REAL-TIME NEURAL VOICE CAMOUFLAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic speech recognition systems have created exciting possibilities for applications, however they also enable opportunities for systematic eavesdropping. We propose a method to camouflage a person's voice over-the-air from these systems without inconveniencing the conversation between people in the room. Standard adversarial attacks are not effective in real-time streaming situations because the characteristics of the signal will have changed by the time the attack is executed. We introduce predictive attacks, which achieve real-time performance by forecasting the attack that will be the most effective in the future. Under real-time constraints, our method jams the established speech recognition system DeepSpeech 3.97x more than baselines as measured through word error rate, and 6.87x more as measured through character error rate. We furthermore demonstrate our approach is practically effective in realistic environments over physical distances.

1 INTRODUCTION

Nowadays, automatic speech recognition models are embedded in nearly all smart devices. Although these models have many exciting applications, the concern for the potential of these devices to eavesdrop is significant. It is becoming increasingly important to develop methods that give users the autonomy to safeguard their speech from voice processing software.

Fortunately, over the last decade, there has been work demonstrating that neural networks models are easily fooled. For example, they remain vulnerable to small additive perturbations (Carlini & Wagner, 2018), ambient noise (Xu et al., 2020), and unusual examples (Nguyen et al., 2015). Predominant methods such as gradient-based methods and their variants have remained the standard approach to generating challenging examples for deep neural networks (Madry et al., 2019). However, to achieve this, these methods require the full input upfront, and thus users can not practically use them as they continuously speak.

Therefore, the community has increasingly been focusing on researching general, robust methods of breaking neural networks that can be used in real-time. We define *robust* to mean an obstruction that can not be easily removed, *real-time* to mean an obstruction that is generated continuously as speech is spoken, and *general* to mean applicable to the majority of vocabulary in a language. Existing prior work has successfully tackled at least one of these three requirements, but none all three. While some work is real-time (Chen et al., 2020; Schönherr et al., 2018), these disruptions can be filtered out as

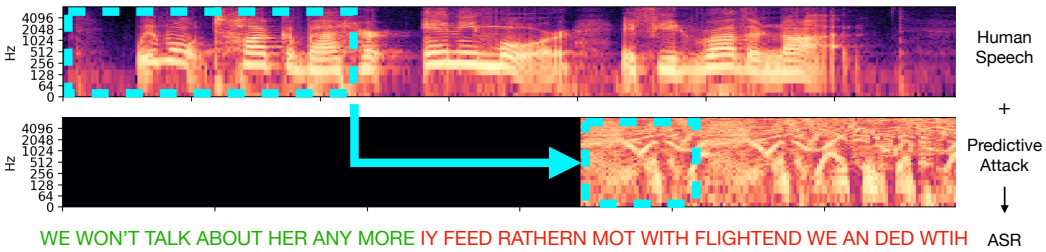

Figure 1: We introduce "Neural Voice Camouflage," an approach that disrupts automatic speech recognition systems in real time. To operate on live speech, our approach must predict corruptions into the future so that they may be played in real-time. The method works for the majority of the English language. Green/red indicates correct/incorrect transcription respectively.

they are constrained to specific frequency ranges. Universal attacks (Lu et al., 2021) can be similarly subtracted. Gong et al. (2019) achieved both real-time and robust obstructions, but are limited to a predefined set of ten words.

Streaming audio is a particularly demanding domain to disrupt because the calculation needs to be performed in real-time. By the time a sound is computed, time will have passed and the streaming signal will have changed, making standard generative methods obsolete. The sampling rate of audio is at least 16 kHz, meaning the corruption for a given input must be estimated and played over a speaker within milliseconds, which is currently infeasible. Additionally, when attacks are played over-the-air, the attack needs to be loud enough to disrupt any rogue microphone that could be far away. The attack sound needs to carry the same distance as the voice.

We introduce predictive attacks, which are able to disrupt any word that automatic speech recognition models are trained to transcribe. Our approach achieves real-time performance by *forecasting* an attack on the future of the signal, conditioned on two seconds of input speech. Our attack is optimized to have a volume similar to normal background noise, allowing people in a room to converse naturally and without monitoring from an automatic speech recognition system.

Forecasting with deep neural networks has already been successfully used in other domains to achieve real-time performance, for instance in packet loss concealment (Pascual et al., 2021). In this paper, we demonstrate how and why this approach lends itself particularly well to developing general, robust and real-time attacks for automatic speech recognition models. Our experiments show that predictive attacks are able to largely disrupt the established DeepSpeech recognition system. On the standard, large-scale dataset LibriSpeech, our approach causes at least a three fold increase in word error rate over baselines, and at least a six fold increase in character error rate.

Our method is practical and straightforward to implement on commodity hardware. We additionally demonstrate the method works inside real-world rooms with natural ambient noise and complex scene geometries. We call our method Neural Voice Camouflage.

2 Related Work

Breaking Neural Networks: Szegedy et al. (2014) first discovered adversarial attacks in computer vision. Since then, a large number of methods to break neural networks have been introduced (Madry et al., 2019; Kurakin et al., 2017; Carlini & Wagner, 2017; Croce & Hein, 2020; Moosavi-Dezfooli et al., 2016; Goodfellow et al., 2014) , where noise optimized by gradient descent fool the state-of-the-art models. Audio adversarial attacks (Carlini & Wagner, 2018; Qin et al., 2019; Yakura & Sakuma, 2019; Schönherr et al., 2018) have also been constructed. Gradient based iterative adversarial attacks, while effective, are computationally intensive, and need to see the whole example first before launching the attack. Faster adversarial attacks uses generators to generate attacks (Xiao et al., 2019). However, the attacks are still offline. To make the adversarial attack reliable for live speech, the attacker needs to anticipate the future in an online manner.

Online Attacks: Real-time attacks are an emerging area of research in machine learning and there have been several initial works. For example, Gong et al. (2019) develop a reinforcement learning based approach to balance the trade-off between number of samples seen and attack deployment duration. They also optimize a volume trade-off to achieve over-the-air performance. While they learn to disrupt spoken keyword detection (a predefined set of ten words), our approach is able to obfuscate entire sentences. Further, attacks for streaming data with bayesian approaches have been proposed (Braverman et al., 2021; Seraphim & Poovammal, 2021). However, they are unable to tackle high-dimensional data such as audio. Another direction prior work has taken to create online attacks is to constantly be attacking a certain word (Li et al., 2019). Although this works in real-time, it only targets the wake word, and not full sentences. There also have been a few methods that jam microphones by emitting sound outside the range of human hearing. For example, Chen et al. (2020) developed an approach to emit ultrasonic attacks and Schönherr et al. (2018) also generate attacks outside the human hearing range. However, by limiting the attack to specific frequencies, a defender can design a microphone that filters this set of frequencies out.

Robustness: Due to the importance of this problem, there has been extensive research into learning robust models (Madry et al., 2019; Carmon et al., 2019; Wang et al., 2020; Mao et al., 2019; 2020; 2021). However, building defenses is challenging, and work has even shown that many basic de-

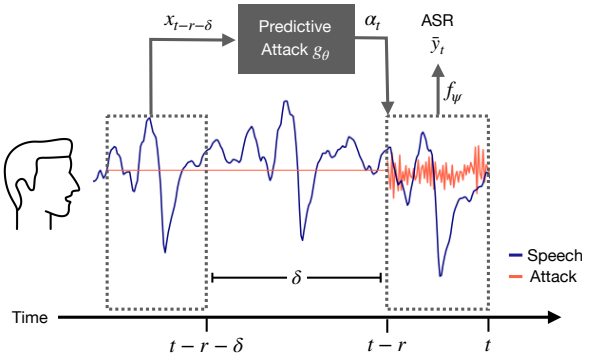

Figure 2: We illustrate our problem set-up for predictive attacks. In order to attack the audio starting at time $t - r$, we need to start computing the attack by time $t - r - \delta$, assuming it takes δ time to record, compute and play the attack. Our approach is able to obtain real-time performance by predicting this attack in the future, given the previous observations of the stream.

fenses, such as adding randomness to the input, are not effective (Athalye et al., 2018). Among all the defense strategies, adversarial training proposed by Madry et al. (2019) is the standard defense that has been most widely used. However, adversarial training has the drawback that it improves robustness accuracy at the cost of reducing the original accuracy (Tsipras et al., 2019), which is the reason that adversarial training is not used in most real-world applications. In this paper, we show our approach is still effective against these established defenses.

Real-time Machine Learning: Interest in real-time artifical intelligence dates back to 1996, starting with anytime algorithms, which return a solution at any given point in time (Zilberstein, 1996). More recently, there have been challenges to evaluate vision models in real-time (Kristan et al., 2017). Generally, there has been a focus on speeding up forward passes to allow for faster inference, thereby approaching real-time (Howard et al., 2017). Recently, the community has recently taken an interest in establishing robust metrics and evaluations for real-time inference (Li et al., 2020). In addition, there has been recent work in leveraging deep neural network predictions to achieve real-time performance has been applied to speech packet loss concealment (Pascual et al., 2021).

3 METHOD

We present our approach for creating real-time obstructions to automatic speech recognition (ASR) systems. We first motivate the background for real-time attacks, then introduce our approach that achieves online performance through predictive attack models.

3.1 STREAMING SPEECH RECOGNITION

Let x_t be a streaming signal that represents the input speech up until time t. The goal of ASR is to transcribe this signal into the corresponding text y_t. The field often estimates this mapping through a neural network $\hat{y}_t = f_\psi(x_t)$ where the parameters ψ are optimized to minimize the empirical risk $\min_\psi \mathbb{E}_{(x,y)} \left[\mathcal{L}\left(\hat{y}_t, y_t\right)\right]$. For modeling sequences, the CTC loss function is a common choice for $\mathcal{L}$ (Graves et al., 2006).

In offline setups, we can corrupt the neural network f_ψ with a standard adversarial attack. These attacks work by finding a minimal additive perturbation vector α_t that, when added to the input signal, produces a high loss: $\arg\max_{\alpha_t} \mathcal{L}\left(f_\psi\left(x_t + \alpha_t\right), y_t\right)$ subject to a bound on the norm of the perturbation $\|\alpha_t\|_\infty < \epsilon$. Adversarial attacks, such as projected gradient descent (PGD) or fast gradient descent, have been widely effective on vision and speech datasets (Madry et al., 2019; Goodfellow et al., 2014; Carlini & Wagner, 2018). Defending against them both empirically and theoretically remains an active area of research today.

Standard adversarial attacks will optimize the perturbation vector α_t conditioned on the current position of the stream x_t. However, by the time the solution α_t is found for a stream, the attack will be obsolete because time will have passed and the condition will have almost certainly changed. Audio is a particularly demanding domain because the high sampling rate (as high as 48 kHz) would require attacks to be computed nearly instantaneously (less than 20 microseconds). Furthermore, applying the stale α_t to the future $x_{t+\delta}$ will not work because the attack vectors are optimized to corrupt the features of their input, which may vary over time.

3.2 PREDICTIVE REAL-TIME ATTACKS

We propose a class of predictive attacks, which enable real-time performance by forecasting the attack vector that will be effective in future time steps. It will invariably take some time for the attack to be computed. For attacks to operate in real-time environments, this means the attack needs to be optimized not for the observed signal, but for the unobserved signal in the future. If our observation of the signal x_t is captured at time t and our algorithm takes δ seconds to compute an attack and play it, then we need to attack the signal starting at $x_{t+\delta}$. However, creating these attacks is challenging practically because real-world signals are stochastic and multi-modal. Due to the high uncertainty, generating future speech $x_{t+\delta}$ for the purpose of computing attacks is infeasible.

Rather than forecasting the signal, we will learn to forecast the attack vector, which encloses all possible variations of the next utterances conditioned on the current input. This attack will learn to "hedge the bet" by finding a single, minimal pattern that robustly obstructs all upcoming possibilities. Under the perturbation bound ϵ, we model predictive attacks as:

$$\alpha_{t+\delta+r} = g_\theta\left(x_t\right) \quad \text{s.t.} \quad \left\|g_\theta\left(x_t\right)\right\|_\infty < \epsilon, \tag{1}$$

where g_θ is a predictive model conditioned on the present input speech and paramaterized by θ. To be consistent with our notation, which represents x_t as the signal until time t, we include an additional offset r to represent the temporal duration of the attack. To satisfy the constraint on the perturbation bound, we use the tanh activation function to squash the range to the interval $[-1, 1]$ before multiplying the result by a scalar ϵ. This ϵ is equal to the product of a predetermined multiplier m and the maximum of the absolute value of the input speech waveform.

With predictive attacks, the algorithm for generating obstructions in real-time becomes straightforward. After the microphone observes x_t, the speakers need to play $\alpha_{t+\delta+r}$ exactly δ seconds later. Since sound is additive modulo reverberation, this will cause a third-party microphone to receive the corrupted signal $x_{t+\delta+r} + g_\theta(x_t)$. We found modeling the room acoustics was unnecessary because significant reverberation already breaks state-of-the-art ASR models.

We will use neural networks to instantiate the predictive model g. To obtain real-time performance, our feed forward calculation needs to be less than the delay δ into the future. On commodity hardware today, this calculation is on the order of 50 milliseconds.

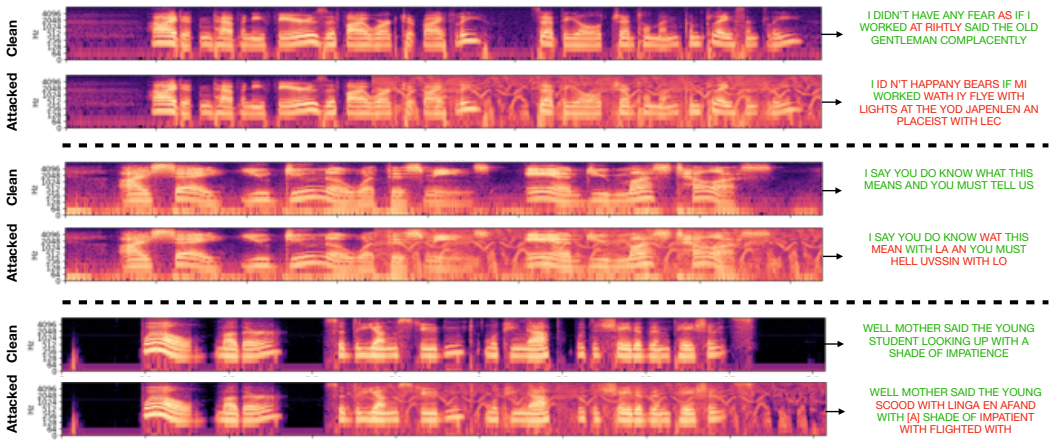

Figure 3: We illustrate three examples of our attack in action. The first row in a pair is the clean spectrogram of the input speech signal. The second row in the pair is the attacked spectrogram of the speech signal. We note that the difference becomes visible after 2.5s of the speech signal, since our method requires 2s of input and has a delay of (δ) 0.5s before it can predict an attack. As seen, the predicted attack resembles that of speech formants.

3.3 LEARNING

We will learn the parameters θ of our predictive model from a large-scale labelled speech dataset, for a specific automatic speech recognition model f_ψ. We formulate this as the maximization problem:

$$\max_\theta \; \mathbb{E}_{(x_t, y_t)} \left[\mathcal{L}\left(\bar{y}_t, y_t\right) \right] \quad \text{s.t.} \quad \bar{y}_t = f_\psi\left(x_t + g_\theta\left(x_{t-r-\delta}\right)\right) \quad \text{and} \quad \left\| g_\theta\left(x_t\right) \right\|_\infty < \epsilon \tag{2}$$

where $\bar{y}_t$ is the result of the ASR model after our attack to x_t. The objective will drive the model to find attacks that, in the future of the signal, will maximize the expected loss of the ASR model. We optimize θ using stochastic gradient descent while keeping ψ fixed. Once training is performed offline, inference is efficient, requiring just a single feed-forward computation.

3.4 IMPLEMENTATION DETAILS

The input to our network g_θ is the Short-Term Fourier Transform (STFT) of the last 2 seconds of the speech signal. The network outputs a waveform of 0.5 seconds, sampled at 16kHz. To calculate the STFT, we use a hamming window length of 320 samples, hop length of 160 samples, and FFT size of 320, resulting in an input dimension of $2 \times 161 \times 204$. We use a 13 layer convolutional network. The appendix has full network details.

We also experimented with a network that ouputs an STFT instead of waveform. However, regressing an STFT has no guarantee that there will be a corresponding waveform to it, which means we can not actually play it in practice. In order to prevent this from happening, there has been work that adds an additional term to minimize the loss between the predicted STFT and the nearest valid STFT (Marafioti et al., 2019). However, we found that predicting the waveform directly was both simpler and more effective.

Speech datasets generally do not have time stamps to their transcriptions. In order to train our model, we need to compute the loss between the predicted speech and the ground-truth speech, meaning that in training, we need to attack the entire speech signal, not just a small segment. We therefore need to schedule our forward and backward passes such that we have computed the attack for the entire segment before we calculate the gradients. We optimized our predictive network g_θ for 4 epochs with batch size 32 across 8 NVIDIA RTX 2080 Ti GPUs on the 100-hour LibriSpeech dataset. This computation took approximately 2 days. The learning rate started at $1.5 \cdot 10^{-4}$ and decreased using an exponential learning rate scheduler, with a learning anneal gamma value of 0.99. Our code was written in PyTorch (Paszke et al., 2019) and PyTorch-Lightning (Falcon & .al, 2019).

4 EXPERIMENTS

The objective of our experiments is to analyze predictive attacks under the constraints of real time speech streams. We first introduce the experimental setup, baselines, and defense models. We then present our experimental results with both quantitative and qualitative evidence.

We train on the LibriSpeech clean 100 hour dataset, validate on the LibriSpeech clean development set, and test on the LibriSpeech test set. For our approach, we restrict the amplitude of our predicted attack to be 0.008 times the maximum of the absolute value of the amplitude of the speech signal. We call this the relative amplitude throughout the paper. Intuitively, our attack sounds similar to the sound of a quiet air-conditioner in the background. We additionally evaluate on several baselines, including various levels of white noise as well as projected gradient descent. For some of baselines, we experimented with making the amplitude louder, but never below the amplitude of our predicted attack. In order to measure the time taken fairly, we measured the time necessary to create the attack vector for an input of two seconds averaged over 200 runs.

4.1 ATTACK METHODS AND METRICS

To evaluate our approach, we compare against several methods to obstruct the speech signal.

Uniform Noise: One straight-forward way to obstruct speech is to play white noise. We use the same amplitude that our attack uses. We also experimented with amplitudes that are an order of magnitude louder than our attack.

Table 1: Under real-time constraints, we quantitatively evaluate our attack method and baselines with and without defense mechanisms.

Approach	Running Time (s)	Power of Noise	DeepSpeech		+Denoiser		+AdvTrain	
			WER	CER	WER	CER	WER	CER
No Attack	0	0	11.3	3.6	12.1	4.0	18.7	6.7
Uniform Noise	0.0006	0.008	12.8	3.9	12.2	4.0	19.4	4.4
Uniform Noise	0.0006	0.05	28.4	12.0	12.2	4.1	19.3	4.3
Uniform Noise	0.0006	0.1	47.1	23.3	12.2	4.1	19.3	4.4
Online PGD	3.13†	0.008	20.5	7.8	27.7	11.8	22.5	8.4
Our Approach	0.014	0.008	**81.5**	**53.6**	**47.0**	**24.5**	**52.5**	**29.0**
Offline PGD	3.13†	0.008	100.9	68.4	28.0	12.0	82.8	52.5

The † indicates a lower bound because running PGD on the denoiser takes twice the amount of time.

Offline Projected Gradient Descent (PGD): Projected gradient descent is the standard method for attacking speech samples (Madry et al., 2019). It calculates the gradient of the attack using back-propagation, adds this gradient multiplied by a step size to the attack vector, and projects this sum back into the valid bound by clipping if the gradient exceeds the designated range. For the DeepSpeech model, we run 10 iterations of projected gradient descent on the input speech signal with step size equal to 20% of the bound. For the denoiser, we ran gradient descent for 30 iterations. Since projected gradient descent requires access to the entire signal and cannot be conducted online, we use this baseline to understand what the best attack could be if we had access to the future.

Online Projected Gradient Descent (PGD): Offline projected gradient descent does not work in real-time, since PGD requires the entire input signal in order to optimize the attack vector, and by the time the input signal is recorded, it has already passed. To make an online version of PGD, we calculate the PGD from the window of the input stream in the same manner as described for the offline method, but apply it to the future time. We note that this is unfair to our own approach, because PGD is at least two orders of magnitude slower than our approach.

Our Approach: We finally evaluate our approach, which requires just a single forward pass per half second of input speech. Our attack takes 0.014 seconds for a single forward pass, meaning that we need to be forecasting at least that amount into the future. We experimented with several options for how far into the future we forecast, using larger delays to allow time for recording speech and play back of attack (0.5s, 0.75s, 1.0s). We found that 0.5s performed best empirically.

The most common way to evaluate speech recognition models is through word and character error rates. We evaluate our attacks at their capability to *increase* errors. **Word Error Rate (WER)** measures the proportion of words that were incorrectly predicted, defined as $(S_w + D_w + I_w)/N_w$, where S_w represents the number of word substitutions, D_w the number of word deletions, I the number of word insertions, and N_w is the number of words. **Character Error Rate (CER)** measures the proportion of characters that were incorrectly predicted, which is important to analyze because our attack might just disrupt a single letter but the word is still intact. It is defined as $(S_c + D_c + I_c)/C_c$, where S_c represents the number of character substitutions, D_c the number of character deletions, I the number of character insertions, and N_c is the number of characters.

4.2 Robust Models

We evaluate our approach with both standard automatic speech recognition models as well as their robust counterparts. There are two main methods to make models robust: input preprocessing methods and adversarial training (Zelasko et al., 2021). The former fortifies the mode by attempting to clean the data of the attack, and the latter by strengthening the robustness of the model against attacks. We implement both methods to evaluate our approach.

Audio Denoiser: The standard way to suppress noise is to denoise the input signal. We use a state-of-the-art audio denoiser on the attacked inputs (Xu et al., 2020). In order to make our attacks robust to this form of preprocessing, we retrain our predictive model g_θ, this time passing $\hat{y}_{x_t}$ through the denoiser model h_ϕ, before passing it the automatic speech recognition model: $\bar{y}_t = f_\psi\left(h_\phi\left(x_t + g_\theta\left(x_{t-r-\delta}\right)\right)\right)$. Once again, we keep the automatic speech recognition model f_ψ and the denoiser model h_ϕ fixed, while updating our predictive model g_θ.

Adversarial Training: In addition, we use adversarial training to create a robust speech recognition system. We fine-tune the automatic speech recognition model on the adversarial examples. To maintain the performance on clean examples, we also add regular inputs in our training, following standard practice (Zhang et al., 2019). We train the DeepSpeech model f_ψ, and every batch contains half clean inputs, and half attacked inputs with 3 steps of projected gradient descent. This is already more attack than the fast-adversarial training approaches (Wong et al., 2020), as they use 1 step, making this model very robust. We call the robust model f'_ψ. There is always a trade-off between robustness and clean accuracy (Tsipras et al., 2019). We stop training once the WER on the attacked inputs dropped sufficiently, from 100.9% to 82.8%, and when the WER on the clean inputs increased from 11.3% to 18.7%. As is standard practice with adversarial training, we retrained our predictive model g_θ with the new robust f'_ψ, giving g'_θ.

4.3 Quantitative Analysis

Table 1 shows that our predictive attack is able to significantly disrupt automatic speech recognition systems. When we evaluate with the standard model (DeepSpeech column), the predictive attack is able to produce a WER that is nearly four times more effective than a standard online PGD attack. The white noise is able to corrupt the signal, but it requires substantially more amplitude than our approach. Even when the white noise amplitude is an order of magnitude larger than ours, our method is still more effective. We see a similar rate with the CER, suggesting that its completely corrupting words. The performance of the offline attack shows that observing the future is able to further improve the error rate, at the cost of not being able to be real-time.

We next evaluate our model when the ASR system has a defense mechanism. As expected, the defense mechanisms cause the WER for the attacked inputs to go down for all attack approaches. However, our attack still outperforms the baselines by nearly double in both cases. The denoiser is particularly effective at removing the white noise, which is expected as this is what it is trained to do. Moreover, the state of the art methods for audio denoising, published just last year (Xu et al., 2020), are actually very effective at removing PGD attacks. However, our approach still manages to fool the denoiser.

We also ran inference on the adversarially trained DeepSpeech without retraining our predictive model. Training with the defense improves the WER by only 8.4% and the CER 7.8%. This shows that our method is still strong even without updating our own predictive model when the automatic speech recognition model has been retrained to be more robust.

4.4 Characterizing the Attack

In order to analyze how our model produces attacks, we performed several quantitative experiments.

Does the model attack vocal timbers? Our attack model learns which frequencies to produce at each time in order to maximize the error rate of the ASR system. One possible strategy the model could learn is to detect key features about the person's vocal timbers and prioritize those during the attack. To investigate this, we swapped attacks from different speakers. Since input speech signals have varying lengths, the attacks will also have varying lengths. For the attack that is shorter than the speech, we repeat the attack until the entire speech covered. Conversely, if the attack is longer than the speech, then we cut it short. Our results shows that by swapping attacks for speakers, the WER

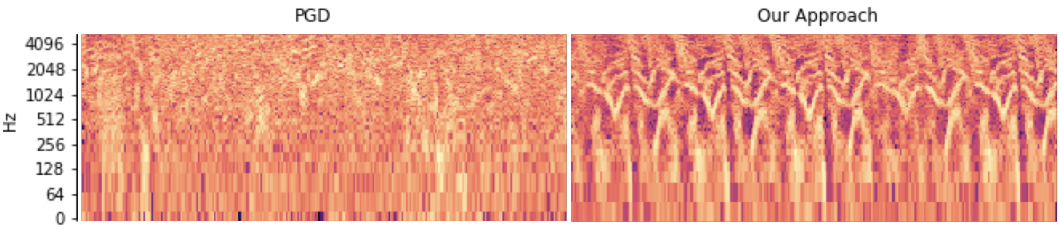

Figure 4: For a given input speech, we display both the PGD attack and our predictive attack. Unlike PGD, we note that our attack has features resembling formants.

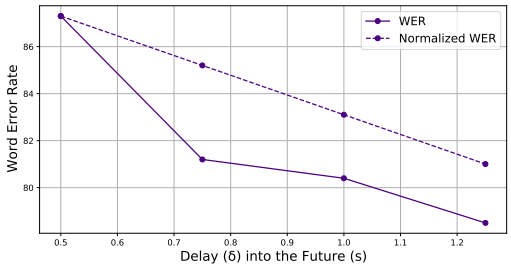

Figure 5: Delay vs. Word Error Rate

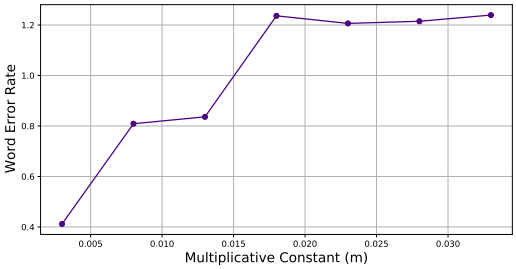

Figure 6: Multiplier vs. Word Error Rate

and CER both drop. In doing so, the WER drops from 81.5% to 35.0% and the CER drops from 53.6% to 19.9%. This indicates that the model is predicting corruptions that are speaker specific.

How robust is the attack to temporal shifts? In practical settings, the attack may not launch at *exactly* the right time due to various delays in software and hardware systems. Therefore, we analyze how the delay δ influences the success of our attack. The larger the delay δ, the further into the future our model needs to predict. We train a model to predict $\delta = 0$ into the future, and Figure 5 shows that, when it is applied for a larger delay δ, the WER drops. There are two factors that could explain this drop. The first is that as the delay increases, there is a shorter amount of time for which the speech is attacked. The second factor is that as the delay increases, we are predicting further into the future, thus the future becomes more uncertain. In order to disentangle these two factors, for each different delay, we linearly scale the error rate in proportion to the decrease of time for which the attack is active (shown in the dashed line). The decreasing plot shows that the attack is sensitive to the timing even when we factor in the reduced time to deploy it. This suggests the model is learning to predict key features about the upcoming speech. However, the performance drop is not severe, showing our approach is relatively robust during inaccurate timing.

How does power impact attack performance? In offline settings, normal adversarial attacks aim to reduce the volume as much as possible. However, this is more challenging in over-the-air settings because there may be background ambient noise and the rogue recording device may be far away. To investigate which level of amplitude a person should select, Figure 6 shows there is a linear

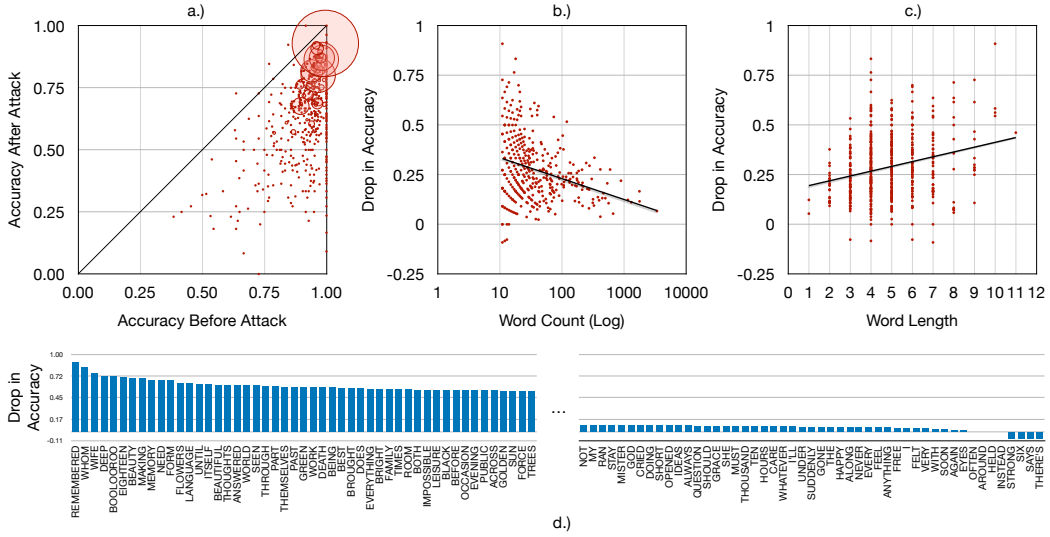

Figure 7: We analyze which words are the easiest and hardest to attack. **a**) For each word, we plot clean accuracy against the attacked accuracy. The size of the circle is proportional to the count of the word. **b**) We plot the log word count versus the drop in accuracy, with a black logarithmic trend line. **c**) We plot the drop in accuracy (before attack minus after attack) versus the word length. **d**) We plot a histogram of the easiest 50 words to attack and the hardest 50 words to attack.

ASR with No Attack:
DID YOU SEE THAT THE LAD GOT RENEVATED IT LOOKEDS FANTASTIC I ALSO JUST GOT A NEW MONATOR

With With Noise (0.016):
DID YOU SEE THAT THE LAD GOT RENOVATED IT LOOKES FANTASTIC I ALSO JUST GOT AND NEW MANATOR

With Our Attack (0.008):
DID YOU SEE THAT THE LAD GOT RENEVATID WITH REVENT LOOKES FANTASTED WITH REASONS WITH THEY ALSO TOSCAT AND NEUMANITOR

ASR with No Attack:
DO YOU WANT TO GO SEE SOME ART THIS WEEK THEN WE CAN GO TO THER RESTAURANT

With White Noise (0.016):
DO YOU WANT TO GO SEE SOME ART THIS WEEK THEN WE CAN GO TO THE RESTAURANE

With Our Attack (0.008):
DO YOU WANT TO GO SE[e]SOME AR[T] THIS HORK LIGHTED WITH HANY YTRIS ARE TO THER REST REMANT

Figure 8: We show how our attack works in realistic rooms with diverse acoustic environments. For our attack, we use a relative amplitude of 0.008. For the white noise, we found that we needed to increase the relative amplitude to 0.016 in order for the baseline to be effective.

relation between the power and the error rate until a critical point at about 0.02. In our real world experiments, we found the value of 0.008 to be effective in real room environments.

What makes a word easy or hard to attack? For each word, Figure 7a plots the recognition accuracy both before and after the attack. Since most of the points are below the diagonal line, this shows our attack is effective for most words. However, some words drop in accuracy more than others. Figure 7b compares this drop in accuracy to how often the word is spoken in the dataset. The results show that the most common words (e.g. "the", "our", "they") are the most difficult to disrupt. However, by definition, the common words carry low information content, thus making them less crucial to attack. Figure 7c compares the drop in accuracy versus the word length in characters, showing a positive correlation. This result suggests that longer words are generally easier for our model to attack, possibly because they have more temporal structure to predict.

Which words are guarded by our attack? Figure 7d displays a histogram of the drop in accuracy for the top and bottom 50 words. The word "remembered" experiences the most significant change in accuracy, nearly becoming unrecognizable in our dataset. On the other hand, some shorter words, such as "held" or "often", are not impacted by our attack. There are only four out of 1631 cases where our attack improves the recognition rate of the word.

How are predictive attacks qualitatively different from PGD attacks? Finally, we also visualize our predictive model's attack as well as the offline projected gradient attack (PGD) in Figure 4. While PGD is difficult to interpret, we notice that our attack takes a form similar to forments in speech, as can be seen by the wave-like features in the attack rows.

4.5 Real Room Effectiveness

Although our attack is optimized without factoring in the room impulse response function, we find it generalizes well to real-world settings. We record a person speaking in two different areas, a workspace and a kitchen. As a person speaks, there will be reverberation and ambient noise in addition to our attack. To understand how this impacts our approach, we played the attack through speakers in the same room. A third-party device consequently receives the sum of the attack, the speech, and the ambient noise convolved with the room impulse response function. Figure 8 shows a few examples of our attack in acoustic environments.

5 Conclusion

In order to corrupt automatic speech recognition models in practice, attacks need to be real-time. To achieve this, our approach predicts an attack for the future unobserved speech, after a delay δ, allowing time for the system to generate an attack and play it. We found that our approach works even when the automatic speech recognition model is robust.

ETHICAL CONSIDERATIONS

Our research is founded on ethical considerations. We are excited about the potential for automatic speech recognition to push the frontier of technology, such as in human-user interaction, telecommunications, accessibility, and education. We are also keenly aware that there can be negative consequences from the deployment of machine learning models in practice. Notably, automatic speech recognition systems have raised concerns with respect to privacy. Our method is designed to protect user privacy, and return the control of user speech data back to users.

One potential limitation of our approach is that it is trained on Western speech data, and may not generalize to other cultures that are linguistically and phonetically different. The method has also not been validated on different languages or people with speech impediments. As such, the dataset and results are not representative of the population. Deeper understanding of this issue requires future studies in tandem with linguistic and socio-cultural insights.

The authors attest that they have reviewed the ICLR Code of Ethics and we acknowledge that this code applies to our submission. We are committed in our work to abide by the ethical principles from the ICLR Code of Ethics.

REFERENCES

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 274–283. PMLR, 2018.

Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. Adversarial robustness of streaming algorithms through importance sampling. *arXiv preprint arXiv:2106.14952*, 2021.

Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text, 2018.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pp. 39–57, 2017.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/32e0bd1497aa43e02a42f47d9d6515ad-Paper.pdf`.

Yuxin Chen, Huiying Li, Shan-Yuan Teng, Steven Nagels, Zhijing Li, Pedro Lopes, Ben Y Zhao, and Haitao Zheng. Wearable microphone jamming. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2020.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

WA Falcon and .al. Pytorch lightning, 2019.

Yuan Gong, Boyang Li, Christian Poellabauer, and Yiyu Shi. Real-time adversarial attacks. *arXiv preprint arXiv:1905.13399*, 2019.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka ˇCehovin Zajc, Tomas Vojir, Gustav Hager, Alan Lukezic, Abdelrahman Eldesokey, et al. The visual object tracking vot2017 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 1949–1972, 2017.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2017.

Juncheng B Li, Shuhui Qu, Xinjian Li, Joseph Szurley, J Zico Kolter, and Florian Metze. Adversarial music: Real world audio adversary against wake-word detection system. *arXiv preprint arXiv:1911.00126*, 2019.

Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *European Conference on Computer Vision*, pp. 473–488. Springer, 2020.

Zhiyun Lu, Wei Han, Yu Zhang, and Liangliang Cao. Exploring targeted universal adversarial perturbations to end-to-end asr models. *arXiv preprint arXiv:2104.02757*, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Chengzhi Mao, Amogh Gupta, Vikram Nitin, Baishakhi Ray, Shuran Song, Junfeng Yang, and Carl Vondrick. Multitask learning strengthens adversarial robustness. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision – ECCV 2020*, pp. 158–174, Cham, 2020. Springer International Publishing.

Chengzhi Mao, Mia Chiquer, Hao Wang, Junfeng Yang, and Carl Vondrick. Adversarial attacks are reversible with natural supervision, 2021.

Andres Marafioti, Nicki Holighaus, Nathanael Perraudin, and Piotr Majdak. Adversarial generation of time-frequency features with application in audio synthesis, 2019.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2016.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

Santiago Pascual, Joan Serrà, and Jordi Pons. Adversarial auto-encoding for packet loss concealment. *arXiv preprint arXiv:2107.03100*, 2021.

Adam Paszke, Sam Gross, 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, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Yao Qin, Nicholas Carlini, Ian Goodfellow, Garrison Cottrell, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition, 2019.

Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *arXiv preprint arXiv:1808.05665*, 2018.

Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding, 2018.

B Ida Seraphim and E Poovammal. Adversarial attack by inducing drift in streaming data. *Wireless Personal Communications*, pp. 1–25, 2021.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy, 2019.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*, 2020.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training, 2020.

Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks, 2019.

Ruilin Xu, Rundi Wu, Yuko Ishiwaka, Carl Vondrick, and Changxi Zheng. Listening to sounds of silence for speech denoising, 2020.

Hiromu Yakura and Jun Sakuma. Robust audio adversarial example for a physical attack. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Aug 2019. doi: 10.24963/ijcai.2019/741. URL `http://dx.doi.org/10.24963/ijcai.2019/741`.

Piotr Zelasko, Sonal Joshi, Yiwen Shao, Jesus Villalba, Jan Trmal, and Najim Dehak oband Sanjeev Khudanpur. Adversarial attacks and defenses for speech recognition systems, 2021.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy, 2019.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–73, 1996.

A Appendix

A.1 Network Architecture

The architecture is comprised of 8 down-sampling convolutional blocks, followed by 4 up-sampling convolutional blocks, followed by a linear layer.

The down-sampling convolutional block is comprised of a reflection padding, followed a 2d convolution layer, followed by a 2d batch norm, followed by a prelu activation function. The first downsampling block has 64 channels, the second through the seventh have 128, and finally the last one has 256 channels. The last conv block also has a leaky relu instead of prelu. Here the signal is reshaped into a one-hot vector.

The upsampling blocks are comprised of 1-dimensional ConvTranspose 1d, and a leaky relu activation function. The first has 64 channels, the second 32, the third 16, the fourth, 1. Finally, the linear layer is followed by a tanh activation function.