SpecPatch: Human-in-the-Loop Adversarial Audio Spectrogram Patch Attack on Speech Recognition

Hanqing Guo
SEIT Lab
Michigan State University
East Lansing, MI, USA
guohanqi@msu.edu

Yuanda Wang
SEIT Lab
Michigan State University
East Lansing, MI, USA
wangy208@msu.edu

Nikolay Ivanov
SEIT Lab
Michigan State University
East Lansing, MI, USA
ivanovn1@msu.edu

Li Xiao
Michigan State University
East Lansing, MI, USA
lxiao@cse.msu.edu

Qiben Yan
SEIT Lab
Michigan State University
East Lansing, MI, USA
qyan@msu.edu

ABSTRACT

In this paper, we propose SpecPatch, a human-in-the loop adversarial audio attack on automated speech recognition (ASR) systems. Existing audio adversarial attacker assumes that the users cannot notice the adversarial audios, and hence allows the successful delivery of the crafted adversarial examples or perturbations. However, in a practical attack scenario, the users of intelligent voice-controlled systems (e.g., smartwatches, smart speakers, smartphones) have constant vigilance for suspicious voice, especially when they are delivering their voice commands. Once the user is alerted by a suspicious audio, they intend to correct the falsely-recognized commands by interrupting the adversarial audios and giving more powerful voice commands to overshadow the malicious voice. This makes the existing attacks ineffective in the typical scenario when the user's interaction and the delivery of adversarial audio coincide. To truly enable the imperceptible and robust adversarial attack and handle the possible arrival of user interruption, we design SpecPatch, a practical voice attack that uses a sub-second audio patch signal to deliver an attack command and utilize periodical noises to break down the communication between the user and ASR systems. We analyze the CTC (Connectionist Temporal Classification) loss forwarding and backwarding process and exploit the weakness of CTC to achieve our attack goal. Compared with the existing attacks, we extend the attack impact length (i.e., the length of attack target command) by 287%. Furthermore, we show that our attack achieves 100% success rate in both over-the-line and over-the-air scenarios amid user intervention.

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11…$15.00
https://doi.org/10.1145/3548606.3560660

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Machine learning**;

KEYWORDS

Automated Speech Recognition; Adversarial Audio Attack; Human-In-The-Loop

ACM Reference Format:
Hanqing Guo, Yuanda Wang, Nikolay Ivanov, Li Xiao, and Qiben Yan. 2022. SpecPatch: Human-in-the-Loop Adversarial Audio Spectrogram Patch Attack on Speech Recognition. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3548606.3560660

1 INTRODUCTION

Speech is a major interface for human to communicate with an intelligent agent. Voice communication is a human-computer interaction approach that enables hands-free operation and offers opportunities for visually impaired users. Recently, with the thriving development of Artificial Intelligence (AI) and deep learning models, the performance of Automatic Speech Recognition (ASR) has improved significantly, resulting in a growing product market. For example, tech companies developed their online ASR systems and provided those services to the public, including Amazon Transcribe [7], Google Cloud Speech-to-Text [21], IBM Watson Speech to Text [25], and Microsoft Azure Speech Service [30]. Furthermore, they also integrated their ASR APIs to the Intelligent Voice Control (IVC) devices to offer voice assistant services (e.g., Siri [39], Google Assistant [19], or smart speaker systems such as Google Home [20] and Amazon Echo [6]). Besides that, more and more companies deliver their customer service using intelligent voice systems, which are empowered by ASR models to understand customers' questions and improve the efficiency of the customer support.

With the increasing number of deployed ASR systems, their security issues are getting more and more attention from researchers. Recent studies have demonstrated the vulnerabilities of modern ASR systems through multiple attack vectors. For example, attackers can launch an inaudible voice command injection attack through

an ultrasound speaker [34, 49], PZT transducer [46], public charging cable [41] or laser source [36] by exploiting the non-linearity effect of microphones. There are also signal processing attacks that analyze the differences between the perceptual sound of a human and an intelligent agent and then craft noisy-like commands via signal processing techniques [3, 4].

Audio Adversarial Attacks: Different from the aforementioned side-channel attacks and signal processing attacks, the adversarial attacks aim to fool the ASR models by introducing small perturbations. The adversarial attack was first found and demonstrated in image recognition tasks [18, 37]. Attackers exploit the vulnerabilities of machine learning (ML) models by searching for unnoticeable perturbations and then impose them on original images to mislead the ML model and yield a false classification. The vulnerabilities of ML models are generally introduced by the linearity of the activation functions and operations at each layer [37]. Since the ASR models are usually built by similar architectures and training processes, they share the vulnerabilities of other ML models. The first attempt at generating audio adversarial examples (AEs) demonstrates that ASR systems are vulnerable to AEs [12, 17], which are crafted while the attackers have the complete knowledge of the victim model. Later, several studies [5, 38] proposed the black-box attacks by utilizing genetic algorithms and gradient estimation techniques. However, all of the aforementioned attacks fail to attack over the air due to the fact that perturbation itself is fragile and easy to deform through the real-world acoustic channel. To circumvent this problem and enable the physical attack over the air, Li et al. [27] and Yakura et al. [45] incorporate over-the-air transformations to the process of AE generation (e.g., by adding a band-pass filter, applying the impulse response, etc.), thereby ensuring the robustness of the AEs. Furthermore, researchers strive to make the AE imperceptible by adding loudness constraints [33] or mixing it with songs [16, 48, 50]. Alternatively, a recent attack called AdvPulse [29] uses a short pulse to deliver malicious commands, which has been regarded as a more dangerous and stealthy attack technique.

Failure Cases of Existing Attacks: Despite the effort of existing over-the-air attacks [16, 27, 33, 45, 48, 50], all of them do not seriously take the human user's presence into account. Here, we showcase three scenarios that could deter a successful delivery of existing attacks, including, **Case A:** *User Interference*; **Case B:** *User Perception*; **Case C:** *User Interaction*, as shown in Figure 1. We use ①, ②, ③, and ④ to denote the sequence of events, red-colored words to denote the targeted attack commands and the responses from the ASR system. The blue-colored words denote benign commands from the user and responses from the ASR system. For every attack case, the adversary prepares the AEs in advance and then plays them via a loudspeaker.

• **Case A:** As shown in Figure 1(a), while the adversary and the user pronounce commands concurrently (e.g., the AE says, "call 911", and the user speaks, "set an alarm at 6 am" at stage ①), the ASR system tends to accept the user's command rather than the AEs; in this case, it will respond with "Alarm has been set". This is because: on one hand, the user's command has a higher sound pressure level when he/she is close to the ASR system, so the ASR system takes the stronger sound; on the other hand, the robustness of AEs is not guaranteed during the crafting procedure, i.e., once the audio

(a) Case A: User Interference

(b) Case B: User Perception

(c) Case C: User Interaction

Figure 1: Failure cases of existing attacks in real human-in-the-loop scenarios.

quality of AEs is degraded by the human-introduced interference, the attack will no longer work.

• **Case B:** Figure 1(b) demonstrates the scenario when the user notices the attack. While some previous attacks [16, 46, 48] stated that the adversary could play the AEs repeatedly (e.g., "Call 911 ... Call 911") at stage ① to ensure the successful delivery of the attack audio, the repeated AEs could raise alert. Although the adversary might craft imperceptible AEs by encoding the adversary command into songs or different speeches, the user is still able to locate the source of skeptical sound because of the long duration and repeated appearance of common audio adversarial attacks.

• **Case C:** In the scenario depicted in Figure 1(c), the adversary launches the attack by playing the "read message" adversarial audio at stage ①, followed by the successful response from the ASR system reading the message containing a personal verification code at stage ②. However, when the user is present, he/she is conscious of the abnormal behavior of the ASR device and tries to interact with the ASR system by sending a halting command (such as "stop reading") at stage ③ to regain the control. Consequently, the ASR system follows the user's benign command and terminates the reading process.

We summarize the existing adversarial attacks in Table 1 in terms of *Attack Model*, *Attack Type*, *Delivery Method*, and *Attack Media*. For the *Attack Model*, we use the acronym ASR to denote the Automated Speech Recognition model, and use SR for the Speaker Recognition model. The *Attack Type* indicates what type of attack samples are crafted when the attackers are preparing for the attack. In typical adversarial attacks, the attack type is either Adversarial Example (AE) or Perturbations (PT). If it is labeled as AE, that

Table 1: Comparison of SpecPatch with other attacks.

Attacks	Attack Model	Attack Type	Delivery Method	Over Line	Over Air
Houdini [17]	ASR	-	-	✓	✗
C&W [12]	ASR	-	-	✓	✗
Adversarial [5]	ASR	-	-	✓	✗
Practical [27]	SR	AE	Speech	✓	✓
Robust [45]	ASR	AE	Song	✓	✓
Fakebob [13]	SR	AE	Speech	✓	✓
Imper. [33]	ASR	AE	Speech	✓	✓
Comm. [48]	ASR	AE	Song	✓	✓
Metamorph [15]	ASR	AE	Speech	✓	✓
Devil's [16]	ASR	AE	Song	✓	✓
AdvPulse [29]	ASR	PT	Pulse	✓	✓
OCCAM [50]	ASR	AE	Song	✓	✓
SpecPatch	**ASR**	PT	Patch	✓	✓

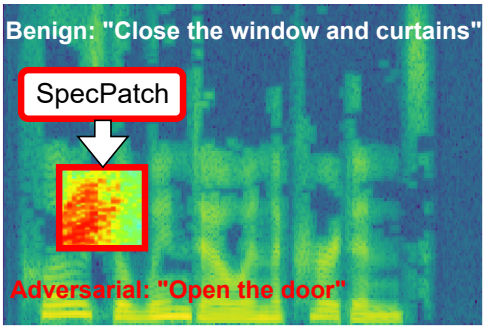


Figure 2: SpecPatch purturbs an audio input with adversarial spectrogram patch.

means the attackers will play the complete AE to launch their attack; otherwise, the attackers use the perturbation to alter the user's original commands. For the *Delivery Method (Deli. Method)*, we describe how the attacker launches their attack (i.e., by playing an adversarial speech, a song, or a pulse to deliver the adversarial commands). *Over Air* and *Over Line* narrate the ability of listed research to attack through different media.

All of the existing attacks, except AdvPulse [29], will fail to execute the attack in **Case A** and **Case B** due to their delivery methods of AEs. AdvPulse, on the other hand, utilizes short pulses to launch audio adversarial attacks that carry short commands. However, they cannot avoid the user interaction scenario (i.e., **Case C**) for two reasons: 1) the proposed universal pulse is only resilient to a single-word distortion because it is trained on a small dataset, and 2) the user's input voice commands out of the time range of a pulse will still be recognized by the ASR model. Therefore, *no existing attacks can launch imperceptible and stealthy physical attacks successfully when human is in the loop, i.e., while the user is presenting and intentionally disrupting the attack.*

New Attack Idea: To make the audio adversarial attacks more realistic in a human-in-the-loop scenario, we propose SpecPatch, the adversarial audio spectrogram patch attack. Inspired by the patch attack in Computer Vision (CV) [10], we aim to inject an adversarial patch into a benign spectrogram. There are three main benefits to leveraging adversarial patches for speech attacks: 1) adversarial patch has a relatively small size compared to the entire spectrogram, which makes it less noticeable; 2) adversarial patch can affect the global interpretation of a long voice command; 3) adversarial patch attack is text-independent, as the attackers can play the adversarial patch sound in any speech context. Figure 2 depicts the attack scenario. The spectrogram corresponds to the benign command: *"close the window and curtains"*. Then, the attacker injects an adversarial perturbation that is sensed by the IVC device. The adversarial perturbation is processed to be an adversarial patch in spectrogram scope, which deceives the ASR model to interpret it as the target command (*"open the door"*). Although the idea is promising, we still need to address the following *four* challenges.

- **Limited Impact Length:** It is challenging to encode long speech commands into a short duration patch. Existing attack [29] demonstrated that a 500ms perturbation could affect single word prediction; even with an increased perturbation length, it can at most impact 2-3 words.
- **Bypassing user's corrections:** Unlike the image classification task that takes a single image as input and predicts a single label, the speech recognition model usually takes many frames as input and predicts the corresponding phonemes. While the later input frames are unaffected by the adversarial patch, undesirably the user's correction commands will be fully understood by the model. It is challenging to disregard the user's followup commands using only a slight modification of benign speech.
- **Universal to any speech context:** Existing audio adversarial attacks [16, 27, 28, 33, 45, 48, 50] rely on the successful delivery of an integral AE constructed from a specific speech context, and hence are fragile to distortions (e.g., noise, user interference). To make SpecPatch robust on any speech context, an intuitive solution would be to train an adversarial patch on every speech content, but it is prohibitively expensive.
- **Perturbation Sync:** To successfully launch our attack, the adversary is expected to play the perturbation at the right timing to assure the adversarial patch posed in the correct location. However, in a real-world scenario, the timing of perturbation is hard to control, which would affect the attack success rate.

Contributions: In this paper, we make the following contributions.

- **New Attack:** We expose the deficiency of existing audio adversarial attacks in a human-in-the-loop scenario. To the best of our knowledge, SpecPatch is the first human-in-the-loop voice adversarial attack that is robust against *user interference*, *user perception* and *user interaction*.
- **New Techniques:** By exploring the internal mechanism of CTC (Connectionist Temporal Classification) loss, we find the root causes that limit the impact length of an adversarial patch on speech tasks. Then, we reconstruct an optimization function to craft an adversarial patch with a longer impact length. Moreover, we propose *Mute* adversarial samples by analyzing the principle of speech sequence input. With the *Mute* samples, we allow SpecPatch to cancel out the user's future interaction, thereby making SpecPatch more stealthy and dangerous.
- **Comprehensive Experiments:** We conduct physical attack experiments in three different places (i.e., indoor home, outdoor street, public dining hall) for speech recognition models. We

demonstrate the feasibility of launching our attack with a human-in-the-loop scenario, and prove its stealthiness via two user studies. Our results show that SpecPatch can achieve 100% attack success rate through both the over-the-air and over-the-line attacks with an adversarial patch.

2 BACKGROUND

2.1 Adversarial Patch

Compared to the traditional adversarial attacks, the adversarial patch attack is more dangerous because the crafted patches can be used to attack any scene in the CV domain [10]. The attackers launch the attack by printing the crafted adversarial patches as stickers and putting the stickers on any benign objects to fool the ML models (e.g., object detection, object localization). To obtain the patch $\hat{p}$, they use Expectation over Transformation (EOT) framework [9] to optimize the following objective function:

$$\hat{p} = \underset{p}{argmax}\mathbb{E}_{x\sim U,t\sim T,l\sim L}[\log \Pr(\hat{y}|A(p,x,l,t))]. \tag{1}$$

Given an image $x \in \mathbb{R}^{W\times H\times C}$ (W, H, C are width, height and channel), a patch p, a patch location l and patch transformation t, the function A works as an operator to apply the patch on the benign image. $\mathbb{E}$ represents EOT. Then, it keeps optimizing $\hat{p}$ to reach high log-probability on predicting the patched image as the target $\hat{y}$. The construction of this objective function ensures the universal and robustness of patch $\hat{p}$, because it considers the expectation ($\mathbb{E}$) over any background image ($x \sim U$), any transformations ($t \sim T$) of the patch (e.g., scaling, rotating, degrading), and any location ($l \sim L$) of the patch placement.

2.2 CTC in Speech Recognition

Unlike the image recognition task in which the model is only required to produce one label, the speech recognition model is more complicated as it needs to merge the sequential letter predictions and produce a sentence. To train a speech recognition model with spectrograms and their transcriptions, one challenge is to align the transcription letters to the input frames. CTC [22] is proposed to resolve this problem. The idea of CTC computation can be summarized as follows: given a sequential model, it takes T frames of spectrograms as input and produces T probability arrays. For example, the probability array at frame t can be represented as $\mathbf{Pr}_t = [\Pr_{t,a}, \Pr_{t,b}, ..., \Pr_{t,\epsilon}]$, where $\Pr_{t,a}$ indicates the probability of predicting the frame t as character "a", and so on and so forth. Let $\mathbb{C}$ be the available character set, which records the appearance probability of 28 characters (a-z, space, and ϵ). For the $|\mathbb{C}| \times T$ probability matrix, CTC counts all paths (i.e., symbol sequences) that can be merged to match the target phrase with two rules: 1) remove all contiguous duplicated characters; 2) remove all ϵ tokens. For example, a path "hheeleɛlo" will be decoded as "hello". After it gets all paths representing the target phrase, the probability of predicting the spectrogram as the target phrase can be computed by summarizing the probability of those paths. This process can be formulated as follows:

$$\Pr(Y|X) = \sum_{\pi\in\pi_{X,Y}}\prod_{t=1}^{T}(\Pr_{t,a_t}|X), \tag{2}$$

where Y is the target phrase, and X is the input spectrogram with T frames. π is the path that includes T characters: $\pi = a_1a_2...a_T$, and $\pi_{X,Y}$ refers to all the paths that can be reduced to Y. If Y is "hi", and $T = 3$, then $\pi_{X,Y}$ includes "ϵhi, $h\epsilon i$, $hi\epsilon$, hhi, hii". For every path belonging to $\pi_{X,Y}$, it computes the product probability of consecutive characters that form π. Formally, consider a_t is the t_{th} character in path π, $\Pr_{t,a_t}$ represents the probability of the appearance of character $a_t \in \mathbb{C}$ at time t. The product represents the path appearance probability, and the sum operation deduces the target phrase probability. To compute the loss, we use:

$$\mathcal{L}_{CTC}(X,Y) = -\log \Pr(Y|X), \tag{3}$$

i.e., given an input spectrogram X and its target phrase Y, the loss can be retrieved by the negative log likelihood of $\Pr(Y|X)$.

2.3 Problem Formulation

The ASR system takes waveform $v \in [-1,1]^N$ as raw input and produces its corresponding label $Y \in \mathbb{A}^m$, where $\mathbb{A}$ is a set that contains all letters from a to z, and space, and m is the length of transcription. When unpacking the ASR system, we use $M(\cdot)$ to denote the speech recognition model that empowers the ASR system. Instead of using waveform as input, the $M(\cdot)$ takes the processed data (e.g., spectrogram) as input because it is more representative and has fewer data samples. The size of the spectrogram depends on the duration of v, the STFT window length, STFT hop length, and the number of FFT points. We use $X \in \mathbb{R}^{T\times F}$ to denote the user's speech spectrogram, which includes F frequency bins and T frames, represented as follows:

$$X(m,\omega) = |\sum_{n=0}^{N} v[n]w[n-m]e^{-j\omega n}|, \tag{4}$$

where m is the frame index and ω is the frequency bin index, w represents the window function, and n denotes the sample index of the waveform. After taking the spectrogram frame by frame, the speech recognition model $M(\cdot)$ fabricates a probability matrix **Pr** as logits output, which is shaped as $|\mathbb{C}| \times T$. Then, based on the probability matrix, it computes the probability of every possible phrase with Eq. (2), selects the phrase which has the highest CTC probability, and finally gives the transcription as $Y = M(X)$.

The attacker's goal is to construct an audio perturbation δ. When it associates with a waveform v, the ASR system will produce a target transcription Y. Unlike the prior audio perturbation, SpecPatch is designed to target the most realistic scenarios (e.g., human-in-the-loop) by leveraging an adversarial patch. As such, the following issues need to be reconsidered.

Audio Adversarial Patch: While the prior audio perturbation usually has the same duration as the benign waveform, the adversarial patch has a limited duration and frequency range. We denote our adversarial audio patch as $p \in \mathbb{R}^{T'\times F'}$, where $T' \ll T$ and $F' \ll F$ denote the small size of the adversarial patch compared to the user's speech spectrogram X.

Transcriptions: Instead of using a single-word label to tag the input, the speech recognition model generates a sentence as output. More specifically, the predicted sentence is the phrase that reaches the highest CTC probabilities, namely, $\underset{Y}{\text{argmax}}\Pr(Y|X)$. In this

case, the transcription Y can be decoded from any paths $\pi \in \pi_{X,Y}$, and the length of Y is less than the number of frames (T).

Universality: Most adversarial attacks assume that the attackers know the users' speech and can deliver the perturbation synchronously at a specific time point. However, the assumption does not always hold in a real-world scenario. SpecPatch expects that the attacker can "place" the audio patch at any time, and over any speech context. Let function $A(X, p, t, f)$ be the "place" operation that puts an adversarial patch p to t_{th} spectrogram frame and f_{th} frequency index with any input spectrogram X. Then, our goal is to attain $\hat{Y} = M(A(X, p, t, f))$ for all X in human speech and p on any place of X.

2.4 Threat Model

SpecPatch entails the novel adversarial patch attack in the audio domain. We circumvent all the three common failure cases mentioned in Figure 1 by introducing the universal adversarial patch and mute signal. The generated adversarial patch is imperceptible and inconspicuous due to the frequency and the time constraint of the spectrogram patch, making SpecPatch a more dangerous and stealthy attack than existing ones.

Adversary's Capability: Unlike the prior work [11–13, 35] that requires the adversaries to know the victims' benign commands in advance to calculate the corresponding audio perturbation, we assume the adversaries have no access to the victim's benign audio and have no knowledge about what the victim will say during their attacks. We assume the adversaries can place a hidden loudspeaker close to the target devices to launch the attack. For the SpecPatch crafting process, we assume the attackers have prior knowledge of the target ASR model. For example, the architecture and model parameters can be found from a public resource. This setting is widely used in most prior work [11, 12, 15, 27, 33], and can be generalized to a black-box scenario [14].

Attack Scenarios: Unlike all the previous studies, we focus on attacking the ASR system when the user is present. More specifically, the adversary crafts adversarial patches offline, and then uses a preset loudspeaker to deliver the adversarial patch, therefore misleading the target ASR system to make wrong prediction/transcription. For example, the adversary can send fake commands to the voice assistants and request them to perform the wrong operation. Moreover, the adversary can fool the telephone voice system by injecting falsified personal information to trick the ASR-based customer service; besides, the adversary can deny the service provided by the target model via simply broadcasting the spectrogram patches. Due to the shortness and imperceptibility of SpecPatch, the attack can be launched in public spaces (e.g., malls, streets, cafes) with nearby loudspeakers (e.g., smartphone, in-ceiling speaker).

3 DESIGN OVERVIEW

Figure 3 illustrates the system flow of SpecPatch. First, we will craft an adversarial patch to generate the malicious command, i.e., using a short patch to affect a longer benign spectrogram. Second, when the user makes a correction, we need to mute the users' correction by denying the users' followup commands. We achieve that with a specially designed signal called **"Mute"** signal. Next, we make SpecPatch universal to any speech context. This step usually requires the adversarial perturbation to traverse all images/audios

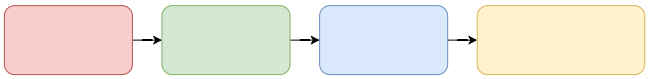


Figure 3: SpecPatch workflow.

in a large dataset to validate the effect of the perturbation on all possible contexts. However, the infinite number of speech contexts makes it computationally infeasible to evaluate a universal perturbation. Rather than optimizing the adversarial patch across different speech contents, we design a *phoneme-level context free optimization* method. We guarantee that SpecPatch can work across any user interference. The final step of our design is to enhance the robustness of SpecPatch in a real-world scenario. To achieve that, we take the transmission loss of a physical attack into account during the optimization of adversarial patches. More details can be found in §7.

4 SPECPATCH DESIGN

This section first analyzes why short perturbations cannot impact long input, and then we describe our strategy to reach our attack goal, i.e., using short patches to attack long commands. After that, we describe the design of the **Mute** signal to deny user's interference. Then, we introduce our *phoneme-level universal* patch crafting process. Finally, we present the techniques to robustify SpecPatch in an over-the-air scenario.

Formulation: Our goal is to craft an adversarial spectrogram patch $\hat{p} \in \mathbb{R}^{T' \times F'}$ that alters all benign spectrogram X and translates them into the target phrase $\hat{Y}$. To achieve this goal, the following expectation needs to be optimized:

$$\hat{p} = \underset{p}{argmin} \mathbb{E}_{X \sim U, t \sim T, f \sim F} \mathcal{L}_{CTC}(A(X, p, t, f), \hat{Y}). \quad (5)$$

Here, we compute the CTC loss of patch p when it is applied anywhere (t~T, f~F) of the benign spectrogram X, based on which we derive the best adversarial patch $\hat{p}$ that reaches minimal expectations of losses.

4.1 Long Command Conversion

4.1.1 Adversarial Patch with CTC Loss. For most adversarial patch attacks in the image domain, the patch will help ensure very high confidence in the target class. Furthermore, recent studies [42, 43] prove that the effectiveness of adversarial patches on deep neural networks (DNNs) is caused by the large receptive fields of CNN layers. As the image classification model maps one image to one label, it connects multiple convolutional layers sequentially. The later convolution layers will have a higher receptive field and will likely include the adversarial patch. Therefore, even a small adversarial patch can be sensed by a later CNN layer and hence affects the global prediction of the image. However, most speech recognition models [8, 24, 44, 51] use a recurrent structure, which usually takes multiple frames as input, produces multiple phoneme predictions for every frame, and then connects the phoneme predictions to form the final sentence prediction. One critical challenge is in applying an adversarial patch to the sequence model. As the adversarial patch could only affect a couple of input frames, the remaining output is barely altered. Therefore, it could be hard to achieve the alteration into a long target sentence.

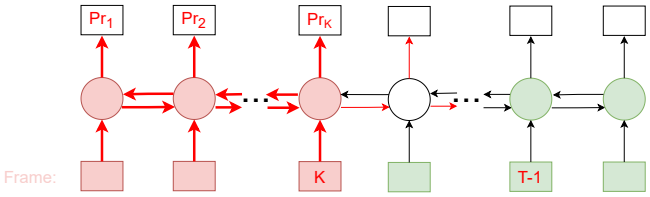


Figure 4: SpecPatch flowchart.

Suppose the adversarial patch $p' \in \mathbb{R}^{T' \times F'}$ overlaps with K input frames (where K is determined by the window size of the speech recognition model). For ease of explanation, we assume the patch is placed at the left corner of the benign spectrogram, which means $t = 0$ and $f = 0$. While the benign speech has T frames and $T > T'$, there will be a limited number of output probabilities affected by the adversarial patch.

Figure 4 demonstrates the workflow of SpecPatch that uses an adversarial patch to attack a sequence model. The bottom blocks show the input frames, while the middle nodes are computational cells of $M(\cdot)$, usually implemented by the LSTM or RNN cells. The top row represents every node's logits output (also known as the probability array of 26 letters). We use red color to mark the frames, nodes, and logits output directly affected by patch p and let the green color label the benign frames and nodes. To demonstrate the data forwarding process of the sequence-to-sequence model, we use red arrows to denote how frames affect the hidden state of nodes and further alter the probabilities up to the K^{th} frame. It can be seen that $\Pr_1, \Pr_2, ..., \Pr_K$ are determined by the frames 1 to K, the intermediate output of previous/next nodes, and the hidden state of the current node. When crafting the adversarial patch, the model $M(\cdot)$ parameters are fixed, so we can only control the value of p' to meet the target transcription. Let X' denote the spectrogram after applying an adversarial patch p. Our goal is to optimize the following objective function:

$$\begin{aligned} \hat{p} &= \underset{p}{argmin} \mathcal{L}_{CTC}(X', \hat{Y}), \\ X' &= X + p. \end{aligned} \tag{6}$$

Insight 1: *The restricted length of adversarial patch affects the convergence of the objective function.*

Observation 1: When optimizing the objective function above, it requires the tuning of p and Pr to match the target phrase. However, limited by the short length of adversarial patch p, the later input frames are untouched during the optimization process, and therefore the values of $\Pr_{k+1}$ to $\Pr_T$ remain the same. This will make it hard for $\mathcal{L}_{CTC}$ to converge. To explain it in more details, we break down the probability equation into two parts:

$$\Pr(\hat{Y}|X') = \sum_{\pi \in \pi_{X',\hat{Y}}} [\prod_{t=1}^{K} (\Pr_{t,a_t}|X') * \prod_{t=K+1}^{T} (\Pr_{t,a_t}|X)]. \tag{7}$$

To minimize $L_{CTC}(X', \hat{Y})$, we aim to maximize $\Pr(\hat{Y}, X')$ as shown in Eq. (3). The probability can be separated into two parts in Eq. (7). The first term $\prod_{t=1}^{K}(\Pr_{t,a_t}|X')$ denotes the probability that is directly affected by the adversarial patch, which will be fine-tuned continuously by adapting the adversarial patch value. However, the second term $\prod_{t=K+1}^{T}(\Pr_{t,a_t}|X)$ takes the benign X as input, and hence the later probability will remain in low value as it does not match the target letter a_t and has a low chance to be affected by the adversarial patch. This is due to the limited length K of the patch. Therefore, the second term is barely affected as $t > K+1$. Therefore, when we compute the gradient of $L_{CTC}(X, Y)$, we take the second term into account, but after we update the adversarial patch according to the gradient, we will still get a similar result of the second product term. In short, no matter how to update X', we will have the second term of the gradients remaining the same, which will mislead the direction of optimization of X. In other words, we will not be able to achieve our attack goal if you use the global gradient to update local changes.

Insight 2: *The mismatch length of the target phrase and benign phrase affects CTC loss.*

Observation 2: Besides the shape and value of the adversarial patch, the other critical factor that affects the CTC optimization process is the target phrase. Let us revisit Eq. (7): the probability $\Pr(\hat{Y}|X')$ is determined by all the paths $\pi \in \pi_{X,Y}$ that can be merged to the target phrase. While replacing the benign target Y with the target phrase $\hat{Y}$, the number of paths will change accordingly, which will influence the computational cost for CTC loss. For example, if the target phrase $\hat{Y}$ has a length of $l_{\hat{Y}}$, and we assume the length of X is T, we will have total number of paths as follows:

$$\binom{T + l_{\hat{Y}}}{T - l_{\hat{Y}}} = \frac{(T + l_{\hat{Y}})!}{(T - l_{\hat{Y}})!(2l_{\hat{Y}})!}. \tag{8}$$

If we have long input and short target phrases, the number of paths for the target phrases will exponentially grow. For example, when T is 15 and $l_{\hat{Y}}$ is 5, the total number of paths would be $\binom{20}{10} = 184,756$. Even though the loss computation can be efficiently computed with dynamic programming [22], it will still result in redundant gradients due to the constrained adversarial length.

4.1.2 Extend the Adversarial Patch Impact. With the previous observations, we find that it is challenging to craft an adversarial patch to alter the recognition of a complete spectrogram. To address the challenges, we propose a novel method called *partial matching*. The basic idea of *partial matching* is allowing the target label to include a portion of the benign label, such that the optimization can focus on the tunable variables. Formally, instead of assigning $\hat{Y}$ as the target when crafting an adversarial patch, we use Y_t to concatenate the target phrase and the benign phrase as: $Y_t = \hat{Y}||Y_{tail}$, where Y_{tail} is the trailing benign phrase.

Figure 5 demonstrates the strategy of *partial matching*. Given the benign spectrogram and the adversarial patch as input, the attacker aims to mislead the transcription from "Close the window and curtains" to "open the door." At the bottom of Figure 5, we have a benign spectrogram that spans from left to right. Inside the benign spectrogram, there is an adversarial patch (in red color). When we feed the spectrogram to a model, it is divided into frames by a fixed window and a preset hop size. In the middle layer, we use three different colors to denote the state of the nodes. Red represents the nodes that have adversarial input; green depicts the nodes that have benign input but are immediately affected by the previous node's output; Blue means the nodes have a very low possibility of being impacted by the adversarial patch. Every node produces a probability array that records the probabilities of every letter and eventually generates the transcription based on the decoding

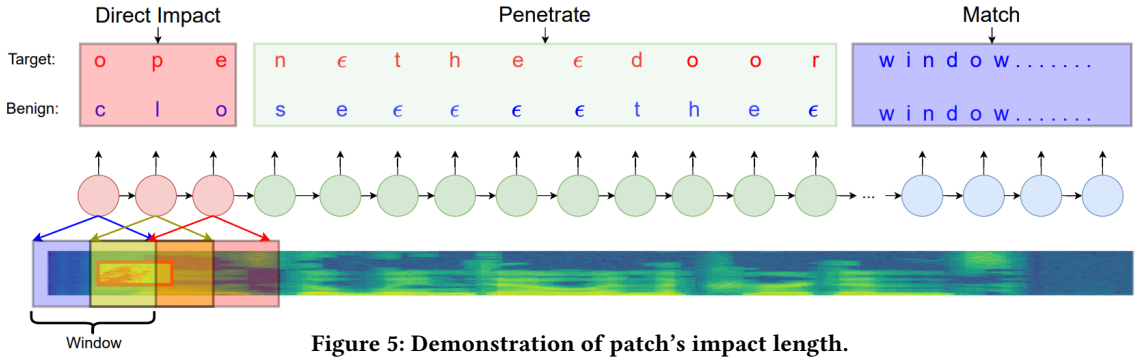


Figure 5: Demonstration of patch's impact length.

method (e.g., greedy decoding [23], beam searching decoding [22]). From the top layer, it shows the benign output is "Close the window ...". The target phrase is a concatenation of $\hat{Y}$ ("open the door") and the partial benign label ("window ..."). We use the red region to denote that part of the target can be achieved directly by tuning adversarial patch. The green region of the target phrase can be achieved by extending the impact of the adversarial patch via the internal links between nodes. The blue region is the benign output that ensures the optimization can converge despite of the limited length of the adversarial patch.

Validation of *Partial Matching*: Next, we experimentally validate the effectiveness of *partial matching*. The goal is to convert the benign transcription "Close the window and curtains" to the malicious command "open the door" by applying an adversarial patch in the beginning of the audio. We follow the optimization function in Eq. (6) to craft an adversarial patch in two different scenarios, i.e., without the partial matching and with the partial matching. In the first scenario, we set $\hat{Y}$ as "open the door". In the second scenario, we use "open the door window and curtains" as our target phrase, which contains a trailing (partial) benign command. Figure 6 shows the optimization result up to 500 epochs. At the very beginning, both cases start with the benign label at epoch 0. As the optimizing step proceeds, the first approach (i.e., without partial matching) only alters a single word (red color "open") to match the target. The result remains the same after 400 epochs, which indicates that the optimization converges but does not achieve the attacker's goal (i.e.,, delivering the target command "open the door"). In comparison, the *partial matching* approach converges faster and meets the target phrase within 300 epochs. This experiment shows that, without modifying any parameters or optimization scheme, the *partial matching* improves the convergence speech in crafting an adversarial perturbation. Next, we visualize the adversarial patch in Figure 7. For ease of explanation, we assume the adversarial patch starts at the beginning of the benign input and spans all the frequency ranges, i.e., the adversarial patch (the red portion) lasts 500 ms and has 8 kHz bandwidth. The benign label is shown in the top blue field and the concatenated target phrase is in the middle red field. We find that the length of the target command ("open the door") exceeds the range of the adversarial patch, which indicates that the *partial matching* helps achieve the attack goal in extending the impact of adversarial patch and outputting the target command.

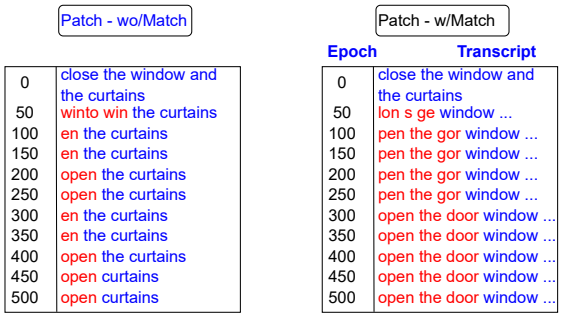


Figure 6: Comparison between SpecPatch with and without partial matching.

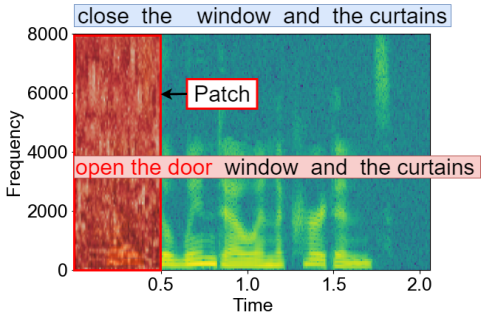
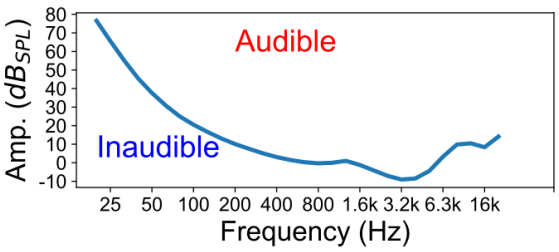


Figure 7: The effect of patch towards a long command.

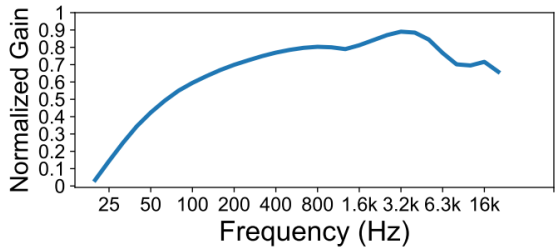
4.2 Patches to Deny User Input

The proposed *partial matching* mechanism successfully extends the adversarial patch impact length beyond its own duration. However, we still face two challenges to fulfill our attack goal. First, we have no knowledge of the benign phrases in advance, so it is impractical to adjust the optimization of the target phrase for every possible benign phrase. Second, the human factor (e.g., user interaction or long user commands) cannot be resolved because the adversarial patch cannot affect the speech transcription that is far away from the patch position. To overcome the challenges, we propose *Mute Patches* by exploiting the *discrepancy* of the ASR model's input and output mapping.

Design Mute Patches: The design goal of mute patches is to disrupt the user's commands without attracting their attention. Specifically, we aim to inject a few adversarial samples with low volume to mislead the ASR model to produce empty transcriptions.



(a) Hearing Curve of Human



(b) Reflection of Hearing Curve

Figure 8: Optimization of patch frequency based on the auditory property of human.

To design such mute patches, we review the complete speech recognition process and find the opportunity to meet our design goal. As described in §2.3, the waveform serves as the raw input, which is converted into spectrogram to be fed into the ASR models. Next, every node of the ASR model takes a couple of spectrogram frames and outputs a letter prediction. By reviewing the whole process, we realize that every node of the ASR model perceives a large scope of waveform samples. A similar phenomenon has been observed by prior studies [42, 43] in image recognition models, and the authors conclude that a large receptive field of the neural node is responsible for the adversarial patch attack because a small patch in an image can be perceived and misinterpreted by a neural node. Inspired by their findings, we are motivated to inject sampled adversarial audio signals into the neural nodes. To craft the mute signals, we formulate the following problem:

$$\begin{aligned} p^m &= \underset{p}{argmin}\mathcal{L}_{CTC}(X', Y_b), \\ X' &= X + Tp, \end{aligned} \tag{9}$$

where Y_b is a phrase that only contains blank symbols, and p^m is the mute patch that is composed of multiple patches such as $p^m = [p^1, p^2, ..., p^L]$. For every patch that has $1 \leq l \leq L$, we have $p^l \in \mathbb{R}^{1\times F'}$. The size of the mute patch is $1 \times F'$ because a single adversarial sample can only affect one bin of the spectrogram. We set the length of the mute patch as T, and X' is the resulting spectrogram. By optimizing Eq. (9), we can craft the mute signal in the time domain with minimal loss value. The choice of T is determined by the hop length of STFT and the input size of the ASR model. In practice, we can set the value of T to be the same as the W_{STFT}, such that we can ensure every vertical spectrogram bin contains adversarial information.

4.3 Imperceptible Patch

Almost all prior adversarial audio attacks (e.g., [12, 29, 33, 48]) aim to minimize the amplitude of the perturbation (i.e., minimize $dB(\delta)$), e.g., by including the perturbation amplitude in the loss function. However, we find that although these perturbations are well-optimized, they are still audible when performing the physical attacks.

In our attack scenario, we expect to launch an imperceptible attack when the victim user is close to the adversary. Since this goal is hard to achieve by the optimization method (i.e., penalizing the amplitude of perturbation), we design a new approach to satisfy the imperceptible attack goal. In a nutshell, the imperceptibility of SpecPatch is ensured by the short duration of the adversarial spectrogram patch and further secured by the narrow frequency band of SpecPatch.

In the prior optimization settings, the crafted perturbations are audible because the victim microphone is sensitive to a certain input amplitude. Here, we focus on yielding the perturbation inaudible without dropping its amplitude. To achieve this goal, we investigate the human hearing sensitivity curve and find that the human ear has uneven sensitivity to different frequencies. We depict the hearing curve in Figure 8(a). Formally, the hearing curve can be represented by a function with f, and we denote it as $H(f)$. The source data is measured by prior auditory research on equal loudness contours [26]. In the figure, the blue line indicates the required amplitude for pure continuous tones at a specific frequency that can be heard by humans. Above the curve, we can feel the sound at such loudness, while below the curve, the sound intensity is insufficient. For example, one can hear continuous audio with frequency at 100 Hz as long as it has more than $20dB_{SPL}$. Once the volume is decreased to less than $20dB_{SPL}$, the human can no longer perceive it. From the shape of the curve, we find that the human auditory system is more sensitive to a frequency between $1.6kHz$ and $4kHz$. In comparison, we are unperceptive to sound below $1.6kHz$, as the lower frequency stimulates less attention from human ears. Therefore, we can design low-frequency patches (e.g., $< 1.6KHz$) to diminish the perceptual level of human hearing. To reach this goal, we add a *frequency selective penalty* term to the objective function in Eq. (9). The updated function is presented below:

$$\begin{aligned} \hat{p} &= arg\min_{f_l<p<f_h} \mathcal{L}_{CTC}(X', Y_t) + ||p * \widetilde{H(f)}||_2, \\ \widetilde{H(f)} &= Normalize(-H(f)). \end{aligned} \tag{10}$$

This new term $||p * \widetilde{H(f)}||_2$ is composed of the patch p and a frequency response function $\widetilde{H(f)}$, and we multiply them together to compute the L_2 norm result. f_l and f_h indicate the low and high-frequency boundaries of the learned patch. Compared with existing attacks [12, 29] that assume a constant value in the penalty term, we design a frequency response function as a adjustable coefficient. The goal of this term is to selectively penalize the human sensitive frequency portion (e.g., $1.6kHz$ and $4kHz$) and retain the insensitive components (e.g., $< 1.6kHz$) in the adversarial patch. We design the frequency response function $\widetilde{H(f)}$ based on the human hearing curve shown in Figure 8(a). By performing a reflection operation and normalize the result in the range of 0 to 1, we can obtain the

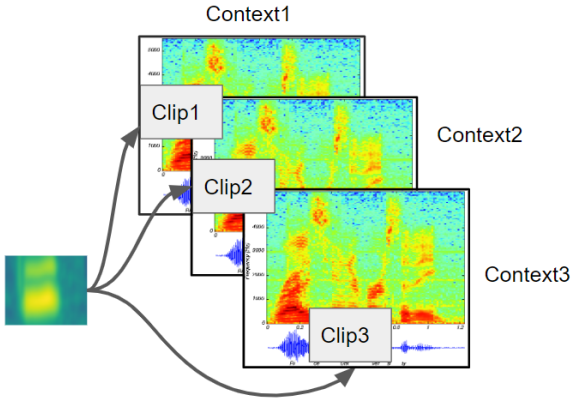


Figure 9: Universal perturbation.

$\widetilde{H(f)}$ as shown in Figure 8(b). The rationale of such an operation is to reduce the human-sensitive energy while retaining the inaudible portion of the adversarial patch.

4.4 Universal Patch

The prior adversarial attacks either use AE or Perturbation to launch the attack. As shown in Table 1, most prior work leverages AE to deliver the adversarial commands. However, there is a major concern with this method, as the user's intervention could disrupt the AE. Because the AE needs to be crafted with known benign commands, it requires the adversary to predict the incoming benign commands in advance. In a real attack scenario, it is hard to predict incoming commands. Therefore, such unpredictable user intervention could disrupt the performance of AE. The only exception among the previous attacks is AdvPulse [29], which utilizes a *universal perturbation* to launch the physical attack. They use the iterative greedy algorithm [31] to generate a perturbation that works with any given contexts. However, AdvPulse is proved effective on the single-word commands. Compared with single-word commands, diverse speech contents make it more difficult to compute the universal patch. Generally, a universal perturbation is crafted by iterating over all the benign contexts [29, 31]. Formally, the perturbation can be represented by the following formula:

$$\begin{aligned} &\hat{p} = \underset{p}{argmin}\mathbb{E}_{X\sim U}\mathcal{L}((X+p),\hat{Y}), \\ &U = \{X \mid X \text{ is speech with arbitrary contents}\}, \end{aligned} \tag{11}$$

where $\hat{p}$ is the universal perturbation, X is the benign context (e.g., background sound), and U is the set that includes all possible benign samples. Due to the diverse speech contents, U will become a large set, resulting in a substantial computational cost. To reduce the cost, we propose *phoneme-level universal* optimization scheme, and the logic of this approach is depicted in Figure 9. In short, the proposed *phoneme-level universal* scheme reduces the size of U by introducing clips. One clip contains one single phoneme, and we added padding to ensure every clip has the same duration as the patch as shown in Figure 9. Since there is a limited number of phonemes (i.e., 44 phonemes in English) in the speech context, the clip set is much smaller than the speech set. The clips can be simply retrieved from the speech dataset. We formalize the proposed *phoneme-level universal optimization* as follows:

$$\begin{aligned} &\hat{p} = arg\min_{fl<p<fh} \mathbb{E}_{X\sim U}\mathcal{L}_{CTC}(X',\hat{Y}) + ||p * \widetilde{H(f)}||_2, \\ &U = \{X \mid X \text{ is clip with fixed duration}\}. \end{aligned} \tag{12}$$

4.5 Overall Physical Over-the-Air Attack

As mentioned above, we guarantee that SpecPatch is able to deliver long commands over users' intervention in any speech context. Specifically, we first apply a universal mute signal to override the entire original speech, which will result in a blank-transcription. Partial matching then takes a part of blank transcription as input to generate an imperceptible adversarial patch for the target phrase. Note that, similar to the generation of adversarial patch, the generation of mute signals also does not require knowledge of original phrases. For example, suppose the benign command is "open the door", the target is "close the door", we first apply the mute patches to the benign audio to convert "open the door" into a blank transcription ("—-...–"). Then, we generate the adversarial patch based on the muted benign input. If the benign command is longer than our target, after applying the mute patches, we will set "close the door———" (with trailing blank symbols) as target to generate patch.

However, to launch the attack in a real-world scenario, we need to resolve the patch distortion during the over-the-air transmission. We follow the design in [29, 45] due to its simplicity. In general, three operations are considered during the crafting process: 1) band-pass filtering, 2) room impulse response, and 3) ambient noise mitigation. The bandpass filter is designed to cope with the uneven frequency response of the speaker and microphone. The room impulse response (RIR) is introduced to compensate for the absorption and reverberation in the environment. Finally, the ambient noise is considered to craft a robust perturbation that resists environmental noise. In practice, we form the following final objective function to include the operations mentioned above:

$$\begin{aligned} &\hat{p} = arg\min_{fl<p<fh} \mathbb{E}_{X\sim U}\mathcal{L}_{CTC}(X',\hat{Y}) + ||p * \widetilde{H(f)}||_2, \\ &X' = X + BPF(p) * R(f) + W. \end{aligned} \tag{13}$$

The *BPF* refers to the band pass filter, and we follow the setting in [45] to configure the cut-off frequency as $50 \sim 4,000Hz$. The $R(f)$ represents the spectrum of the room impulse response. We use the RWCP dataset [32] to enrich the RIR measurements and further compute the spectrogram of the RIR audios. The noise spectrogram W is chosen from another ambient noise dataset NOISEX-92 [40], which contains various noises (babble noise, factory noise, HF radio channel noise, pink noise, white noise, vehicle noise).

5 EVALUATION

5.1 Experiment settings

We implement SpecPatch using the Tensorflow [2] framework. We craft adversarial patches following Eq. (13). The experiments are conducted on a desktop with Intel i7-7700k CPUs, 64GB RAM, and NVIDIA 1080Ti GPU, running 64-bit Ubuntu 18.04 LTS operating system. In the evaluation, we launch our attack in two scenarios: *over-the-line* and *over-the-air*. In the over-the-line attack, we pass the SpecPatch directly to the model as a Waveform Audio file.

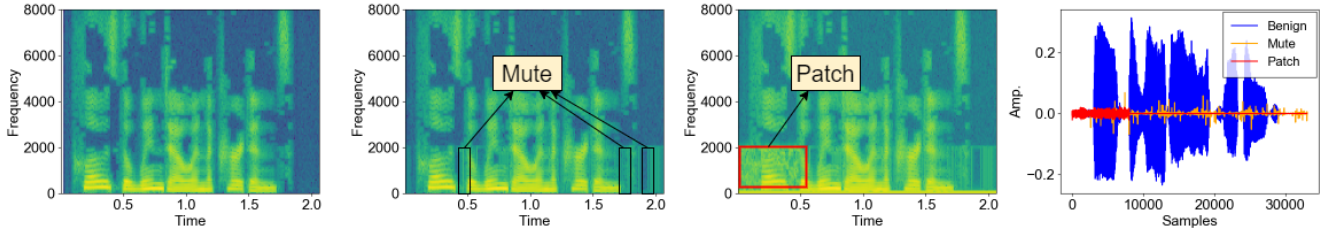


Figure 10: Spectrogram and time-domain signals of SpecPatch.

In the over-the-air setting, we attack the victim's phone using a speech-to-text service. In our implementation, we set up a server that runs our target ASR model and allows the victim's phone to request service through the local area network.

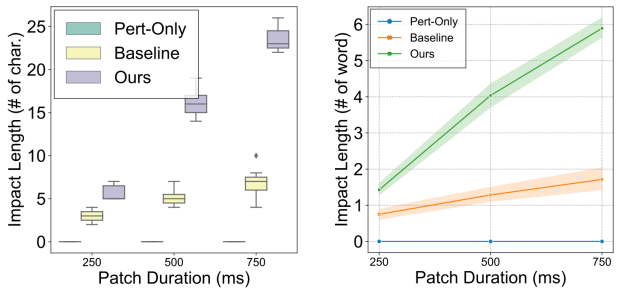
Target Model Selection: As our attack target is the speech recognition model, we will examine the effectiveness of SpecPatch on the most popular ASR models. Specifically, we select DeepSpeech2 [8] as the target ASR model. DeepSpeech2 leverages CTC loss and recurrent cells to improve the recognition performance.

Metrics: We use the following metrics to quantify the effectiveness of our attack: *(1) Success Rate:* this metric is the ratio of successful attacks and the total attempts. We report success only when the prediction matches the targeted command in a targeted attack. In terms of the untargeted attack, we measure the success rate of mis-transcribing the victim user's benign commands. *(2) Impact Length:* This metric is to identify how many characters/words can be affected by our SpecPatch. Given a long benign spectrogram, we inject an adversarial patch and measure the length of characters/words that are different from the original transcription. *(3) L2 Distortion:* the L2 distortion $||p||^2$ indicates the amplitude of adversarial patches. Prior to the launch of a physical attack, we can measure the distortion value by summarizing the squared amplitude of the generated perturbations.

Datasets: The dataset we choose as benign audio is TIMIT [1]. This dataset contains four types of corpora designed jointly by the Massachusetts Institute of Technology (MIT), SRI International (SRI), and Texas Instruments, Inc. (TI). It contains 6,300 audios from 630 speakers. The duration of each audio is around 5 seconds and contains approximately ten words. For our target commands, we collect them from the website ok-google.io, which provides commonly used commands on Google Assistant. We select ten sentences as our attack goal, e.g., "find my phone" and "turn on the lights." For the phoneme clip dataset, we construct it manually by following the annotations in TIMIT [1]). In total, we obtain 50,487 phoneme clips that cover 44 phonemes. We added padding to ensure every clip to have fixed length as 500 ms, which matches with the length of the adversarial patch.

5.2 Over-The-Line Attack

Over-the-line SpecPatch: We first showcase the capability of SpecPatch in converting the benign audio into our target transcription by injecting adversarial patches. As shown in Figure 10, the first spectrogram represents the benign audio "close the window and curtains". We first apply mute patches to translate it into consecutive blank symbols which results in an empty transcription. The mute patches are crafted by Eq. (9). We use rectangle boxes to mark those mute patches. It can be observed that each mute patch only occupies a single frame and periodically appears as vertical lines below $2kHz$. After applying the mute patches, we can craft an adversarial patch to meet our target goal by leveraging the *partial matching* strategy (i.e., concatenating the target command with trailing blank symbols). The third figure depicts the adversarial patch on the muted spectrogram. We use a red rectangle box to highlight the position of the adversarial patch. We can see that the adversarial patch occupies 0.5 second within $50Hz \sim 2kHz$ and is placed at the beginning of the spectrogram. The frequency band of this patch is learned from Eq. (10) and further constrained by Eq. (13). To investigate the amplitude of the benign audio, mute signal, and adversarial patch, we visualize the waveform of the complete AE in the rightmost figure. Compared to the benign audio, the adversarial and mute patches have a very low volume (~%5 of the benign audio).



(a) Impact length in characters (b) Impact length in words

Figure 11: Comparison of impact length.

Evaluate the impact length: To evaluate the effectiveness of *partial matching* strategy, we conduct experiments with three different strategies to attack a long command (~ 10 words) with different patch duration. The first strategy, perturbation-only (Pert-Only) strategy searches for a short patch that delivers long commands without considering the benign audio in crafting the AE; the second strategy utilizes a long empty audio as the background sound to increase the logits output-length, and then craft a patch based on the benign audio. This approach is adopted by the prior work [29]. The third strategy leverages the *partial matching* strategy. Similar to the second strategy, this approach uses an empty audio as background. However, it configures the target command to include trailing blank symbols, as illustrated in Section 4.5.

We randomly select 20 sentences in TIMIT dataset as our target and use three different lengths of the patch (250ms, 500ms, 750ms) to achieve the attack goal. Figure 11 presents the comparison of impact lengths among the three strategies. The "Perturbation-Only"

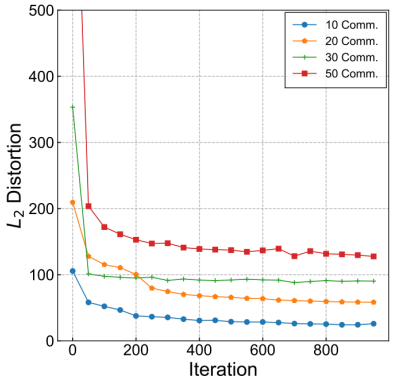


Figure 12: Universal mute patches.

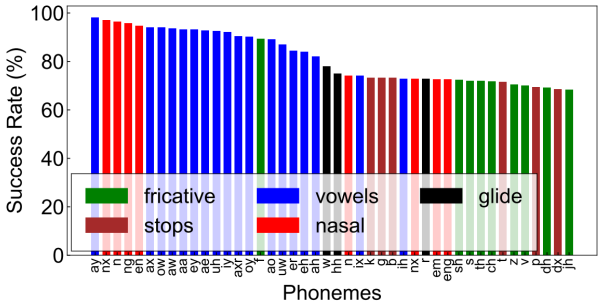


Figure 13: Universal attacks across different phonemes.

label represents the first strategy as it does not consider any benign audio. The "Baseline" label symbolizes the second strategy, and the "Ours" represents the *partial matching* strategy. From Figure 11(a), we find the first strategy failed to craft a short patch (impact length is 0) that delivers a long command for all the three lengths of patch configurations. This can be attributed to the lack of logits output for a short variable (perturbation), leading to the error of *"INVALID ARGUMENT"* caused by *"not enough time for target transition sequence"*. In contrast, the baseline strategy can proceed with the optimization process with a long benign input. However, it can only affect a couple of characters (< 10) (see Figure 11(a)) and a limited number of words (< 2) (see Figure 11(b)) even with the longest patch configuration. Using the proposed partial matching strategy, we can use a patch with the same duration to attack longer sentences, and the impact length reaches $\sim$ 15 characters and $\sim$ 4 words for 500*ms* patch, and this can be further extended to $\sim$ 6 words with a longer patch (e.g., 750*ms*). In summary, using the same length of patch, we extend the impact length by 200% in characters and 287% in words, which proves the effectiveness of the *partial matching* strategy.

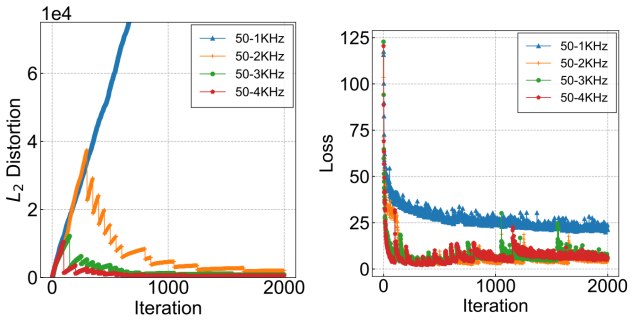
Evaluate mute patches: Next, we evaluate the generation of *mute patches* with different speech contexts. Since we expect that the mute patches could disrupt any user interference, we craft the patches based on a variety of sentences. More specifically, we randomly select multiple sentences from the benign audio dataset and craft one series of mute patches that are applicable to all sentences. We report the L_2 distortion of the mute patches in Figure 12.

We use a variable *"X Comm."* to represent the situation that mute patches can change the transcription to empty symbols for all "X" number of selected commands. The X value ranges from 10 to 50. The results show that the mute patches converge speedily for all the situations and reach a steady value in 200 iterations. Moreover, we notice that the more general mute patches we request, the higher power of mute patches we will get. For example, to craft a series of mute patches that are suitable for 50 commands, it triples the value of L_2 distortion after convergence compared to the solution with 10 commands.

Evaluate phoneme-level universal perturbation: After analyzing the generality of mute patches, we then investigate the universality of adversarial patches on different background audio contexts. The adversarial command patch is obtained by optimizing the objective function in Eq. (12). In this evaluation, we craft adversarial patches over 50,487 phoneme clips and report the success rate when using an adversarial patch on top of the specific phonemes. As depicted in Figure 13, the universal adversarial patch achieves 68% success rate on all the five types of phoneme clips. Furthermore, among the different types of phonemes, we observe that vowels and nasal are more compatible with the adversarial patch. In contrast, the fricatives and stops are more resilient against adversarial patches. This is because our adversarial patches contain more low-frequency energy due to Eq. (12), and the vowels and nasal present rich low-frequency components. Therefore, these phonemes will be affected much easier. In contrast, the fricative and stops contain more high-frequency energy than that of the patch.

Evaluate patch with different frequencies: One key benefit of using spectrogram patch rather than short pulses is that our patch can be more imperceptible. Generally, a patch with a wider bandwidth can be found much faster and can have a lower amplitude. In comparison, a patch with a narrow bandwidth could raise less attention in human auditory system, however, it will fail to deliver certain commands because of the lack of some specific frequency energy. Therefore, there is a trade-off between the frequency band, the perceptual level, and the successful rate of the patch. To investigate the best frequency band of the adversarial patch, we conduct the following experiment with four different frequency settings as follows: $50Hz \sim 1kHz$, $50Hz \sim 2kHz$, $50Hz \sim 3kHz$, and $50Hz \sim 4kHz$. We aim to retain the low-frequency components as they are more likely neglected by human ears (see Figure 8(a)). Our goal is to find a narrow frequency patch that has more low-frequency components and with lower overall amplitude. We craft 10 adversarial patches with every frequency setting, and record the average L_2 distortion and loss during the process. We present the results in Figure 14. As shown in Figure 14(a), if the patch only includes $50 \sim 1KHz$, it fails to achieve the target goal as the L_2 distortion keeps growing, and the loss in Figure 14(b) implies that this setting leads to the highest loss. For the $50 \sim 2KHz$ setting, the distortion rises at the first 400 iteration, and then it satisfies the target goal and starts dropping the amplitude of patch by iterations. The rest two settings converge faster. These results indicate that patch with a wider bandwidth is more likely to meet the target. To summarize, the $50 \sim 2KHz$ is the best choice, because it reaches the low amplitude after 1,000 iteration and has comparable L_2 distortion with other cases, and it also occupies less bandwidth. As a result, we use this setting for all the rest experiments.

Evaluate overall performance: In this experiment, we train 10 universal patches along with a series of mute patches. Then, we



(a) The distortion of SpecPatch (b) The loss of SpecPatch

Figure 14: Comparison of different frequency ranges

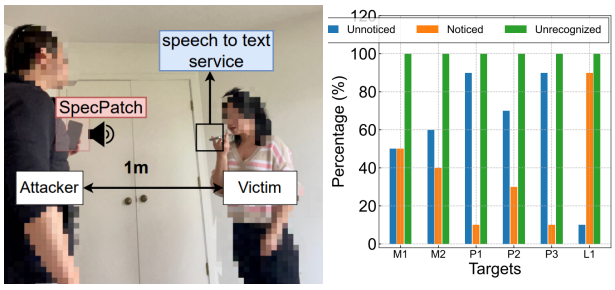
Target	Success Rate	Mis-Trans. Rate
"turn on the lights"	94.1%	100%
"find my phone"	96.5%	100%
"Turn off the kids' wifi"	87.1%	100%
"What is my password?"	86.5%	100%
"stop the music"	97.2%	100%
"open the door"	100%	100%
"lock the front door"	91.3%	100%
"Listen to the news"	92.5%	100%
"call 911 now"	98.5%	100%
"Cancel my alarm"	94.6%	100%

Table 2: Over-the-line attack performance

apply these SpecPatch to all of the benign audios in TIMIT dataset at random positions to validate the effectiveness of our attack. Table 2 reports the success rate after adding 10 different adversarial patches to 6,300 benign audios, resulting in 63,000 AEs. We count the success cases when the target command is the only output in the transcription. Otherwise, if the benign label is transcribed wrongly, we regard it as the *mis-transcription* case. We find that, SpecPatch could cause 100% mis-transcription rate for any audio. This means SpecPatch could almost surely deny the service of users. In terms of the target success rate, we achieve > 90% success rate for 8 out of 10 patches. The success rate for longer commands that have 4 to 5 words is lower than those of shorter commands, this is reasonable since longer target commands are hard to achieve in a noisy background.

5.3 Over-The-Air Attack

Attack Scenario: Figure 15(a) depicts the attack scenario. The victim is using the speech-to-text service while the adversary uses a smartphone to play SpecPatch to deceive the ASR model. Note that the adversary can play the attack audio at anytime, and once he/she launches the attack, the victim's commands will be denied by the consecutive mute signals. In our experiment, the adversary is 1 meter away from the victim, and SpecPatch is played at different volumes. We measure the loudness of the user's interference and the SpecPatch audios by a decibel meter. We conduct the experiments in three places: indoor room, outdoor street, and public dining hall.



(a) Real world attack scenario (b) User study results

Figure 15: Over-the-air SpecPatch attack.

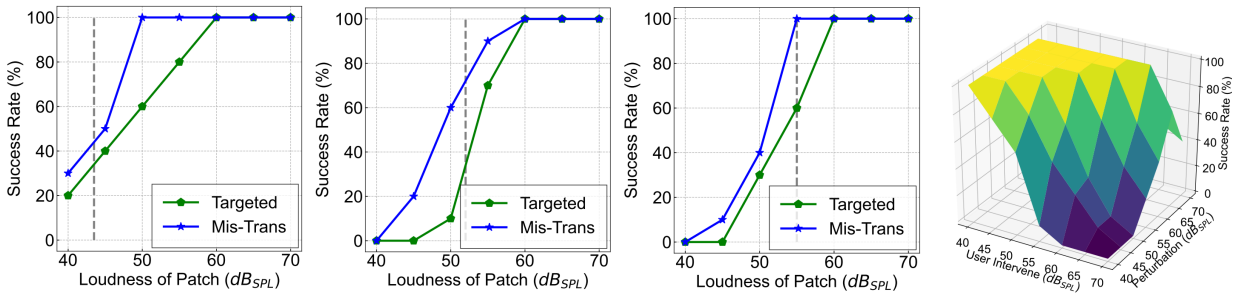
Attack Performance: In this experiment, we play a crafted patch of "open the door" 10 times for each volume, attempting to deliver this command to victim's phone. The victim is holding their smartphone and speaking at the volume of $55dB_{SPL}$. The ambient noise levels of the three places are $43.5dB_{SPL}$, $52dB_{SPL}$, $55dB_{SPL}$ for indoor room, outdoor street, and dining hall, respectively. We present the success rate of targeted attack and the mis-transcription attack in Figure 16. The grey dot line indicates the ambient noise level. As can be seen from the Figure 16(a), when the perceived patch volume is lower than the ambient noise level, there are 8 out of 10 attempts failed in the targeted attack scenario. Once the victim device perceives a comparable power (e.g., 45dB) from the patch audio, the success rate increases to 40% for the targeted attack. When the volume is $10dB$ greater than the ambient noise, we can achieve 80% success rate, and 100% in denying the user's input. We observe the similar results in Figure 16(b) and Figure 16(c). These results indicate that SpecPatch can successfully attack the ASR system with a limited power profile. Typically, SpecPatch achieves successful attacks when there is $< 5dB_{SPL}$ power difference between the patch and the ambient noise. If we raise the attack power, the success rate can be assured to 100%.

To better understand the relationship between user's volume and the loudness of patch, we conduct another experiment to control those two factors. We play the same patch 10 times at 7 volumes (from $40dB_{SPL}$ to $70dB_{SPL}$ with $5dB$ increments). For every volume, we use another speaker to play a benign audio with increasing volumes. This experiment is conducted in the same indoor place, and the result is present in Figure 16(d). We find that when the patch has same volume of the benign audio, it achieves 100% success rate. If the patch is $20dB$ less than the benign audio, SpecPatch no longer works. In general, a louder patch can achieve a higher success rate. Noticeably, when both the patch and the benign audio have high power, the success rate reduces to 40%.

5.4 User Study

To evaluate the stealthiness of SpecPatch in a real-world attack, we conduct two online user studies that involve ten volunteers to investigate the users' perception level of SpecPatch.

Study 1: In this study, the users are requested to hear four AEs that include the crafted patches. Then, we ask the volunteers about the contents they heard. The benign and adversarial transcriptions are described in Table 3. For the same benign sample, we add three different patches to achieve three goals (one of them is an empty transcription). The result shows that 10 out of 10 volunteers are



(a) Indoor scenario (b) Outdoor scenario (c) Public dining hall scenario (d) User's speech volume vs. patch loudness

Figure 16: Attack success rate across four different scenarios.

deceived by our attack, as all of them consider the benign label as their heard content. Surprisingly, we can inject the patch to a silent benign audio, and this implies the possibility of a hidden attack. Similarly, none of the volunteer can perceive the hidden patch, as 10 out of 10 considered the malicious "turn on the wifi" patch as a silent audio.

Study 2: To further validate SpecPatch, we conduct the second user study. The volunteers are asked to *pretend to speak to their voice assistants* while hearing the six different patches, these patches are played through their smartphones with a medium volume. The distance between the volunteer and the their smartphones is 0.5 meter. After that, they will answer questions to describe their comprehension of the heard patch. The options of perception levels include: *Unnoticed*, *Noticed*, and *Unrecognized*. *Unnoticed* indicates that the volunteer cannot hear a patch; *Noticed* implies that the volunteer can hear a patch but regard it as a normal noise; and *Unrecognized* stipulates that they cannot understand the meaning of the heard sound. We report the experimental result in Figure 15(b). The labels in x-axis represent different patches, namely, (M1 and M2 are two mute patches, P1-P3 are short patches, while L1 is a long patch that is composed of 3 short patches). It shows that most of participants (> 70%) cannot even notice our short patch attack (P1-P3). For the consecutive mute patches, there are around 50% of volunteers can perceive it. For the long patch, 9 out of 10 participants can clearly feel it. It is noteworthy that none of the patches can be understood by volunteers.

	Benign	Adversarial	Deceive
	"turn on the lights"	"open the door"	10/10
	"turn on the lights"	""	10/10
	"turn on the lights"	"open the window"	10/10
	""	"turn on the wifi"	10/10

Table 3: User case study

6 DISCUSSION

Limitations: SpecPatch has the following limitations: 1) the attack is model dependent; 2) the attack could not successfully attack very long sentences; 3) the attack distance is relatively short. For the first limitation, this attack can only attack the recurrent neural network, since our attack is established by exploiting the vulnerability of inter-connection between each cells. The second limitation can be addressed by introducing a longer patch, however, it might raise the alert of the victims. The third limitation can be possibly addressed by amplifying the power of the patch, but the adversary needs to handle both the distortion from the amplifier and the long-distance induced attenuation.

Defense: Prior studies [29, 47, 48] reveal that signal processing techniques can defend the adversarial audio attack since the adversarial perturbations are delicately crafted and hence are deemed fragile. The signal processing techniques, however, can reduce the fidelity of perturbations and protect the ASR models. Typical signal processing defense methods include 1) *Down sampling (DS)*: decreasing the sampling rate of AEs to degrade the quality of AEs [29, 47, 48]; 2) *Quantization*: this approach rounds the 16-bit precise value to its nearest integer multiple of constant Q, which has been adopted to defend against the attacks [29, 47]. 3) *Low pass filtering (LFP)*: this defense can use a Butterworth low-pass filter with different cutoff frequencies to remove the high-frequency components of the perturbations [29]. We will evaluate different defense approaches against SpecPatch in our future work.

7 CONCLUSION

In this work, we proposed SpecPatch, a human-in-the-loop adversarial patch attack on ASR systems. SpecPatch considers the scenarios when the users are presenting or intentionally disrupting the adversarial audio attacks against ASR systems. SpecPatch optimizes the adversarial patch to increase the length of the target commands. SpecPatch also includes Mute adversarial samples that can ensure the user interference does not affect the adversarial perturbation. Moreover, we further enhance SpecPatch to make it imperceptible and robust in both over-the-line and over-the-air attack scenarios. Our extensive real-world experiments show that SpecPatch can unnoticeably deliver the malicious commands in a noisy environment amid user's interference.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for providing valuable feedback on our work. This work was supported in part by National Science Foundation grants CNS-1950171, CNS-2226888, and CCF-2007159.

REFERENCES

[1] 1993. TIMIT Acoustic-Phonetic Continuous Speech Corpus. https://catalog.ldc.upenn.edu/LDC93S1. Accessed: 2021-11-04.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 265–283.

[3] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin RB Butler, and Joseph Wilson. 2019. Practical hidden voice attacks against speech and speaker recognition systems. *arXiv preprint arXiv:1904.05734* (2019).

[4] Hadi Abdullah, Muhammad Sajidur Rahman, Washington Garcia, Kevin Warren, Anurag Swarnim Yadav, Tom Shrimpton, and Patrick Traynor. 2021. Hear" No Evil", See" Kenansville"*: Efficient and Transferable Black-Box Attacks on Speech Recognition and Voice Identification Systems. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 712–729.

[5] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. 2017. Did you hear that? adversarial examples against automatic speech recognition. In *31st Conference on Neural Information Processing Systems (NIPS)*.

[6] Amazon. 2021. Amazon Echo. https://www.amazon.com/All-New-Echo-4th-Gen/dp/B07XKF5RM3.

[7] Amazon. 2021. Amazon Transcribe. https://aws.amazon.com/transcribe/.

[8] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*. PMLR, 173–182.

[9] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing robust adversarial examples. In *International conference on machine learning*. PMLR, 284–293.

[10] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665* (2017).

[11] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.

[12] Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 1–7.

[13] Guangke Chen, Sen Chenb, Lingling Fan, Xiaoning Du, Zhe Zhao, Fu Song, and Yang Liu. 2021. Who is real bob? adversarial attacks on speaker recognition systems. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 694–711.

[14] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. 2020. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1277–1294.

[15] Tao Chen, Longfei Shangguan, Zhenjiang Li, and Kyle Jamieson. 2020. Metamorph: Injecting inaudible commands into over-the-air voice controlled systems. In *Network and Distributed Systems Security (NDSS) Symposium*.

[16] Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng Wang. 2020. Devil's whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices. In *29th USENIX Security Symposium (USENIX Security 20)*. 2667–2684.

[17] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. 2017. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. *Advances in neural information processing systems* 30 (2017).

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[19] Google. 2021. Google Assistant. https://assistant.google.com/.

[20] Google. 2021. Google Home/Nest. https://store.google.com/product.

[21] Google. 2021. Google Speech. https://cloud.google.com/speech-to-text.

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

[23] A. Hannun. 2017. Sequence modeling with ctc. https://distill.pub/2017/ctc.

[24] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567* (2014).

[25] IBM. 2021. IBM Speeche. https://www.ibm.com/cloud/watson-speech-to-text.

[26] ISO 226:2003 2003. *Acoustics — Normal equal-loudness-level contours*. Standard.

[27] Zhuohang Li, Cong Shi, Yi Xie, Jian Liu, Bo Yuan, and Yingying Chen. 2020. Practical adversarial attacks against speaker recognition systems. In *Proceedings of the 21st international workshop on mobile computing systems and applications*. 9–14.

[28] Zhuohang Li, Cong Shi, Tianfang Zhang, Yi Xie, Jian Liu, Bo Yuan, and Yingying Chen. 2021. Robust Detection of Machine-induced Audio Attacks in Intelligent Audio Systems with Microphone Array. (2021).

[29] Zhuohang Li, Yi Wu, Jian Liu, Yingying Chen, and Bo Yuan. 2020. AdvPulse: Universal, Synchronization-free, and Targeted Audio Adversarial Attacks via Subsecond Perturbations. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1121–1134.

[30] Microsoft. 2021. Microsoft Azure. https://azure.microsoft.com/en-us/.

[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.

[32] Satoshi Nakamura, Kazuo Hiyane, Futoshi Asano, Takanobu Nishiura, and Takeshi Yamada. 2000. Acoustical sound database in real environments for sound scene understanding and hands-free speech recognition. (2000).

[33] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. 2019. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International conference on machine learning*. PMLR, 5231–5240.

[34] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. 2018. Inaudible voice commands: The long-range attack and defense. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 547–560.

[35] Lea Schönherr, Thorsten Eisenhofer, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. 2020. Imperio: Robust over-the-air adversarial examples for automatic speech recognition systems. In *Annual Computer Security Applications Conference*. 843–855.

[36] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. 2020. Light commands: laser-based audio injection attacks on voice-controllable systems. In *29th USENIX Security Symposium (USENIX Security 20)*. 2631–2648.

[37] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[38] Rohan Taori, Amog Kamsetty, Brenton Chu, and Nikita Vemuri. 2019. Targeted adversarial examples for black box audio systems. In *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 15–20.

[39] Siri Team. 2017. Hey siri: An on-device dnn-powered voice trigger for apple's personal assistant. *Apple Machine Learning Journal* 1, 6 (2017).

[40] Andrew Varga and Herman JM Steeneken. 1993. Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems. *Speech communication* (1993).

[41] Yuanda Wang, Hanqing Guo, and Qiben Yan. 2022. GhostTalk: Interactive Attack on Smartphone Voice System Through Power Line. In *Network and Distributed Systems Security (NDSS) Symposium*.

[42] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. 2021. PatchGuard: A Provably Robust Defense against Adversarial Patches via Small Receptive Fields and Masking. In *30th USENIX Security Symposium (USENIX Security 21)*. 2237–2254.

[43] Chong Xiang and Prateek Mittal. 2021. DetectorGuard: Provably Securing Object Detectors against Localized Patch Hiding Attacks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 3177–3196.

[44] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. 2018. The Microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 5934–5938.

[45] Hiromu Yakura and Jun Sakuma. 2018. Robust audio adversarial example for a physical attack. *arXiv preprint arXiv:1810.11793* (2018).

[46] Qiben Yan, Kehai Liu, Qin Zhou, Hanqing Guo, and Ning Zhang. 2020. Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided waves. In *Network and Distributed Systems Security (NDSS) Symposium*.

[47] Zhuolin Yang, Bo Li, Pin-Yu Chen, and Dawn Song. 2018. Characterizing audio adversarial examples using temporal dependency. *arXiv preprint arXiv:1809.10875* (2018).

[48] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. 2018. Commandersong: A systematic approach for practical adversarial voice recognition. In *27th USENIX Security Symposium (USENIX Security 18)*. 49–64.

[49] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 103–117.

[50] Baolin Zheng, Peipei Jiang, Qian Wang, Qi Li, Chao Shen, Cong Wang, Yunjie Ge, Qingyang Teng, and Shenyi Zhang. 2021. Black-box Adversarial Attacks on Commercial Speech Platforms with Minimal Information. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*.

[51] Tehseen Zia and Usman Zahid. 2019. Long short-term memory recurrent neural network architectures for Urdu acoustic modeling. *International Journal of Speech Technology* 22, 1 (2019), 21–30.