detected. This effectively deters the spread of fake data.

While most efforts to date have focused on detecting fake images, in this paper, we focus on detecting fake speech syn-

[1] https://sites.google.com/view/fakespeechdetection.
[2] https://zenodo.org/records/14722182.
[3] https://www.wsj.com/articles/fraudsters-use-ai-to-mimic-ceos-voice-in-unusual-cybercrime-case-11567157402

thesized using a TTS model. Our goal is to protect publicly accessible speech from being exploited by an adversary to maliciously impersonate a person using voice cloning techniques. Our proposed watermarking technique can be categorized as detecting modifications by generative models (i.e., TTS). In particular, modification by a TTS model refers to changing the linguistic content while preserving a speaker's voice, i.e., voice cloning.

In this research, we consider the scenario where watermarks are embedded in speech data before it is released to the public. If an adversary applies speaker adaptation (i.e., fine-tuning TTS) using the watermarked speech, the generated fake speech will also contain our predefined watermarks. In this manner, the detection of a watermark in fake speech makes the detection outcomes explainable and also transforms fake speech detection into a simple task of detecting the existence of watermarks.

Speaker adaptation is crucial and widely used to improve the fidelity of voice cloning. It is especially useful when there is a distribution shift between the target voice and the training data [5]. Commercial voice cloning services usually require users to upload samples of a target voice to fine-tune their internal TTS models. When the models are ready, a user can generate fake speech using the target voice by simply entering a string of text.

Our method is conceptually similar to a dataset poisoning attack in the sense that our watermarks "poison" the training set used for fine-tuning TTS models. Our watermarks are specially designed to cause a TTS model to learn predefined patterns if it is fine-tuned on the "poisoned" data. This is not a trivial task because the linguistic content output by a TTS model can completely differ from the original speech. To demonstrate the real-world practicality of our method, we demonstrate that our watermarks can be learned successfully by commercial online voice cloning services, of which their internal workings are unknown.

Our contributions are summarized as follows:

- This work proposes a watermarking technique for protecting original speech from being exploited by voice cloning. This understudied research direction is orthogonal to existing reactive detection works.
- Our watermarks successfully transferred to TTS models with different architectures and trained on different data.
- Our watermarks also successfully transferred to commercial online voice cloning services. Thus, demonstrating the practicality of our method in the real world.
- Our watermarks are robust against attacks that are commonly used for evaluating audio watermarks.
- We also show the robustness of our watermarks against adaptive attacks, specifically designed to defeat our method.

2 Related Work

The study of detecting fake images [25, 37] significantly inspired work on detecting fake speech. In contrast to the image domain, where there is a huge body of work, research on detecting fake speech is limited. One line of work focuses on liveness detection, which relies on physical properties of human speech [39].

In contrast, another line of work focuses on reactively detecting artifacts in fake speech, which has attracted much attention [3, 18, 33–35]. By exploiting large pre-trained models for calculating embeddings of speech, fake speech detection was able to achieve high accuracy [3]. However, continuous improvements in deep generative models and the lack of explainability, impede further development in this direction.

The recently proposed Antifake [38] exploited adversarial perturbations to prevent the synthesis of fake speech. The idea was to generate adversarial examples to fool synthesizers such that generated speech would not be perceived as being spoken by the victim. Antifake is fundamentally different from our work because Antifake aims to fool speaker embedding models using adversarial examples, while our method aims to allow TTS models to learn our watermarks.

Similar to fake image detection, researchers have also proposed to watermark speech generative models [7, 23]. For instance, Ren et al. [23] embedded source speaker identities as watermarks in the vocoder component of a target system. As discussed, this type of defense is ineffective if an adversary can train models from scratch or fine-tune pre-trained models that are publicly available. To date, the use of watermarks to protect speech from being exploited by voice cloning remains underexplored.

WavMark [6] and AudioSeal [26] are recently proposed audio watermarking techniques that focus on embedding watermarks into already generated deepfake speech. Our work differs from these because the objective of our method is to transfer our watermarks to TTS models.

Liu et al. [17] recently proposed Timbre Watermarking, which uses an encoder-decoder architecture to embed audio watermarks. This method shares a similar idea with our work and the generated watermarks can also transfer to TTS models. However, Timbre Watermarking extracts watermarks by calculating the average of features along the time axis, resulting in complex watermark patterns. In contrast, our method extracts watermarks from each 1-second section, resulting in simple watermark patterns (a visual example can be found in Section 10.3 in the Appendix). While the complex Timbre Watermarking patterns may be less noticeable, they are vulnerable to noise. In Section 7.11, we show that a simple ensemble attack of Gaussian noise and resampling results in Timbre Watermarking failing to detect fake speech, whereas our method was still effective.

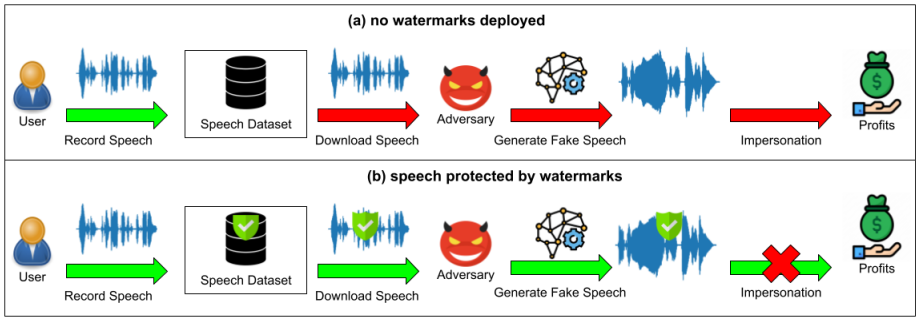


Figure 1: Our watermarking technique allows for detecting voice cloning by a TTS model. Resulting fake speech will contain watermarks embedded in the original speech, which enables the detection of fake speech in a reliable and explainable manner.

3 Threat Model

In this study, we focus on developing a watermarking scheme that detects voice cloning by a TTS model. The attacker's primary objective is to use samples of a victim's voice to generate speech using a TTS model to impersonate the victim. The source of the victim's speech samples can be public datasets or video/audio-sharing websites. Similar to other watermarking approaches, to protect speech against voice cloning, our watermarks must be embedded in speech samples before they are released to the public. Fig. 1 illustrates a practical scenario where our watermarks can be used to protect publicly accessible speech data from exploitation.

Fig. 1(a) shows the case where no defense is deployed. A user records speech and uploads it to the Internet. For example, the user may be a voice actor contributing to an open speech dataset. An adversary then downloads the user's speech and fine-tunes a TTS model using the speech (i.e., speaker adaptation). The adversary can then exploit the generated fake speech for profit, e.g., uploading the fake speech to a social media platform to impersonate the target.

In contrast, Fig. 1(b) shows an example scenario of how our watermarking scheme can be employed to deter the spread of fake speech. At the start, our watermarks are embedded in a user's original recorded speech to uniquely identify the user before the speech is released to the public. If an adversary downloads the watermarked speech and clones the victim's voice using a TTS model, the watermarks embedded in the original speech will transfer to the generated fake speech. Hence, attempts to impersonate the victim can be detected based on the presence of our watermarks in the fake speech. In this manner, our watermarks can deter the spread of fake speech.

4 Problem Definition and Assumptions

Let t represent a well-trained TTS model that transforms text input into waveforms. An adversary uses t for generating fake speech. A key assumption of our work is that the adversary applies speaker adaption, i.e., fine-tuning t using watermarked speech. Otherwise, it is impossible for generated speech to contain watermarks because the training of t is independent of the watermarks.

A speech sample is represented with a sequence of floating-point values, where each value ranges between $[-1,1]$. Our watermarks are represented with a bit string, i.e., a sequence of 0s and 1s. Mathematically, let $f : [-1,1]^n, \{0,1\}^m \rightarrow [-1,1]^n$ be the function to embed watermarks of length m into speech of length n, where n indicates the number of floating-point values in the sequence. Let $g : [-1,1]^n \rightarrow \{0,1\}^m$ be the function to retrieve watermarks of length m from speech of length n. f and g must fulfil the following requirements:

- *The effect of embedded watermarks on the intelligibility of speech is negligible*. Given original speech $x \in [-1,1]^n$ and a watermark $w \in \{0,1\}^m$, we require that:

$$p \circ f(x,w) \approx p(x), \tag{1}$$

  where p represents the human perception function. If the embedded watermarks affect intelligibility, it will decrease the value of the watermarked speech.

- *The original watermarks can be retrieved correctly from fake speech*. After an adversary fine-tunes t using watermarked speech $x' = f(x,w)$, t learns the embedded watermark w. This results in the generated fake speech also containing w:

$$g \circ t(s) = w, \tag{2}$$

  where s is the fake speech's text.

- *No false positives for speech without watermarks*. Let $\mathbb{W}$ be the set of registered watermarks. The output of decoded speech without watermarks must not be in $\mathbb{W}$:

$$g(x) \notin \mathbb{W}, \tag{3}$$

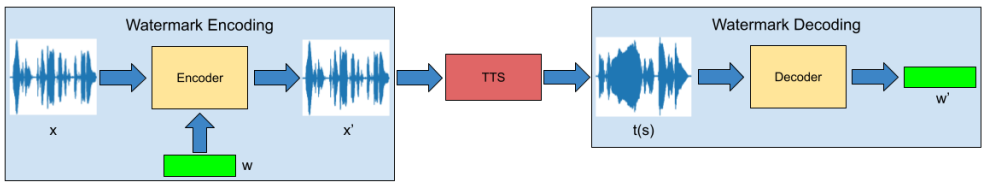


Figure 2: Overview of our proposed watermarking scheme. Message w is the watermark to embed in original speech x using the encoder. Fake speech $t(s)$ is input into the decoder to retrieve any embedded watermarks.

5 Proposed Method

5.1 Overview

We adopt the widely used encoder-decoder architecture for embedding and retrieving watermarks. Fig. 2 illustrates an overview of our method. Given a message w, which can be a user's unique identifier, we embed it into a set of original speech x to obtain watermarked speech x'. If an adversary fine-tunes a TTS model using x', the TTS model will learn our watermarks such that generated fake speech will contain our watermarks. When fake speech $t(s)$ is input into the decoder, the watermark w' will be retrieved. Fake speech is successfully detected if $w' = w$.

5.2 Insight

Other than unpredictable linguistic content, another challenge to overcome in detecting fake speech is variability in the duration of fake speech. This makes retrieving watermarks a non-trivial task. The vast variability in the duration of fake speech also makes detecting fake audio more difficult than detecting fake images, which generally take certain sizes.

Hence, we propose to divide original speech into sections that are 1 second in duration. Watermarks are then embedded into all such sections. When retrieving watermarks, fake speech is also divided into 1-second sections, where we attempt to retrieve watermarks from all these sections. This strategy enables watermark retrieval from fake speech of any duration and also allows our method to be applied to speech streamed over the Internet.

As previously discussed, our method is analogous to a dataset poisoning attack that guides a deep learning model to learn desired properties by modifying the training data [4]. The desired properties aim to make a target model distinguishable from other independently trained models and they are normally not correlated with the model's task. Therefore, these properties must be specially designed such that a target model can easily learn them. To successfully "poison" a TTS model, we embed watermarks in every frame of speech. This makes a target model frequently encounter our watermarks during speaker adaptation, which facilitates the learning of watermark patterns.

5.3 Architecture

5.3.1 Encoder

The common practice in an encoder-decoder architecture for embedding watermarks is to let the encoder fully generate watermarked data [32, 36]. Not only must the encoder embed watermarks but it also needs to make watermarked data indistinguishable from the original data.

Inspired by the design of TrojanModel [40], we argue that reconstructing original data unnecessarily increases the difficulty of training the encoder. Instead, we design the encoder to only learn perturbations applied to original speech. We do not include any generative network simulator to represent a potential TTS architecture used by an adversary when training the encoder. This ensures that our watermarks are not overfitted to a specific TTS architecture.

Our encoder f is a deep neural network (DNN) composed of wav2vec 2.0 [2], a Long Short-Term Memory (LSTM) layer, and fully connected (FC) layers. The workflow of our encoder is depicted in Fig. 3. There are 4 steps for embedding a watermark in speech. The first is to transform speech from the time domain into the frequency domain via the short-time Fourier transform (STFT). The second step exploits wav2vec 2.0 to extract features from raw waveforms. Message w is transformed by a linear layer to make its dimension consistent with the extracted features. The results are then added to the extracted features. This stabilizes training because it includes more information on the intended watermarks.

The third step is to calculate perturbations representing the watermark to embed. An LSTM layer and multiple FC layers are stacked to transform features extracted in the last step into frame-level perturbations. These perturbations are added to the original STFT results. In practice, perturbations are only added to the low-frequency components. There are two reasons for this. First, although the human voice spreads over a large range of frequencies, its power is mainly concentrated at low frequencies. Hence, forcing perturbations to be at low frequencies will intertwine our watermarks with the human voice so that they will be difficult to remove. Otherwise, if watermarks are not intertwined with speech in the frequency domain, an adversary can easily apply a filter to remove watermarks while negligibly affecting the quality of speech. Second, the masking threshold in acoustics suggests

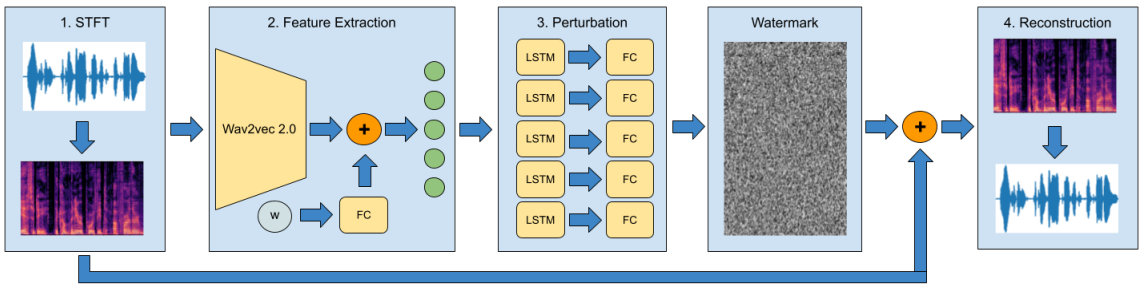


Figure 3: The encoder consists of wav2vec 2.0, an LSTM layer, and FC layers. The watermark embedding workflow has 4 steps. The first uses STFT to transform input speech into the frequency domain. Then, wav2vec 2.0 extracts speech features and message w is added to these features. The third step calculates perturbations representing the watermark to be embedded. The final step adds the perturbations to STFT results from the first step and applies inverse STFT to reconstruct watermarked speech in the time domain.

that a loud sound will make a soft sound inaudible if they are close in frequency [28]. This means small perturbations at the low frequencies will be masked by speech. This improves the imperceptibility of our watermarks. The final step is to add the watermark to original STFT results, generated in the first step, and apply inverse STFT to transform data in the frequency domain back to waveforms in the time domain.

5.3.2 Decoder

In comparison with the encoder, the architecture of the decoder g is simpler. It stacks multiple convolutional layers and 2 FC layers. The first FC layer increases the feature dimension from the output of convolutional layers. The second FC layer then projects hidden representations to the desired dimension. We empirically determined that appending a dropout layer to the first FC layer improves performance. Input to the decoder is the log-scaled magnitude of STFT results calculated from 1-second speech sections.

5.3.3 Normalization

After each convolutional layer in the encoder and decoder, we append an instance normalization layer [29] instead of the widely used batch normalization layer. A batch normalization layer compares all data in a batch, which makes sense for images. As an example, if an image is brighter (or darker) than other images in the same batch, that image may be a daytime (or nighttime) scene. However, comparing speech spectrograms in a batch may not be meaningful. For instance, a loud voice does not mean that the speaker is male or female. Therefore, we use instance normalization layers that normalize each data point individually along a specific channel.

5.4 Loss Function

Fulfilling the requirements shown in Equations 1, 2, and 3 are key to designing our loss function. To meet the requirement in Equation 1, we need perturbations applied to the STFT results of original speech to be small. Let δ represent the calculated perturbations, the loss for optimizing δ is shown as follows:

$$\ell_\delta = \|\delta\|_2^2 \tag{4}$$

Minimizing ℓ_δ decreases squared ℓ_2 norm of δ which will lead to watermarked speech being similar to the original speech.

To meet the requirement in Equation 2, we propose to optimize the squared difference between output logits of the decoder g and target watermarks:

$$\ell_{wm} = \|g(x') - h(w)\|_2^2, \tag{5}$$

where x' is watermarked speech and $w \in \{0,1\}^m$ is the target watermark. The function h is defined as:

$$h(w) = (w * 2 - 1) * \tau \tag{6}$$

The purpose is to change the target values from $\{0,1\}$ to $\{-\tau,\tau\}$. This enlarges the gap between different values to improve model performance. τ represents the desired logit values output by the decoder. It is preferable to set τ to a small number. This choice aligns with common DNN training pipelines that apply weight decay regularization to encourage the resulting DNN to contain small weights. In practice, setting τ to 5 produces good empirical results.

To fulfil the requirement in Equation 3, we predefine a watermark $w_0 \notin \mathbb{W}$, where $\mathbb{W}$ is the set of registered watermarks. The idea is for original speech to be decoded as a fixed watermark:

$$\ell_{org} = \|g(x) - h(w_0)\|_2^2, \tag{7}$$

where x represents original speech without watermarks.

Finally, the loss function to optimize is simply the weighted sum of the losses from Equations 4, 5, and 7:

$$\ell = \alpha \cdot l_\delta + \beta \cdot l_{wm} + \gamma \cdot l_{org} \tag{8}$$

5.5 Training

We train the encoder and decoder simultaneously in an end-to-end way. During each iteration, original speech is randomly divided into 1-second sections for embedding watermarks. Then, the decoder attempts to retrieve watermarks from watermarked speech. The decoder is also trained to extract the fixed watermark w_0 from original speech. Finally, the encoder and decoder that achieve the smallest loss are saved. To improve the robustness of our watermarks, we occasionally add Gaussian noise to both watermarked and original speech before inputting them into the decoder. Adding Gaussian noise prevents the decoder from relying on very subtle features, since such features will be destroyed by Gaussian noise. This in turn encourages the encoder to produce robust watermarks that are not easily removed by Gaussian noise. We empirically observed that this simple data augmentation strategy was sufficient to achieve good performance.

5.6 Generating Watermarked Speech

After the encoder and decoder are well-trained, we embed watermarks into the entire original speech. Specifically, original speech is sequentially divided into 1-second sections. Then, the encoder is applied to embed watermarks into all these sections.

5.7 Inference

Given a suspect speech, we first divide it into 1-second sections before applying the decoder to retrieve watermarks in each section. The decoded watermark is compared with a pool of registered watermarks. If a registered watermark matches the decoded watermark, that section is determined to be fake speech.

6 Evaluation Metrics

We use the following metrics to evaluate the detection of fake speech:

- **Undetected Fake Length (UFL)** measures the length of time at which fake speech continuously escapes detection in the context of multi-class classification:

$$UFL = len(g \circ t(s) \neq w), \tag{9}$$

  where $t(s)$ is fake speech. The function $len(\cdot)$ measures duration in seconds. Unlike images, speech conveys information over time. Given predictions $\mathbb{P}$ and the target watermark w, UFL is an array of positive values. Each element in UFL measures the length of time that fake speech can escape detection, this indicates how much information fake speech can convey without being detected. If UFL contains large values, it means fake speech may successfully spread false information without being detected, particularly when a user only accesses part of it and not the full speech. In contrast, if elements in UFL are all close to 0, it means fake speech cannot convey meaningful information before it is detected. The calculation of UFL can be found in our open source code.

- **Binary Undetected Fake Length (BUFL)** measures the length of time at which fake speech continuously escapes detection in the context of binary classification:

$$BUFL = len(g \circ t(s) \notin \mathbb{W}), \tag{10}$$

  where $\mathbb{W}$ is the set of registered watermarks. In practice, one may only be interested in knowing whether a suspect speech is genuine or fake, and not bothering about the specifics of the embedded watermark. Hence, we also measure BUFL to complement UFL. Theoretically, elements in BUFL should not be larger than the corresponding elements in UFL.

  UFL and BUFL can be interpreted as potential delays in detection since these metrics calculate the length of time that fake speech continuously bypasses detection. For example, a BUFL of 10, means a 10-second segment of fake speech will bypass detection.

- **False Positive Rate (FPR)** measures the probability that original speech, without watermarks, is incorrectly detected as fake speech:

$$FPR = \frac{len(g \circ x \in \mathbb{W})}{len(x)} \tag{11}$$

- **True Positive Rate (TPR)** is the probability of successful watermark extraction from watermarked speech in a multi-class classification context. TPR is calculated in terms of 1-second sections since watermarks are embedded in each 1-second section of original speech.

- **Binary True Positive Rate (BTPR)** is similar to TPR. The difference is that BTPR is calculated in the context of binary classification.

- **Equal Error Rate (EER)** is a widely used sentence-level metric for evaluating fake speech detection performance [34]. We calculate EER at the sentence-level in the multi-class classification context. For each sentence, we use the probability of watermarks being extracted from 1-second sections as the metric to calculate EER.

- **Binary Equal Error Rate (BEER)** is similar to EER. The difference is that BEER is calculated in the context of binary classification.

We use Perceptual Evaluation of Speech Quality (PESQ) and Signal-to-noise Ratio (SNR) to evaluate the quality of watermarked speech. These are standard metrics for evaluating distortion introduced into audio. Details of these metrics can be found in Section 10.1 in the Appendix.

7 Experimental Results

All experiments were conducted on a Ubuntu server with 64G RAM and an Nvidia A100 GPU. We report mean values and standard deviation in the form of $a \pm b$ where appropriate. α, β and γ in Equation 8 were all set to 1.0. Other hyperparameters not specified in this section can be found in our open source code. Wav2vec 2.0 was initialized using the pre-trained weights provided by Pytorch.

7.1 TTS Models

To demonstrate that our watermarks can generalize to different TTS architectures and different training data, we evaluated our method against two open source TTS models, namely, SV2TTS [11] and YourTTS [5] and two commercial TTS models, namely, PlayHT[4] and Speechify[5]. The two open source TTS models have fundamentally different architectures and were trained on different data. The architectures of the two commercial TTS models are proprietary and therefore unknown to the public. It is unlikely that they have the same architecture and use the same set training data. Our fine-tuned models are private and do not affect other users. Details of these models are presented in Section 10.2 in the Appendix.

7.2 Speech to Protect

The testing set of YourTTS contains 11 speakers (7 females and 4 males) in the VCTK dataset. We selected all these 11 speakers as victims and applied our watermarks to protect their speech. Each speaker was assigned a unique watermark.

7.3 Watermarks

In the experiments, we used 16-bit watermarks, which can represent 65,536 different users. This capacity is sufficient for large scale speech datasets. For example, the widely used large scale speaker recognition dataset Voxceleb2 [8] only contains 6,112 speakers. Our watermarks were applied to frequency components between 100 Hz to 1,000 Hz where the power of the human voice is concentrated.

7.4 Watermarked Speech

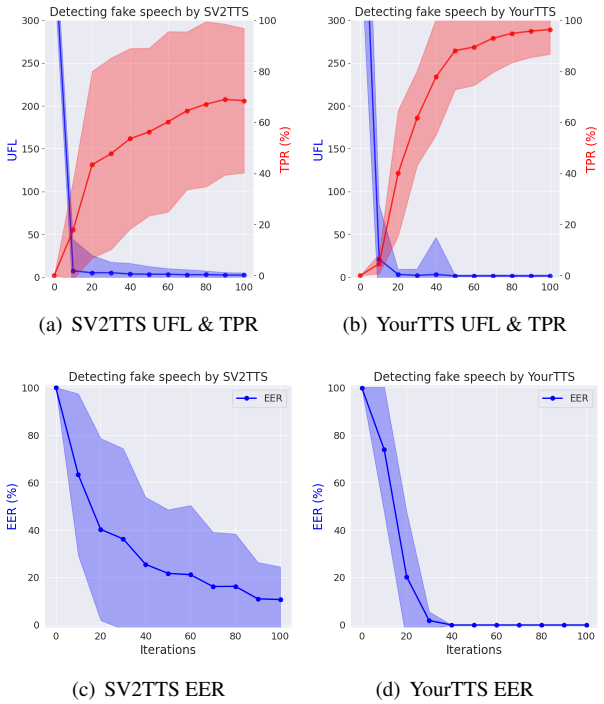
We applied our method to all speech recordings of victim speakers. PESQ and SNR values of watermarked speech were 3.52 ± 0.43 and 33.68 ± 5.54, respectively. Table 1 shows the effectiveness of our method. The PESQ and SNR values are within an acceptable range. This indicates that the intelligibility of our watermarked speech was preserved. Moreover, both UFL and BUFL of our method were small. This indicates that our method was effective in detecting watermarked speech. It is expected that only 1 second of watermarked speech can be played before it is detected. Section 10.3 in the Appendix visualizes our watermarked speech. It also visualizes fake speech generated by TTS models that are fine-tuned on our watermarked speech.

Although the FPR of our method was not 0, the value was small enough for practical usage. Only 1 second of speech, over a 2.5 hour average, was falsely detected as watermarked speech. A defender can prevent false alarms by setting an appropriate threshold for the percentage of watermarked speech detected. A 0% EER further shows that an appropriate threshold can result in excellent watermark detection.

Table 1: Effectiveness of our watermarks.

UFL	BUFL	TPR (%)	BTPR (%)
1.0 ± 0.0	1.0 ± 0.0	99.9 ± 0.2	99.9 ± 0.2

EER (%)	BEER (%)	FPR (%)
0.0 ± 0.0	0.0 ± 0.0	0.01 ± 0.02



(a) SV2TTS UFL & TPR (b) YourTTS UFL & TPR

(c) SV2TTS EER (d) YourTTS EER

Figure 4: UFL, TPR and EER comparisons when fine-tuning SV2TTS and YourTTS on our watermarked data. The figure shows mean and standard deviation values.

[4] https://play.ht/

[5] https://speechify.com/voice-cloning/

7.5 Detecting Fake Speech from SV2TTS and YourTTS

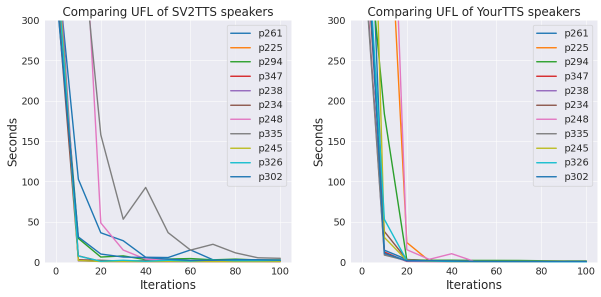
Before generating fake speech, we applied speaker adaptation to the pre-trained SV2TTS and YourTTS models. Specifically, we fine-tuned them on our watermarked speech for at most 100 iterations. We calculated UFL and BUFL every 10 iterations. After speaker adaptation, we selected 100 sentences from the "test-clean" subset of LibriSpeech [22] and used the fine-tuned models to generate fake speech for each sentence. In the experiments, we considered fake speech directly generated by the pre-trained models without speaker adaptation as benign speech. This is because the pre-trained SV2TTS and YourTTS were independent of our watermarks. Experimental results show that the FPR for SV2TTS and YourTTS were $0.05 \pm 0.10\%$ and 0%, respectively. This indicates a rare occurrence of false alarms.

Fig 4 compares the UFL, TPR and EER when fine-tuning SV2TTS and YourTTS. The experimental results of BUFL, BTPR and BEER show similar characteristics and are provided in Fig 10 in the Appendix. As previously discussed, UFL is theoretically the upper bound of BUFL, and TPR is the lower bound of BTPR. This means BUFL and BTPR are the better metrics for measuring detection performance.

Overall, it is obvious that SV2TTS and YourTTS both managed to learn our watermarks because the UFL and EER decreased while the TPR increased significantly as more iterations were run. YourTTS learned our watermarks more efficiently than SV2TTS, which can be seen from its faster decrease in UFL and EER and faster increase in TPR.

Before SV2TTS and YourTTS were fine-tuned, UFL values for SV2TTS and YourTTS were 391.1 ± 35.9 and 564.9 ± 171.1, respectively. It should be noted that the calculation of UFL depends on audio length. The duration of fake speech generated by SV2TTS and YourTTS were different. When an adversary fine-tuned SV2TTS and YourTTS for 20 iterations on our watermarked speech, UFL decreased to 5.2 ± 20.1 and 3.3 ± 5.8, respectively. When 100 iterations were run, the UFL further decreased to 2.5 ± 2.4 and 1.6 ± 0.9, respectively. This means it is highly likely that fake speech generated by SV2TTS and YourTTS will be detected within 3 and 2 seconds, respectively.

The EER values for detecting fake speech generated by SV2TTS and YourTTS were $10.73 \pm 13.78\%$ and $0.00 \pm 0.00\%$, respectively, after fine-tuning for 100 iterations. This indicates the success of our watermarks in terms of sentence-level detection. In particular, the EER values for YourTTS fake speech watermarks stabilized at 0%, which means absolute detection after 40 iterations. Although the performance of SV2TTS fake speech was worse compared to YourTTS, defenders can potentially improve the detection performance by concatenating multiple sentences into a long sentence. This is because the FPR of our method is low, i.e. only $0.05 \pm 0.10\%$ and 0% for SV2TTS and YourTTS, respectively. Without the TPR changing, it is more likely for watermarks to be detected in longer fake sentences, potentially resulting in lower EER values.



(a) SV2TTS (b) YourTTS

Figure 5: UFL change comparison during fine-tuning SV2TTS and YourTTS for each speaker. The difficulty of learning watermarks varies for different speakers.

Table 2: Multi-class classification results using PlayHT and Speechify.

	UFL	**TPR (%)**	**EER (%)***
PlayHT	23.3 ± 19.3	2.4 ± 2.6	45.5
Speechify	44.6 ± 24.3	0.8 ± 1.9	81.8

*: results are calculated based on the speaker level.

An interesting observation is the fluctuation in UFL values for YourTTS after 40 iterations. This was because watermarks for speaker "p248" were incorrectly recognized as watermarks for another speaker. However, this fluctuation did not occur when BUFL was used, since BUFL is for binary classification without differentiating between different speakers.

To investigate the speed at which our watermarks for each speaker were learned, Fig. 5 compares detailed changes in UFL for different speakers. It is interesting to see that the difficulty of learning watermarks for different speakers varied. For example, watermarks for speaker "p335" were the most difficult to learn by SV2TTS. After 40 iterations, the corresponding UFL was still 92.8, and it took 80 iterations to decrease the UFL to 11.8. In contrast, watermarks for speaker "p248" were the most difficult to learn by YourTTS. How-

Table 3: Binary classification results using PlayHT and Speechify.

	BUFL	**BTPR (%)**	**BEER (%)***
PlayHT	17.7 ± 16.5	3.7 ± 3.4	18.2
Speechify	33.2 ± 25.2	1.5 ± 2.8	54.6

*: results are calculated based on the speaker level.

Table 4: UFL for time domain attacks.

	WN	**RS**	**MF**	**SS**	**AS**	**QT**	**EA**
Watermarked	1.4 ± 0.9	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.1	1.0 ± 0.0	$\mathbf{3.6 \pm 5.6}$	1.1 ± 0.3
SV2TTS	$\mathbf{3.6 \pm 5.0}$	2.5 ± 2.4	2.7 ± 3.0	2.8 ± 3.7	2.6 ± 2.6	3.5 ± 5.4	2.6 ± 2.7
YourTTS	1.7 ± 1.2	1.6 ± 0.9	1.5 ± 0.9	1.7 ± 1.0	1.7 ± 1.0	$\mathbf{3.9 \pm 11.0}$	1.5 ± 1.1
PlayHT	30.2 ± 21.7	24.2 ± 19.2	25.2 ± 20.6	25.2 ± 21.7	27.5 ± 20.4	$\mathbf{49.4 \pm 17.3}$	18.8 ± 17.8
Speechify	39.2 ± 25.3	44.6 ± 24.3	33.2 ± 28.3	44.6 ± 24.3	$\mathbf{47.9 \pm 22.3}$	$\mathbf{47.9 \pm 21.2}$	17.4 ± 19.1

Table 5: BUFL for time domain attacks.

	WN	**RS**	**MF**	**SS**	**AS**	**QT**	**EA**
Watermarked	1.4 ± 0.8	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.1	1.0 ± 0.0	$\mathbf{3.6 \pm 5.6}$	1.1 ± 0.3
SV2TTS	$\mathbf{2.9 \pm 4.5}$	2.3 ± 2.2	2.4 ± 2.8	2.5 ± 3.1	2.5 ± 2.6	2.8 ± 4.6	2.4 ± 2.6
YourTTS	1.6 ± 1.2	1.6 ± 0.9	1.5 ± 0.9	1.7 ± 1.0	1.7 ± 1.0	$\mathbf{3.9 \pm 11.0}$	1.6 ± 1.1
PlayHT	23.3 ± 22.3	18.3 ± 16.5	20.1 ± 18.7	19.5 ± 20.1	20.8 ± 19.4	$\mathbf{42.7 \pm 21.2}$	12.6 ± 13.3
Speechify	28.7 ± 24.2	33.2 ± 25.2	25.3 ± 25.5	35.0 ± 26.0	$\mathbf{39.2 \pm 26.2}$	35.0 ± 23.9	12.9 ± 16.5

ever, it took only 20 iterations to decrease the corresponding UFL to 15.7. TPR and EER values for individual speakers can be found in Fig. 11 in the Appendix, which also shows the varying difficulty in learning watermarks for different speakers.

7.6 Detecting Fake Speech from PlayHT and Speechify

We created accounts on the PlayHT and Speechify websites under their free voice cloning plan. Both services require users to first upload the recorded voice of a target before voice cloning. They then used the uploaded voice to fine-tune their underlying TTS models. It takes less than a minute for both services to be ready. Users can then input their intended text and the services will generate corresponding fake speech using the target voice. PlayHT allows users to adjust 3 advanced voice controls: "Stability", "Similarity", and "Intensity", using slider bars. We set the slider bars to the middle position to generate fake speech with an average setting.

The sampling rates of PlayHT and Speechify were 44.1 kHz and 48 kHz, respectively. As these differ from the 16 kHz sampling rate of our model, we up-sampled audio files from 16 kHz to 44.1 kHz before uploading them to PlayHT and Speechify. After the resulting fake speech files were downloaded, they were down-sampled back to 16 kHz before inputting them into our model. PlayHT only allows a single file of less than 50 MB to be uploaded. For a specific speaker, we combined multiple speech files into a single file such that the file size was slightly less than 50 MB. The duration of each combined file was about 4 minutes and 30 seconds. Speechify requires the number of characters for a single piece of fake speech to be less than 1,000. Hence, we only selected 14 sentences from the "test-clean" subset of LibriSpeech to generate, instead of the 100 sentences that we did for SV2TTS and YourTTS. The overall duration of each fake speech was about 57 seconds.

The experimental results are shown in Tables 2 and 3. We consider fake speech generated by fine-tuning PlayHT and Speechify on original speech, i.e. without our watermarks, as benign speech. The resulting generated speech were concatenated into a long speech for each speaker. This was done separately for benign speech and fake speech. The EER and BEER were then calculated over the long speech of all speakers. In this manner, EER and BEER show the performance based on the speaker level. One can see that BEER performs significantly better than EER for both PlayHT and Speechify. Although these values are worse compared to SV2TTS and YourTTS, the results show that our watermarks managed to transfer to both PlayHT and Speechify. This demonstrates the effectiveness of our method for black-box commercial TTS models.

Using BUFL, our method detects fake speech from PlayHT after 17 seconds, whereas fake speech from Speechify was detected after 33 seconds. Furthermore, the FPR for PlayHT and Speechify were both 0, indicating that a defender would reliably detect fake speech using our method. If BTPR > 0 is used as the detection criteria of fake speech, our watermarks for 9 speakers transferred to PlayHT, and our watermarks for 5 speakers transferred to Speechify.

The experimental results also show that PlayHT learned our watermarks more efficiently than Speechify. We cannot determine the exact cause of this difference because the internal workings of these commercial models are proprietary. It may be due to PlayHT running more iterations than Speechify, or PlayHT using a more advanced TTS model than Speechify.

Table 6: UFL for frequency domain attacks.

	MP3	**Opus**	**LP**	**HP**	**NR**
Watermarked	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	$\mathbf{4.2 \pm 23.5}$	1.4 ± 0.7
SV2TTS	2.5 ± 2.4	2.7 ± 2.9	2.5 ± 2.4	$\mathbf{6.2 \pm 33.7}$	3.9 ± 16.4
YourTTS	1.6 ± 0.9	1.7 ± 1.0	1.6 ± 0.9	$\mathbf{4.9 \pm 22.7}$	1.8 ± 1.3
PlayHT	23.3 ± 19.3	27.5 ± 21.4	23.3 ± 19.3	$\mathbf{45.8 \pm 18.7}$	$\mathbf{45.8 \pm 19.4}$
Speechify	41.8 ± 24.0	41.8 ± 24.1	44.6 ± 24.3	51.7 ± 19.1	$\mathbf{61.3 \pm 4.0}$

Table 7: BUFL for frequency domain attacks.

	MP3	**Opus**	**LP**	**HP**	**NR**
Watermarked	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	$\mathbf{1.7 \pm 1.3}$	1.4 ± 0.7
SV2TTS	2.3 ± 2.3	2.4 ± 2.7	2.3 ± 2.2	2.4 ± 2.6	$\mathbf{3.8 \pm 16.5}$
YourTTS	1.6 ± 0.9	1.7 ± 1.0	1.6 ± 0.9	$\mathbf{1.8 \pm 1.3}$	$\mathbf{1.8 \pm 1.3}$
PlayHT	17.2 ± 16.5	22.4 ± 20.7	17.7 ± 16.5	21.6 ± 18.1	$\mathbf{35.4 \pm 22.5}$
Speechify	31.6 ± 24.1	35.0 ± 25.0	33.2 ± 25.2	44.7 ± 23.9	$\mathbf{47.9 \pm 21.6}$

7.7 Robustness to Non-adaptive Attacks

In addition to being effective, a practical watermarking scheme should be robust to attacks. This section evaluates our watermarks against attacks commonly used in the literature to evaluate the robustness of watermarks. These attacks are grouped into time domain attacks and frequency domain attacks. Time domain attacks include White Noise (WN), Resampling (RS), Median Filter (MF), Sample Suppression (SS), Amplitude Scaling (AS), Quantization (QT), and Echo Addition (EA). Frequency domain attacks include MP3, Opus, Low-pass Filtering (LP), High-pass Filtering (HP), and Noise Reduction (NR). The details of these attacks can be found in Section 10.4 in the Appendix.

Experimental results on the robustness of our watermarked speech in terms of UFL and BUFL are shown in Tables 4, 5, 6 and 7. The results on fake speech generated by fine-tuning the open source and commercial TTSs using our watermarked speech are also presented. It should be noted that SV2TTS and YourTTS were fine-tuned for 100 iterations. We marked the worst results for our watermarked speech for each TTS in bold. The experimental results and discussion for EER and BEER can be found in Section 10.5 in the Appendix. The EER and BEER results are consistent with UFL and BUFL results. We focus on discussing UFL and BUFL results in this section.

Overall, our method was robust to the time domain attacks shown in Tables 4 and 5. Although the quantization attack was the most effective in decreasing the performance of our method, our method was still effective since the watermarks are likely to be detected within 5 seconds for our watermarked speech and fake speech generated by SV2TTS and YourTTS. An exception was the quantization attack against fake speech from PlayHT, of which the effectiveness of our watermark decreased. Nevertheless, our watermarks can still be detected in fake speech from PlayHT. Overall, the detection of fake speech from PlayHT and Speechify was robust to the other time domain attacks. This is because the UFL and BUFL values were similar to their corresponding values when no attack was applied, shown in Tables 2 and 3.

Tables 6 and 7 show experimental results for frequency domain attacks. Although the high-pass filter and noise reduction attacks were the most effective in removing our watermarks, our watermarked speech and fake speech generated by SV2TTS and YourTTS can still be detected within an average of 7 seconds. In terms of UFL, our watermarks in fake speech from Speechify were destroyed by the noise reduction attack. Nonetheless, a defender can still rely on BUFL to detect watermarks in fake speech processed by noise reduction from Speechify. For other frequency domain attacks, the detection of fake speech from PlayHT and Speechify was robust as their UFL and BUFL values were similar to their corresponding values, shown in Tables 2 and 3.

7.8 Robustness to Adaptive Attacks

In addition to non-adaptive attacks, a practical watermarking scheme must be robust to adaptive attacks, of which an adversary is aware of the underlying watermark mechanisms. In this work, we considered 3 adaptive attacks. Details and experimental results of these adaptive attacks are presented in the following subsections. We focus on discussing UFL and BUFL results. The experimental results for EER and BEER are overall consistent with UFL and BUFL results and they can be found in Section 10.6 in the Appendix.

Table 8: UFL for the band-stop filter attack.

	100 Hz	200 Hz	400 Hz
Watermarked	1.8 ± 1.9	14.9 ± 122.6	$\mathbf{17.1 \pm 138.1}$
SV2TTS	2.6 ± 2.7	4.2 ± 16.4	$\mathbf{20.5 \pm 71.1}$
YourTTS	3.2 ± 31.1	4.9 ± 41.0	$\mathbf{5.5 \pm 45.1}$
PlayHT	25.2 ± 22.0	25.2 ± 23.4	$\mathbf{45.8 \pm 21.3}$
Speechify	$\mathbf{41.8 \pm 25.1}$	30.1 ± 28.6	27.0 ± 28.9

Table 9: BUFL for the band-stop filter attack.

	100 Hz	200 Hz	400 Hz
Watermarked	1.9 ± 2.2	6.6 ± 87.1	$\mathbf{10.9 \pm 109.6}$
SV2TTS	2.2 ± 2.4	3.7 ± 15.0	$\mathbf{4.8 \pm 25.4}$
YourTTS	3.4 ± 33.6	3.9 ± 31.6	$\mathbf{4.3 \pm 38.7}$
PlayHT	$\mathbf{22.4 \pm 19.4}$	21.6 ± 20.2	11.3 ± 17.0
Speechify	$\mathbf{36.9 \pm 23.7}$	23.4 ± 25.8	17.4 ± 25.6

Table 10: Results for the denoising autoencoder attack.

	Before Denoising	After Denoising
SNR Training	33.86 ± 2.70	47.37 ± 3.14
SNR Victim	33.68 ± 5.54	41.34 ± 3.97
PESQ Training	3.67 ± 0.31	4.31 ± 0.16
PESQ Victim	3.52 ± 0.43	3.88 ± 0.28
UFL Training	1.0 ± 0.1	27.5 ± 101.0
UFL Victim	1.0 ± 0.0	2.1 ± 1.9
UFL YourTTS	1.6 ± 0.9	2.6 ± 2.8
BUFL Training	1.0 ± 0.0	10.6 ± 16.5
BUFL Victim	1.0 ± 0.0	1.6 ± 1.2
BUFL YourTTS	1.6 ± 0.9	2.0 ± 1.4
EER (%) Training	0.0 ± 0.0	95.0 ± 9.0
EER (%) Victim	0.0 ± 0.0	9.3 ± 13.2
EER (%) YourTTS	0.0 ± 0.0	8.6 ± 14.0
BEER (%) Training	0.0 ± 0.0	74.6 ± 24.0
BEER (%) Victim	0.0 ± 0.0	2.0 ± 2.8
BEER (%) YourTTS	0.0 ± 0.0	3.0 ± 8.3

7.8.1 Band-stop Filters

As our watermarks were applied within the 100 Hz to 1,000 Hz frequency components, this adaptive attack randomly selects a bandwidth at 100 Hz, 200 Hz or 400 Hz and filters out all signals in the selected frequency band. This is analogous to a cropping attack in the image domain that crops out up to 44% of watermarks.

The experimental results are presented in Tables 8 and 9. The worst results for our watermarked speech for each TTS are shown in bold. Overall, our watermarks were effective against this adaptive attack. In general, the effectiveness of our watermarks decreased as larger portions of our watermarks were cropped out.

However, fake speech generated by PlayHT and Speechify were exceptions since the effectiveness of our watermarks improved, especially in terms of BUFL, when larger portions of watermarks were cropped out. A potential reason for this may be because fake speech from PlayHT and Speechify introduced large distortions to our watermarks. Hence, cropping out part of the signal in the frequency domain may partially remove these distortions, which facilitated the detection of watermarks.

7.8.2 Denoising Autoencoder

Given a set of original speech, an adversary can use our method to generate watermarked speech. The purpose is to construct a dataset consisting of clean and watermarked speech pairs. The adversary then trains a denoising autoencoder on these data pairs. After the denoising autoencoder is well trained, it is used to remove watermarks from protected speech. The purified protected speech is finally passed to a TTS for voice cloning. The details of training the denoising autoencoder can be found in Section 10.7 in the Appendix.

In the experiments, we used a different set of 11 speakers from the training set of YourTTS to train our watermarking network from scratch. These 11 speakers were different from the 11 victims because if an adversary has access to the victims' original speech, this adaptive attack becomes pointless. We ran speaker adaptation using YourTTS for 100 iterations.

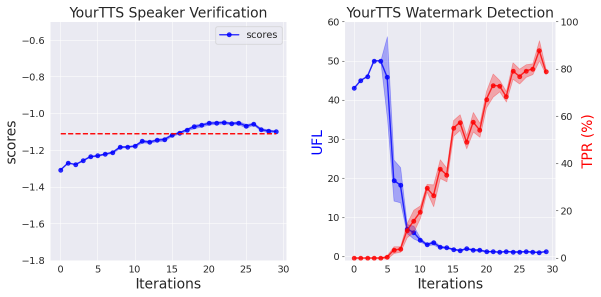
The experimental results in Table 10 show that the denoising autoencoder effectively inversed our watermarks in the training set. This is because SNR and PESQ values of watermarked training speech improved significantly. This reduced the effectiveness of our watermarks for the watermarked training speech, because UFL, BUFL, EER and BEER increased significantly. However, the watermarks for victim speech were not affected. Although SNR and PESQ values for watermarked victim speech increased, there was only a small increase in the corresponding UFL, BUFL, EER and BEER values.

For fake speech generated by YourTTS, UFL, BUFL, EER and BEER values only increased slightly. This means this adaptive attack failed to prevent our watermarks from transferring to YourTTS. On average, fake speech from YourTTS was detected in under 3 seconds.

7.8.3 Adaptive Iteration Attack

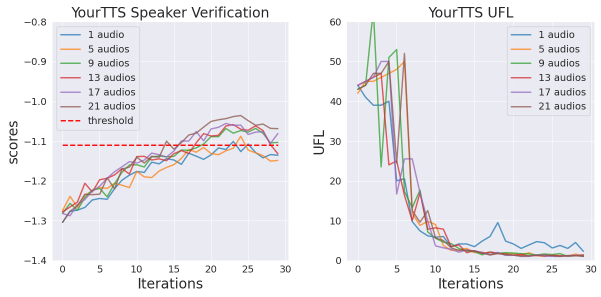
In this adaptive attack, the goal of an adversary is to prevent a TTS model from learning our watermarks by limiting the number of training iterations. Specifically, the adversary immediately stops fine-tuning when the generated fake speech manages to fool a speaker verification system.

In the experiments, we evaluate this attack using YourTTS because we cannot control training iterations for commercial



(a) Speaker Verification (b) UFL and TPR

Figure 6: Speaker verification, UFL, and TPR comparisons for fake speech generated by YourTTS. The experiments were repeated 5 times and the figures plot mean and standard deviations.



(a) Speaker Verification (b) UFL

Figure 7: Speaker verification and UFL comparisons for fake speech generated by YourTTS. The number of audio files range between 1 to 21.

TTSs. We included 5 minutes of speech from the former US president Obama[6] to train our watermarking network and focused on protecting Obama's speech. This is because Obama's voice is quite different from the YourTTS's training set. Otherwise, YourTTS may fool speaker verification without speaker adaptation, which breaks the key assumption of our method. We trained our watermarking model on this training set from scratch and applied watermarks to Obama's speech. We used the same 14 sentences to generate fake speech as we did for evaluating PlayHT and Speechify.

For speaker verification, we used the recently proposed RawNet3 model [13], which achieved state-of-the-art results on the VoxCeleb dataset with an EER of 0.89%. The VoxCeleb dataset [20] is a large-scale speaker verification dataset containing real-world utterances from over 1,000 celebrities. In the experiments, we used a threshold of -1.11 that corresponds to a 0.1% false positive rate and a 4.9% false negative rate. This setting is practical for a real-world scenario in which false positives will cause serious repercussions while a few false negatives are tolerable. We used the first 11 seconds of Obama's speech as enrolment for speaker verification.

We repeated the experiments 5 times. Fig. 6 shows that fine-tuning YourTTS for 16 or more iterations on Obama's speech can reliably fool RawNet3 as the verification scores were above the threshold. However, the UFL values decreased below 5 when 16 or more iterations were run. This means our method was able to detect fake speech even when an adversary immediately stopped speaker verification after 16 iterations.

From the results, we can see that if the target's speech is different from the TTS's training set, our method is robust to an adaptive iteration attack. However, if the target's speech is similar to the training set, our method will fail because a TTS model can fool speaker verification in a zero-shot setting or within a few iterations. Improving the learnability of our watermarks is an interesting topic for future work.

[6] https://www.americanrhetoric.com/speeches/barackobama/barackobamaweeklytransition7.htm

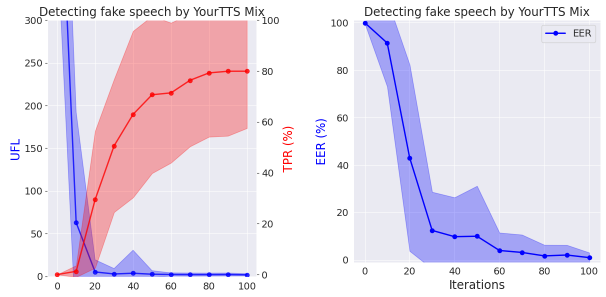
Table 11: YouTTS speaker adaptation for 100 iterations using a mixture of original and watermarked speech.

Ratio*	**BUFL**	**BTPR (%)**	**BEER (%)**	**Succ$^+$**
80%	1.7 ± 1.2	82.8 ± 18.9	0.00 ± 0.00	11/11
60%	7.3 ± 42.8	17.8 ± 14.1	38.6 ± 38.9	9/11
40%	23.8 ± 88.8	5.4 ± 10.9	81.3 ± 32.7	6/11
20%	29.0 ± 115.1	2.4 ± 7.5	99.8 ± 0.4	2/11

*: percentage of watermarked speech.
$^+$: number of speakers for which our watermarks transferred to YourTTS in terms of BUFL.

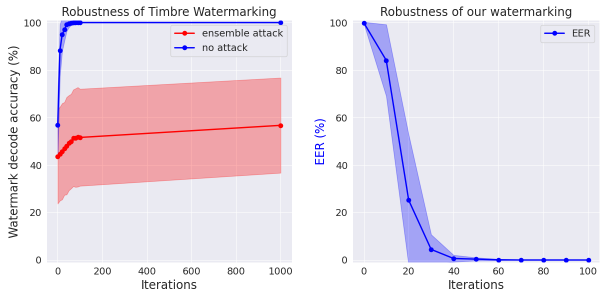
In the preceeding experiments, we used speech with a 5 minutes and 7 seconds duration that we split into 85 files. We then investigated whether adversaries can use a smaller number of files to fine-tune YourTTS to avoid learning our watermarks. These experimental results are shown in Fig. 7. Specifically, adversaries used 1 to 21 audio files to fine-tune YourTTS. The total audio duration ranges between 4.9 seconds to 73.5 seconds.

Overall, with a larger number of files, adversaries needed fewer iterations to fool speaker verification. In particular, fine-tuning with only 58.5 seconds of audio (17 audio samples) was sufficient for achieving comparable fooling performance with fine-tuning with 5 minutes of audio. An interesting observation is that adversaries can potentially fool state-of-the-art speaker verification even when fine-tuning with only 4.9 seconds of audio. Regardless, all fake speech could successfully be detected since all UFL values consistently decreased to less than 5 seconds when speaker verification was fooled by the fake speech.



(a) UFL and TPR (b) EER

Figure 8: Evaluation of our method when an adversary mixes in an arbitrary watermark.



(a) Timbre watermarking (b) Our watermarking

Figure 9: Comparison with Timbre watermarking using a simple ensemble attack on fake speech from YourTTS. Timbre watermarking used watermark decode accuracy in their paper.

7.9 Watermark Mixture

This section evaluates our method against an adversary that mixes in an arbitrary watermark. Specifically, given the already watermarked speeches of 11 VTCK speakers, we trained our model on them from scratch with a different set of watermarks. Then, we embedded the learned watermarks into the already watermarked speeches. These "double" watermarked speeches were then used to fine-tune YourTTS.

Experimental results in Fig. 8 show that the original watermarks can still effectively be detected, despite a slight decrease in performance. When YourTTS was fine-tuned for 100 iterations, the detection of original watermarks achieved a UFL of 1.8 ± 1.3 and an EER of $1.00\% \pm 2.09\%$.

7.10 Different Watermarking Rates

Our watermarks need to be applied before the speech is released to the public. However, original speech from a speaker, e.g., a celebrity, may already be in the public domain. In this section, we evaluate the effectiveness of our method in the case that an adversary has access to original speech samples. We assume that an adversary collects a mixture of original speech and watermarked speech. The experimental results for BUFL, BTPR and BEER in Table 11 show that the greater the number of watermarked speech samples, the easier it is for a TTS model to learn our watermarks. Results for UFL, TPR and EER show similar characteristics and are provided in Table 18 in the Appendix. When 40% of the samples were watermarked speech, our watermarks managed to transfer to YourTTS for more than half of the speakers. Even if only 20% of speech is watermarked, our watermarks still succeeded for 2 of the speakers. This experiment demonstrates that our watermarks can be effective when an adversary has access to a mixture of original speech and watermarked speech. It also shows the percentage of watermarked samples required to effectively fine-tune a TTS model to learn our watermarks.

7.11 Comparison with Timbre Watermarking

As discussed in Section 2, the watermark patterns of Timbre Watermarking [17] are complex. This complexity potentially makes it less noticeable but vulnerable to noise. We compare Timbre Watermarking and our method using a simple ensemble attack, which first adds white noise, then down-samples audio to an 8,000 sampling rate, before up-sampling it back to the original sampling rate.

We applied Timbre Watermarking to embed watermarks in all audio of the 11 VCTK speakers and achieved 100% decoding accuracy. The SNR and PESQ values of watermarked audio were 26.95 ± 1.82 and 3.49 ± 0.30, respectively, which are comparable to our watermarks. We fine-tuned YourTTS on watermarked audio and applied the ensemble attack to the generated fake speech. The fake speech quality decreased in terms of SNR to 21.67 ± 2.95.

Fig. 9 shows the results. Our method is robust to this ensemble attack as the EER decreased to $0.64\% \pm 1.43\%$ when YourTTS was fine-tuned for 40 iterations. Although Timbre Watermarking successfully transferred to YourTTS, which can be seen from the high decoding accuracy when no attack was applied, the watermarks in fake speech were easily destroyed by this ensemble attack. Even when YourTTS was fine-tuned for 1,000 iterations, Timbre Watermarking was still ineffective with a decoding accuracy of below 60%.

7.12 Comparison with AASIST

In this section, we compare our method with AASIST [12], which represents the state-of-the-art in reactive fake speech detection and is a widely used benchmark in the literature. We evaluated pre-trained AASIST on a testing set consisting of the original speech of 11 VTCK speakers and fake speech generated by YourTTS without speaker adaptation. The EER of ASSIST was above 30% because YourTTS fake speech was not included in the training set of AASIST. Under the same settings, our method surpassed AASIST when YourTTS

was fine-tuned for 30 iterations because our BEER was less than 1% after 30 iterations.

It should be noted that it may not be fair to compare watermarking techniques with reactive detection because watermarking techniques require TTS to be fine-tuned on watermarked data while this is not necessary for reactive detection. The main purpose of this section is to show that watermarking techniques can generalize to unseen TTS models, while it is a challenge to generalize reactive detection techniques.

7.13 Watermark Stealthiness

Despite achieving high PESQ and SNR values, our watermarks may be noticable in silent sections of audio. This is because we watermark each 1-second section, including silent sections. If original audio contains background noise, our watermarks are less perceptible when interwined with the background noise, e.g., the Obama's speech sample used in our experiments. Results in Table 10 suggest that the stealthiness of our watermarks can potentially be improved.

To evaluate whether our watermarks affect normal usage, we used RawNet3 to determine if watermarked speech was recognized as being associated with their corresponding speaker. We conducted experiments for all 11 VCTK speakers. For each speaker, 4 sentences were used as reference for speech verification. We maintained the use of a -1.11 threshold, which corresponds to a 0.1% false positive rate and a 4.9% false negative rate.

The experimental results show that RawNet3 successfully associated 99.57% of audio samples from the 11 VCTK speakers with their respective speakers. The lowest recognition rate was 98.73% for speaker "p261". These results suggest that our watermarks do not affect normal speech usage.

7.14 Computational Overhead

Our architecture includes wav2vec 2.0, which is a large model. Nevertheless, wav2vec 2.0 is only involved in the training stage, whereas the decoding process uses a simple convolutional neural network. The number of floating-point operations per second (FLOPS) of our decoder is 1.2375 GFLOPS. This is slightly lower than Alexnet, which is 1.4297 GFLOPS when its input dimensions are 3*224*224. Training our model for 200 epochs for the 11 VCTK speakers took 138 minutes. In contrast, watermark decoding is efficient because the extraction process can be parallelized. In 1 second, the decoder can extract watermarks from 2 hours of audio.

8 Discussion

8.1 Watermark Adaptability

The reason TTS models can preserve our watermarks can theoretically be explained by the TTS training objective. Specifically, most, if not all, TTS model training minimizes the frequency domain difference between real and generated speech. For example, YourTTS and SV2TTS both minimize the ℓ_1 norm between real and generated mel spectrograms. This enables TTS models to learn frequency domain watermarks after fine-tuning and perserve the watermarks in synthesized speech. The ability of TTS models to learn frequency domain watermarks also explains the adaptability of our watermarks to different TTS models even though our training pipeline does not involve any TTS models.

8.2 Limitations and Future Work

In this work, we focused on protecting speakers from voice cloning in a closed set. While this strategy can protect voice sources, e.g., public voice datasets, which contain a fixed number of speakers, our current method cannot protect unseen speakers. This presents limitations in a scenario with new speakers that requires real-time watermarking. For example, a phone company may need to watermark phone conversations in real time to prevent adversaries from using recorded conversations to generate fake speech.

To improve the generalizability of our method, a potential approach is to incorporate speaker embeddings in the watermark generation pipeline. Speaker embeddings generalize to speakers not included in the training data and they can be obtained from well-trained speaker verification models, e.g., RawNet3. Hence, speaker embeddings can be exploited for generating watermarks for unseen speakers.

We have not investigated the scenario where watermarked speech is played through a physical speaker and received by a microphone. Existing work [40] has demonstrated that using Room Impulse Response (RIR) to augment the training data improves robustness in a physical environment. We leave these as topics for future work.

9 Conclusion

In this work, we propose a novel watermarking technique to detect fake speech generated by TTS models. Experimental results show that our watermarks transfer to TTS models if an adversary applies speaker adaption using our watermarked speech. This is because fine-tuning a TTS model using our watermarked speech results in the TTS model learning our watermark patterns. This results in the generated fake speech containing our watermarks, which enables fake speech detection in a reliable and explainable manner. In addition to open source TTS models, our watermarks are also transferrable to commercial TTS models that provide voice cloning services. This demonstrates the generalizability and practicality of our method in the real world. We intend to improve the learnability of our watermarks and the generalization to unseen speakers in future work.

Ethics Considerations

To conduct experiments with the live systems, namely, PlayHT and Speechify, we created valid accounts under their free voice cloning plan and all experiments were conducted using the allowed quota. Although our watermarks transferred to the vendors' models, this does not affect other users because our fine-tuned models are private and not shared with other users. Hence, this research did not affect other users. Moreover, while the results of our study could impact the service providers' business, if there are significant use cases that violate free speech, we believe in the importance of free speech. Furthermore, our research did not reveal any technical flaws with the live systems, nor did we intend to do so.

On the other hand, the community can benefit from utilizing our method to deter adversaries from generating fake speech on live systems for malicious intent. This is especially helpful for people who contribute to public voice datasets. Service providers may also incorporate our techniques in their products to ensure the responsible disclosure of their generative models.

Based on the discussion above, we believe that all experiments in this work were conducted ethically and the potential benefits of this research outweigh the potential harms.

Open Science

This research complies with the open science policy. We have made the following artifacts available online to replicate our work[7]. More details about the repository can be found in "readme.txt".

- Code and scripts: Python code to create and train our model from scratch; Python code to preprocess the VCTK dataset; Python code to generate all the results reported in this paper.
- Data sets: Preprocessed Obama's speech; examples of fake speech generated by YourTTS and SV2TTS; all the fake speech generated by Speechify and PlayHT.
- Binaries: Pre-trained parameters of our model.
- Secondary artefacts: Pre-trained YourTTS and SV2TTS parameters; Python code to fine-tune YourTTS and SV2TTS given a list of speech; pre-trained parameters of 10-bit Timbre Watermarking; code to evaluate Timbre Watermarking on VCTK testing data; pre-trained RawNet3 parameters; Python code to apply RawNet3 for speaker verification; Pre-trained AASIST parameters; Python code to apply AASIST for fake speech detection.

We will not release the original VCTK dataset because it is publicly available, but we will release Python code to preprocess it.

[7] https://zenodo.org/records/14722182.

We will not release the complete set of fake speech generated by YourTTS and SV2TTS because the complete set, including fake speech corresponding to various fine-tuning iterations, is too large to be uploaded online. Instead, we will release the Python code for generating fake speech using YourTTS and SV2TTS.

The following is a checklist to replicate all reported results in this work.

- Tables 1, 2 and 3: all watermarked VCTK data and fake speech generated by PlayHT and Speechify will be released. Python code and pre-trained parameters to replicate the results will be released.
- Tables 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18: Python code, data and pre-trained parameters to replicate all the results will be released.
- Figures 4, 5, 8, 9, 10 and 11: the pre-trained model parameters will be released. Python code to replicate the results will be released.
- Figures 6 and 7: Python code, preprocessed Obama's speech and pre-trained model parameters to replicate the results will be released.
- Figure 12: the corresponding audio files will be released.
- Python code and pre-trained model parameters to replicate results that are not reported in figures or tables will be released, i.e., results reported in Section 7.12, Section 7.13 and Section 7.14.

References

[1] Vishal Asnani, Xi Yin, Tal Hassner, Sijia Liu, and Xiaoming Liu. Proactive image manipulation detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15386–15395, 2022.

[2] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[3] Zexin Cai, Weiqing Wang, and Ming Li. Waveform boundary detection for partially spoofed audio. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[4] Nicholas Carlini. Poisoning the unlabeled dataset of {Semi-Supervised} learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1577–1592, 2021.

[5] Edresson Casanova, Julian Weber, Christopher D Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A Ponti. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International Conference on Machine Learning*, pages 2709–2720. PMLR, 2022.

[6] Guangyu Chen, Yu Wu, Shujie Liu, Tao Liu, Xiaoyong Du, and Furu Wei. Wavmark: Watermarking for audio generation. *arXiv preprint arXiv:2308.12770*, 2023.

[7] Yongbaek Cho, Changhoon Kim, Yezhou Yang, and Yi Ren. Attributable watermarking of speech generative models. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3069–3073. IEEE, 2022.

[8] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

[9] Dr Lance B Eliot. The need for explainable ai (xai) is especially crucial in the law. *Available at SSRN 3975778*, 2021.

[10] Yihao Huang, Felix Juefei-Xu, Qing Guo, Yang Liu, and Geguang Pu. Fakelocator: Robust localization of gan-based face manipulations. *IEEE Transactions on Information Forensics and Security*, 17:2657–2672, 2022.

[11] Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*, 31, 2018.

[12] 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 *ICASSP 2022-2022 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6367–6371. IEEE, 2022.

[13] Jee-weon Jung, You Jin Kim, Hee-Soo Heo, Bong-Jin Lee, Youngki Kwon, and Joon Son Chung. Pushing the limits of raw waveform speaker recognition. In Hanseok Ko and John H. L. Hansen, editors, *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, pages 2228–2232. ISCA, 2022.

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

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

[17] Chang Liu, Jie Zhang, Tianwei Zhang, Xi Yang, Weiming Zhang, and Nenghai Yu. Detecting voice cloning attacks via timbre watermarking. In *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society, 2024.

[18] Zhiqiang Lv, Shanshan Zhang, Kai Tang, and Pengfei Hu. Fake audio detection based on unsupervised pretraining models. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 9231–9235. IEEE, 2022.

[19] Momina Masood, Mariam Nawaz, Khalid Mahmood Malik, Ali Javed, Aun Irtaza, and Hafiz Malik. Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward. *Applied Intelligence*, 53(4):3974–4026, 2023.

[20] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: A large-scale speaker identification dataset. In Francisco Lacerda, editor, *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 2616–2620. ISCA, 2017.

[21] 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.

[22] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[23] Yanzhen Ren, Hongcheng Zhu, Liming Zhai, Zongkun Sun, Rubing Shen, and Lina Wang. Who is speaking actually? robust and versatile speaker traceability for voice conversion. *arXiv preprint arXiv:2305.05152*, 2023.

[24] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, volume 2, pages 749–752. IEEE, 2001.

[25] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1–11, 2019.

[26] Robin San Roman, Pierre Fernandez, Hady Elsahar, Alexandre Défossez, Teddy Furon, and Tuan Tran. Proactive detection of voice cloning with localized watermarking. In *International Conference on Machine Learning*, volume 235, 2024.

[27] 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 *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE, 2018.

[28] Jonathan Sterne. *MP3: The meaning of a format*. Duke University Press, 2012.

[29] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[30] Jean-Marc Valin, Gregory Maxwell, Timothy B Terriberry, and Koen Vos. High-quality, low-delay music coding in the opus codec. *arXiv preprint arXiv:1602.04845*, 2016.

[31] Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al. Superseded-cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. 2016.

[32] Run Wang, Felix Juefei-Xu, Meng Luo, Yang Liu, and Lina Wang. Faketagger: Robust safeguards against deepfake dissemination via provenance tracking. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3546–3555, 2021.

[33] 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. *arXiv preprint arXiv:2109.00537*, 2021.

[34] Jiangyan Yi, Ruibo Fu, Jianhua Tao, Shuai Nie, Haoxin Ma, Chenglong Wang, Tao Wang, Zhengkun Tian, Ye Bai, Cunhang Fan, et al. Add 2022: the first audio deep synthesis detection challenge. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 9216–9220. IEEE, 2022.

[35] Jiangyan Yi, Jianhua Tao, Ruibo Fu, Xinrui Yan, Chenglong Wang, Tao Wang, Chu Yuan Zhang, Xiaohui Zhang, Yan Zhao, Yong Ren, Le Xu, Junzuo Zhou, Hao Gu, Zhengqi Wen, Shan Liang, Zheng Lian, Shuai Nie, and Haizhou Li. Add 2023: the second audio deepfake detection challenge, 2023.

[36] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 14448–14457, 2021.

[37] Ning Yu, Vladislav Skripniuk, Dingfan Chen, Larry S. Davis, and Mario Fritz. Responsible disclosure of generative models using scalable fingerprinting. In *International Conference on Learning Representations*, 2022.

[38] Zhiyuan Yu, Shixuan Zhai, and Ning Zhang. Antifake: Using adversarial audio to prevent unauthorized speech synthesis. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 460–474, 2023.

[39] Linghan Zhang, Sheng Tan, and Jie Yang. Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 57–71, 2017.

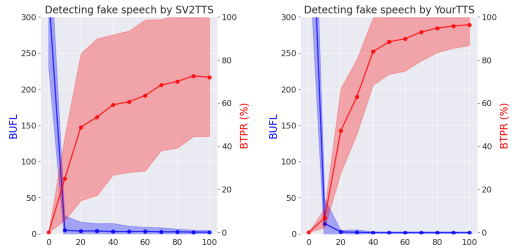
[40] Wei Zong, Yang-Wai Chow, Willy Susilo, Kien Do, and Svetha Venkatesh. Trojanmodel: A practical trojan attack against automatic speech recognition systems. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1667–1683. IEEE, 2023.

10 Appendix

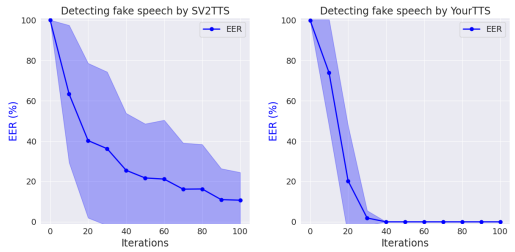
10.1 Speech Quality

Perceptual Evaluation of Speech Quality (PESQ) was initially developed as an evaluation metric for automatically measuring degradation in speech in the context of telephony [24]. The values of PESQ range between 1.0 to 4.5. Higher PESQ values represent better quality.

Signal-to-noise Ratio (SNR) is another metric widely used by researchers for evaluating distortions introduced by noise [6]. Thus, we used SNR to complement PESQ.

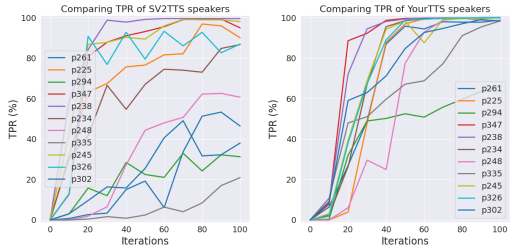


(a) SV2TTS BUFL & BTPR (b) YourTTS BUFL & BTPR

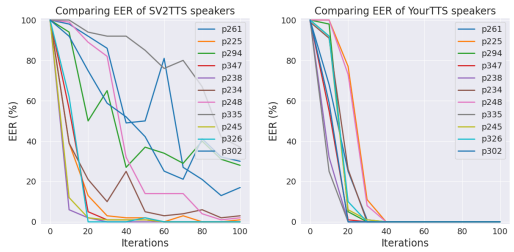


(c) SV2TTS BEER (d) YourTTS BEER

Figure 10: BUFL, BTPR, and BEER comparisons when fine-tuning SV2TTS and YourTTS on our watermarked data. This figure shows mean and standard deviation values.



(a) SV2TTS TPR (b) YourTTS TPR



(c) SV2TTS EER (d) YourTTS EER

Figure 11: TPR and EER change comparisons when fine-tuning SV2TTS and YourTTS for each speaker. The difficulty of learning watermarks varies for different speakers.

10.2 TTS Models

SV2TTS uses a pipelined TTS architecture to generate fake speech. It consists of 3 sub models: a LSTM-based encoder, Tacotron 2 [27] as the synthesizer, and WaveNet vocoder [21] for converting mel spectrograms to waveforms. Our experiments relied on the widely used open source implementation and pretrained models[8].

YourTTS was recently proposed by Casanova et al. [5]. Unlike the pipelined architecture of SV2TTS, YourTTS is an end-to-end TTS that does not split speech generation into different stages. YourTTS is a complex TTS system consisting of multiple sub models, such as a transformer-based text encoder [14], a HiFi-GAN vocoder [16], and a variational autoencoder (VAE) [15]. We used the YourTTS model pretrained on the VCTK dataset [31] provided by Casanova et al. [5] for replicating their "System 1"[9].

PlayHT and Speechify are commercial TTS services with state-of-the-art performance. They both allow users to clone a target voice for free, albeit with a character limit. The internal workings of these 2 models are not released to the public, but it is highly unlikely that they have the same architecture and the same training data set.

[8] https://github.com/CorentinJ/Real-Time-Voice-Cloning
[9] https://github.com/Edresson/YourTTS/

10.3 Visualization

Fig. 12 visualizes how our watermark patterns are preserved in fake speech. SV2TTS and YourTTS were fine-tuned on our watermarked speech for 100 iterations, while the settings at which PlayHT and Speechify adapted to our watermarked speech are not disclosed to the public. Looking closely, changes in our watermark patterns along the time axis are small. The simple and consistent patterns are favorable for a TTS model to learn our watermarks.

10.4 Non-adaptive Attacks

The details of time domain attacks are as follows:

White Noise (WN) attack: adds Gaussian noise, with a mean of 0 and standard deviation of 0.005, to speech.

Resampling (RS) attack: first down-samples audio to an 8,000 sampling rate, before up-sampling it back to a 16,000 sampling rate.

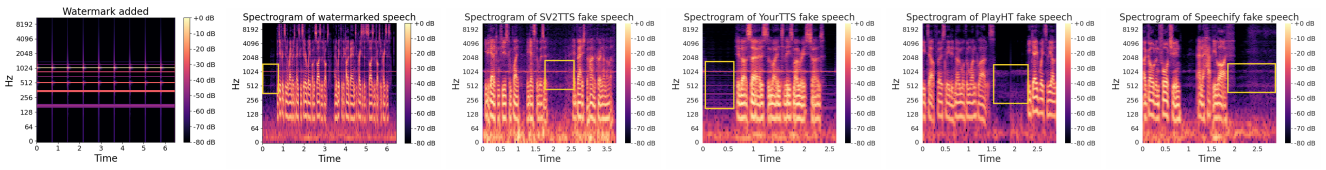
Median Filter (MF) attack: applies a filter, with a kernel size of 5, to smoothen audio.

Sample Suppression (SS) attack: randomly set 3% of samples to 0.

Amplitude Scaling (AS) attack: scales audio amplitudes to 70% of their original values.

Quantization (QT) attack: quantizes samples to the 2^8 level.

Echo Addition (EA) attack: scales the original samples to 30%, delays the scaled samples by 0.1 seconds, and adds



(a) Watermarks (b) Watermarked Speech (c) SV2TTS (d) YourTTS (e) PlayHT (f) Speechify

Figure 12: Watermarked and fake speech of speaker "p326". Our watermark patterns are clearly present in the generated speech from all TTS models (shown in the yellow rectangles). Fake speech generated by Speechify is an exception because our watermark patterns are not clear.

Table 12: EER (%) for time domain attacks.

	WN	**RS**	**MF**	**SS**	**AS**	**QT**	**EA**
Watermarked	2.74 ± 5.16	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	**28.06 ± 26.97**	0.09 ± 0.12
SV2TTS	21.73 ± 26.56	10.73 ± 13.78	14.91 ± 17.63	13.64 ± 17.44	11.91 ± 15.63	**28.09 ± 29.69**	16.73 ± 19.21
YourTTS	1.91 ± 3.63	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	**24.27 ± 30.36**	0.00 ± 0.00
PlayHT*	63.64	45.45	45.45	54.55	45.45	**90.91**	36.36
Speechify*	63.64	**81.82**	**81.82**	63.64	**81.82**	**81.82**	27.27

*: results are calculated based on the speaker level.

Table 13: BEER (%) for time domain attacks.

	WN	**RS**	**MF**	**SS**	**AS**	**QT**	**EA**
Watermarked	2.12 ± 3.81	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	**27.22 ± 26.88**	0.05 ± 0.10
SV2TTS	19.00 ± 26.02	9.82 ± 12.78	13.73 ± 17.21	12.00 ± 16.36	11.73 ± 15.54	**23.18 ± 27.83**	15.73 ± 18.66
YourTTS	1.73 ± 3.67	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	**23.82 ± 30.38**	0.00 ± 0.00
PlayHT*	54.55	18.18	27.27	36.36	36.36	**81.82**	9.09
Speechify*	45.45	54.55	54.55	45.45	**72.73**	54.55	27.27

*: results are calculated based on the speaker level.

Table 14: EER (%) for frequency domain attacks.

	MP3	**Opus**	**LP**	**HP**	**NR**
Watermarked	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	**15.34 ± 30.60**	0.02 ± 0.07
SV2TTS	10.73 ± 13.68	14.73 ± 17.75	10.82 ± 13.90	**48.55 ± 41.14**	37.00 ± 35.86
YourTTS	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	**15.82 ± 31.88**	0.00 ± 0.00
PlayHT*	45.45	54.55	45.45	**81.82**	72.73
Speechify*	72.73	72.73	81.82	81.82	**100.00**

*: results are calculated based on the speaker level.

Table 15: BEER (%) for frequency domain attacks.

	MP3	**Opus**	**LP**	**HP**	**NR**
Watermarked	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	**2.66 ± 6.56**	0.02 ± 0.07
SV2TTS	10.00 ± 13.16	13.18 ± 16.56	9.91 ± 12.91	16.36 ± 16.41	**36.18 ± 36.29**
YourTTS	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	**1.55 ± 4.58**	0.00 ± 0.00
PlayHT*	18.18	**54.55**	18.18	27.27	**54.55**
Speechify*	45.45	63.64	54.55	72.73	**81.82**

*: results are calculated based on the speaker level.

Table 16: EER (%) for the band-stop filter attack.

	100 Hz	200 Hz	400 Hz
Watermarked	11.58 ± 4.06	36.30 ± 11.41	**64.09 ± 22.86**
SV2TTS	31.09 ± 16.88	57.45 ± 14.66	**81.82 ± 15.94**
YourTTS	9.36 ± 3.70	36.36 ± 9.22	**65.45 ± 22.51**
PlayHT	45.45	**90.91**	81.82
Speechify	**81.82**	63.64	**81.82**

Table 17: BEER (%) for the band-stop filter attack.

	100 Hz	200 Hz	400 Hz
Watermarked	5.85 ± 3.75	19.68 ± 8.49	**36.86 ± 11.30**
SV2TTS	21.82 ± 16.25	38.18 ± 13.12	**57.55 ± 11.70**
YourTTS	4.82 ± 3.93	18.00 ± 8.75	**31.82 ± 10.79**
PlayHT	27.27	**63.64**	54.55
Speechify	**54.55**	45.45	45.45

them back to the original samples.

The details of frequency domain attacks are as follows:

Lossy Compression (LC) attack: compresses audio with MP3 [28] or Opus [30] compression. These two compression schemes are widely used in industry. In the experiments, we used low-quality compression, i.e., 96 kbits/s for MP3 and 24 kbits/s for Opus.

Low-pass Filtering (LP) attack: uses a low-pass filter to filter out signals above 3.7 kHz.

High-pass Filtering (HP) attack: uses a high-pass filter to filter out signals below 200 Hz. It should be noted that the power of the human voice is concentrated at low frequencies. Hence, a high-pass filter with a cutoff frequency larger than 200 Hz can make voices noticeably different, especially deeper voices, e.g., male voices.

Noise Reduction (NR) attack: reduces background noise in input audio by computing its spectrograms and a noise threshold for each frequency band. Our experiments relied on an open source implementation of noise reduction algorithms[10].

10.5 Additional Results for Non-adaptive Attacks

The EER and BEER results shown in Tables 12, 13, 14 and 15 are consistent with their corresponding UFL and BUFL results. Overall, using BEER resulted in better robustness to all the attacks. QT was the most detrimental time domain attack against our watermarks. Of the frequency domain attacks, HP and NR attacks were the most effective at destroying our watermarks. Eventhough NR destroyed our watermarks in relation to EER, defenders can still use BEER to potentially detect watermarks in processed speech, since it managed to detect fake speech for 2 out of the 11 speakers.

[10] https://pypi.org/project/noisereduce/

Table 18: YouTTS speaker adaptation for 100 iterations using a mixture of original and watermarked speech.

Ratio*	UFL	TPR (%)	EER (%)	Succ$^+$
80%	1.8 ± 1.2	82.57 ± 19.01	0.00 ± 0.00	11/11
60%	7.4 ± 43.1	17.45 ± 13.98	39.55 ± 38.37	9/11
40%	24.6 ± 95.2	5.05 ± 10.49	82.73 ± 33.00	6/11
20%	29.3 ± 116.7	2.37 ± 7.49	99.91 ± 0.29	2/11

*: percentage of watermarked speech.
$^+$: number of speakers for which our watermarks managed to transfer to YourTTS in terms of BUFL.

10.6 Additional Results for Adaptive Attacks

Additional results for adaptive attacks are shown in Tables 16, 17 and 18.

10.7 Training Denoising Autoencoder

We let the denoising autoencoder learn an inverse function of our watermarking scheme:

$$g(x' + p(x')) = g(x) \tag{12}$$

where p represents the denoising autoencoder. x and x' are original and watermarked speech, respectively. g represents the decoder in our watermarking scheme. The denoising autoencoder only focuses on frequency components between 100 Hz and 1,000 Hz because we only apply watermarks within this range. Equation 12 shows that adding output from the denoising autoencoder to the watermarked speech can remove our watermarks.

The loss function to train the denoising autoencoder is:

$$\ell_{ae} = \|\mathcal{F}(x' + p(x')) - \mathcal{F}(x)\|_2 \tag{13}$$

where $\mathcal{F}$ is the function to transform waveform into features accepted by decoder g. Equation 13 shows that optimizing ℓ_{ae} makes the denoising autoencoder able to inverse our watermarks in terms of ℓ_2 norm.

The denoising autoencoder was trained on the paired data for 200 epochs. The model with the lowest loss was saved for evaluation. The adversary then uses this denoising autoencoder to remove watermarks from the victims' watermarked speech.