VoiceBlock: Privacy through Real-Time Adversarial Attacks with Audio-to-Audio Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

As governments and corporations adopt deep learning systems to collect and analyze user-generated audio data, concerns about security and privacy naturally emerge in areas such as automatic speaker recognition. While audio adversarial examples offer one route to mislead or evade these invasive systems, they are typically crafted through time-intensive offline optimization, limiting their usefulness in streaming contexts. Inspired by architectures for audio-to-audio tasks such as denoising and speech enhancement, we propose a neural network model capable of adversarially modifying a user's audio stream in real-time. Our model learns to apply a time-varying finite impulse response (FIR) filter to outgoing audio, allowing for effective and inconspicuous perturbations on a small fixed delay suitable for streaming tasks. We demonstrate our model is highly effective at de-identifying user speech from speaker recognition and able to transfer to an unseen recognition system. We conduct a perceptual study and find that our method produces perturbations significantly less perceptible than baseline anonymization methods, when controlling for effectiveness. Finally, we provide an implementation of our model capable of running in real-time on a single CPU thread. Audio examples and code can be found at `https://master.d3hvhbnf7qxjtf.amplifyapp.com/`.

1 Introduction

Mass surveillance of voice communications is an ongoing and pervasive issue. While section 702 of the United States Foreign Intelligence Surveillance Act allows the government to perform targeted monitoring of foreign communications, bulk collection practices have resulted in the warrantless surveillance of large numbers of "incidental" foreign and domestic individuals [17]. Despite the fact that millions of these communications are obtained without warrants, they have been used in ordinary criminal investigations [61], undermining a core purpose of the Fourth Amendment of the US constitution [60]: to protect the people against searches without probable cause. Many corporations also possess the capability for large-scale collection of voice data [23], which may be leveraged to profile users for advertising or accessed by government entities through upstream and downstream surveillance [16]. As individuals are faced with these growing surveillance apparati, it is worth remembering that routine surveillance of private voice communications is also corrosive to free speech and association and tends to disproportionately affect marginalized groups [29, 18].

In the absence of identifying metadata, automatic speaker recognition systems can facilitate mass surveillance by allowing an operator to search a database of recorded voice data for utterances from a chosen speaker [7] or to diarise (assign utterances to individuals) transcripts of recordings [45]. Prior to the advent of automatic speaker recognition these tasks required human analysts, forming a natural check on surveillance overreach. We seek to restore this check by degrading the efficacy of speaker recognition models while maintaining the original perceptual quality of the voice communication, a step that could grant users a measure of privacy from mass surveillance.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Modern speaker recognition systems rely on deep neural networks [6]. Deep networks have been shown to be vulnerable to adversarial examples—natural instances (e.g. a recording of "Bob" speaking) modified to cause a model to make an incorrect prediction (the recording's speaker is labeled as "Maria") [54]. This presents opportunities for privacy-minded individuals to mislead or evade surveillance systems with adversarially-crafted inputs. Researchers have proposed adversarial attacks against a variety of audio systems, including speaker recognition [63, 52].

To fool a given neural network-based system, many audio attacks modify a recording by adding a perturbation signal directly at the waveform representation [8]. This perturbation is typically crafted using gradient information obtained from the system's neural network—or from a similarly-constructed surrogate—and requires iterative optimization. Once the optimization is complete, the modified recording is played to the system in an effort to induce arbitrary incorrect predictions (an *untargeted* attack) or a specific incorrect prediction chosen by the attacker (a *targeted* attack).

In theory, such attacks might allow individuals to evade systems that surveil and analyze voice communications, thereby protecting privacy. However, many existing attack algorithms require a costly optimization for each audio recording to which they are applied. Continuous, real-time voice communication precludes the adoption of such approaches through online optimization (which is typically too slow) or through the use of a set of precomputed adversarial examples (which would necessarily limit the user's interaction). This suggests a need for algorithms capable of modifying speech on-the-fly.

Recent years have seen the development of models for audio-to-audio tasks such as denoising, voice conversion, and musical timbre transfer [13, 49, 14]. Such models have sufficient expressive power to modify audio in coherent and task-specific ways, and are often designed to run in real-time. Inspired by these models, we propose **VoiceBlock**, a deep network that learns to apply a time-varying finite impulse response (FIR) filter to outgoing audio, producing highly effective adversarial perturbations on a small fixed delay suitable for streaming. The main contributions of this work are:

- A highly effective method for de-identifying speech in real-time that is more perceptually inconspicuous than existing methods of equal effectiveness
- Objective and subjective experimental results that validate these claims
- A system that embodies the claimed advances, and that runs in real-time on a single CPU thread

Through this work, we hope to encourage further exploration of audio-to-audio models for protecting user privacy. Audio examples and code, including our streaming implementation, can be found at `https://master.d3hvhbnf7qxjtf.amplifyapp.com/`.

2 Related work

Previous works in **speech de-identification** have proposed methods for obfuscating speech to evade surveilling speaker-recognition systems. One such method is voice conversion, which modifies the speech of a source speaker to sound like that of another target speaker, both to humans and machines. Jin et al. [24] and Alegre et al. [4] analyze the vulnerability of speaker recognition systems against de-identification attacks performed with voice conversion models. However, the conversion models considered are incapable of real-time operation. While the recent availability of low-latency voice conversion models [49, 50, 32] may make such approaches more practical, our aim is to modify speech in a way that is inconspicuous and minimally invasive to the user experience, and that has the potential to generalize to tasks beyond speaker recognition. This also rules out recent anonymization approaches catalogued through the VoicePrivacy Challenge, as all evaluated submissions to date noticeably alter speaker characteristics [56]. Similarly, we do not consider obvious transformations like pitch-shifting that significantly alter or degrade recorded audio.

Real-time audio attacks have recently been explored. Chiquier et al. [10] propose an attack on automatic speech recognition systems that is capable of inducing significant transcription errors in an *over-the-air* environment (where attack audio is played to the system in a physical space), but which introduces conspicuous noise due to the use of additive perturbations at the audio waveform. Their approach also necessitates both a large model (requiring approximately 16 GPU-days to train) and lengthy receptive field, resulting in an initial "idle" period of 2.5s during which the attack does not

affect speech. By contrast, we focus on an *over-the-line* setting, where attacks are passed to the victim model over a digital channel, and propose a lightweight network capable of producing inconspicuous perturbations through the use of filtering, with an initial delay of miliseconds rather than seconds.

Universal adversarial attacks optimize a short perturbation that can be played alongside arbitrary speech, optionally in real-time, to evade a victim model [33, 63, 28]. However, these attacks tend to introduce conspicuous noise due to the use of additive perturbations at the audio waveform. The same holds for the attack of Xie et al. [62], which uses a Wave-U-Net [53] model to efficiently generate universal perturbations. Rather than generate additive perturbations, we perform multiplicative attacks in the frequency domain through the use of filtering, and in doing so avoid the bias towards noise-like artifacts typical of additive attacks.

Other works have proposed crafting **adversarial attacks in the frequency domain**. The "Kenansville" attack of Abdullah et al. [1] performs spectral gating, removing low-energy frequency components in an effort to hinder classification. The authors demonstrate the effectiveness of their approach against seen and unseen speaker- and speech-recognition systems, and the proposed method is fast and gradient-free. However, the spectral gating must be constrained to a small number of key audio frames to avoid conspicuous artifacts, which requires offline optimization using word- or phoneme-level alignment information.

Ahmed et al. [3] optimize bandpass filters to perform targeted impersonation attacks on speaker-recognition models; these filters can then be physically realized as resonant tubes through which an attacker can speak to the victim system, allowing real-time operation. However, once optimized and realized, the bandpass filters are fixed in time and cannot adapt to the attacker's speech. The attack is significantly less effective than traditional approaches, even in controlled acoustic environments, and requires an active intervention on the part of the user to both realize and apply.

Finally, O'Reilly et al. [42] adversarially optimize the parameters of a time-varying finite impulse response (FIR) filter in order to avoid noisy artifacts. However, the attack operates offline and must be optimized separately for every instance to which it is applied.

In summary, we are aware of no current approach that is simultaneously capable of performing de-identification while remaining inconspicuous to human listeners and operating in real-time on-device.

3 VoiceBlock

We propose VoiceBlock, a system that applies adversarial time-varying filtering to user audio in real-time. VoiceBlock modifies speech by specifying the frequency magnitude response of a standard finite impulse response (FIR) filter. By carefully varying the filter's response many times per second, the speech is de-identified to an automated system while speaker identity is preserved for human listeners. This is accomplished without introducing noisy artifacts at the waveform, as filtering can only amplify or attenuate frequency energy present in the original signal through multiplication in the Fourier domain (see Section 3.3 for details of the filtering implementation used in our experiments).

VoiceBlock consists of three main modules, shown in Figure 1: (1) an **encoder** module extracts acoustic features from input audio frames, (2) a recurrent **bottleneck** module incorporates context from past frames to predict a set of adversarial filter controls for each frame, and (3) a **decoder** module regularizes the predicted filter controls and applies them to the corresponding frame of input audio. We provide an overview of each component below, and further details in Appendix A.

3.1 Encoder

The encoder module extracts acoustic features from 16kHz audio (double the sample rate of telephone speech) segmented into frames of 256 samples with 50% overlap. Though our model can operate in fully causal fashion, we find we achieve stronger attacks using a lookahead of five frames (see Section 3.2), resulting in a theoretical latency of 56ms. Verizon reports 25ms or less is typical latency for voice-over-IP communication in North America and Europe [57]. Experimental data [27] show that 130ms of latency is acceptable for high-quality Voice communications. Therefore, adding VoiceBlocks to the signal chain leaves latency well within this limit.

Audio frames entering the encoder are passed to four sub-modules in parallel to obtain the following features.

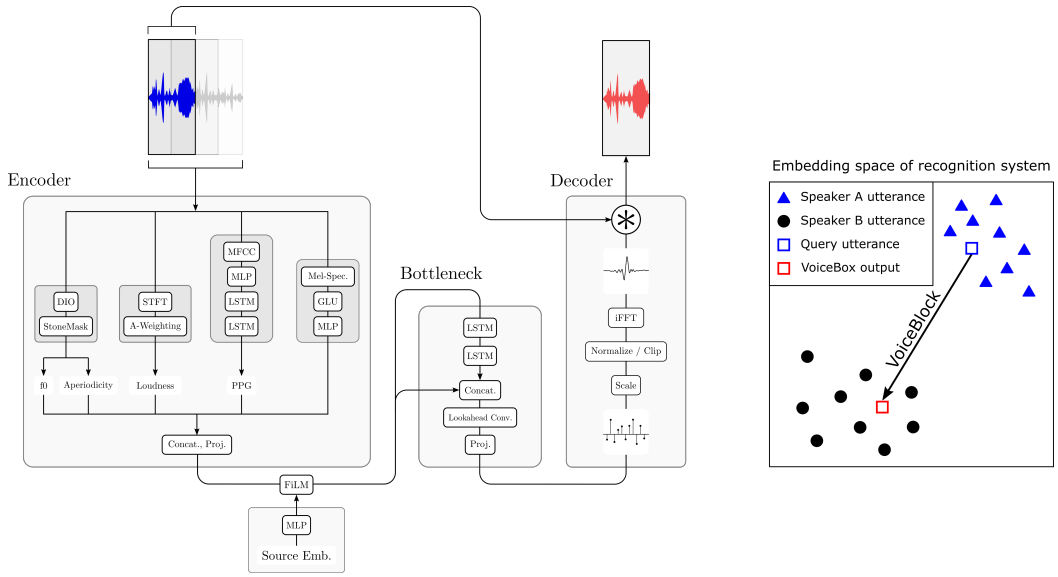

Figure 1: **Left**: The proposed VoiceBlock architecture. Acoustic features extracted by the encoder are fed to the recurrent bottleneck to predict filtering controls, which are regularized and applied to the input by the decoder to obtain adversarial audio. **Right**: VoiceBlock adversarially perturbs the user's audio stream such that any extracted queries are scored by the system as dissimilar to the user's enrolled utterances, hampering identification or retrieval.

Pitch features: Given the known sensitivity of speaker-recognition models to pitch [3] and the importance of spectral structure in delineating linguistic content, we extract fundamental frequency and aperiodicity estimates for each frame using the DIO algorithm [37]. Pitch estimates are refined by the StoneMask algorithm [38]. For both stages, we use the `pyworld` [22] implementation of the WORLD vocoder [38] (MIT license).

Loudness: We take an A-weighted average of each frame's log-magnitude spectrum to obtain a loudness estimate [36].

Phonetic posteriorgrams: Following the method of Ronssin & Cernak [49], we use a trained phoneme classifier with frozen weights to encode linguistic content. The classifier consists of a multi-layer perceptron followed by two LSTM [21] layers and a linear classification layer, and takes as input 13 mel-frequency cepstral coefficients (MFCC) with first- and second-order deltas for each frame. We train our phoneme classifier on the "train-clean-100" subset of the LibriSpeech dataset [43] using frame-aligned phoneme labels [34, 35] (CC-BY 4.0 license). Rather than directly use the classifier's predicted distributions over phoneme labels – known as phonetic posteriorgrams (PPGs) [19] – we discard the classification layer after training and pass along the output of the final LSTM layer, which Ronssin & Cernak also refer to as PPGs in their architecture.

Spectrogram features: Finally, we use a simple network consisting of a gated linear unit and multi-layer perceptron operating independently on each mel-spectrogram frame—a widely used speech representation [26]—to capture any residual information. We note that Engel et al. [14] also make use of a residual spectrogram encoder, alongside pitch and loudness features, to control differentiable signal-processing components.

For each frame, the above features are concatenated and projected linearly to obtain a low-dimensional encoding. We then introduce additional speaker information by passing a fixed, pre-computed embedding of the source speaker through a FiLM layer [46] to modulate the encoder output. This embedding helps to guide the de-identification task, allowing us to train a single model capable of de-identifying arbitrary users. This imposes a requirement that users must record a small amount of speech to obtain an embedding before first using VoiceBlock. We find that less than a minute of speech is sufficient, and use a pre-trained ResNetSE34V2 model [20] to compute embeddings (see sections 4.1, 4.2).

3.2 Bottleneck

Encodings entering the bottleneck are passed through two LSTM layers, and the outputs are concatenated with a skip connection from the encoder. To enable streaming, we use unidirectional LSTM layers and pass concatenated outputs through a small lookahead convolutional network [58] to incorporate information from future frames at the expense of a small fixed delay. We find a lookahead of 5 frames (48 ms) is sufficient to craft strong de-identification attacks. Finally, a linear projection layer maps the concatenated representations to a vector of filter controls, with each entry representing the unnormalized frequency magnitude response of a filter band for the current frame.

3.3 Decoder

Our decoder applies time-varying filtering to the input audio based on frame-wise controls obtained from the bottleneck. For our task, we find that a filtering-based decoder affords a number of advantages over the synthesis-based (e.g. transposed-convolutional) decoder architectures present in many audio-to-audio models [13]:

- The filtering module is not prone to the periodic upsampling artifacts that transposed-convolutional architectures often introduce [47].
- Our decoder is a simple deterministic module with no trainable parameters, keeping VoiceBlock lightweight.
- The capacity of the decoder and conspicuousness of perturbations can easily be constrained in terms of the number of filter bands or their allowed range of motion, providing interpretable control to the user.

To regularize and apply filter controls to each frame of audio, we use a method similar to that of O'Reilly et al. and Engel et al. [42, 14]. We apply sigmoid scaling to bound filter controls to the range $[0, 2]$. We clip deviations from unity beyond a fixed bound ϵ. Each set of scaled filter controls is transformed into a time-domain impulse response via the inverse Fourier transform. We shift the impulse response to zero-phase (symmetric) form, apply a Hann window, and finally convolve with the corresponding input audio frame by taking the Fourier transform and performing element-wise multiplication. After compensating for shift, we overlap-add the resulting frames with a Hann window to obtain the final filtered audio. We discuss the details of our buffered streaming implementation in Appendix A.3. Our VoiceBlock model is implemented in PyTorch [2] and contains 6.3m trainable parameters, and 7.5m in total counting the frozen phoneme encoder.

3.4 Training objective

To demonstrate the ability of VoiceBlock to perform inconspicuous privacy-preserving audio transformations, we train our model to attack speaker recognition systems. Given a large database of speech recordings and access to a user's audio stream, a surveilling entity may seek to (a) identify the user by matching their speech against recordings of known provenance in the database, or (b) retrieve other utterances of the user from the database. Systems designed for these tasks – speaker recognition and retrieval, respectively – often rely on neural network models to map speech utterances to a low-dimensional embedding space in which distance corresponds to speaker similarity [15]. In this work, we consider models f for which the speaker distance D_f between utterances u and v can be measured via a cosine distance between embeddings $f(u)$ and $f(v)$:

$$D_f(u, v) = 1 - \frac{f(u) \cdot f(v)}{\|f(u)\|_2 \, \|f(v)\|_2} \tag{1}$$

At inference time, *query* audio from the user is embedded and scored for similarity against *enrolled* utterances – pre-computed embeddings stored in the database. Scores may be evaluated for all enrolled embeddings, or for only a representative of each unique speaker (e.g. speaker centroids). The highest-ranking result or results (e.g. the closest speaker identity) are then returned. We do not distinguish between recognition and retrieval tasks, and refer to both under the umbrella of "speaker recognition." This is because from a privacy standpoint, the objective in each task is identical: alter the query audio to prevent valid matches, thereby de-identifying the user.

A variety of methods have been proposed to efficiently compute and search over low-dimensional speaker representations. Generally, a hashing algorithm is applied to speaker embedding vectors

to reduce storage and search costs [51, 31, 15]. Because the resulting hash representation merely serves as an efficient point of access for embedding-space distances, we omit hashing algorithms from consideration and define our attack objectives on the embedding space directly.

Given a speaker embedding model, we aim to modify query audio drawn from a user's stream such that its embedding-space distance from any enrolled utterances of the user is large, and its evaluated similarity is small. We quantify this de-identification in terms of distance thresholds in the embedding space, set according to percentiles of the estimated distribution of all inter-speaker embedding distances. Let P_r represent the distance corresponding to the rth percentile of this distribution; then for query audio u and enrolled utterance v, $D_f(u, v) > P_r$ implies that roughly r percent of database entries should be scored as more similar to u than v. Thus, we can set distance thresholds that correspond directly to the strength of de-identification applied to user audio, and construct a loss function that penalizes our model when the embedding-space cosine distance between query and enrolled utterances falls below the threshold. To do so, we use a variant of the adversarial loss proposed by Zhang et al. [63]. Let f represent the victim model, g our VoiceBlock network, u an utterance from our user's audio stream, and P_r our de-identification threshold; then

$$\mathcal{L}_{adv}(f, g, u) = (P_r - D_f(g(u), u) + \kappa)^+ \tag{2}$$

where $(\cdot)^+ = \max(\cdot, 0)$ and κ is a confidence parameter encouraging the attack to fully cross the threshold. To ensure that our VoiceBlock model learns to perturb user audio inconspicuously, we incorporate an additional loss function to penalize perceptible filtering artifacts. We compute the combined waveform L_1 and multi-resolution spectrogram losses proposed by Defosséz et al. [13] on the clean and adversarial audio:

$$\mathcal{L}_{aux}(g, u) = \mathcal{L}_{stft}(u, g(u)) + ||u - g(u)||_1 \tag{3}$$

where $\mathcal{L}_{stft}$ is given by a sum of magnitude and spectral convergence losses computed over several spectrogram resolutions. We provide further details in Appendix A.2. Combining the above adversarial and auxiliary losses, we obtain our final attack objective:

$$\mathcal{L} = \mathcal{L}_{adv} + \mathcal{L}_{aux} \tag{4}$$

4 Experimental design

We describe experiments used to validate the claimed advances of our work, namely that VoiceBlock can de-identify speech in real-time while remaining significantly less conspicuous than existing methods of similar effectiveness. We first introduce the models, datasets, and attacks considered in our experiments. Following this, we detail our experiment configurations and present the results of both objective and subjective evaluations.

4.1 Speaker recognition models

ResNetSE34v2: We train attacks against the ResNetSE34v2 model [20] provided in the `VoxCeleb Trainer` repository [12] (MIT License). The model takes mel-spectrogram inputs and uses 2D convolutions with residual connections, squeeze-and-excitation, and attentive statistics pooling to generate frame-level features and aggregate them into 512-dimensional speaker embeddings.

Y-Vector: To examine the transferability of our approach against unseen systems, we evaluate trained attacks against the Y-Vector model proposed by Zhu et al. [64]. The model uses a multiscale 1D-convolutional waveform encoder to extract acoustic features, followed by squeeze-and-excitation blocks, feature aggregation, and a time-delayed neural network to map variable-length utterances to 128-dimensional speaker embeddings.

Both the ResNetSE34v2 and Y-Vector models were trained on the development set of the VoxCeleb2 dataset [11]. Note that while our attack is causal—modulo a short, five-frame lookahead—we do not impose the same restriction on a hypothetical surveillance system; instead, we evaluate our attack against strong, non-causal systems capable of aggregating speaker characteristics from across full utterances before rendering predictions.

4.2 Datasets

LibriSpeech: (CC-BY 4.0) We use both the `train-clean-100` and `test-clean` subsets of the LibriSpeech dataset [43] for training VoiceBlock. The former comprises 28,539 utterances from 251

speakers while the latter comprises 2,620 utterances from 40 speakers. We trim or pad all utterances to 4 seconds.

VoxCeleb1: (CC-BY 4.0) To simulate large-scale surveiling speaker recognition, we evaluate attacks on the VoxCeleb1 dataset [40], comprising 153,516 utterances from 1,251 speakers. This also ensures that no evaluation speakers are seen during training. As with the LibriSpeech dataset, we trim or pad all utterances to 4 seconds.

For all experiments, we carefully divide the data to imitate a realistic attack setting. During training, we select fifteen utterances (one minute total) from each source speaker in the training set and compute embeddings using the ResNetSE34v2 (MIT License) model. The centroid of these embeddings is then used as an enrolled target for the computation of the adversarial loss with all utterances of that speaker. We find that this produces stronger attacks than using individual utterance embeddings as targets, possibly by ensuring more consistent gradient information across the optimization. For our VoiceBlock attack (see Section 4.3), we select a further ten utterances (40s total) from each source speaker in the training set and again compute embeddings using the ResNetSE34v2 model. The centroid of these embeddings is then fed as a fixed conditioning vector to the VoiceBlock model alongside all utterances from that speaker (see Section 3.1). Similar to training, during evaluation we select fifteen utterances per speaker as a query set. We again select a further ten utterances to serve as conditioning for the VoiceBlock attack. Finally, twenty utterances of each speaker are enrolled in the speaker recognition system, serving as the database against which query utterances are matched.

4.3 Attack algorithms

We perform untargeted attacks using the following algorithms. **VoiceBlock:** We implement the proposed VoiceBlock attack as described in Section 3 and Appendix A and train for 10 epochs on 3 NVidia RTX 2080 Ti GPUs; this takes approximately 40 minutes. **Universal:** We optimize a short (2s) universal additive perturbation for 10 epochs using the established penalty method [33, 63, 28] and the same adversarial objective as VoiceBlock. On 3 NVidia RTX 2080 Ti GPUs, this takes approximately 16 minutes. To limit the perceptibility of the attack, we scale the perturbation to have L_∞ norm 0.08 times that of the unperturbed speech. During training and evaluation, the perturbation is aligned arbitrarily with query utterances and looped to match durations, serving as a constant adversarial "background" signal. **White noise:** We add Gaussian noise to utterances at the waveform representation at a signal-to-noise ratio of -10dB. **Spectral gating:** We modify the "Kenansville" attack of Abdullah et al. [1] to allow for streaming use by performing spectral gating at all frames, using a threshold of 4dB relative to the maximum-energy spectral bin of each frame.

4.4 Objective evaluation

We evaluate attacks in a large-scale closed-set speaker recognition task. First, we use the `test-clean` LibriSpeech subset to obtain a rough estimate of the distribution of distances between an individual utterance and the centroids of each distinct speaker in the embedding space of the ResNetSE34V2 model. To encourage strong de-identification attacks, we take the distance corresponding to the 25th percentile of this distribution as the target threshold P_{25} for our training loss (see Section 3.4). We train each attack as discussed above.

We evaluate the closed-set recognition of all attacks over the VoxCeleb1 dataset. We compute the distance between each query embedding and the centroid of all embeddings of each speaker; recogntion is then performed by returning the speaker identity of the nearest speaker centroid. We find this is slightly more accurate and robust to attack than using the identity of the nearest embedded utterance. We perform exact nearest-neighbors search over the embedding space using `FAISS` [25] (MIT license), and report the top-1 (T-1) and top-10 (T-10) accuracy of the relevant speaker recognition model given both clean and adversarial queries. Additionally, we compute the following objective speech quality metrics over the clean and adversarial query audio as a proxy measure of the imperceptibility of attacks: Perceptual Evaluation of Speech Quality (PESQ) [48], operating in the wide-band configuration and using the `python-pesq` implementation [59] (MIT license); and Short-Time Objective Intelligibility (STOI) [55], using the `pystoi` implementation [44] (MIT license).The results of our evaluation are presented in table 1.

We evaluate the real-time performance of the streaming implementation of the VoiceBlock attack described in Appendix A.3 by measuring its average real-time factor (RTF). We measure performance

Table 1: Results of our objective evaluation. We perform attacks on the VoxCeleb1 dataset against seen (ResNetSe34v2) and unseen (Y-Vector) speaker recognition models, and compute both top-1 and top-10 recognition accuracies. Additionally, we compute a set of objective speech quality metrics for each attack.

	Speech Quality Metrics		**ResNetSe34V2**		**Y-Vector**	
Approach	**PESQ** ↑	**STOI** ↑	**T-1**↓	**T-10**↓	**T-1**↓	**T-10**↓
White noise	1.03	0.23	0.13	0.40	0.00	0.01
Spectral gating	1.12	0.56	0.02	0.11	0.02	0.12
Universal	2.36	0.82	0.14	0.22	0.44	0.64
VoiceBlock	3.77	0.90	0.02	0.10	0.32	0.62
No attack	4.64	0.99	0.97	0.99	0.93	0.98

on a single thread on two different CPUs, an Intel i7-5600U @ 3.2 GHz and an Apple M1 Chip. In the streaming configuration, VoiceBlock processes a chunk of 4 overlapping frames at a time, equivalent to 640 samples / 40 ms. We compute the average RTF for processing a chunk over all chunks in a 4 second audio clip, and find that VoiceBlock has an average RTF of .255 on the Intel i7-5600U and .200 on the Apple M1.

4.5 Subjective evaluation

For our subjective evaluation, we sampled 100 clean speech recordings from the VoxCeleb1 dataset. Each of the four attacks described in Section 4.3 was applied to all 100 recordings, resulting in 100 comparison sets with five recordings per set: the original clean speech and the speech modified by each of the four attacks. These sets were then evaluated in a MUSHRA-style listening study [9] deployed on Amazon Mechanical Turk using the open-source, MIT-licensed Reproducible Subjective Evaluation (ReSEval) [39] system. IRB approval was obtained prior to conducting this study, and there were no known risks to the participants in this study.

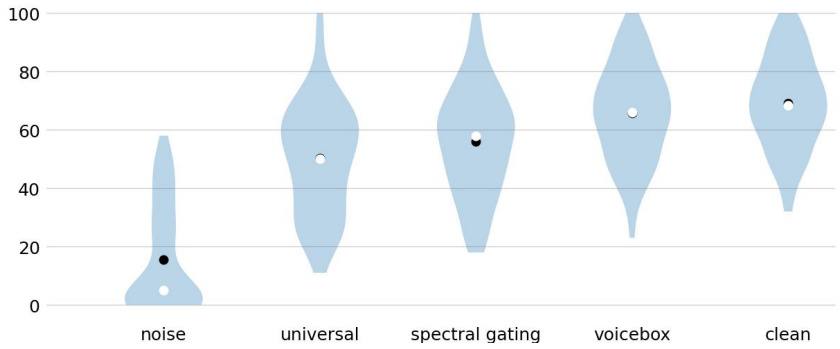

Figure 2: Distributions of quality ratings from our crowdsourced subjective listening MUSHRA-style test on audio quality. Higher numbers are better. Black dots are means and white dots are medians. Wilcoxon signed-ranked tests between all pairs of conditions show statistical significance at $p < 0.05$.

We recruited 20 participants. Participants were screened with a listening test prior to beginning the study; those that passed rated 20 comparison sets. For each comparison set, the participant was asked to listen to and rate the relative audio quality of each of the five audio files on a scale from 0 to 100. We omitted responses by four participants who failed our prescreening listening test and nine who rated the white noise attack (our low anchor) as superior in quality to ground-truth clean audio (our high anchor), giving us a total of 140 five-way comparisons. For more details of our crowdsourced subjective evaluation, see Appendix C.

The results of our crowdsourced subjective evaluation can be found in Figure 2. We find VoiceBlock is preferred to all other methods, and exhibits similar perceptual quality to ground-truth speech recordings. The difference between VoiceBlock and ground-truth audio is significant ($p < 0.05$) using a Wilcoxon signed-rank test ($p = 0.024$) but not significant using Welch's T-test ($p = 0.071$),

which assumes that human perceptual scores are normally distributed but with potentially different variances. All other pairs of conditions are significantly different using either test.

4.6 Additional experiments

We conduct supplementary experiments demonstrating the robustness of VoiceBlock against a deep network-based speech enhancement model, examining its performance under the assumption that adversarial queries are enrolled by the surveilling speaker recognition system, and evaluating the intelligibility of processed audio using an automatic system. These experiments are detailed in appendices B.1, B.2, and B.3 respectively.

5 Ethics

The main goal of our work is to show that large-scale, untargeted automated surveillance can be hindered by introducing inconspicuous perturbations to audio in real-time. Because VoiceBlock does not conceal speaker identity from human listeners, it still allows high-effort targeted surveillance (e.g. authorized human-attended wiretaps of criminal enterprises). In this way, we hope to return to the status quo of the 20th and early 21st centuries – in which the need for human listeners provided an important check on mass surveillance.

Those wishing to avoid all voice identification by human or machine may use existing voice conversion technologies. Voice conversion is not, however, appropriate in many contexts where some measure of privacy from mass surveillance may be desired. Consider telehealth: a medical expert or therapist discussing sensitive topics with a patient may gain great insight from hearing the nuance of the patient's voice. In this setting, methods that alter one's voice beyond human recognition (e.g. voice conversion) would necessarily degrade the interaction.

The approach of VoiceBlock could also, in concept, be used to fraudulenty access information or services protected by speaker verification. However, VoiceBlock performs perceptually inconspicuous filtering and the speech produced would still sound like the original speaker. As such, it could not pass human inspection. Both voice-conversion and speech synthesis-based attacks are much more suited to this purpose, as both produce speech in a voice perceptually similar to the target speaker.

Our approach can be thought of as leveraging the asymmetry between system and attacker attention inherent to mass surveillance, using inconspicuous perturbations to evade large-scale systems that must render predictions in bulk. This same asymmetry is not necessarily present when trying to bypass authentication mechanisms and access tightly-guarded services.

6 Conclusions

Our experimental results indicate VoiceBlock can de-identify speech from arbitrary unseen users in real-time on a standard M1 CPU. The de-identified speech is of significantly higher audio quality than competing methods, as reported by a subjective listener study and as measured through standard metrics of speech intelligibility. Our method also achieves nontrivial de-identification results against a system it was not trained on, outperforming a far more perceptible universal attack. This is notable given that we take no explicit steps to improve the transferability of our method. While pure signal-processing approaches such as spectral gating and white noise transfer between systems more successfully, they severely degrade audio quality, resulting in much lower listener quality ratings. This limits their practicality in real-world voice communications. By contrast, our method produces adversarial audio that is virtually indistinguishable from the clean source, as indicated by our subjective evaluation.

We view our method as an initial step towards protecting users from indiscriminate mass surveillance. A number of obvious directions for future work stand out, such as improving the transferability of adversarial examples [41], and evaluating attacks against real-world speaker recognition systems and over real-world communication channels. We hope this work encourages further exploration of the applications of audio-to-audio models for protecting user privacy.

References

[1] Hadi Abdullah, Muhammad Rahman, Washington Garcia, Kevin Warren, Anurag Yadav, Tom Shrimpton, and Patrick Traynor. Hear "no evil", see "kenansville"*: Efficient and transferable black-box attacks on speech recognition and voice identification systems. In *IEEE Symposium on Security and Privacy*, 2021.

[2] Francisco Massa Adam Lerer James Bradbury Gregory Chanan Trevor Killeen Zeming Lin Natalia Gimelshein Luca Antiga Alban Desmaison Andreas Kopf Edward Yang Zachary DeVito Martin Raison Alykhan Tejani Sasank Chilamkurthy Benoit Steiner Lu Fang Junjie Bai Adam Paszke, Sam Gross and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[3] Shimaa Ahmed, Yash Wani, Ali Shahin Shamsabadi, Mohammad Yaghini, Ilia Shumailov, Nicolas Papernot, and Kassem Fawaz. Pipe overflow: Smashing voice authentication for fun and profit. *arXiv preprint arXiv:2107.14642*, 2022.

[4] Federico Alegre, Giovanni Soldi, Nicholas Evans, Benoit Fauve, and Jasmin Liu. Evasion and obfuscation in speaker recognition surveillance and forensics. In *International Workshop on Biometrics and Forensics*, 2014.

[5] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[6] Zhongxin Bai and Xiao-Lei Zhang. Speaker recognition based on deep learning: An overview. *Neural Networks*, 140:65–99, 2021.

[7] Peter J. Barger and Sridha Sridharan. On the performance and use of speaker recognition systems for surveillance. In *International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2006.

[8] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *IEEE Security and Privacy Workshops*, 2018.

[9] Mark Cartwright, Bryan Pardo, Gautham J Mysore, and Matt Hoffman. Fast and easy crowd-sourced perceptual audio evaluation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[10] Mia Chiquier, Chengzhi Mao, and Carl Vondrick. Real-time neural voice camouflage. In *International Conference on Learning Representations (ICLR)*, 2022.

[11] J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. In *Interspeech*, 2018.

[12] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han. In defence of metric learning for speaker recognition. In *Interspeech*, 2020.

[13] Alexandre Défossez, Gabriel Synnaeve, and Yossi Adi. Real time speech enhancement in the waveform domain. In *Interspeech*, 2020.

[14] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing. In *International Conference on Learning Representations (ICLR)*, 2020.

[15] Lei Fan, Qing-Yuan Jiang, Ya-Qi Yu, and Wu-Jun Li. Deep hashing for speaker identification and retrieval. In *Interspeech*, 2019.

[16] Electronic Frontier Foundation. Upstream vs. prism. `https://www.eff.org/702-spying`, 2018.

[17] Barton Gellman and Ashkan Soltani. Nsa surveillance program reaches 'into the past' to retrieve, replay phone calls. *The Washington Post*, 2014.

[18] Hannah Giorgis. When the fbi spied on mlk. *The Atlantic*, 2021.

[19] Timothy J Hazen, Wade Shen, and Christopher White. Query-by-example spoken term detection using phonetic posteriorgram templates. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, 2009.

[20] Hee Soo Heo, Bong-Jin Lee, Jaesung Huh, and Joon Son Chung. Clova baseline system for the voxceleb speaker recognition challenge 2020. *arXiv preprint arXiv:2009.14153*, 2020.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[22] J. Hsu. pyworld. `https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder`, 2021.

[23] Umar Iqbal, Pouneh Nikkhah Bahrami, Rahmadi Trimananda, Hao Cui, Alexander Gamero-Garrido, Daniel Dubois, David Choffnes, Athina Markopoulou, Franziska Roesner, and Zubair Shafiq. Your echos are heard: Tracking, profiling, and ad targeting in the amazon smart speaker ecosystem. *arXiv preprint arXiv:2204.10920*, 2022.

[24] Qin Jin, Arthur R. Toth, Tanja Schultz, and Alan W Black. Voice convergin: Speaker de-identification by voice transformation. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Apr. 2009.

[25] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. In *IEEE Transactions on Big Data*, volume 7, pages 535–547, 2019.

[26] Takuhiro Kaneko, Hirokazu Kameoka, Kou Tanaka, and Nobukatsu Hojo. Cyclegan-vc3: Examining and improving cyclegan-vcs for mel-spectrogram conversion. In *Interspeech*, 2020.

[27] Murizah Kassim, Ruhani Ab Rahman, Mohamad Azrai A Aziz, Azlina Idris, and Mat Ikram Yusof. Performance analysis of voip over 3g and 4g lte network. In *2017 International Conference on Electrical, Electronics and System Engineering (ICEESE)*, pages 37–41. IEEE, 2017.

[28] Andre Kassis and Urs Hengartner. Practical attacks on voice spoofing countermeasures. *arXiv preprint arXiv:2107.14642*, 2021.

[29] Dia Kayyali. The history of surveillance and the black community. `https://www.eff.org/deeplinks/2014/02/history-surveillance-and-black-community`, 2014.

[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Lantian Li, Chao Xing, Dong Wang, Kaimin Yu, and Thomas Fang Zheng. Binary speaker embedding. In *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2016.

[32] Yinghao Aaron Li, Ali Zare, and Nima Mesgarani. Starganv2-vc: A diverse, unsupervised, non-parallel framework for natural-sounding voice conversion. In *Interspeech*, 2021.

[33] Zhuohang Li, Yi Wu, Jian Liu, Yingying Chen, and Bo Yuan. Advpulse: Universal, synchronization-free, and targeted audio adversarial attacks via subsecond perturbations. In *CCS*, 2020.

[34] Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. Speech model pre-training for end-to-end spoken language understanding. In *Interspeech*, 2019.

[35] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: trainable text-speech alignment using kaldi. In *Interspeech*, 2017.

[36] RG McCurdy. Tentative standards for sound level meters. *Electrical Engineering*, 55(3):260–263, 1936.

[37] Masanori Morise, Hideki Kawahara, and Haruhiro Katayose. Fast and reliable f0 estimation method based on the period extraction of vocal fold vibration of singing voice and speech. *Journal of the Audio Engineering Society*, February 2009.

[38] Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. World: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, E99.D(7):1877–1884, 2016.

[39] Max Morrison, Brian Tang, Gefei Tan, and Bryan Pardo. Reproducible subjective evaluation. In *ICLR Workshop on ML Evaluation Standards*, April 2022.

[40] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *Interspeech*, 2017.

[41] Krishna Kanth Nakka and Mathieu Salzmann. Learning transferable adversarial perturbations. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[42] Patrick O'Reilly, Pranjal Awasthi, Aravindan Vijayaraghavan, and Bryan Pardo. Effective and inconspicuous over-the-air adversarial examples with adaptive filtering. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[43] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015.

[44] Manuel Pariente. pystoi. `https://github.com/mpariente/pystoi`, 2021.

[45] Tae Jin Park, Naoyuki Kanda, Dimitrios Dimitriadis, Kyu J. Han, Shinji Watanabe, and Shrikanth Narayanan. A review of speaker diarization: Recent advances with deep learning. *Computer Speech Language*, 72, November 2021.

[46] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[47] Jordi Pons, Santiago Pascual, Giulio Cengarle, and Joan Serrà. Upsampling artifacts in neural audio synthesis. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[48] A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

[49] Damien Ronssin and Milos Cernak. Ac-vc: Non-parallel low latency phonetic posteriorgrams based voice conversion. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021.

[50] Takaaki Saeki, Yuki Saito, Shinnnosuke Takamichi, , and Hiroshi Saruwatari. Real-time, full-band, online dnn-based voice conversion system using a single cpu. In *Interspeech*, 2020.

[51] L. Schmidt, M. Sharifi, and I. Lopez-Moreno. Large-scale speaker identification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[52] Ali Shahin Shamsabadi, Francisco Sepúlveda Teixeira, Alberto Abad, Bhiksha Raj, Andrea Cavallaro, and Isabel Trancoso. Foolhd: Fooling speaker identification by highly imperceptible adversarial disturbances. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[53] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *International Society for Music Information Retrieval (ISMIR)*, 2018.

[54] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[55] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2125–2136, 2011.

[56] Natalia Tomashenko, Xin Wang, Emmanuel Vincent, Jose Patino, Brij Mohan Lal Srivastava, Paul-Gauthier Noé, Andreas Nautsch, Nicholas Evans, Junichi Yamagishi, Benjamin O'Brien, Anaïs Chanclu, Jean-François Bonastre, Massimiliano Todisco, and Mohamed Maouche. The VoicePrivacy 2020 challenge: Results and findings. *Computer Speech &amp Language*, 74:101362, 2022.

[57] Verizon. Monthly ip latency statistics. https://www.verizon.com/business/terms/latency/, 2022.

[58] Chongshun Wang, Dani Yogatama, Adam Coates, Tony Han, Awni Y. Hannun, and Bo Xiao. Lookahead convolution layer for unidirectional recurrent neural networks. In *ICLR Workshop*, 2016.

[59] Miao Wang, Christoph Boeddeker, Rafael G. Dantas, and Ananda Seelan. pesq. `https://github.com/ludlows/python-pesq`, 2022.

[60] Human Rights Watch. Q & a: Us warrantless surveillance under section 702 of the foreign intelligence surveillance act. `https://www.hrw.org/news/2017/09/14/q-us-warrantless-surveillance-under-section-702-foreign-intelligence-surveillance`, 2017.

[61] Human Rights Watch. Secret evidence and the threat of more warrantless surveillance. `https://www.hrw.org/news/2018/01/11/secret-evidence-and-threat-more-warrantless-surveillance`, 2018.

[62] Yi Xie, Zhuohang Li, Cong Shi, Jian Liu, Yingying Chen, and Bo Yuan. Enabling fast and universal audio adversarial attack using generative model. In *AAAI*, 2021.

[63] Weiyi Zhang, Shuning Zhao, Le Liu3, Jianmin Li, Xingliang Cheng, Thomas Fang Zheng, and Xiaolin Hu. Attack on practical speaker verification system using universal adversarial perturbations. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[64] Ge Zhu, Fei Jiang, and Zhiyao Duan. Y-vector: Multiscale waveform encoder for speaker embedding. In *Interspeech*, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We claim to have designed a system to make highly effective, inconspicuous perturbations that can transfer to unforseen architectures in real-time. 1. We show this in our objective 4.4and subjective evaluations 4.5

(b) Did you describe the limitations of your work? [Yes] See Section 6

(c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [N/A]

(b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] It will be anonymized in the supplemental materials

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We provide a distribution of subjective quality ratings for each attack in in 2. We, however, do not provide error bars over random seeds for training.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Stated in 4.3

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We cite authors of LibriSpeech, VoxCeleb, ResNetSE34v2, ReSEval and Y-Vector

(b) Did you mention the license of the assets? [Yes] We mention the licenses for datasets and models in 4.2 and for ReSEval in 4.5. We note that the Y-Vector model does not have a License, to our knowledge.

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] In the Supplementary Materials

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [Yes] We obtained IRB approval. There are no known risks to the participants in out subjective evaluation. 4.5

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes] In C

A VoiceBlock implementation

A.1 Architecture

We provide additional details of the VoiceBlock architecture used in our experiments.

Phoneme encoder: The phoneme encoder consists of a feedforward network with two hidden layers of size 256 and ReLU activations, followed by two LSTM layers with hidden size 256. Using the LibriSpeech "train-clean-100" dataset and aligned phoneme labels provided by Lugosch et al. [34], we train the phoneme encoder with cross entropy loss for 25 epochs using the Adam optimizer with learning rate 1e-3. During training, a linear layer is appended to the phoneme encoder to render predictions over phoneme labels; we discard this layer after training.

Spectrogram encoder: We pass 64-bin mel-spectrograms through a 1×1 convolutional layer and gated linear unit (GLU) operating along the frequency axis, followed by a two linear layers with leaky ReLU activations and hidden size 512. The convolution layer upsamples the channel (frequency) dimension to 1024, from which the GLU obtains a 512-dimensional representation to pass to the linear layers.

Source speaker conditioning: For each frame of audio, we concatenate the 256-dimensional output of the phoneme encoder, the 512-dimensional output of the spectrogram encoder, and the one-dimensional pitch / periodicity / loudness features and linearly project to obtain a 512-dimensional vector. A fixed 512-dimensional source speaker embedding obtained from the ResNetSE34V2 model is passed through a 2 linear layers with leaky ReLU activations and hidden size 512, followed by batch normalization and a final linear layer to obtain a 1024-dimensional vector holding scales and biases for a feature-wise affine transformation (FiLM) of the encoder output. The affine transformation is applied and the resulting vector is passed to the bottleneck.

Bottleneck: Our bottleneck consists of two LSTM layers with hidden size 512. The encoder output is passed through the LSTM layers with a skip connection concatenated at the output. Following the application of a lookahead convolutional network, the resulting vector is projected linearly to obtain a 128-dimensional vector of filter controls for the current frame.

Decoder: We perform filtering with 128 bands per frame, and set L_∞ bound $\epsilon = 0.5$ to further constrain deviations from a "neutral" control configuration of $\vec{1}$. We experimented with alternative methods of regularizing predicted filter controls, such as L_2 normalization, but found that simple L_∞ clipping yielded satisfactory results.

Training: We train both VoiceBlock and the universal baseline attack using the Adam [30] optimizer with learning rate $1\mathrm{e}{-4}$ for 10 epochs over the LibriSpeech "train-clean" subset.

A.2 Auxiliary loss

We briefly summarize the multi-resolution STFT loss of Defosséz et al. [13] used in the training of our attacks. The loss is given by

$$\mathcal{L}_{aux}(g, u) = \mathcal{L}_{stft}(u, g(u)) + ||u - g(u)||_1 \tag{5}$$

where $\mathcal{L}_{stft}$ is given by a sum of magnitude and spectral convergence losses computed over M spectrogram resolutions:

$$\mathcal{L}_{stft}(a, b) = \sum_{i=1}^{M} \left[\mathcal{L}_{sc}^{(i)}(a, b) + \mathcal{L}_{mag}^{(i)}(a, b) \right] \tag{6}$$

$$\mathcal{L}_{sc}(a, b) = \frac{||\ |STFT(a)| - |STFT(b)|\ ||_F}{||\ |STFT(a)|\ ||_F} \tag{7}$$

$$\mathcal{L}_{mag}(a, b) = ||\ \log|STFT(a)| - \log|STFT(b)|\ ||_1 \tag{8}$$

Here $||\cdot||_F$ and $||\cdot||_1$ refer to the Frobenius and L_1 norms, respectively. Following Defosséz et al., we use $M = 3$ spectrogram resolutions with hop sizes of 50, 120, and 240 samples, FFT lengths of 512, 1024, and 2048, and window lengths of 240, 600, and 1200, respectively.

A.3 Streamer

We implement a version of VoiceBlock for processing audio streams in real time. All components are modified to support the computation of perturbations on overlapping windows of audio while the input and output streams process non-overlapping chunks. Where possible, components are complied

Table 2: Top-1 (T-1) and top-10 (T-10) recognition accuracy of the ResNetSE34v2 system on the de-identification attacks described in Section 4.3 when all query audio is passed through a Demucs [13] speech enhancement preprocessing stage

	ResNetSe34V2		**+Demucs**	
Approach	**T-1** ↓	**T-10** ↓	**T-1** ↓	**T-10** ↓
White noise	0.13	0.40	0.02	0.09
Spectral gating	0.02	0.11	0.02	0.11
Universal	0.14	0.22	0.87	0.97
VoiceBlock	0.02	0.10	0.04	0.15
No attack	0.97	0.99	0.96	0.99

to TorchScript to improve performance. Three buffers are added to process a stream of audio into overlapping chunks:

- Last Frame Input Buffer: Contains the last 128 samples of the last input audio to be appended to the oncoming chunk, before computing the overlapping windows.
- Lookahead Buffer: Contains overlapping frames to be used as lookahead.
- Output Buffer: Contains the last 128 samples output by VoiceBlock, to be overlap-added with the next processed chunk.

Code for our streaming implementation can be found at `https://master.d3hvhbnf7qxjtf.amplifyapp.com/`.

B Additional experiments

B.1 Attack robustness to preprocessing

In real-world settings, user audio may travel through various preprocessing stages before reaching a speaker recognition model. To simulate the presence of such stages, we perform attacks using the experiment configuration discussed in Section 4.3 but pass query utterances through a pretrained Demucs [13] speech enhancement model en route to the ResNetSE34v2 recognition system; results are reported in table 2. The speaker recognition system maintains its top-1 and top-10 accuracy on clean queries. In contrast to Chiquier et al. [10] we do not incorporate the enhancement model into our adversarial optimization, meaning that Demucs essentially functions as an unseen adversarial defense. We find that VoiceBlock loses almost no effectiveness when queries are passed through Demucs; by comparison, the universal attack loses most of its effectiveness, as Demucs is able to separate the low-magnitue but noisy perturbation from clean speech. Both the white noise and spectral gating attacks retain their effectiveness, presumably because they degrade audio beyond Demucs' capability to provide coherent reconstructions of the original speech. We leave as future work the exploration of the robustness of VoiceBlock against more sophisticated preprocessing pipelines and adversarial defenses.

B.2 Enrollment of adversarial queries

It is possible that adversarially-perturbed query utterances extracted from a VoiceBlock user's audio stream may themselves be enrolled by a surveiling speaker recognition system. In such cases, there are two possibilities:

1. VoiceBlock successfully de-identifies user speech, and adversarial queries are enrolled as a separate speaker profile from any existing clean utterances of the user
2. VoiceBlock fails to de-identify user speech, and adversarial queries are incorporated into an existing profile of the user containing clean utterances.

We examine both scenarios, again using the VoiceBlock attack discussed in Section 4.3. Results are presented in table 3. We find that while VoiceBlock is highly effective at de-identifying users against

Table 3: For sets of 15 clean and adversarial query utterances, we compare the top-1 (T-1) and top-10 (T-10) accuracy of speaker recognition with the ResNetSE34v2 model under three conditions. **Clean profile**: 20 clean utterances are enrolled per speaker. **Mixed profile**: 10 clean and 10 adversarial (VoiceBlock) utterances are enrolled per speaker. **Adversarial profile**: 20 adversarial (VoiceBlock) utterances are enrolled per speaker.

	Clean profile		**Mixed profile**		**Adversarial profile**	
Query processing	**T-1**	**T-10**	**T-1**	**T-10**	**T-1**	**T-10**
VoiceBlock	0.02	0.10	0.51	0.80	0.78	0.93
None	0.97	0.99	0.83	0.95	0.09	0.28

Table 4: De-identification performance of proposed and baseline models over the LibriSpeech `train-clean-360` dataset. To evaluate the intelligibility of the resulting audio, we ASR compute word- and character-error rates in addition to PESQ and STOI scores.

	Speech Quality Metrics		**ResNetSe34V2**		**Wav2Vec 2.0**	
Approach	**PESQ** $\uparrow$	**STOI** $\uparrow$	**T-1** $\downarrow$	**T-10** $\downarrow$	**WER** $\downarrow$	**CER** $\downarrow$
White noise	1.03	0.52	0.24	0.58	1.00	1.00
Spectral gating	1.13	0.64	0.03	0.18	0.50	0.29
Universal	1.38	0.86	0.09	0.24	0.16	0.08
VoiceBlock	3.72	0.94	0.05	0.22	0.02	0.01
No attack	4.64	0.99	0.99	0.99	0.01	0.01

a profile constructed from clean (unperturbed) utterances, its de-identification performance suffers significantly under both of the aforementioned conditions. This suggests that additional work is required to ensure that a VoiceBlock user remains a "moving target" to surveilling speaker recognition systems. One possible solution is to leverage targeted rather than untargeted attacks: by proactively "spoofing" specific (random) locations in the embedding space, VoiceBlock may hamper the creation of a single matching enrolled profile.

B.3 Word error rate evaluation

To measure the intelligibility of speech processed by the proposed and baseline methods, we evaluate the word- and character-error rates of each attack described in Section 4.3 on the LibriSpeech `train-clean-360` subset using a Wav2Vec 2.0 automatic speech recognition model [5]. The `train-clean-360` dataset contains utterances and accompanying transcriptions from 921 speakers not seen during training. We do not use the VoxCeleb dataset because it does not have the necessary transcriptions. We partition each speaker's data into query, enrollment, and conditioning partitions, as described in Section 4.2. Our results are reported in Table 4.

Our results demonstrate that pure signal-processing attacks such as spectral gating and white noise result in word- and character-error rates 20+ times that of VoiceBlock. VoiceBlock also outperforms all methods except spectral gating on the de-ientification task, with which it performs competitively. This indicates our method's ability to preserve the intelligibility of speech while adversarially modifying speaker characteristics.

C Listening Study

The listening study consists of 20 evaluation tasks, where the participants are shown the following text, along with 5 four second long audio examples and 5 corresponding sliders with values from 0 to 100:

> Listen to all recordings of a person speaking. Then, move the sliders to **rate the quality of each audio file from 0 (worst) to 100 (best)**. The higher-quality audio files are the ones that are more natural sounding, or have fewer audio artifacts (e.g., clicks, pops, noise, or otherwise sound 'unnatural'). **Note** - Each slider cannot be moved until its corresponding audio file has been listened to in its entirety.

It takes approximately 15 minutes to complete all the evaluation tasks. Participants are also asked to complete a listening test before proceeding to the audio evaluation. The listening test involves the participant listening to two audio files, and reporting the number of tones they heard. For each audio file, the participants are given 3 tries to correctly report the number of tones played. If a participant succeeds in the listening test and completes the evaluation tasks, then they are paid 3.00 USD, or equivalently, 12.00 USD / hour. If the participant fails the listening test, then they are paid 0.50 USD.