Who Are You (I Really Wanna Know)? Detecting Audio DeepFakes Through Vocal Tract Reconstruction

Logan Blue, Kevin Warren, Hadi Abdullah, Cassidy Gibson, Luis Vargas, Jessica O'Dell, Kevin Butler, Patrick Traynor

University of Florida, Gainesville, FL
Email: {bluel, kwarren9413, hadi10102, c.gibson, lfvargas14, odelljessica, butler, traynor}@ufl.edu

Abstract

Generative machine learning models have made convincing voice synthesis a reality. While such tools can be extremely useful in applications where people consent to their voices being cloned (e.g., patients losing the ability to speak, actors not wanting to have to redo dialog, etc), they also allow for the creation of nonconsensual content known as deepfakes. This malicious audio is problematic not only because it can convincingly be used to impersonate arbitrary users, but because detecting deepfakes is challenging and generally requires knowledge of the specific deepfake generator. In this paper, we develop a new mechanism for detecting audio deepfakes using techniques from the field of articulatory phonetics. Specifically, we apply fluid dynamics to estimate the arrangement of the human vocal tract during speech generation and show that deepfakes often model impossible or highly-unlikely anatomical arrangements. When parameterized to achieve 99.9% precision, our detection mechanism achieves a recall of 99.5%, correctly identifying all but one deepfake sample in our dataset. We then discuss the limitations of this approach, and how deepfake models fail to reproduce all aspects of speech equally. In so doing, we demonstrate that subtle, but biologically constrained aspects of how humans generate speech are not captured by current models, and can therefore act as a powerful tool to detect audio deepfakes.

1 Introduction

The ability to generate synthetic human voices has long been a dream of scientists and engineers. Over the past 50 years, techniques have included comprehensive dictionaries of spoken words and formant synthesis models that create new sounds through the combination of frequencies. While such techniques have made important progress, their outputs are generally considered robotic-sounding and easily distinguishable from organic speech. Recent advances in generative machine-learning models have led to dramatic improvements in synthetic speech quality, with convincing voice reconstruction now available to groups including patients suffering from the loss of speech due to medical conditions and grieving family members of the recently deceased [1, 2].

While these speech models are a powerful and important enabler of communication, they also create significant problems for users who have not given their consent. Specifically, generative machine learning models now make it possible to create audio *deepfakes*, which allow an adversary to simulate a targeted individual speaking arbitrary phrases. While public individuals have long been impersonated, such tools make impersonation scalable, putting the general population at risk. Such attacks have reportedly been observed in the wild, including a company that allowed an attacker to instruct funds to be sent to them using generated audio of the victim company's CEO's voice [3]. In response, researchers have developed detection techniques using bispectral analysis (i.e., inconsistencies in the higher-order correlations in audio) [4,5] and training machine learning models as discriminators [6]; however, both are highly dependent on the specific, previously observed generation techniques to be effective.

In this paper, we develop techniques to detect deepfake audio samples by solely relying on limitations of human speech that are the results of biological constraints. Specifically, we leverage research in articulatory phonetics to apply fluid dynamic models that estimate the arrangement of the human vocal tract during speech. Our analysis shows that deepfake audio samples are not fundamentally constrained in this fashion, resulting in vocal tract arrangements that are inconsistent with human anatomy. Our work demonstrates that this inconsistency is a reliable detector for deepfake audio samples.

We make the following contributions:

- **Identify inconsistent vocal tract behavior:** Using a combination of fluid dynamics and articulatory phonetics, we identify the inconsistent behavior exhibited by deepfaked audio samples (e.g., unnatural vocal tract diameters). We develop a technique to estimate the vocal tract during speech to prove this phenomenon.
- **Constructing a deepfake detector:** After proving the existence of the phenomena, we construct a deepfake

detector that is capable of detecting deepfake audio (Precision: 99.9%, Recall: 99.5%) from a large dataset we create using the Real-Time Voice Cloning generator [7]. Finally, we also demonstrate that entries from the ASVSpoof2019 dataset are easily detectable in the prefiltering portion of our mechanism due to a high word error rate in automatic transcription.

- **Analysis of deepfake detector:** We further analyze which vocal tract features and portions of speech cause the deepfakes to be detectable. From this analysis, we determine that on average our detector only requires a single sentence to detect a deepfake with a true positive rate (TPR) of 92.4%.
- **Analysis of Potential Adaptive Adversaries:** We conducted two large-scale experiments emulating both a naïve and an advanced adaptive adversary. Our experiments consist of training 28 different models and show that in the best case, an adaptive adversary faces greater than a 26x increase in training time, increasing the approximate time necessary to train the model to over 130 days.

We note that the lack of anatomical constraints is consistent across all deepfake techniques. Without modeling the anatomy or forcing the model to operate within these constraints, the likelihood that a model will learn a biologically appropriate representation of speech is near zero. Our detector, therefore, drastically reduces the number of possible models that can practically evade detection.

The paper is organized as follows: Section 2 provides context by discussing related work; Section 3 gives background on relative topics used throughout this paper; Section 4 discusses our underlying hypothesis; Section 5 details our threat model; Section 6 explains our methodology and detection method; Section 7 describes our data collection and experimental design; Section 8 discusses the results of our experiments; Section 9 details the intricacies and consequences of our work; and Section 10 provides concluding remarks.

2 Related Work

Advances in Generative Adversarial Networks (GANs) have enabled the generation of synthetic "human-like" audio that is virtually indistinguishable from audio produced by a human speaker. In some cases, the high quality of GAN-generated audio has made it difficult to ascertain whether the audio heard (e.g., over a phone call) was organic [8]. This has enabled personalized services such as Google Assistant [9], Amazon Alexa [10], and Siri [11], which use GAN-generated audio to communicate with users. GANs can also be trained to impersonate a specific person's audio, this kind of audio is known as a deepfake [12].

The dangerous applications of deepfake audio have spurred the need to automatically identify human audio samples from deepfakes. Some of the current work in this area has focused on identifying subtle spectral differences that are otherwise imperceptible to the human ear [4,5]. In some cases, the deepfake audio will be played over a mechanical speaker, which will itself leak artifacts into the audio sample. These artifacts can be detected using a variety of techniques such as machine learning models [13, 14], additional hardware sensors [15], or spectral decomposition [16]. Researchers have also tried to detect these artifacts by using mobile devices. They use differences in the time-of-arrival in phoneme sequences, effectively turning the mobile devices into a Doppler Radar that could verify the audio source [17, 18]. These techniques fall within the category of liveness detection and have spawned major competitions such as the ASV Spoof Challenge [19]. However, these methods have certain limitations including the distance of the speaker from the recording microphone, accuracy, additional hardware requirements, and large training sets. Phonetics (the scientific study of speech sounds) is commonly used by language models for machine learning systems built for speech to text [20,21] or speaker recognition [22]. Speech recognition and audio detection tools also use phonetics to increase their overall performance and capabilities [23, 24]. While articulatory phonetics is not commonly used in security, this has been used in past work, such as reconstructing encrypted VoIP calls by identifying phonemes [25].

Using concepts of articulatory phonetics, our work attempts to extract the physical characteristics of a speaker from a given audio sample; these characteristics would otherwise not be present in deepfake audio. Human or organic speech is created using a framework of muscles and ligaments around the vocal tract. The unique sound of each of our voices is directly tied to our anatomy [26]. This has enabled researchers to use voice samples of a speaker to extract the dimensions of their anatomical structures such as vocal tract length [27–31], age [32], or height [33, 34] of the speaker. These works attempt to derive an acoustical pipe configuration by modeling the human pharynx. This configuration can then be used as a proxy for the human anatomy to retrieve the physical characteristics of the speaker. Since deepfakes are generated using GANs, the physical dimensions are likely inconsistent. This inconsistency can be measured and help differentiate between deepfake and human audio samples.

3 Background

3.1 Phonemes

Phonemes are the fundamental building blocks of speech. Each unique phoneme sound is a result of different configurations of the vocal tract components shown in Figure 1. Phonemes that comprise the English language are categorized into vowels, fricatives, stops, affricates, nasals, glides, and diphthongs (Table 1).

Vowels (e.g., "/ɪ/" in ship) are created using different ar-

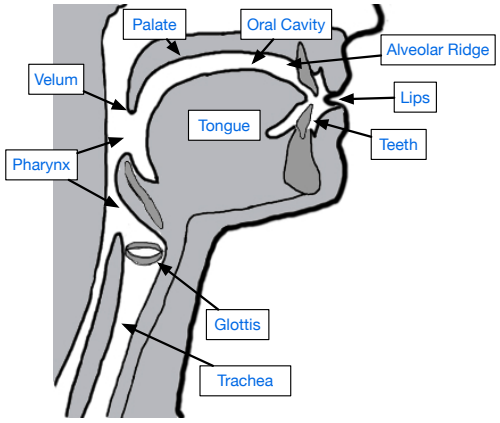


Figure 1: The vocal tract is composed of various components that act together to produce sounds. Distinct phonemes are articulated based on the path the air travels, which is determined by how the components are positioned.

rangements of the tongue and jaw, which result in resonance chambers within the vocal tract. For a given vowel, these chambers produce frequencies known as formants whose relationship determines the actual sound. Vowels are the most commonly used phoneme type in the English language, making up approximately 38% of all phonemes [35]. Fricatives (e.g.,"/s/" in sun) are generated by turbulent flow caused by a constriction in the airway, while stops (e.g.,"/g/" in gate) are created by briefly halting and then quickly releasing the airflow in the vocal tract. Affricatives (e.g.,"/tʃ/" in church) are a concatenation of a fricative with a stop. Nasals (e.g.,"/n/" in nice) are created by forcing air through the nasal cavity and tend to be at a lower amplitude than the other phonemes. Glides (e.g.,"/l/" in lie) act as a transition between different phonemes, and diphthongs (e.g.,"/eɪ/" in wait) refer to the vowel sound that comes from the lips and tongue transitioning between two different vowel positions.

Phonemes alone do not encapsulate how humans speak. The transitions between two phonemes are also important for speech since it is a continuous process. Breaking speech down into pairs of phonemes (i.e., bigrams) preserves the individual information of each phoneme as well as transitions between them. These bigrams generate a more accurate depiction of the vocal tract dynamics during the speech process.

3.2 Organic Speech

Human speech production results from the interactions between different anatomical components, such as the lungs, larynx (i.e., the vocal cords), and the articulators (e.g., the tongue, cheeks, lips), that work in conjunction to produce sound. The production of sound[1] starts with the lungs forcing air through the vocal cords, which induces an acoustic resonance that contains the fundamental (lowest) frequency of a speaker's voice. The resonating air then moves through the vocal cords and into the vocal tract (Figure 1). At this point, different configurations of the articulators (e.g., where the tongue is placed, how large the mouth is) shape the path for the air to flow, which creates constructive/destructive interference that produces the unique sounds of each phoneme.

[1]This process is similar to how trumpets create a sound as air flows through various pipe configurations.

Phoneme Type	Phoneme	Example
Vowel	/ɪ/	ship
Fricative	/s/	sun
Stop	/g/	gate
Affricative	/tʃ/	church
Nasal	/n/	nice
Glide	/l/	lie
Diphthong	/eɪ/	wait

Table 1: English is composed of these seven categories of phonemes. Their pronunciation is dependent on the configuration of the various vocal tract components and the airflow that goes through it.

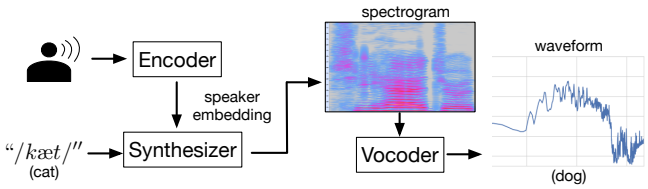


Figure 2: Deepfake generation has several stages to create a fake audio sample. The encoder generates an embedding of the speaker, the synthesizer creates a spectrogram for a targeted phrase using the speaker embedding, and the vocoder converts the spectrogram into the synthetic waveform.

3.3 Deepfake Audio

Deepfakes are digitally produced speech samples that are intended to sound like a specific individual. Currently, deepfakes are produced via the use of machine learning (ML) algorithms. While there are numerous deepfake ML algorithms in existence, the overall framework the techniques are built on are similar. As shown in Figure 2, the framework is comprised of three stages: encoder, synthesizer, and vocoder.

Encoder: The encoder learns the unique representation of the speaker's voice, known as the speaker embedding. These can be learned using a model architecture similar to that of speaker verification systems [36]. The embedding is derived from a short utterance using the target speaker's voice. The accuracy of the embedding can be increased by giving the encoder more utterances, with diminishing returns. The output embedding from the encoder stage is passed as an input into the following synthesizer stage.

Synthesizer: A synthesizer generates a Mel Spectrogram

from a given text and the speaker embedding. A Mel Spectrogram is a spectrogram that has its frequencies scaled using the Mel scale, which is designed to model audio perception of the human ear. Some synthesizers can produce spectrograms solely from a sequence of characters or phonemes [37].

Vocoder: Lastly, the vocoder converts the Mel Spectrogram to retrieve the corresponding audio waveform. This newly generated audio waveform will ideally sound like a target individual uttering a specific sentence. A commonly used vocoder model is some variation of WaveNet [38], which uses a deep convolutional neural network that uses surrounding contextual information to generate its waveform.

Although the landscape of audio generation tools is ever-changing, these three stages are the foundational components of the generation pipeline. The uniqueness of each tool is derived mainly from the quality of models (one for each stage) and the exact design of their system architecture.

4 Hypothesis

Human-created speech is fundamentally bound to the anatomical structures that are used to generate it. Only certain arrangements of the vocal tract are physically possible for a speaker to create. The number of possible acoustic models that can accurately reflect both the anatomy and the acoustic waveform of a speaker is therefore limited. Alternatively, synthetic audio is not restricted by any physical structures during its generation. Therefore, an infinite set of acoustic models could have generated the synthetic audio. The details of this phenomenon will be discussed shortly in Section 6. It is highly improbable that models used to generate synthetic audio will mimic an acoustic model that is consistent with that of an organic speaker. As such, synthetic audio can be detected by modeling the acoustic behavior of a speaker's vocal tract.

5 Security Model

Our security model consists of an adversary, a victim, and a defender. The goal of the adversary is to create a deepfake audio sample of the victim uttering a specific phrase. We assume a powerful adversary, one who has access to enough of the victim's audio and enough computing power to generate a deepfake sample.

The adversary offers the defender either the deepfake or an organic audio sample. The defender is tasked with ascertaining whether the adversary-provided sample is deepfake or organic audio. If the defender makes the correct assessment, the adversary loses.

The defender does not have knowledge of, or audio data from, the victim the adversary will attempt to impersonate (i.e., no user-specific audio samples of the victim). The defender also has no knowledge of, or access to, the attacker's audio generation algorithm. This is a stronger threat model than existing works in the area, which often use very large training data sets (order of thousands of audio samples) [6]. Lastly, we assume that the defender wants an explanation as to why their detection system flagged a sample as either deepfake or organic.

A practical example of this scenario is as follows. An adversary creates a deepfake of a local politician and releases it to the media to further some goal. The media is the defender in this scenario and must decide whether the adversary's audio sample is authentic. Once the authenticity of the audio sample has been checked the media can choose to either ignore or publish the audio. If the media publishes a synthetically generated audio sample, then the adversary has successfully victimized the politician by making them appear to have said something they did not. Additionally, any media outlet which publishes a synthetic audio sample could have its reputation damaged if it is later discovered that the audio sample was inauthentic. By leveraging our technique, the media outlet can prevent reporting inauthentic audio samples, thus preventing their loss of reputation and the victimization of the politician.

6 Methodology

Our technique requires a training set of organic audio and a small set of deepfake audio samples generated by the deepfake algorithm.[2] The process of determining the source of an audio sample (i.e., organic vs deepfake) can then be broken down into two logical steps:

- **Vocal Tract Feature Estimator:** First, we construct a mathematical model of the speaker's vocal tract based on the amplitudes of certain frequencies (commonly referred to as the frequency response) present in their voice during a specific pair of adjacent phonemes (i.e., bigram). This model allows us to estimate the cross-sectional area of the vocal tract at various points along the speaker's airway.
- **Deepfake Audio Detector:** Next, we aggregate the range of values found for each bigram-feature pair in our organic dataset. These values determine if the audio sample can be realistically produced by an organic speaker. This enables our system to discriminate between organic and deepfake samples. Additionally, we can isolate the bigram-feature pairs that best determine the source of an audio sample to create a set of ideal discriminators. These ideal discriminators allow us to optimally determine whether an unseen audio sample is a deepfake.

[2]The training data is a general set that does not require samples of the specific victim that is being targeted. Additionally, the training data required for this technique is significantly less than the data required with prior ML-based techniques.

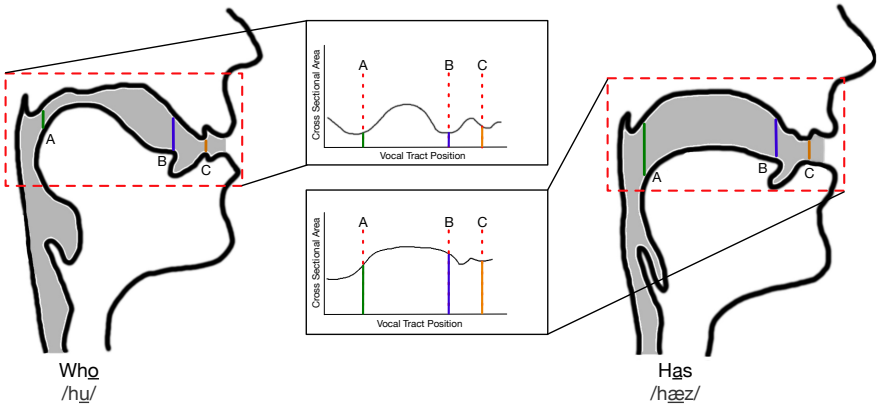


Figure 3: The sound produced by a phoneme is highly dependent on the structure of the vocal tract. Constriction made by tongue movement or jaw angle filters different frequencies.

6.1 Reader Participation

Before we go further into the details of these two steps, we would like to help the reader develops a deeper intuition of phonemes and speech generation.

For speech, air must move from the lungs to the mouth while passing through various components of the vocal tract. To understand the intuition behind our technique, we invite the reader to speak out loud the words "who" (phonetically spelled "/hu/") and "has" (phonetically spelled "/hæz/") while paying close attention to how the mouth is positioned during the pronunciation of each vowel phoneme (i.e., "/u/" in "who" and "/æ/" in "has").

Figure 3 shows how the components are arranged during the pronunciation of the vowel phonemes for each word mentioned above. Notice that during the pronunciation of the phoneme "/u/" in "who" the tongue compresses to the back of the mouth (i.e., away from the teeth) (A) at the same time, the lower jaw is held predominately closed. The closed jaw position lifts the tongue so that it is closer to the roof of the mouth (B). Both of these movements create a specific pathway through which the air must flow as it leaves the mouth. Conversely, the vowel phoneme "/æ/" in "has" elongates the tongue into a more forward position (A) while the lower jaw distends, causing there to be more space between the tongue and the roof of the mouth. This tongue position results in a different path for the air to flow through, and thus creates a different sound. In addition to tongue and jaw movements, the position of the lips also differs for both phonemes. For "/u/", the lips round to create a smaller more circular opening (C). Alternatively, "/æ/" has the lips unrounded, leaving a larger, more elliptical opening. Just as the tongue and jaw position, the shape of the lips also impacts the sound created.

One additional component that affects the sounds of a phoneme is the other phonemes that are adjacent to it. For example, take the words "ball" (phonetically spelled "/bɔl/"') and "thought" (phonetically spelled "/θɔt/"). Both words contain the phoneme "/ɔ/," however the "/ɔ/" in "thought" is affected by the adjacent phonemes differently than how "/ɔ/" in "ball" is. In particular "thought" ends with the plosive "/t/" which requires a break in airflow, thus causing the speaker to abruptly end the "/ɔ/" phoneme. In contrast, the "/ɔ/" in "ball" is followed by the lateral approximant "/l/," which does not require a break-in airflow, leading the speaker to gradually transition between the two phonemes.

6.2 Vocal Tract Feature Estimator

Based on the intuition built in the previous subsection, our modeling technique needs to be able to extract the shape of the vocal tract present during the articulation of a specific bigram. To do this, we use a fluid dynamic concatenated tube model to estimate the speaker's vocal tract that is similar to Rabiner et al.'s technique [27]. Before we go into the details of this model, it is important to discuss the assumption the model makes.

- **Lossless Model:** Our model ignores energy losses that result from the fluid viscosity (i.e., the friction losses between molecules of the air), the elastic nature of the vocal tract (i.e., the cross-sectional area changing due to a change in internal pressure), and friction between the fluid and the walls of the vocal tract. Ignoring these energy losses will result in our model having acoustic dampening, causing the lower formant frequencies to increase in value[3] and an increase in the bandwidth of all formant frequency spikes[4]. Additionally, we assume the walls of the vocal tract have an infinitely high acoustic impedance (i.e., sound can only exit the speaker from their mouth) which will result in our model missing trace amounts of low bass frequencies. Overall, these assumptions simplify the modeling processing while decreasing the accuracy of our technique by a marginal amount and are consistent with prior work [27].

[3] This effect is mainly caused by the elastic nature of the vocal tract walls.

[4] The viscosity and friction losses predominately effect frequencies above 4 kHz [27].

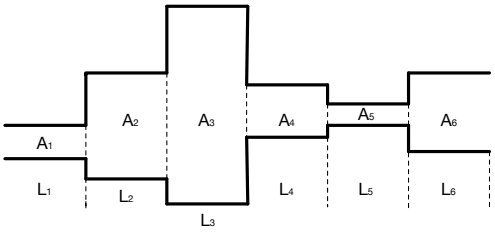


Figure 4: The cross sectional area of each pipe is calculated to characterize the speaker's voice tract.

- **Unidirectional Traveling Waves:** We assume that, within the model, we will only have traveling waves along the centerline of the tube. It stands to reason that this assumption is accurate enough for our model given the small diameter of our tubes (i.e., vocal tract). This assumption should not affect our results since any error caused by this assumption will most likely occur in frequencies greater than 20 kHz (far above human speech). As we will discuss later in this section, our model is most accurate for lower frequencies and thus we only analyze frequencies beneath 5 kHz.[5]
- **Vowel Limitation:** The model used in this paper was only designed to accurately model vowel phonemes. Other phonemes are generated via fundamentally different and more difficult model mechanisms such as turbulent flow. Despite this, we apply the same model across all bigrams throughout this work for reasons discussed in Section 9.1.

Our concatenated tube model consists of a series of open pipe resonators that vary in diameters but share the same length. A simplified representation can be seen in Figure 4.

To estimate the acoustics of an individual tube at a specific time during a bigram, we need to understand the behavior of pressure waves within the resonator. The simplest way to do this is to model the net volumetric flow rate of the fluid (i.e., the air in the vocal tract) within the resonator. We can model the acoustics of a resonator via the flow rate since the volumetric flow rate and the pressure (i.e., sound) within the resonator are directly related [27].

Modeling the interaction between two consecutive tubes is accomplished by balancing the volumetric inflows and outflows of the two tubes at their connection. Since the volumetric flow rate between two consecutive tubes must be equal, but the cross-sectional areas (and thus the volumes) may differ, there may exist a difference in fluid pressure between them. This pressure difference at the boundary results in a reflection coefficient, which affects the fluid flow rates between the two tubes. A schematic of the intersection between two tubes can be seen in Figure 5. Mathematically, the interactions between

[5] It is worth noting that most information in human speech is found below 5 kHz. It is also the reason why cellular codecs, such as those used in GSM networks, filter out noise in higher frequencies [39].

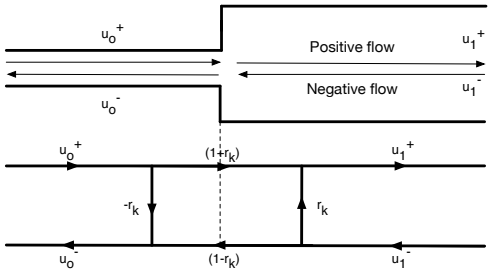


Figure 5: In our model we must account for airwaves being able to move in different directions within the vocal tract and anticipate how they interact with each other.

two consecutive pipes can be written as follows:

$$u_1^+ = u_0^+(1+r_k) + U_0^-(r_k) \tag{1}$$

$$u_0^- = u_1^-(1-r_k) + U_0^+(-r_k) \tag{2}$$

where u_0^+ and u_0^- is the forward and reverse volumetric flow rate in the left pipe, u_1^+ and u_1^- is the forward and reverse volumetric flow rate in the right pipe and r_k is the reflection coefficient between the two consecutive pipes.

The reflection coefficient r_k can be expressed as follows:

$$r_k = \frac{A_{k+1} - A_k}{A_{k+1} + A_k} \tag{3}$$

where A_{k+1} is the cross-sectional area of the tube that is downstream (i.e., further from the pressure source) in the tube series and A_k is the cross-sectional area of the tube that is upstream (i.e., closer to the pressure source) in the tube series. It should be noted that r_k is mathematically bound between -1 and 1. This bounding represents the scenarios where either A_k or A_{k+1} is infinitely larger than the next pipe adjacent to it.

Between these three equations, we can fully describe a single intersection between two tubes. Our vocal tract model consists of various tubes with multiple intersections being concatenated to form a series. To model this, we need to expand these equations to incorporate additional tube segments and intersections. In particular, we need to incorporate N connected tubes with $N-1$ intersections between them. The resulting differential equation is the transfer function of our N-segment tube series and when simplified is the following:

$$V(\omega) = \frac{0.5(1+r_G)\prod_{k=1}^{N}(1+r_k)e^{-LCNj\omega}}{D(\omega)} \tag{4}$$

$$D(\omega) = [1, -r_G]\begin{bmatrix} 1 & -r_1 \\ -r_1 e^{-2LCj\omega} & e^{-2LCj\omega} \end{bmatrix} \cdots \begin{bmatrix} 1 & -r_N \\ -r_N e^{-2LCj\omega} & e^{-2LCj\omega} \end{bmatrix} \begin{bmatrix} 1 \\ r_{Atm} \end{bmatrix} \tag{5}$$

where r_G is the reflection coefficient at the glottis, $r_1 \ldots r_N$ are the reflection coefficients for every consecutive tube pair in

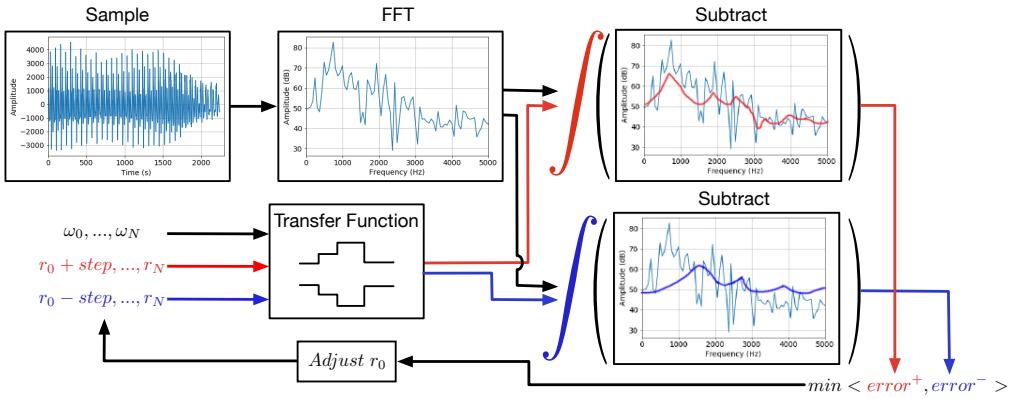


Figure 6: High-level overview of how the vocal tract feature estimator works. A speaker's audio sample (a single-window from a bigram) has its frequency data extracted using an FFT, the output of which is used as the target for our transfer function to reproduce. The transfer function is run twice over a range of frequencies $w_0, ..., w_N$. The first application of the transfer function uses the current reflection coefficients $r_0, ..., r_N$ with a step size offset added to a single coefficient. The second application instead subtracts the step size offset from the same single coefficient. The estimated frequency response curve calculated for both series is subtracted from the target curve. Whichever reflection coefficient results in a lower area under the resulting curve will be selected for the next iteration. This process continues (applying the step size offset to all of the reflection coefficients) until the area under the subtracted curves approach zero, indicating that we have found a reflection coefficient series that approximately replicates the original speaker's vocal tract.

the series, r_{Atm} is the reflection coefficient at the mouth, L is the length of each tube, C is the speed of sound (34,300 cm/s), j is the imaginary constant, and ω is the frequency of the waveform in rad/s. $V(\omega)$ is the volumetric flow rate at the lips during the pronunciation of a certain frequency, which is directly related to acoustic pressure (i.e., amplitude of the voice at frequency ω). We separate the denominator of Equation 4 out separately into Equation 5 for increased readability.

These equations together are a simplified representation of a system of $2N$ equations (N Equation 1's and N Equation 2's) that represents a series of N connected tube intersections (Figure 6). Since the volumetric flow rate through every tube within this series must be equal, we can simplify the $2N$ equations to Equation 4 and 5.

We refer the reader to Rabiner et al.'s work for a full derivation of these equations [27].

It is important to note that this differential equation lacks a closed-form solution and thus, we must specify several boundary conditions before solving the equation. Specifically, we must fix the number of tubes used in the series (N) and the reflection coefficients at both the beginning (r_G) and end of the series (r_{Atm}). This helps to more closely bind our equation to the physical anatomy from which it is modeled.

We can determine the number of tubes necessary for our model by taking the average human vocal tract length (approximately 15.5 cm [40]) and dividing by the length of each tube. This length, L, can be determined by the following equation (derivation of this equation can be found in Section 3.4.1 of Rabiner et al.'s work [27]):

$$L = \frac{TC}{2} \tag{6}$$

where T is the period between samples in our audio recordings. In our study, all of our audio samples had a sampling rate of 16 kHz. This sampling rate was selected since it captures the most important frequencies for speech comprehension and is also the most commonly found sampling rate for voice-based systems [41]. By sampling at 16 kHz, our vocal tract model will be made up of 15 distinct pipe resonators.

Next, we can use our understanding of human anatomy to fix the first reflection coefficient in the series (r_G in Equation 5). This reflection coefficient represents the fluid reflection that occurs at the speaker's glottis. During large portions of speech (e.g., during vowels) the glottis is actively being engaged. This means that the vocal folds are actively vibrating and thus preventing fluid flow in the reverse direction. With this in mind, we can set r_G to 1, symbolizing only fluid flow in the forward direction. Finally, the last reflection coefficient r_{Atm} is representing the behavior of the flow at the opening of the mouth. Here, once again, we can expect to see predominately only positive flow. This is because, during speech, the vocal tract is raised to a higher than atmospheric pressure, preventing flow from moving from the atmosphere back into the vocal tract. We can, therefore, set the last reflection coefficient r_{Atm} equal to 1.

With these boundary conditions, we now have a solvable differential equation that describes the acoustic behavior of our concatenated tube model. Using this equation we are now able to accurately estimate the amplitude of a certain frequency ω during a bigram for a known speaker (that has

a known $r_0, ..., r_N$ series). However, in our case, we do not know the dimensions of the speaker's vocal tract and cannot simply apply the transfer function. We do, however, have access to samples of the speaker's voice. Thus, we can use these audio samples and our transfer function to estimate the speaker's vocal tract during various articulations. The process of estimating a speaker's vocal tract can be seen in Figure 6.

The estimation is done by running a segment of a speaker's speech through a Fast Fourier Transform (FFT) to get the relative amplitudes for the frequencies that make up their voice. The found frequency response curve is effectively the output we would expect from the transfer function if we knew the speaker's $r_0, ..., r_N$ values. We can use the frequency response curve found with the FFT to check if a certain $r_0, ...r_N$ series correctly matches our speaker. We can, therefore, find an accurate approximation of a speaker's vocal tract by finding a $r_0, ..., r_N$ series that accurately reproduces the speaker's frequency response curve.

To avoid naïvely searching the entire $r_0, ..., r_N$ space for a match, we can instead construct an error function that can be optimized with gradient descent to find a good solution. Since gradient descent searches for a local minimum, we subtract the outputs from our transfer function from the frequency response curve found using the FFT. The transfer function is initially run with all reflection coefficients $r_0, ..., r_N$ set to zero. This is analogous to a constant diameter tube, which is a configuration achievable by the human vocal tract.[6] We then integrate the resulting curve to find the overall error between the two curves. As the output of our transfer function approaches the frequency response curve, the area between the two curves will approach zero and result in a local minimum. The $r_0, ..., r_N$ values used in the transfer function should approximate the speaker's vocal tract during that bigram.

Once we have found the optimal series of reflection coefficients, we can convert them into cross-sectional area estimates using Equation 3. This step requires us to make one last assumption about the vocal tract since there is one more cross-sectional area measurement than there are reflection coefficients (i.e., $N-1$ tube intersections). To mitigate this, we set the cross-sectional area at the glottis to the average size of a human glottis of $3.7cm^2$ [40]. With this assumption, we can then calculate the cross-sectional area series $a_0, ..., a_N$ that closely approximates the human vocal tract.

6.3 Deepfake Audio Detector

Using the vocal tract estimator we can design a generalized detector for deepfake audio. Our detector has two phases. First, it extracts the acceptable organic ranges for bigram-feature pairs that describe organic speech. Next, the detector will use these acceptable organic ranges to classify whole samples of audio as either deepfake or organic. The associated code for this paper is available at https://github.com/blue-logan/who_are_you/.

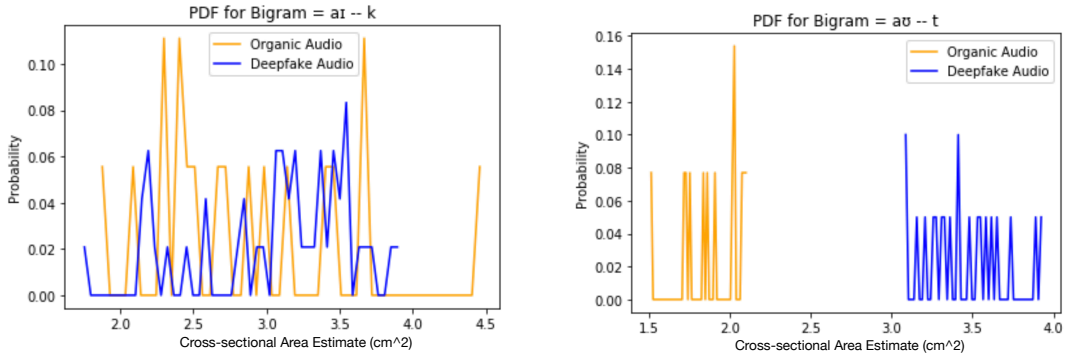
6.3.1 Organic Range Extraction

The organic range extraction phase begins with the detector ingesting known organic audio samples. These audio samples also have associated metadata containing timestamps for both the words and individual phonemes that make up the sample. The phoneme metadata is then augmented to create the necessary bigram timing information. For this, we need to define which phonemes are considered to be adjacent to one another. We define two phonemes as being adjacent if they are both in the same word and occur one after the other. For example, the word cat (phonetically spelled "/kæt/") contains two bigram pairs, "/k – æ/" and "/æ– t/". We consider a bigram to begin at the start of the first phoneme and stop at the end of the second phoneme. The bigram timing information will later be associated to estimate features from processing the audio.

Each bigram audio sample was divided using a sliding window of 565 samples with an overlap between windows of 115 samples. To find these values, we found the minimum and maximum duration for any bigram that existed in our feature extraction set (more detail of the feature extraction set in Section 7). We then selected sliding window parameters (565 samples per window with an overlap of 115 samples) that ensure that every bigram would have between three and seven windowed samplings taken from them. This ensured that we capture the temporal behavior of every bigram. Since speech is not discrete, each bigram captures the transitional behavior that exists as the speaker moves from the initial phoneme to the final phoneme. Windowing the audio allows us to examine individual stages of these transitions (e.g., beginning, middle, end). This forces deepfake audio samples to be generated correctly throughout the transition between phonemes in order to evade detection. Without windowing, it would be possible to evade detection by merely generating correct phonemes without a transition. Each windowed segment of audio is then passed through our vocal tract estimator and assigned a feature vector of 15 cross-sectional areas. The 15 different cross-sectional areas are estimates of the vocal tract at different points between the glottis and the oral cavity opening. Each windowed segment is then labeled with the word, bigram, and window index which corresponds to it.

We are now able to determine which bigrams and features best differentiate organic and deepfake audio. This is done by finding divergences in the distributions of features in specific bigrams between deepfake and organic audio samples. In other words, we look for differences in the distribution of the cross-sectional area estimations found for organic and deepfake audio. The detection of a difference in the cross-sectional area distributions found for the two types of audio indicates that the deepfake audio is being created by an inor-

[6] During the "/ə/" phoneme the vocal tract roughly resembles a constant diameter tube.



(a) High Overlap

(b) No Overlap

Figure 7: The divergence of the cross-sectional area estimate distributions for each bigram can be used to help identify deepfake samples from organic ones. The plots show the distributions of the cross-sectional area estimates for the bigrams (a) "aɪ – k" as in "like" and (b) "aʊ – t" as in "out". Because the distributions in (a) overlap, "aɪ – k" is a poor indicator of whether a sample is deepfake or not. In contrast, the distributions in (b) do not overlap at all. This means "aʊ – t" is a good indicator to differentiate between deepfake and organic audio samples. Our technique would then select a threshold value, such as $2.1cm^2$, that divided the two distributions.

ganic source. These divergences exist because the biological framework of the vocal tract limits organic speech, whereas GAN-generated audio is not limited in the same way. Thus, we are then able to identify deepfake audio samples from organic ones by looking for irregular (i.e., inorganic) cross-sectional area estimates. We, therefore, record the distributions of the cross-sectional area estimates extracted from the organic audio set to be used for future comparison. One distribution is recorded for each unique set of a bigram, window index, and vocal tract position.

6.3.2 Whole Sample Detection

The second phase of our detector is used to determine whether whole audio samples were GAN or organically generated. This phase begins similarly to the organic range extraction phase described in the previous section.

This phase begins by creating the necessary bigram timing information from the sample's metadata. Next, it windows and evaluates the audio samples using the vocal tract estimator. Finally, it associates the estimated vocal tract features to specific bigrams and words just as in the ideal feature selection phase. At this point, our whole sample detection phase deviates from the organic range extraction phase.

Instead of calculating the cross-sectional area distributions for all bigram-features pairs in the data, this phase checks whether every bigram-feature pair falls within the previously determined organic ranges. More specifically, we extract every bigram-feature pair from the sample that exists in both itself and the organic ranges (our set of organic ranges has no guarantee of containing all possible bigram-feature pairs). Next, each feature is compared against the maximum and minimum values found in the distribution of previously extracted organic samples. If the majority of the bigram-feature pairs found in the audio sample are outside the organic distributions, the audio sample is labeled as a deepfake.

6.4 Detector Optimization

Although the previously described detector can differentiate between organic and deepfake audio samples, it is inefficient. Not all bigram-feature pairs act as effective discriminators since deepfake audio models might be accidentally learning the correct distribution for some bigram-feature values. This scenario is likely possible as these models do produce high-quality "human-like" audio. Thus, our detector is estimating large numbers of bigram-feature pairs that are unlikely to indicate the origin of the audio sample. To prevent this, we construct an ideal feature set that contains only bigram-feature pairs that act as strong indicates of the audio authenticity.

6.4.1 Ideal Feature Set Creation

To determine the ideal bigram-feature pairs that act as good discriminators, we initially follow the same procedures laid out in our organic range extraction phase. Namely, extract the timing information from the sample's metadata, window the audio, evaluate it using the vocal tract estimator, and construct an association between the vocal tract features and specific bigrams. We then plot the probability density function (PDF) (Figure 7) for each bigram-feature pair. The PDF represents the likelihood of the random variable, in this case, the bigram-feature pair, having a certain value. If there is a large overlap between the PDF curves for organic and deepfake audio (Figure 7a), then that feature is a poor discriminator, which indicates that the model has learned the correct distribution of the bigram-feature pair. In contrast, if there is little-to-no overlap between the PDF curves (Figure 7b), then that bigram-

feature pair is an ideal discriminator (i.e., can be used to help differentiate between deepfake and organic audio).

Our set of ideal features consists of bigram-feature pairs that can differentiate between deepfake and organic audio samples with a precision-recall of at least 0.9. We determine these threshold values by a sweep through a range of potential values for each bigram-feature pairs. This process continues until a threshold value (the threshold k, which is chosen on a per-feature basis) achieves the desired precision-recall values. This results in a triplet bigram-feature-threshold that we refer to as an ideal feature. Next, we weigh each bigram-feature pair to avoid outlying bigram-features pairs that meet our requirements but only contain a relatively small number of samples. This weight is the number of samples used in our selection criteria calculations. We then filter our bigram-feature pairs using these weights so that only the pairs whose weight are equal to or greater than the average weight of the set are kept. The collection of all the resulting triplets is hereby referred to as our ideal feature set.

It is worth noting that our thresholds singularly divide the PDF. That is, thresholds in our ideal set will label all bigram-feature pairs as organic only if it shares a certain relationship with their threshold (i.e., less than or greater than). Therefore it is possible to create two ideal feature sets, one where values below the thresholds are labeled as organic and one where values above the thresholds are labeled as organic. Unless otherwise stated we will be referring to the ideal feature set as the set of thresholds where values less than the threshold are labeled as deepfakes and the values greater than or equal to the threshold are labeled as organic.

6.4.2 Optimized Detector

Finally, we construct an optimized detector that only computes and analyzes bigram-features that have been shown to act as strong indicators (i.e., our ideal feature set). This detector will follow the same initial operation as our whole sample detector, decorating the audio data with its corresponding timing and phoneme data. However, unlike the whole sample detector, our optimized detector will only check the bigram-features that best indicate whether the audio sample is organic or deepfake. More specifically, we extract every feature from the sample that exists in both itself and the ideal feature set. For every one of these features, we compare the previously found threshold from the ideal feature set with the value found in the current sample. We count the number of times the values from the test audio samples cross the threshold. If more bigram-feature values cross the threshold than do not, we label the audio sample as a deepfake (i.e., majority voting).

7 Datasets

We now discuss the datasets that our technique was tested against as well as the process that was performed in generating deepfakes. For our transfer function, we use the TIMIT Acoustic-Phonetic Continuous Speech Corpus [42] as it is the standard in acoustic-phonetic studies and is manually verified by the National Institute of Standard and Technology (NIST). [7][8]

Organic Audio The TIMIT dataset is a speech corpus that is used in phonetic studies and is commonly used in the training of ML models for speech recognition systems [45]. Despite its age, the TIMIT dataset is widely used in research today with over 1,900 citations since 2015 according to Google Scholar. TIMIT provides documentation of the time alignments for the phonemes and words in each audio file, which is information that is essential for developing our modeling process. The TIMIT dataset is comprised of 630 speakers of 8 different American English dialects speaking phonetically balanced sentences [42]. Each speaker has 10 sentences recorded with a sampling rate of 16 kHz. For our experiments, we randomly sampled 300 speakers from the TIMIT dataset to have a similarly-sized dataset as that of previous work [46].

For consistency, we also performed our time alignments using an open-source forced aligner called Gentle [47] which time-aligns words of transcription with phonemes in an audio sample. Gentle is built on the Kaldi model [48], which is a toolkit frequently used for automatic speech recognition. We use Gentle since any audio samples outside the TIMIT dataset will not have the phoneme time-alignment information needed for extraction. By performing our time-alignments on the TIMIT dataset, we can keep any error in alignments consistent across all samples (i.e., organic and deepfake).

Deepfake Audio We derived our own set of synthetic TIMIT audio samples using the open-source Real-Time-Voice-Cloning (RTVC) tool from Jemine [7, 49], the most widely used publicly available deepfake generation tool. RTCV is an implementation of Tacotron 2 by Liu et al., which uses Tacotron as the synthesizer and Wavenet as a vocoder [50]. For each of our 300 TIMIT speakers, we trained an RTCV model on a concatenation of all 10 TIMIT audio recordings (approximately 30 seconds). Each RTCV model was then used to create a deepfake version of every TIMIT sentence spoken by each speaker. In total, this creates 2,986 usable synthetic audio samples of our 300 original speakers. The 14 missing audio samples were too noisy for Gentle to process and were thus unable to be used in our experiments.

Additionally, we contacted several commercial companies with Deepfake generation tools in an attempt to test our technique against other systems. Most of these companies never returned our requests to use their products in our research. The few companies that responded would only give us extremely

[7] External factors such as face coverings during audio recording do not affect the features our technique needs to operate [43].

[8] Additional experiments were conducted using the ASVspoof2019 dataset [44]. These can be found in Appendix A.

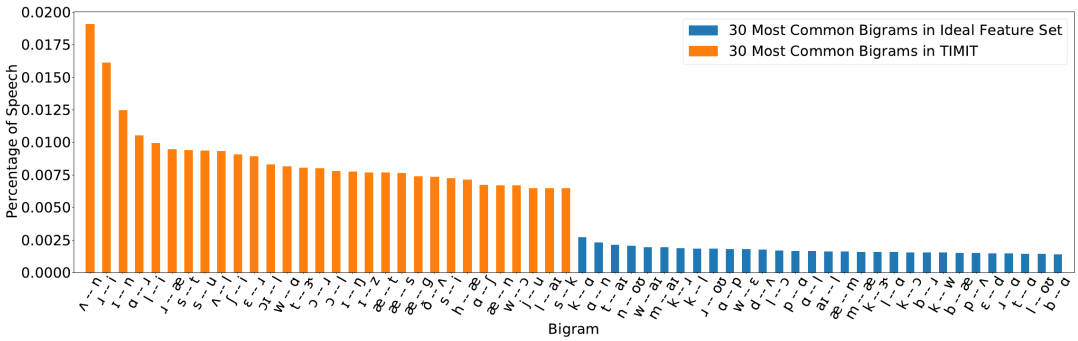


Figure 8: The bigrams found in the ideal feature set were not the most common bigrams found in speech. However, the bigrams in the ideal feature set still made up approximately 30.9% of bigrams in our dataset.

limiting access to their product after purchase. Their restrictions would have limited us to at most 5 different speakers compared to the 300 speakers present in the TIMIT Deepfakes we generated. We, therefore, took the largest available of such datasets, published by Lyrebird [51], and evaluated it. We note that the generation of these samples is black box and represents a reasonable test against unknown models.

Feature Extraction and Evaluation Sets To evaluate and test our technique, we subdivided both the organic and deepfake TIMIT samples into a feature extraction set (51 speakers) and an evaluation set (249 speakers). The feature extraction set is used to determine the ideal bigram-feature pairs and their corresponding thresholds k using the ideal feature extractor outlined in Section 6.3.1. Conversely, the evaluation set is used to evaluate the efficacy of our technique. Both datasets contain all of the organic and deepfake audio samples for their respective speakers. Our security model (Section 5) assumes no knowledge of a speaker is known to the defender. As such, both sets were selected so that they did not share any speakers. This demonstrates that our technique is extracting useful features that are inherent to deepfake audio as a class, rather than features specific to the deepfake of an individual speaker. *This captures a stronger threat model as we do not have any information about the speaker who will be impersonated.*

The feature extraction set contains 1,020 audio files from 51 speakers, which contain a total of 702 unique bigrams. Of these, 510 audio files from 9 speakers are deepfake samples and 510 audio files are organic. The evaluation set consists of 4,966 audio files from 249 speakers, which contain 835 unique bigrams. Of these, 2,476 audio files are deepfake samples and 2,490 audio files are organic. It is important to note that our evaluation set is five times as large as our feature extraction set. We used a smaller feature extraction set and a larger evaluation set to showcase the efficiency of our technique. Traditional ML models require large datasets, orders of magnitude larger than what is used here, to learn from and capture data intricacies that improve the model's generalization [6]. Generating large datasets of DeepFakes can be difficult, inherently limiting the effectiveness of an ML-based detector. In contrast, our technique does not require a large dataset to learn from since we leverage the knowledge of human anatomy. As a result, our technique requires a significantly smaller dataset to learn from while still being able to generalize over a much larger evaluation set.

8 Evaluation

In this section, we discuss the performance of our deepfake detection technique and explain the results.

8.1 Detector Performance

We first need to find the ideal feature set using the process detailed in Section 6.3.1. The feature extraction dataset was used to find the set of ideal features that consisted of 865 bigram-feature-threshold triples.

To evaluate the performance of our detector, we classified all the audio samples in the evaluation dataset. To do this, we concatenated all the sentences for each speaker together to form a single audio sample. We then ran each audio sample through our whole sample detection phase outlined in Section 6.3.2. Overall, we extract and compared 12,525 bigram-features pairs to the values found in our ideal feature set. Finally, our detector was able to achieve a 99.9% precision, a 99.5% recall, and a false-positive rate (FPR) of 2.5% using our ideal feature set.

8.2 Bigram Frequency Analysis

We now explain why the detector performed so well by analyzing the bigram results. The 865 bigram-feature pairs of the ideal feature came from 253 distinct bigrams that had 3.4 features on average within the set. These bigrams make up approximately 30.9% of the 820 unique bigrams present in the TIMIT dataset we tested. Since TIMIT is a phonetically balanced dataset, it accurately represents the distribution of phonemes in spoken English. In Figure 8, we show the 30 most common bigrams in both the TIMIT dataset and our ideal feature set. While most of the bigrams in the ideal feature set are not in the top 30 bigrams, the total ideal feature set

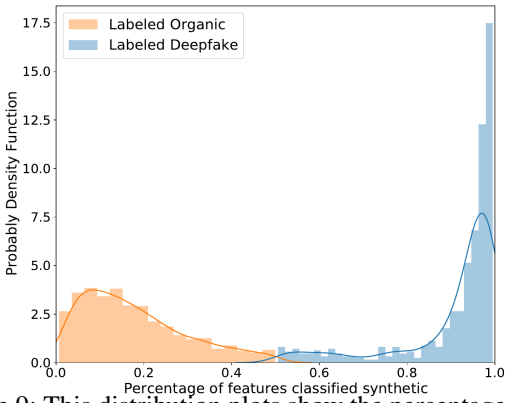


Figure 9: This distribution plots show the percentage of features classified as deepfakes per sentence. We can see that the majority of the time the decision to classify a sentence as a deepfake is not near the decision boundary of 0.5.

still accounts for about 15.3% of the total bigram occurrences extracted from our evaluation set, implying that even though our ideal features are not the most common bigrams, they still account for a sizable portion of the speech. This makes selecting a phrase that does not contain multiple occurrences of the bigrams in our ideal feature set difficult for longer phrases, especially when considering most words are constructed from multiple bigrams. As such, an English sentence will likely contain bigrams that are included in our ideal feature set.

With this knowledge, we next explore the likelihood that our detector will misclassify a sentence. Figure 9 shows the PDF and histogram of the percentage of features labeled deepfake for every sentence in our dataset. The figure shows that most features evaluated in the deepfake samples are individually labeled as deepfakes, which informs us that classification is dependent on multiple features rather than a few prominent ones. This implies that an adversary's model performance would need to increase by a considerable margin before the model could trivially beat our detection technique.

8.3 Fundamental Phenomena Confirmation

To observe the fundamental difference between deepfakes and organic audio that our detector is based around, we conducted an in-depth analysis of the incorrect behavior of the vocal tract estimates found for deepfake audio in a single phoneme ("/d – oʊ/", pronounced doh). Figure 10 shows the estimated cross-sectional area for one of the bigrams from the ideal feature set. For comparison, we use a disjoint set of the TIMIT data to create a secondary set of audio samples (labeled TIMIT Test) that has not been previously used. The box plots (a) represent the estimated cross-sectional area found by our estimator described in Section 6.2. The dimensions represent the multiple tubes our transfer function used to estimate the vocal tract with, as previously seen in Figure 4. We then converted these cross-sectional area estimates to their approximate diameters (b). It is clear at this point that the deepfake audio is not behaving in a manner that is similar to the organically spoken data. The final segment of the figure (c) shows that *the vocal tract estimates found for deepfake audio are approximately the size and shape of a drinking straw.*

In addition to the bigram deep dive, we conducted a small-scale Principle Component Analysis (PCA) experiment, the results of which are visible in Figure 11. Our PCA experiment was conducted using all bigram pairs from the organic samples in the feature extraction set (labeled TIMIT Evaluation), the organic samples in the evaluation set (labeled TIMIT Testing), and the deepfake samples in the evaluation set (labeled Deepfake). We treated each feature vector as a point in a 15 dimensional space and then used PCA to reduce the data down to a single dimension that accounted for the most variance within the data. This single dimension accounts for approximately 48% of all the variance in the dataset. As shown in Figure 11, deepfake audio samples are much less variable than their organic counterparts, which demonstrates that the "drinking straw" vocal tract observed in the bigram deep dive is not an outlier, but rather more likely the norm.

8.4 Transferability Experiments

As discussed in Section 7, the availability of public deepfake datasets is limited, meaning that it was not possible to test our technique against models at the same scale done using the RTVC tool. However, we were able to collect a limited dataset from Lyrebird [51]. These nine synthetic audio samples of former presidents Obama and Trump were generated using tweeted messages and publicly released as marketing material. While our dataset is small, the internal workings of the Lyrebird model are not public and thus we can perform a black box test. We collected eight true-negative sentence samples from previous State of the Union addresses for both speakers. In total, the synthetic audio samples contained 1,914 bigram pairs representing 220 unique bigrams and the organic audio samples contained 4,262 bigram pairs representing 270 unique bigrams. The organic and synthetic audio samples had 136 bigrams in common.

We then evaluated these samples using the ideal features derived from our original dataset and found that they were unable to successfully detect the Lyrebird audio. Following this, we wanted to see if a different ideal feature set capable of detecting Lyrebird audio existed. We proceeded to use the organic and deepfaked audio samples of Presidents Trump and Obama to extract an ideal feature set using the process described in Section 6.3.1. After extraction, we found that *every one of the 136 bigrams* shared between the organic and synthetic audio sets would qualify as an ideal discriminator (i.e., an ideal feature). This means that while the ideal feature from one deepfake model failed to transfer to another, our hypothesis that deepfake models are failing to correctly mimic the acoustic behavior of the human vocal tract, remains correct. Because of this, we were still able to detect every synthetic

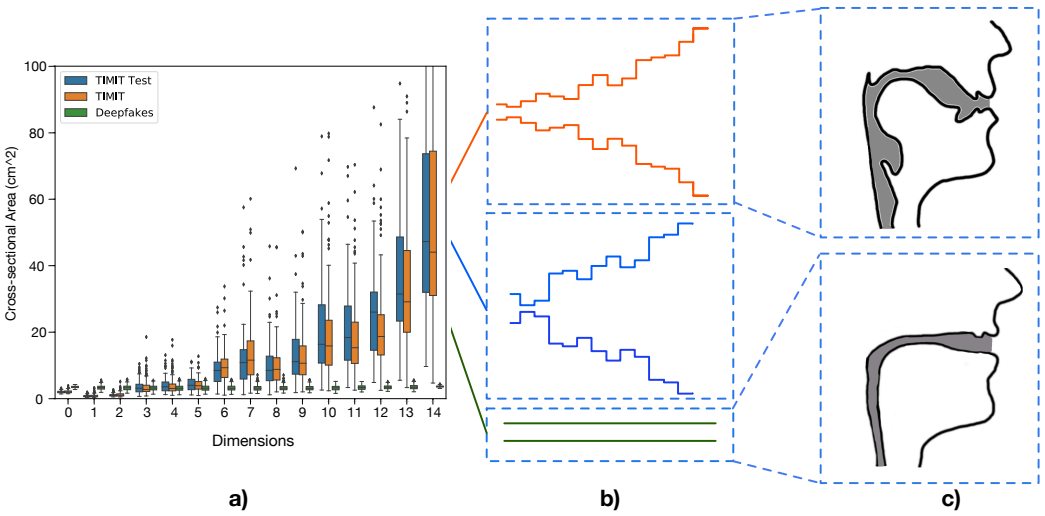


Figure 10: a) The cross-sectional area estimates output by the transfer function for bigram "/d – oʊ/." pronounced "doh" b) The approximate vocal tracts used to create each of the datasets. c) An anatomical approximation of a deepfaked model (bottom), which no longer represents a regular human vocal tract (top) and instead is approximately the dimensions of a drinking straw. **This inconsistency is prevalent across more than 350 observed bigrams.**

audio sample generated by Lyrebird by using an ideal feature set that was sensitive to the Lyrebird model. We believe that this indicates that the RTVC and Lyrebird deepfake generation models are failing to mimic human acoustics in different ways. It, therefore, appears that the ideal features extracted from one deepfake generation model will not necessarily apply to other models. However, the lack of overlapping ideal features between models can be potentially circumvented by having a defender check all bigrams within an audio sample. This would allow a defender to practically check all possible ideal feature sets simultaneously. However, the thoroughness provided by checking all bigrams will result in a considerable increase in the processing time for the detector and would potentially require a different process for whole sample detection than what was presented in Section 6.3.2. We leave the further exploration of the concept of an all-bigram processing method to future work. To conclude, a defender who is concerned about detecting previously unknown deepfake generation models will likely not be able to benefit from the performance increases provided by the creation of an ideal feature set. Furthermore, they will likely need to rely on a different metric than majority voting when evaluating the set of all bigrams within the sample.

9 Discussion

9.1 Limitations

Acoustic Model While our acoustic modeling can process all phonemes for a given speech sample, the pipe series are only anatomically correct for the vocal tract while the speaker is pronouncing a vowel. This means that our technique is less accurate when processing non-vowel phonemes. That being said, vowels make up 38% of all phonemes, meaning most bigrams should contain at least one vowel phoneme. Therefore, our use of bigrams also helps to minimize the number of processed necessary samples.

Preprocessing During the preprocessing stage of our pipeline, we use Gentle to automatically timestamp the audio files according to their words and phonemes. Gentle requires sample transcriptions, which we generate using the Google Speech API. Thus the accuracy of the timestamps (and the following stages of the pipeline) are directly tied to the accuracy of Gentle and the Google Speech API. While some phonemes are only a few milliseconds long, Gentle's precision is to the nearest hundredth of a second. This forces Gentle to overestimate the timestamps for short phonemes, which introduces rounding errors. The use of bigrams helped to mitigate this problem, since using pairs gave us more appropriate target lengths for Gentle's precision levels.

The noisiness of synthetically generated audio can also cause mistranscriptions in the Google Speech API. However, the mistranscriptions are usually phonetically similar to the correct ones. As a result, Gentle's timestamps will contain little error. This limits any major impact that a mistranscription could have on our results.

Data Access There does not exist many large publicly available corpora of deepfake audio samples or generation tools. While we would have liked to test our technique against a larger variety of samples, this was not possible. Our dataset is limited to the data and tools that are currently publicly

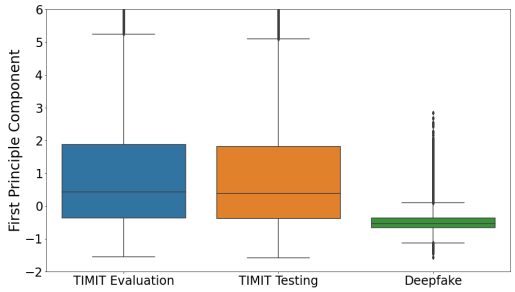


Figure 11: The first PCA dimension shows that deepfake audio samples fundamentally lack the same amount of variability found in organic speech.

available. Startups in deepfake generation have propriety algorithms that are either not available for purchase or otherwise made inaccessible for use in academic research. Despite these issues, our technique appears to be generalizable and not targeting any specific deepfake audio generator, although additional research is needed to verify this.

9.2 Deployment Considerations

Our optimized detector leverages a one-time preprocessing step to amortize the cost of processing individual audio samples later. We preprocess each bigram during the feature extraction phase to determine the acceptable organic ranges and ideal features (e.g., a 60 bigram sample would take approximately 15 seconds to process). However, during the evaluation phase, which simulates our detection process, the preprocessing is performed only on a discriminative set of bigrams (i.e., the ideal features extracted during feature extraction). By greedily selecting the ideal features, the preprocessing time for similarly-sized samples during evaluation is approximately reduced by an order of magnitude, allowing us to perform detection in real-time. For example, the average sentence in the TIMIT dataset contains 25 bigrams over 2.7 seconds of audio. If our technique processed every bigram in the sentence, it would take approximately 6.25 seconds to complete. However, as discussed in Section 8.2, our ideal feature set makes up about approximately 15.3% of bigram occurrences. This means that by only processing the ideal features our technique could evaluate the 2.7 seconds of audio in approximately 1 second. Other techniques that use audio analysis (e.g., Pindr0p [52]) require on the order of 15 seconds to work and are therefore not usually performed in realtime.

9.3 Performance Trade-off

Our optimized technique has a 99.9% precision rate resulting in a minor decrease in recall to 99.5%. A high precision rate will ensure that a deepfake audio is not accidentally labeled as organic by our system. This is specifically designed to protect the victim of a deepfake attack. It is far more dangerous for a deepfake audio to be believed as real, than the converse. For example, it is a greater threat to democracy if the population believes that a deepfake audio of a politician making incendiary remarks is real.

Furthermore, to achieve such a high precision rate, our detector must also sacrifice its false-positive performance. The achieved FPR of 2.5% could be seen as higher than ideal for automated systems that process hundreds or thousands of audio samples per day. However, we believe this trade-off is still better overall. If an organic sample is falsely identified as a deepfake, it is trivial for the original speaker to verify the authenticity of the sample if they choose. However, a malicious speaker could use the false-positive as an opportunity to disassociate from a comment they had previously made. While ideally our technique could prevent this kind of disassociation, we believe that allowing an individual to disassociate from previous a comment is less of a security risk than allowing a deepfake to go undetected. For this reason, we accept our FPR of 2.5% as an acceptable value.

9.4 Advantages over other techniques

Our technique offers several distinct advantages over some techniques in the current literature. Researchers have explored the use of Deep Neural Nets (DNNs) for detecting deepfake audio [6]. These require large training data sets (thousands of audio samples), which is extremely limiting as generating large amounts of deepfake audio data is not easy. If the training data is not large enough to capture the full distribution, the trained DNN will fail to generalize. As a result, the DNN will perform poorly on the test set. Our method only requires a few dozen audio samples and can generalize to a much larger evaluation set. Also, since DNNs are black boxes, they do not provide explanations for the predicted labels. On the other hand, our method leverages a deep understanding of the human anatomy to explain the predicted labels.

9.5 Robustness against Adaptive Adversaries

Attacks and defenses are in a constant arms race. An attacker with knowledge of our technique may try to adapt their attack to defeat it. We explore two different approaches an adaptive adversary could use to evade our detector. The first approach follows the general best practice as described by Tramèr et al. Ideally, a defense should only be attacked end-to-end (i.e., used as a loss function) if the entire defense is differentiable and an inspection of the technique reveals that the defense is not likely to fail due to the exploitation of one or two subcomponents. However, our technique is not end-to-end differentiable due to Eqns. 4 and 5. In line with Tramèr et al.'s work, however, an adversary can still attempt to evade our technique by identifying the critical subcomponent of the model that needs to be exploited. In the case of our technique, this subcomponent is the underlying vocal tract transfer function represented by Eqns. 4 and 5.

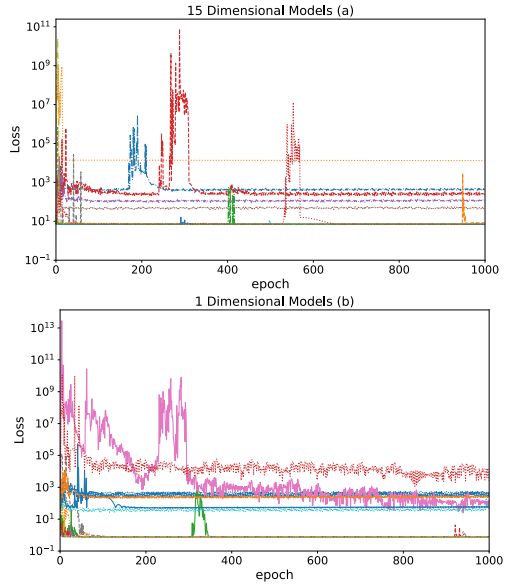


Figure 12: While attempting to overfit to a small dataset, the best models were only able to achieve loss values of approximately 10^{-1}. This is several orders of magnitude greater than what would be expected if the models were well suited to learning our technique's mapping. This indicates that these models struggle to mimic our technique and significant domain-specific knowledge is needed to overcome our defense. Additional information on which model is represented by each line can be found in Table 2.

To exploit this segment, an adversary needs to learn the mapping of our technique between the Fourier Transform of a bigram's audio and the 15 cross-sectional area estimates of the vocal tract. Simply put, an adversary would have to train an ML model to predict the cross-sectional area of a speaker's vocal tract from the frequencies present in the audio sample. By minimizing the error between the model's prediction and the extracted vocal tract estimates, the model would attempt to learn some mapping similar to that of our technique. This trained model would then be used to generate adversarial audio, using a technique such as a Generative Adversarial Network, to evade detection.

To measure the difficulty of this task, we constructed a naïve adversary that uses out-of-the-box Deep Learning models to mimic our technique. Before training on a full dataset, we took an initial step and trained on a small sample set (i.e., 16 random audio samples) to try to overfit the model and get near-zero loss (e.g., around 10^{-5} [53]). Doing so would suggest that the models are indeed learning some mapping between the frequencies in the audio and the 15-dimensional output, indicating that training on a larger dataset would generalize better. However, a non-near-zero loss suggests that, even with a small sample set, the model struggles with finding a correct mapping. To further simplify the problem, we also attempted to train the models to output only the first cross-sectional area estimate, rather than all 15.

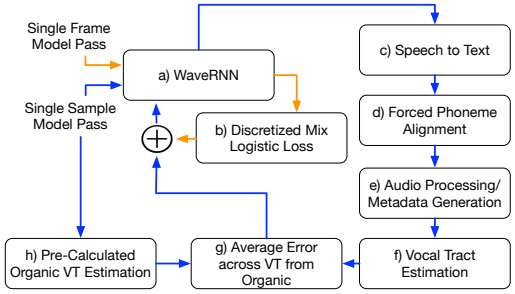


Figure 13: The modifications made to the WaveRNN model (a) to integrate our vocal tract estimation technique as a loss function. Every batch during training the model now generate a full audio sample, transcribe it to text (c), align the necessary metadata (d, e) before estimating the vocal tract (f) and calculating the effective error this has incurred (g) from a set of pre-calculated organic vocal tracts (h). This error is then combined with original loss used by WaveRNN (b).

As shown in Figure 12, the loss (calculated by mean absolute error) of a wide range of models,[9] detailed in Table 2, remained flat and did not converge to near-zero after training for multiple epochs. Our best models had their loss values converge to between 10^{0} and 10^{-1}, *putting them four to five orders of magnitude greater than standard practice* [53].

Thus, we believe that a naïve adversary would be unsuccessful in evading our defense mechanism, and our results suggest that generalizing our technique using ML is non-trivial and requires significant domain-specific knowledge.

The second adaptive adversary approach leverages part of the loss function during the deepfake generator training. As discussed in Section 3.3, deepfake generators compromise three stages; the encoder, synthesizer, and vocoder. We choose to modify the vocoder training process as it is the final step and is responsible for creating the life-like audio.

For this test, we modified the training process of the WaveRNN [54] vocoder model used by RTVC [7]. These modification can be seen in Figure 13. The WaveRNN model's original loss function (b) was calculated on mini-batches of 32 single frames of training audio at a time. However, these single frames of audio are not full audio samples and therefore do not contain any full bigrams our detection method uses. Thus, during every mini-batch, we used the model to generate a single complete audio sample. This is then run through the transcription and phoneme aligner, discussed in Section 7 (c & d). We then create the necessary metadata (e) which is passed along with the audio to the vocal tract estimator (f). The distance between the vocal tract estimation for the model's generated audio is then compared (g) to a precalculated set of organic vocal tracts (h) to find the rela-

[9] While there are other architectures to consider (e.g., RNNs, LSTMs, and transformers), those architectures tend to rely on temporal dependencies. Since our input was the Fourier Transform of a bigram, which lacks temporal data, testing those models would not be appropriate.

tive loss incurred by the vocal tract analysis. This loss value is then added to the original WaveRNN loss. Training then occurs as normal from there.

In addition, we made a major optimization to the training process. Early in the training phase, the model will not output transcribable audio samples. Thus, our technique can not estimate the vocal tract of the audio. In this case, we simply set the component of the loss found from the vocal tract estimation to zero and prevent any related calculations from occurring (d, e, f, or g). However, the model will still need to generate a complete audio sample to determine if it is capable of creating transcribable audio.

All testing was conducted on a local desktop machine running a 6th generation Intel i7-6770HQ with 32 GB of RAM and a GTX 1080 with 8 GB of V-RAM. The WaveRNN model was constructed using PyTorch 1.9.1 and Cuda 11.4.

We use the WaveRNN model's default parameters while training on all 4,622 samples of the TIMIT corpus as a baseline. The model defaulted to a learning rate of 0.0001, 1,000,000 steps, a batch size of 32, and would complete 6,994 epochs. This is similar to the pretained model included in RTVC that was trained for 1,159,000 steps at a batch size of 50. In this configuration, the unedited model was able to complete a batch every 0.50 seconds and an epoch every 71 seconds. The model would require approximately 5.75 days to train.

After the inclusion of our new loss function, we saw a considerable slowdown. During the training phases, before the model is producing transcribable audio, the model was able to process a single batch every 8.86 seconds and an epoch every 1,267 seconds or 21 minutes, a slow down of 17.8x. This slow down was a result of the model having to generate whole audio samples at every epoch, regardless of audio transcription. In this state, the model would take a minimum of approximately 100 days to complete $\sim$90% of its training. This estimate does not include the additional time necessary for running the vocal tract estimator. We are likely to see a further slow down at the end of the model training when the full vocal tract estimator needs to run. At this stage, we assumed that the adversary has already calculated the vocal tracts for a large number of existing organic audio samples to use as ground truth to avoid having to do the vocal tract estimation twice per batch. For TIMIT this process would take approximately 77 hours but only needs to be completed one time. Once the vocal tract estimation loss function was fully engaged the model was able to complete one batch every 31.52 seconds and one epoch every 1.25 hours. Assuming the model will only be producing transcribable audio for the final 10% of its total epochs, the model would complete this phase of training in approximately 36.4 days. Overall, when the vocal tract estimation loss function is fully engaged, the model training is slowed by approximately a factor of 63x. In total, training the WaveRNN model using our vocal tract estimator in this way would require approximately 130 days. Furthermore, since adversarial samples do not transfer across RNN based models [41], this adversarial sample could not be used against any other iteration of our system. Therefore, an attacker will need to spend 130 for every single audio sample, making this attack completely impractical.

10 Conclusion

Deepfake audio generators can now enable attackers to impersonate any person of their choosing. Existing techniques to detect deepfake audio often require knowledge of the specific generator. In our work, we present a novel detection mechanism that is independent of any generator. Our method leverages the knowledge of the human anatomy, fluid dynamics, and the articulatory system to detect deepfake audio samples with a precision of 99.9% and a recall of 99.5%. In doing so, our work presents a unique lens to view the problem of deepfake detection – one that is explainable, generalizable, and free of the limitations of other ML-based approaches.

Acknowledgments

The authors thank our anonymous reviewers and our shepherd, Stjepan Picek, for their valuable comments and suggestions. This work was supported in part by the Office of Naval Research under grant number ONR-OTA N00014-21-1-2658. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research.

References

[1] T. Mills, H. T. Bunnell, and R. Patel, "Towards Personalized Speech Synthesis for Augmentative and Alternative Comm." *Augmentative and Alternative Communication*, 2014.

[2] J. M. Costello, "Message Banking, Voice Banking and Legacy Messages," Boston Children's Hospital - https://www.childrenshospital.org/~/media/centers-and-services/programs/a_e/augmentative-communication-program/messagebankdefinitionsandvocab201613.ashx?la=en, 2016.

[3] C. Stupp, "Fraudsters Used AI to Mimic CEO's Voice in Unusual Crime," *Wall Street Journal*, 2019.

[4] E. A. AlBadawy, S. Lyu, and H. Farid, "Detecting AI-Synthesized Speech Using Bispectral Analysis," in *CVPR Workshops*, 2019.

[5] H. Malik, "Securing Voice-driven Interfaces against Fake (Cloned) Audio Attacks," in *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2019.

[6] M. Alzantot, Z. Wang, and M. B. Srivastava, "Deep Residual Neural Networks for Audio Spoofing Detection," *arXiv:1907.00501*, 2019.

[7] C. Jemine, "Real-Time Voice Cloning," https://github.com/CorentinJ/Real-Time-Voice-Cloning, 2019.

[8] J. Saunders, "Detecting Deep Fakes With Mice : Machines vs Biology," 2019.

[9] "Google Home," https://madeby.google.com/home/, 2017.

[10] "Amazon Alexa Line," https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011, 2017.

[11] "Apple Siri," https://www.apple.com/ios/siri/, 2017.

[12] J. Lorenzo-Trueba, F. Fang, X. Wang, I. Echizen, J. Yamagishi, and T. Kinnunen, "Can we steal your vocal identity from the Internet?: Initial investigation of cloning Obama's voice using GAN, WaveNet and low-quality found data," *arXiv:1803.00860*, 2018.

[13] Y. Wang, W. Cai, T. Gu, W. Shao, Y. Li, and Y. Yu, "Secure Your Voice: An Oral Airflow-Based Continuous Liveness Detection for Voice Assistants," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2019.

[14] Q. Wang, X. Lin, M. Zhou, Y. Chen, C. Wang, Q. Li, and X. Luo, "Voicepop: A Pop Noise based Anti-spoofing System for Voice Authentication on Smartphones," in *IEEE Conference on Computer Communications*, 2019.

[15] C. Wang, S. A. Anand, J. Liu, P. Walker, Y. Chen, and N. Saxena, "Defeating Hidden Audio Channel Attacks on Voice Assistants via Audio-Induced Surface Vibrations," in *Proceedings of the Annual Computer Security Applications Conference*, 2019.

[16] L. Blue, L. Vargas, and P. Traynor, "Hello, Is It Me You're Looking For? Differentiating Between Human and Electronic Speakers for Voice Interface Security," in *Proceedings of the ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2018.

[17] L. Zhang, S. Tan, J. Yang, and Y. Chen, "VoiceLive: A Phoneme Localization Based Liveness Detection for Voice Authentication on Smartphones," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[18] L. Zhang, S. Tan, and J. Yang, "Hearing Your Voice is Not Enough: An Articulatory Gesture Based Liveness Detection for Voice Authentication," 2017.

[19] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, "Asvspoof 2019: Future horizons in spoofed and fake audio detection," *arXiv:1904.05441*, 2019.

[20] S. Naren, "Speech Recognition using DeepSpeech-2," Last accessed in 2019, https://github.com/SeanNaren/deepspeech.pytorch.

[21] "Google Cloud Speech-to-Text API," Last accessed in 2019, https://cloud.google.com/speech-to-text/.

[22] "Azure Speaker Verification API," Last accessed in 2019, available at https://azure.microsoft.com/en-us/services/cognitive-servic/speaker-recognition/.

[23] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition," in *ICASSP*, 2016.

[24] H. Scheidl, S. Fiel, and R. Sablatnig, "Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm," in *2018 International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[25] A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose, "Phonotactic Reconstruction of Encrypted VoIP Conversations: Hookt on Fon-iks," in *IEEE Symposium on Security and Privacy*, 2011.

[26] Z. Zhang, "Mechanics of human voice production and control," *The Journal of the Acoustical Society of America*, 2016.

[27] L. Rabiner and R. Schafer, *Digital Processing of Speech Signals*. The Journal of the Acoustical Society of America, 1978.

[28] R. Kirlin, "A posteriori estimation of vocal tract length," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1978.

[29] H. Wakita, "Normalization of vowels by vocal-tract length and its application to vowel identification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1977.

[30] A. C. Lammert, S. Narayanan, and C. R. Larson, "On Short-Time Estimation of Vocal Tract Length from Formant Frequencies," in *PloS one*, 2015.

[31] S. Flego, "Estimating vocal tract length by minimizing non-uniformity of cross-sectional area," in *Proceedings of Meetings on Acoustics*. ASA, 2018.

[32] S. Skoog Waller and M. Eriksson, "Vocal Age Disguise: The Role of Fundamental Frequency and Speech Rate and its Perceived Effects," *Frontiers in psychology*, 2016.

[33] H. Cao, Y. Wang, and J. Kong, "Correlations between body heights and formant frequencies in young male speakers: a pilot study," *The 9th International Symposium on Chinese Spoken Language Processing*, 2014.

[34] J. H. Hansen, K. Williams, and H. Bořil, "Speaker height estimation from speech: Fusing spectral regression and statistical acoustic models," *The Journal of the Acoustical Society of America*, 2015.

[35] R. E. Hayden, "The relative frequency of phonemes in General-American English," *Word*, 1950.

[36] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized End-to-End Loss for Speaker Verification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[37] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv:1703.10135*, 2017.

[38] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[39] B. Reaves, L. Blue, and P. Traynor, "Authloop: End-to-end cryptographic authentication for telephony over voice channels," in *USENIX Security Symposium*, 2016.

[40] K. N. Stevens, *Acoustic phonetics*. MIT press, 2000.

[41] H. Abdullah, K. Warren, V. Bindschaedler, N. Papernot, and P. Traynor, "SoK: The Faults in our ASRs: An Overview of Attacks against Automatic Speech Recognition and Speaker Identification Systems," in *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[42] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1," *NASA STI/Recon technical report n*, 1993.

[43] D. D. Nguyen, P. McCabe, D. Thomas, A. Purcell, M. Doble, D. Novakovic, A. Chacon, and C. Madill, "Acoustic voice characteristics with and without wearing a facemask," *Scientific Reports*, 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-85130-8

[44] J. Yamagishi, M. Todisco, M. Sahidullah, H. Delgado, X. Wang, N. Evans, T. Kinnunen, K. A. Lee, V. Vestman, A. Nautsch *et al.*, "ASVspoof 2019: The 3rd Automatic Speaker Verification Spoofing and Countermeasures Challenge database," 2019.

[45] J. Michálek and J. Vanek, "A Survey of Recent DNN Architectures on the TIMIT Phone Recognition Task," *ArXiv*, 2018.

[46] N. Subramani and D. Rao, "Learning Efficient Representations for Fake Speech Detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[47] lowerquality, "Gentle Force Aligner," 2018. [Online]. Available: https://github.com/lowerquality/gentle

[48] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi Speech Recognition Toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 2011.

[49] Y. Jia, Y. Zhang, R. J. Weiss, Q. shan Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. Lopez-Moreno, and Y. Wu, "Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis," *ArXiv*, 2018.

[50] Y. Liu and J. Zheng, "Es-Tacotron2: Multi-Task Tacotron 2 with Pre-Trained Estimated Network for Reducing the Over-Smoothness Problem," *Information*, 2019.

[51] "LyreBird," https://github.com/logant/Lyrebird, 2017.

[52] V. Balasubramaniyan, A. Poonawalla, M. Ahamad, M. Hunter, and P. Traynor, "PinDr0p: Using single-ended audio features to determine call provenance," 2010.

[53] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On Adaptive Attacks to Adversarial Example Defenses," *arXiv:2002.08347*, 2020.

[54] O. McCarthy, "WaveRNN," https://github.com/fatchord/WaveRNN, 2021.

A ASVSpoof Dataset

We explored the potential use of the ASVSpoof2019 dataset to evaluate our deepfake detection technique. The ASVspoof2019 dataset contains a collection of synthetically modified audio samples, none of which are actual deepfakes. Instead, these audio samples are used for speaker verification tasks, such as voice authentication. While our algorithm can still detect these audio samples, they should not be used for evaluating deepfake detection algorithms. This dataset was not designed for this task.

We ran the full dataset against our approach, which required over 1,400 hours of processing time. However, we noticed that such tests produced very high word error (WER) rates of 0.45. This means that nearly half of all words were transcribed incorrectly. Upon manual listening tests, we found these audio samples sounded very robotic, thus resulting in poor transcriptions. Therefore, the lower quality of the audio was the source of these failures, and therefore served as efficient filters. Further investigation revealed that contrary to popular belief, ASVSpoof2019 is not a deepfake audio dataset - the maintainers of this challenge note that deepfake detection is a separate challenge, and have identified it as such in their yet-to-be-released 2021 dataset. Even though our preprocessing stage can detect these audio samples as being abnormal due to the high WER, they were never intended to be used for deepfake detection and instead target the related problem of automatic speaker verification (ASV) (e.g., authentication), hence the name of the challenge.

B Phrases

TIMIT phrase that was converted into our deepfake dataset.

1. Cattle which died from them winter storms were referred to as the winter
2. The odor here was more powerful than that which surrounded the town aborigines. (si1077)
3. No, they could kill him just as easy right now. (si1691)
4. Yet it exists and has an objective reality which can be experienced and known. (si654)
5. I took her word for it, but is she really going with you? (sx395)

Model Name	Output Dimensions	Line Style	Number of Training Parameters	General Architecture
General CNN (Small)	15	(a)	13,996,049	2 Convolutional layers, 2 Dense Fully Connected Layers
General CNN (Small)	1	(b)	6,326,275	2 Convolutional layers, 2 Dense Fully Connected Layers
General CNN (Medium)	15	(a)	29,853,713	6 Convolutional layers, 2 Dense Fully Connected Layers
General CNN (Medium)	1	(b)	23,044,099	6 Convolutional layers, 2 Dense Fully Connected Layers
General CNN (Large)	15	(a)	53,640,209	12 Convolutional layers, 2 Dense Fully Connected Layers
General CNN (Large)	1	(b)	48,120,835	12 Convolutional layers, 2 Dense Fully Connected Layers
VGG-16 (Small)	15	(a)	23,121,873	VGG-16 [55] with several of the repeating sets of layers remove to reduce model size
VGG-16 (Small)	1	(b)	18,265,731	VGG-16 [55] with several of the repeating sets of layers remove to reduce model size
VGG-16 (Medium)	15	(a)	19,167,025	VGG-16 [55] with the less of the repeating sets of layers removed than before
VGG-16 (Medium)	1	(b)	17,446,961	VGG-16 [55] with the less of the repeating sets of layers remove than before
VGG-16 (Large)	15	(a)	17,998,689	VGG-16 [55] with all layers
VGG-16 (Large)	1	(b)	17,310,547	VGG-16 [55] with all layers
General DNN (Small)	15	(a)	13,780,569	4 Dense Fully Connected Layers (2048), 2 Dense Fully Connected Layers (15)
General DNN (Small)	1	(b)	13,780,349	4 Dense Fully Connected Layers (2048), 2 Dense Fully Connected Layers (15)
General DNN (Medium)	15	(a)	30,656,977	8 Dense Fully Connected Layers (2048), 2 Dense Fully Connected Layers (15)
General DNN (Medium)	1	(b)	30,565,753	8 Dense Fully Connected Layers (2048), 2 Dense Fully Connected Layers (15)
General DNN (Large)	15	(a)	55,744,089	14 Dense Fully Connected Layers (2048), 2 Dense Fully Connected Layers (15)
General DNN (Large)	1	(b)	55,743,865	14 Dense Fully Connected Layers (2048), 2 Dense Fully Connected Layers (15)
Auto-encoder-NoConv (Small)	15	(a)	187,025	3 Dense Fully Connected Layers (256, 124, 64)
Auto-encoder-NoConv (Small)	1	(b)	186,115	3 Dense Fully Connected Layers (256, 124, 64)
Auto-encoder-NoConv (Large)	15	(a)	463,249	4 Dense Fully Connected Layers (512, 256, 124, 64)
Auto-encoder-NoConv (Large)	1	(b)	462,339	4 Dense Fully Connected Layers (512, 256, 124, 64)
Auto-encoder-Conv (Small)	15	(a)	9,240,145	2 Convolution Layers, 3 Dense Fully Connected Layers (256, 124, 64)
Auto-encoder-Conv (Small)	1	(b)	9,239,235	2 Convolution Layers, 3 Dense Fully Connected Layers (256, 124, 64)
Auto-encoder-Conv (Large)	15	(a)	18,563,153	2 Convolution Layers, 4 Dense Fully Connected Layers (512, 256, 124, 64)
Auto-encoder-Conv (Large)	1	(b)	18,562,243	2 Convolution Layers, 4 Dense Fully Connected Layers (512, 256, 124, 64)
Alexnet	15	(a)	21,781,841	Based off Alexnet [56]
Alexnet	1	(b)	21,724,483	Based off Alexnet [56]

Table 2: Descriptions and high-level information about the models tested in our adaptive adversary experiments. The corresponding results for each model can be seen in Figure 12 by the line denoted in the Line Style column.