RESEARCH **Open Access**

CommanderUAP: a practical and transferable universal adversarial attacks on speech recognition models

Zheng Sun[1,2], Jinxiao Zhao[1,2], Feng Guo[1,2], Yuxuan Chen[1,2*] and Lei Ju[1,2]

Abstract
Most of the adversarial attacks against speech recognition systems focus on specific adversarial perturbations, which are generated by adversaries for each normal example to achieve the attack. Universal adversarial perturbations (UAPs), which are independent of the examples, have recently received wide attention for their enhanced real-time applicability and expanded threat range. However, most of the UAP research concentrates on the image domain, and less on speech. In this paper, we propose a staged perturbation generation method that constructs Commander-UAP, which achieves a high success rate of universal adversarial attack against speech recognition models. Moreover, we apply some methods from model training to improve the generalization in attack and we control the imperceptibility of the perturbation in both time and frequency domains. In specific scenarios, CommanderUAP can also transfer attack some commercial speech recognition APIs.

Keywords Adversarial examples, Universal adversarial perturbations, Speech recognition

Introduction

Speech is an efficient and rich information carrier, serving as a key mode of human–machine interaction. In recent years, related technologies such as automatic speech recognition (ASR) have attracted attention from academic and industrial communities. Amodei et al. (2016) have achieved accuracy rates beyond human levels in specific tasks. Intelligent voice devices or assistant service software based on ASR have been deeply integrated into our daily life, such as Google Assistant, Microsoft Cortana, Apple Siri, and iFLYTEK. As an important technology in deep learning, ASR will have wider support and application in the future.

Deep Neural Networks (DNN) have rapidly become popular in many tasks due to their powerful learning representation capabilities. At the same time, the security issues inherent in the model have also become a research hotspot, such as adversarial attacks, data poisoning attacks, and model stealing attacks. Adversarial attack is a type of attack during the model inference, which is defined as the adversary add crafted perturbation to the normal example to generate an adversarial example (AE). AE is difficult for humans to perceive as abnormal but will cheat the target model to output incorrect or specified inference results. The concept of AE was first proposed in Szegedy et al. (2013) for the task of handwritten digit recognition, and most of the subsequent work focused on images. Other fields such as language processing and ASR have also emerged some novel AE generation methods, further revealing the security issues of machine learning. The existence of adversarial attacks poses a threat to the development and application of machine learning, casting a shadow over it. Ilyas et al. (2019) pointed out that adversarial attacks are not

*Correspondence:
Yuxuan Chen
chenyuxuan@sdu.edu.cn
[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
[2] Quancheng Laboratory, QCL, Jinan, China

a problem that can be solved, but a flaw inherent in the model. Studying adversarial attacks is significant as it helps us enhance defensive measures and gain a deeper understanding of machine learning.

Although there are numerous studies on adversarial attacks in the field of speech recognition, research on universal attacks remains limited. Common adversarial attack methods against ASR involve generating a small perturbation and adding it to a normal audio example, causing the target model or other model to recognize incorrect text. Typically, this perturbation is generated based on a single example and only affects that example. We refer to this perturbation as a specific adversarial perturbation (SAP). Currently, the majority of work in this area is concentrated on SAP (Carlini and Wagner 2018; Qin et al. 2019; Yuan et al. 2018; Chen et al. 2020; Khare et al. 2018; Alzantot et al. 2018; Du et al. 2020; Taori et al. 2019). In contrast, universal adversarial perturbation (UAP) is more destructive, which is audio-agnostic, meaning it has a high attack success rate when added to any audio example. However, due to the requirement of simultaneously affecting multiple audios, existing methods struggle to maintain high success rates and imperceptibility under such strict constraints. Therefore, there is little research on UAP in ASR and each has its limitations (Neekhara et al. 2019; Zong et al. 2021; Lu et al. 2021; Guo et al. 2022). Furthermore, several studies concentrate on UAP attacks in other speech-related domains, such as speaker recognition (Li et al. 2020b; Xie et al. 2021b), speech command classification (Li et al. 2020b; Vadillo and Santana 2019; Abdoli et al. 2019), and environmental sound classification (Xie et al. 2021b).

This paper proposes a targeted UAP generation method that can launch attacks with a high success rate and great imperceptibility in both online and physical worlds. Our method will generate a CommanderUAP according to the target command, achieving the effect of deceiving the ASR model on a large number of normal audio examples. The process of constructing CommanderUAP adopts a two-stage generation method, with different stages having different purposes. In addition, we have applied some methods used in normal DNN training to optimize the attack effect. Experimental results show that the online attack constructed for 10 commands achieved a maximum of 84.0% and an average of 64.6% attack success rate. The average attack success rate in the physical world is 50.4%. In terms of perturbation imperceptibility, we control from both the time domain and the frequency domain. According to results from a human hearing perceptual survey, 98.8% of the auditory surveys were unable to perceive the presence of commands in CommanderUAP by listeners. Finally, after relaxing the restrictions, CommanderUAP can achieve different degrees of attack success rates on commercial APIs such as iFLYTEK and Baidu.

Contributions:

- **New generation algorithm** We designed a phased perturbation generation algorithm, that is, first eliminate the original semantics of normal examples, and then construct meaningful attacks. In addition, we summarized the control of perturbations in other papers and proposed a new method to improve perturbation imperceptibility from both the time domain and frequency domain.
- **Improved training methods** We linked the generation process of UAP with the training process of normal DNN, which provides a reference for future related research.
- **Transfer attack on commercial APIs** To our knowledge, we are the first to verify the attack effect of targeted UAP on commercial ASR APIs, although this requires sacrificing a certain degree of auditory perceptibility.

Background

Adversarial attacks can be classified according to various criteria. One criterion is the desired outcome of the attack: whether it aims to induce a specific output from the target model (targeted attack) or any output other than the correct one (untargeted attack). Another criterion is the level of access to the target model: whether the adversary requires the model's parameters or architecture (white-box attack), partial information about the model (grey-box attack), or no information at all (black-box attack). And as said before, in terms of whether the perturbations can affect multiple examples simultaneously, we categorize adversarial attacks into SAP attacks and UAP attacks. In this section, we review the problem definitions and existing advances of SAP and UAP attacks on ASR, and we present the threat model and basic concepts of CommanderUAP.

Problem formulation

The definitions of SAP and UAP for targeted attacks are different. Given an audio x_0 selected from the dataset D, the ASR model $f:\mathbb{R}^d \rightarrow T$ outputs the recognition result with the highest prior probability, expressed as $f(x_0) = \text{“what is the weather”}$, where $x_0 \in D \in \mathbb{R}^d$. The SAP generation method $S:\mathbb{R}^d \rightarrow \mathbb{R}^d$ generates a corresponding perturbation δ for each normal audio, based on the target command chosen by the adversary, such that the model f misclassifies the perturbed audio as the target command. This perturbation typically needs to satisfy a p-norm constraint, i.e., $||\delta||_p < \varepsilon$. For instance, $f(x_0^*) = \text{“turn off the light”}$, where $x_0{}^* = x_0 + \delta_0$, and

$\delta_0 = S(x_0)$. To achieve generality, the UAP generation method $U : \mathbb{R}^d \rightarrow \mathbb{R}^d$ trains an audio-agnostic perturbation μ on multiple audio examples $x_{m\sim n}$, which can be added to most normal audio and leads to misrecognition by f as the target statement. Formally, for a given $x_i \in D$, $f(x_i^*) =$ "turn off the light", where $x_i^* = x_i + \mu$, and $\mu = U(x_{m\sim n})$. To distinguish, we use δ and μ to represent SAP and UAP respectively, in fact, they both represent the perturbations added to the audio by the adversary. In this paper, the generation of CommanderUAP can be simply expressed as:

$$\underset{\mu}{argmax} \sum_{x\in D} C(f(x+\mu) = t) \quad \text{subject to } \|\mu\|_p \le \varepsilon \tag{1}$$

where D represents the distribution of normal examples x, μ denotes the resulting CommanderUAP, C is the indicator function, t is the target command specified by the adversary, p represents the norm type (such as L1, L2, or L∞), and ε is the upper constraint on the norm value. In summary, our objective is to train a universal perturbation μ that is acoustically imperceptible, so that the model will output the target command for most audio examples with μ added to them.

Related works

Specific adversarial perturbation The field of SAP attacks has been extensively studied, with the most diversified methods. Carlini and Wagner (2018) introduced an efficient adversarial attack for ASR by designing a loss function using Connectionist Temporal Classification (CTC) loss, achieving nearly 100% attack success rate against DeepSpeech (Amodei et al. 2016). Qin et al. (2019) utilized psychoacoustic masking effects to hide perturbations within the imperceptible range of human hearing while successfully attacking the Lingvo model (Shen et al. 2019). CommanderSong, proposed by Yuan et al. (2018), employed music segments as carriers and achieved a nearly 100% attack success rate against the Kaldi toolkit (Povey et al. 2011). It was also the first SAP attack method to demonstrate effectiveness in the physical world. Chen et al. (2020) proposed a method called Devil's Whisper, a black-box transfer attack on multiple commercial APIs and Intelligent Voice Control (IVC) devices, using substitute models. Other black-box attack methods, such as gradient estimation (Taori et al. 2019), genetic algorithms (Alzantot et al. 2018; Khare et al. 2018; Du et al. 2020), have also achieved high attack success rates. Despite the unique algorithms and applicable scenarios of these SAP methods, they all generate a one-to-one adversarial perturbation corresponding to a specific input audio. Furthermore, most methods assume that the adversary has prior knowledge of the user's input audio to the model or API, whereas, in practical scenarios, the user's input is often unknown. Consequently, SAP methods often lack real-time performance and generality in practical deployments.

Universal adversarial perturbation Most UAP research in ASR lacks effective trade-offs between success rate and imperceptibility and rarely considers the attack scenarios of UAP transferring to non-target models. The earliest UAP attack was proposed in image recognition (Moosavi-Dezfooli et al. 2017). Subsequently, many UAP studies emerged in the image field (Zhang et al. 2021). Neekhara et al. (2019) first proposed an untargeted UAP attack against the ASR model DeepSpeech. Untargeted attacks are less threatening and enlightening than targeted attacks, especially in ASR. In addition, the authors also verified the attack performance of their UAP when transferring to WaveNet. Zong et al. (2021) successfully implemented a targeted UAP attack against DeepSpeech. The author's generation method was divided into two stages: in the first stage, the perturbation value was not penalized, and the UAP obtained in this stage achieved a success rate of about 99% on five commands, but the imperceptibility was very poor; the second phase punished the perturbation maximum, which effectively improved the imperceptibility, but the average success rate dropped to 54.21%. Lu et al. (2021) proposed two types of UAPs: additive perturbation and prefix perturbation. They focused on exploring the robustness of end-to-end models when facing UAPs. The results showed that the LAS model had an attack success rate of over 99% for both types of UAPs; the RNN-T model only showed some low robustness to prefix perturbation, with an attack success rate of about 40%; and the CTC model was robust to both types of UAPs, with an attack success rate of almost 0%. The article only used empty sentences and "thank you" as target commands. In addition, the authors did not impose any restrictions on the perturbation throughout the experiment. The above works are the basis of speech recognition UAP research, which shows different degrees of attack capabilities in specific scenarios, but they have some shortcomings in terms of threat or imperceptibility.

Li et al. (2020b) and Guo et al. (2022) are the latest developments related to UAP in ASR. The target model of Li et al. (2020b) was speaker recognition model X-vectors and lightweight speech command recognition model Speech Command. The authors generated subsecond time-independent targeted UAPs that could work in the physical world. We concentrated on their simple attempt to attack the DeepSpeech. The experimental results showed that their UAPs were hard to penetrate to distant speech inputs. Guo et al. (2022) trained SpecPatch on the segmented phoneme level, achieving good generality and

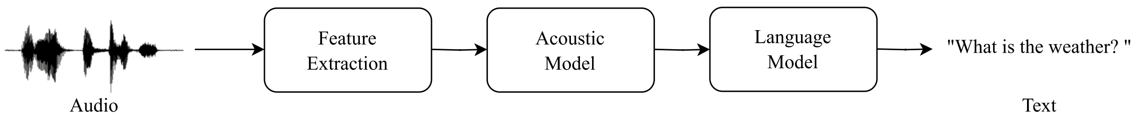

Fig. 1 The workflow of Kaldi

time independence. The method inserts silent frames to disrupt the original user semantics in normal audio. The experimental results showed that SpecPatch achieved online white-box attack success rates of over 90% on 10 commands. Like other targeted works, Guo et al. (2022) did not explore the attack effect of SpecPatch when transferring to other models or commercial APIs.

Threat model

In this paper, we define the adversary's goal as generating concealed UAPs and deceiving the ASR model or commercial API service to output a specified target command on most normal audios. The adversary achieves this by embedding pre-generated UAPs into the user's normal audio input without knowledge of the specific audio. We assume that the adversary can arbitrarily specify the target command, without being limited by the length of the target command or the original audio information.

The target model, i.e., Povey et al. (2011), is a white-box to the adversary. This implies that the adversary has all the knowledge about this model such as model structure and parameters, etc., so the adversary can perform operations such as extracting intermediate results, repeatedly modifying perturbations in the input, and adding constraints on perturbations, which are common assumptions and methods in other adversarial attack methods (Neekhara et al. 2019; Zong et al. 2021; Lu et al. 2021; Li et al. 2020b; Guo et al. 2022; Carlini and Wagner 2018; Qin et al. 2019; Yuan et al. 2018; Alzantot et al. 2018).

Other models, such as speech recognition APIs provided by commercial companies such as iFLYTEK (https://www.iso.org/standard/34222.html), Alibaba (https://www.alibabacloud.com/zh/product/intelligent-speech-interaction), Baidu (https://ai.baidu.com/tech/speech), and Tencent (https://cloud.tencent.com/document/ product/1093), are black-box themselves. Both adversaries and users can only use them by sending audio and getting recognition results. Although adversaries only have access rights to such other models, Yuan et al. (2018), Chen et al. (2020), Xie et al. (2021a), Neekhara et al. (2019) rely on the transferability of AEs to achieve attacks on black-box models.

In addition, we assume that the adversary can launch attacks in the physical world by means of playing through speakers and inputting through microphones. The specific settings will be described in the subsequent experimental process.

Target model

CommanderUAP is a further exploration based on (Yuan et al. 2018), mainly for a universal adversarial attack against the open source speech recognition toolkit Kaldi. Kaldi is one of the most popular high-accuracy ASR platforms, with 12.6k stars and 5.2k forks on GitHub. Figure 1 illustrates the workflow of Kaldi. Initially, the audio undergoes preprocessing such as pre-emphasis, framing, and windowing. Relevant features are then extracted, with Mel Frequency Cepstrum Coefficients (MFCC) being the chosen method in this case. Next, the acoustic model takes the MFCC features of the audio as input and computes the corresponding Hidden Markov Model (HMM) states for each frame of the audio data. These states are then mapped to the corresponding phoneme states, ultimately yielding a phoneme identifier sequence. The language model computes the probability of the text corresponding to the phoneme identifier sequence appearing in the corpus. By integrating the computations of multiple parts, Kaldi produces the recognized text. Currently, popular end-to-end ASR models (Amodei et al. 2016; Graves and Jaitly 2014; Battenberg et al. 2017) no longer explicitly separate the acoustic model and language model components.

Specifically, we select the ASpIRE Chain Model from Kaldi as our target model. ASpIRE utilizes an HMM-DNN acoustic model, where the DNN takes MFCC features as input and computes the posterior probability matrix $\mathbf{M}$. Each element $M_{ij}(1 < i < n, 1 < j < k)$ represents the posterior probability that the i-th frame of MFCC features belongs to the j-th phoneme state. Phoneme states are represented by posterior density function identifiers known as pdf-ids. The range of i depends on the number of feature groups in the MFCC features, which is determined by the total number of frames. The range of j is the number of all phoneme states, i.e., the total number of pdf-ids, determined by the granularity set during model training. By extracting the maximum value from each row of $\mathbf{M}$, we obtain the pdf-id sequence $\mathbf{m}$. The entire process of feature extraction and DNN forward calculation for audio x is represented by $g(x)$, so $g(x) = \mathbf{m} = (m_1, m_2, \ldots, m_n)$, where $m_i = \arg\max_j M_{i,j}$. The computation process of $g(x)$ is differentiable. $\mathbf{m}$ is the most likely pdf-id sequence corresponding to the audio x, and by further transformation, we can obtain the phoneme state sequence. Subsequently, the phoneme state sequence is merged into a phoneme sequence by other components of the HMM and language model, leading to the inference of the corresponding text. In summary, the pdf-id sequence outputted by the DNN serves as a crucial intermediate feature in the entire speech recognition process. Therefore, it is a feasible method to construct AEs by modifying the original audio data from the perspective of the pdf-id sequence.

Pdf-id sequence matching

The pdf-id sequence matching algorithm is the core method of Yuan et al. (2018) to launch attacks on Kaldi. The authors initially refactored a large number of shell scripts and C++ code into Python and then proposed the pdf-id sequence matching algorithm to construct AEs. Specifically, the authors extract MFCC features from the perturbed audio and the target command audio (obtained through Text-to-Speech (TTS)), input them into Kaldi's DNN, and obtain the corresponding pdf-id sequences for each. Then, the authors utilize a custom loss function Eq. 2 to describe the difference between the two pdf-id sequences:

$$Loss_{net} = \left\| g(x+\delta) - \mathbf{b} \right\|_1 \tag{2}$$

where the term $g(x+\delta)$ represents the pdf-id sequence associated with an audio segment x that has been added by a perturbation δ. The term $\mathbf{b}$ represents the pdf-id sequence that corresponds to a pre-selected target command, which is called the optimization reference target in the subsequent context. The objective is to iteratively calculate the $Loss_{net}$ and backpropagates it to the distortion term δ for gradient descent, aiming to minimize the discrepancy between $g(x+\delta)$ and $\mathbf{b}$. The resulting AE $x+\delta$ will then be recognized as the target command (the command corresponding to $\mathbf{b}$).

In addition, Carlini and Wagner (2018) introduced a generic noise model to simulate electronic noise and background noise, forcing the AEs to be generated under such noise effects. Therefore, when deployed to the real physical world, the AEs can resist the influence of the actual environment and still attack successfully. The calculation Eq. 2 after introducing the noise effect is:

$$Loss_{net} = \left\| g(x+\delta+n) - \mathbf{b} \right\|_1 \tag{3}$$

where n represents a random noise vector sampled from a uniform distribution $(-N, N)$. N is the bound of random values, and 100 is usually taken in experiments. Evaluation results demonstrate that the method of incorporating random noise can enhance the robustness of $x+\delta$ against distortions caused by environmental factors, thereby enabling attacks in physical scenarios. In the generation of CommanderUAP, we similarly introduce random noise to improve the robustness of the perturbation.

Overview

The research method in this paper mainly consists of four steps: (1) Random shuffling and grouping of the dataset; (2) Stage 1 generation; (3) Stage 2 generation; (4) Transfer attack. First, we randomly shuffle the dataset to prevent UAP from learning the order of the dataset during generation, which would weaken its generalization ability. Then, the generation of CommanderUAP will be divided into two stages, each with different purposes. In Stage 1, we generate BaseUAP to cover the entire normal audio, eliminating the influence of the original semantics on the recognition result. Subsequently, based on BaseUAP, we continue to generate AttackUAP with practical attack significance, which is Stage 2. The specific differences between the two stages will be discussed in sections "Two-stage generation" and "Regulating the imperceptibility of CommanderUAP". The CommanderUAP is obtained by adding the BaseUAP and AttackUAP, that is, CommanderUAP = BaseUAP + AttackUAP. Finally, to maintain the attack capability when transferring to other models, we relaxed some constraints during the UAP generation process.

Approach

Our attack aims to overcome two challenges: (1) eliminating the influence of the original semantics of normal audios on the recognition results, thereby forcing the model to output the results expected by the adversary; (2) achieving high success rates while making the command as imperceptive as possible. To overcome the first challenge, we propose using a two-stage generation method, as shown in section "Two-stage generation". Additionally, in sections "Batch training and weighted gradient" and "Dropout", we introduce techniques from regular deep learning tasks into the generation of perturbations to accelerate the convergence process and enhance the attack effect. Regarding the second challenge, we employ a method that simultaneously controls the imperceptibility in both the time and frequency domains, which will be detailed in section "Regulating the imperceptibility of CommanderUAP". Finally, Section "Transfer attack on API" outlines the attempts made to launch attacks on non-target models.

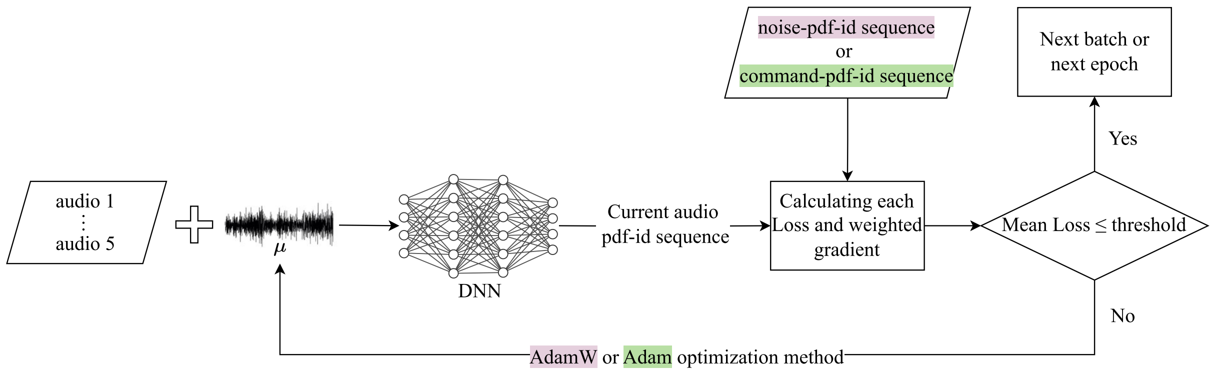

Fig. 2 Process of Stage1 for BaseUAP and Stage2 for AttackUAP

Two-stage generation

The generation of CommanderUAP is divided into two stages, each with different meanings and goals. Both perturbations are initialized as **0** vectors and are trained on the same training dataset using a group parallel method, but the generation algorithms are slightly different. Figure 2 shows the process details of the two stages in the generation of CommanderUAP, where the purple part and the green part respectively represent the choices of Stage 1 and Stage 2. It should be noted that BaseUAP and AttackUAP must be added to normal audio at the same time to deceive the ASR model.

The purpose of Stage 1 is to allow BaseUAP to cover the semantics of normal audio. In each round of epoch in Stage 1, a uniformly distributed noise audio of the same duration as the normal audio is generated, and the noise-pdf-id sequence extracted from this noise audio is used as the optimization reference target for calculating loss in this epoch (**b** in Eq. 3). After multiple epochs of training, a BaseUAP that meets the requirements is obtained. The reason for extracting noise-pdf-id sequences from a new noise file as the optimization reference target in each epoch is to let BaseUAP learn more general features of random noise, and enhance BaseUAP's attack ability to transform normal audios into "[noise]". If we fix a noise-pdf-id sequence as the optimization target in Stage 1, although the entire optimization quickly tends to be stable, the obtained BaseUAP shows an "overfitting" characteristic and poor auditory concealment. Therefore, although the method of changing the optimization target in every epoch in Stage 1 prolongs the training time and difficulty, it effectively improves the success rate and concealment, which also helps the generation of Stage 2.

The BaseUAP and normal examples are combined pairwise to create a new training set, and stage 2 generates the AttackUAP on this basis. The target pdf-id sequence in this stage is called the command-pdf-id and needs to be chosen and fixed according to the adversary's specified attack command. For example, the audio generated by TTS of the command "take a picture" can be used to extract the corresponding command-pdf-id sequence. Since BaseUAP already covers the majority of the original semantics in normal examples, the command-pdf-id sequence selected in Stage 2 can be shorter while still maintaining attack effectiveness. Let's assume the length of the target pdf-id sequence is L. During the training process, it is necessary to align the first L pdf-id values of each example with the target. Therefore, the duration of the final AttackUAP is typically shorter than that of normal audio. The rest of the audio is still influenced by BaseUAP, causing it to be misidentified as "[noise]". As a result, the AE is likely to be recognized as "target command" + "[noise]". which does not affect the feasibility of the UAP attack.

Batch training and weighted gradient

Training a UAP with a high success rate and great imperceptibility in adversarial attack is similar to training a DNN model with high generalization ability and low complexity in deep learning. Therefore, we consider migrating some common basic methods in DNN training to UAP generating, such as batch training and weighted gradient.

In the generation, batch training refers to shuffling the training dataset first in each epoch, and then dividing it into several batches according to the specified batch size (such as 5 audios per batch), and the subsequent training process is carried out in every batch. By reading other papers and codes of UAP about ASR, we find that the batch training method has not been widely used.

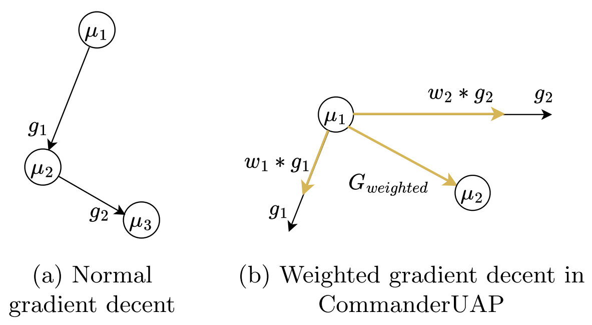

Fig. 3 Comparison of normal and weighted gradient decent

Previous methods were all based on forward propagation and updates on individual normal audio.

Within each batch, the update of CommanderUAP employs a weighted gradient. We first add the UAP to each example in the batch and calculate the loss and gradient for each with respect to the UAP. Then we assign weights to the gradients based on the corresponding loss values. Finally, the UAP is updated using the weighted gradient. In this approach, an example with higher loss values in the batch will have larger weights, which means that the current update will focus more on optimizing the direction for example that is farther away from the target command. UAP is updated only once per batch, and the calculation process is shown in formula (4):

$$\begin{aligned} G_{weighted} &= \sum_{i}^{b} \omega_i * g_i \\ \omega_i &= \frac{Loss_i}{\sum_{i}^{b} Loss_i} \end{aligned} \tag{4}$$

where i represents the index of the current audio within the batch, b represents the batch size, ω_i and g_i represent the weight coefficient and gradient of the current audio.

Figure 3 compares the differences between using batch training with weighted gradients and using single example training with its gradient. To simplify the illustration, the dimensionality of the audio space is reduced to a two-dimensional plane in the figure. Let's assume a batch size of 2 and ignore the learning rate and other update methods for simplicity. Figure 3a represents the traditional approach of using single example gradient descent. Assuming the current UAP is μ_1, the loss and gradient g_1 are computed using example 1, and the UAP is updated from μ_1 to μ_2. This process is repeated for UAP updates. It is similar to setting the batch size to 1 in regular DNN training, resulting in a more erratic path, and a higher risk of getting stuck in local minima. Figure 3b represents the batch training with weighted gradients used in CommanderUAP generation. Starting from the current perturbation μ_1, $Loss_1$ and g_1 are computed for example 1, and similarly, $Loss_2$ and g_2 are computed for example 2. Then, the weighted sum of gradients is calculated according to Eq. 4, and the perturbation is updated from μ_1 to μ_2. This update process is indicated by the yellow arrow in Fig. 3b. The method makes each update of UAP affected by the optimization direction of multiple examples, which not only reduces the chaos of the gradient by considering global updates but also stably advances to the optimal point with methods such as momentum.

Dropout

We have introduced dropout for the first time in the training of UAPs to enhance their generalization across different normal audio and models. Dropout is a DNN training technique that reduces model variance and overfitting risks while enhancing generalization. It achieves the effect of self-ensemble learning by randomly dropping some neurons. In the field of adversarial attacks and defenses, dropout has been used in previous work to achieve different objectives. Wang et al. (2018) uses dropout for a certain layer of the network during inference, which can effectively defend against adversarial attacks. Xie et al. (2021a) proposes a dropout-based gradient iterative attack method to improve transferability and verify the input invariance of dropout from the data level. Inspired by these works, we aim to adapt UAPs during the training process to overcome information loss caused by dropout and improve their generalization across audio and even models.

Different from using dropout at the model level (neuron level), we use dropout at the perturbation data level. In the computational graph, the data first goes through a layer of dropout processing before acoustic feature extraction. Specifically, each value of CommanderUAP has a certain probability of being temporarily set to 0, and this part of values will not be updated during backpropagation. The probability is called the dropout rate. It should be noted that to avoid losing too much information in audio preprocessing, we recommend using a lower dropout rate in the generation of UAP. This saves time and ensures the attack effect of UAP.

Regulating the imperceptibility of CommanderUAP

In the generation of CommanderUAP, we adopt an auditory masking penalty method that controls the perceptibility of perturbations in the frequency domain. On the other hand, most SAP and UAP methods use the $\|\mu\|_2$ as a penalty term in the objective function, or apply clipping to $dB(\mu)$ or $\|\mu\|_\infty$ after each perturbation update,

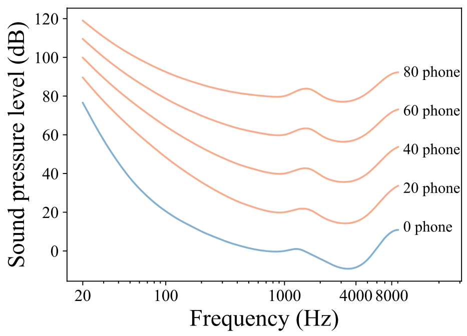

Fig. 4 Equal loudness contours and hearing curve of human

aiming to reduce the perturbation's value. This approach of penalizing can be seen as controlling the perturbation's imperceptibility in the time domain. However, when combined with the Adam optimization method, it may weaken the penalty (Loshchilov and Hutter 2017). By simultaneously controlling the perturbation in both the frequency and time domains, CommanderUAP exhibits excellent imperceptibility properties in human perception.

Controling frequency domain imperceptibility

There have been many works trying to use methods such as high-frequency perturbation (Zhang et al. 2017), spectral modification (Guo et al. 2022; Abdullah et al. 250251), and psychoacoustic masking effect (Qin et al. 2019; Schönherr et al. 2018) to hide the perturbation in the area that human ear cannot perceive. However, we found in our experiments that if UAP is controlled using frequency masking effect manner in Qin et al. (2019), although the imperceptibility of UAP is very good, it will greatly affect its attack capability.

To mitigate the perceptibility of perturbations in the frequency domain, we leverage the equal loudness contours known as the Fletcher–Munson curve (https://www.iso.org/standard/34222.html). The curve graphically represents the sensitivity of human hearing to sounds at different sound pressure levels and frequencies. Figure 4 shows the equal loudness curve from 0 to 80 phones. Each line expresses the sound pressure level required for a sound of different frequencies to be perceived as that loudness by the human ear. It can be observed that the trend of the curves remains basically consistent across different loudness levels. The lowest line in the figure is the auditory curve of quiet conditions (0 loudness), which is called the human ears auditory curve.

By leveraging this characteristic, we utilize the data of the human auditory curve as a vector, denoted as D. Through operations such as linear interpolation, normalization, and reflection, we derive the frequency domain weights FQW. The perturbed frequency domain values are multiplied by FQW, and the L2 norm of the resulting vector is calculated, yielding the $Loss_{hearing}$:

$$Loss_{hearing} = \|FQW * FFT(\mu)\|_2 \tag{5}$$

where $FQW = 1 - Normalize(Interpolate(D))$.

Therefore, in a practical attack, we employ a weighted sum of Eqs. 3 and 5 as our objective function, given by

$$Loss = Loss_{net} + c * Loss_{hearing} \tag{6}$$

where c is a coefficient that controls the correlation between the two components. Under the influence of the $Loss_{hearing}$ penalty term, CommanderUAP weakens the magnitude in the frequency domain during its generation process, with a milder penalty on high-frequency components and a stronger penalty on low-frequency components, aligning with the auditory characteristics of the human ear. We do not intentionally pursue making the loudness of CommanderUAP at all frequencies below a specific standard; instead, we aim to apply penalties through optimization as much as possible. Hence, although it might be considered a more coarse method compared to applying frequency masking on each frame as done in Qin et al. (2019), this paper simply uses the sensitivity of human hearing as a reference for the global frequency domain penalty for CommanderUAP.

Guo et al. (2022) also utilizes the characteristics of auditory curves, but it only penalizes the most sensitive frequency range of the human ear. For example, it applies penalties only to the magnitude of perturbations within the range of 1.6kHz to 4kHz. Since SpecPatch adds small, localized patches to the spectrum of the clean audio, it is feasible to apply penalties to specific frequencies using auditory curve data. However, in the case of CommanderUAP, where the frequency range of BaseUAP and AttackUAP covers the entire spectrum, we choose to apply different levels of penalties to perturbations across the entire frequency range.

Controling time domain imperceptibility

In addition to hiding perturbations in the frequency domain, it is also essential to control the values of CommanderUAP in the time domain. During the training process, we utilized the AdamW method (Loshchilov and Hutter 2017), which improves the regularization of the Adam optimizer by decoupling weight decay and gradient updates. AdamW addresses the issue that when using L2 regularization on input data together with the Adam optimizer, the gradient computation of the regularization term can influence the backpropagation, leading to inadequate penalties for large values. However, CommanderUAP requires more iterations so that the deficiency of inadequate penalties can accumulate. To avoid this problem, the AdamW optimizer does not add an L2 penalty term in the objective function but directly modifies the input data. Specifically, before modifying the data x based on the gradient, x is first multiplied by the decay coefficient. The decay coefficient is less than 1, for example, if 0.995 is taken, then $x = 0.995 * x$. Apart from this modification, the application of AdamW is consistent with Adam regarding momentum and RMSprop methods. Furthermore, AdamW helps mitigate the drawbacks introduced by the L2 norm computation in $Loss_{hearing}$.

It should be noted that another important difference between Stage 1 and Stage 2 is the control methods of imperceptibility. While it is ideal to regulate in the frequency and time domains, after practical validation, we decided to use both the $Loss_{hearing}$ penalty term and the AdamW optimizer only in Stage 1, which is the process of generating BaseUAP. In Stage 2, but just rely on the $Loss_{hearing}$ penalty term and use the regular Adam optimizer in Stage 2. The attack success rate of using AdamW in Stage 2 is approximately 40% lower than that of using Adam. This is because if too much noise control is added in Stage 2, even if the perturbation is not easy to be perceived, the attack success rate will also decrease significantly. Obviously, it is more important for Stage 2 to achieve accurate attack success. On the other hand, the BaseUAP generated in Stage 1 already has a large search space which can achieve the expected attack effect even under double constraints in frequency and time domains. Additionally, although Stage 2 lacks the weakening of numerical values in the time domain, AttackUAP still maintains good imperceptibility based on the auditory properties of BaseUAP. As a result, the final generated AEs are unlikely to be perceived by human ears as containing the target command. The detailed processes of Stage 1 and Stage 2 are presented in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Stage 1

```
Input: dataset $D$, normal distribution $\mathcal{N}(\mu, \sigma^2)$, batch size $S$
Output: BaseUAP $\mu$
1  initialize model $g(x)$, $\mu \leftarrow 0$;
2  randomly divide $D$ into batches;
3  for number of training iterations do
4      for each batch do
5          Generate a noise audio from $\mathcal{N}$ and extract noise-pdf-id;
6          for $i \leftarrow 1$ to $S$ in batch do
7              Compute the $Loss_i$ and $g_i$ using Eq. (4), (5), (6);
8          end
9          Compute weitght of each gradient: $w_i = Loss_i / sum(Loss_i)$;
10         Compute $G_{weighted} = w_i * g_i$;
11         Update perturbation: $\mu = AdamW(\mu, G_{weighted})$;
12     end
13 end
14 return BaseUAP
```

Algorithm 2 Stage 2

```
Input: dataset $D$, target command $t$, batch size $S$, BaseUAP
Output: AttackUAP $\mu$
1  initialize model $g(x)$, $\mu \leftarrow 0$;
2  randomly divide $D$ into batches;
3  for each audio in $D$ do
4      $X_i = x_i + BaseUAP$;
5  end
6  extract command-pdf-id according to $t$;
7  for number of training iterations do
8      for each batch do
9          for $i \leftarrow 1$ to $S$ in batch do
10             Compute the $Loss_i$ and $g_i$ using Eq. (4), (5), (6);
11         end
12         Compute weitght of each gradient: $w_i = Loss_i / sum(Loss_i)$;
13         Compute $G_{weighted} = w_i * g_i$;
14         Update perturbation: $\mu = Adam(\mu, G_{weighted})$;
15     end
16 end
17 return AttackUAP
```

Transfer attack on API

Another important metric for assessing AEs attack capability is the transfer attack success rate in black-box scenarios. The transferability of AEs enables them to pose a threat to models other than the target model. Although some studies in the CV field have enhanced the transferability of UAP (Li et al. 2020a; Hashemi et al. 2020), only (Neekhara et al. 2019) conducted untargeted attacks from DeepSpeech to WaveNet in ASR.

The CommanderUAP generated by the above two-stage method has poor transferability. The reason is that the perturbation overfits the decision space of the Kaldi ASpIRE model in terms of model independence during the training process, resulting in insufficient generalization ability across models. Additionally, the control over perturbation imperceptibility also constrains its feature representation capacity.

By relaxing the constraints on perturbation imperceptibility to some extent and improving the generalization capability, CommanderUAP can achieve a high success rate in black-box transfer attacks against commercial ASR APIs. Specifically, we no longer use the two-stage generation method, but instead, skip the stage 1 step and directly generate perturbations targeting the desired command. This process is similar to stage 2 but with modifications to the algorithm details. Firstly, the optimization reference target is set to the pdf-id sequence of a specific command that matches the length of the normal audio, ensuring that each frame of the normal audio is directly affected by the perturbation. Additionally, the dropout rate is appropriately increased to make the generated perturbations more robust and adaptable to damage, so that it still has a chance to succeed when transferring to commercial APIs. In terms of imperceptibility control, only the $Loss_{hearing}$ penalty term is used to control the frequency domain performance.

Evaluation

This section presents the experimental results of the above methods. We mainly evaluate the attack capability of CommanderUAP generated by the two-stage method in terms of white-box attack and physical attack. In addition, we verify under what conditions AEs can achieve transfer attacks on APIs. Finally, to assess whether the AEs are easy to be perceived by human hearing, we design a human perception survey and collect results on a crowdsourcing platform.

Setup

We used TensorFlow to implement our method. We first designed 10 commands as target commands and used Microsoft TTS to generate the corresponding audio. After ensuring that each audio could be accurately recognized by Kaldi, we extracted the pdf-id sequences corresponding to each audio, which were probably used as optimization reference targets in subsequent experiments. The specific commands are shown in Table 1. The target model remains consistent with (Yuan et al. 2018), which is the ASpIRE Chain model released on October 15, 2016. We experimented on an Ubuntu server equipped with an NVIDIA 2080ti GPU.

Table 1 White-box attack results

Target command	SRoA(%)	WER	SNR(dB)
"take a picture"	84	0.2	0.15
"shut down now"	71	0.19	0.13
"turn on GPS"	58	0.19	– 0.32
"lock phone"	56	0.51	0.34
"open the front door"	59	0.23	– 0.03
"make a credit card payment"	69	0.14	– 0.43
"turn off the light"	63	0.21	0.17
"read mail"	55	0.46	0.34
"airplane mode on"	62	0.27	0
"where is my car"	69	0.2	– 0.05
Average	64.6	0.26	0.03

Dataset and hyperparameters For the choice of normal audio which is the carrier of perturbation, we used the conversational dataset LibriSpeech (Panayotov et al. 2015). Based on the recording background of the audio, the dataset is divided into multiple subsets. We selected the test-clean subset as the normal audio dataset for training and testing CommanderUAP. Since these audios have varying durations, we split all audios into 2.5-second segments and discarded the part of the audio shorter. Finally, we obtained 3986 audio examples. Generating UAP does not require a large amount of training data (Zong et al. 2021; Moosavi-Dezfooli et al. 2017). We randomly selected 50 segmented audios and set the batch size to 10. With a small training set of audios, combined with methods like random shuffling and weighted gradient descent in batch, the perturbation already possessed attack capability. In this scenario, further expanding the dataset or using data augmentation operations would not significantly improve the success rate. Instead, it would only prolong the training time. During the testing phase, we randomly selected 100 audios as carriers for perturbation verification.

Compared to the range of pixel values in images being (0,255), audio data has a much larger value range. In our experiments, we chose 16-bit raw audio without normalization, with a value range of (−32767,32767). Therefore, the learning rate was set to a larger value of 100. The hyperparameters for AdamW were selected according to the recommended default values in Kingma and Ba (2014), such as beta1 = 0.9, beta2 = 0.999, and the weight decay coefficient was set to 0.995.

Metric We used the following metrics to evaluate the effectiveness of our work:

1. Success Rate of Attack (SRoA): It is the proportion of successfully attacked audios in the test set. The criteria for determining the success of the AEs may vary depending on the experimental conditions, and the specific definition will be introduced later in the paper.
2. Word Error Rate (WER): This metric represents the ratio of total errors (substitutions, insertions, and deletions) to the total number of words in the reference text. The adversary's goal is to make the model's recognition result for the AEs match the target command exactly, which corresponds to a WER of 0. Therefore, the lower the WER, the more successful the attack. It should be noted that our strategy allows for the possibility of content such as "[noise]" or "[laughter]" in the recognition results. This irrelevant information does not affect the adversary's expected attack effectiveness but may have some impact on the performance of WER.
3. Signal-to-Noise Ratio (SNR): This metric is widely used to quantify the level of noise relative to the power of an audio signal. It is calculated as SNR(dB)=$10log10(Px/P\mu)$, where $P\ x$ and P_μ represent the average power of the clean audio and the perturbed audio, respectively. A higher SNR indicates lower noise energy, thus indicating a more imperceptive perturbation. Research that uses psychoacoustic characteristics to hide perturbations usually does not use metrics such as SNR or sound pressure level to demonstrate imperceptibility.

White-box attack

White-box online attacks on the target model represent an idealized attack scenario. We first demonstrate the attack capability of CommanderUAP in this simplest setup. Stage 1 and Stage 2 are the same in all parameters except for the way of obtaining the pdf-id sequence as the optimization reference target and the optimization algorithm. In each epoch of stage 1, we generate noise audio within the range of 0 to 2000 to extract the noise-pdf-id sequence.

In the white-box online attack experiments, we consider an AE successful if the recognition result is the complete target command or if it starts with the target command and ends with irrelevant content such as "[noise]" or "[laughter]". Under this setup, the results of white-box attacks are shown in Table 1.

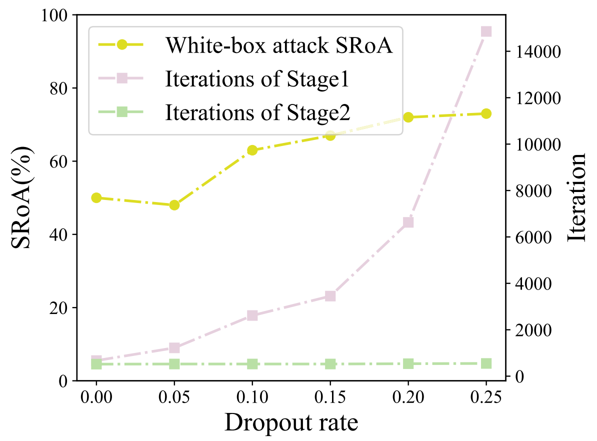

Fig. 5 SRoA and iterations with different dropout rates

Dropout rate In the common DNN training process, the dropout rate is usually fixed at 0.5, or initialized to a smaller value and then gradually increased. Appropriate dropout can effectively reduce the risk of overfitting and improve generalization. The effect of dropout value on SRoA and SNR is shown in Fig. 5. The experimental results show that the increase in dropout rate can only bring a slight improvement in attack success rate. On the other hand, the dropout rate and the number of iterations for both stages are positively correlated. The number of iterations for Stage 1 shows exponential growth, while the number of iterations for Stage 2 increases very slightly. When the dropout rate is set to 0.25, Stage 2 only needs to iterate 30 more times than when no dropout is used. A high dropout rate means that the important information learned by UAP in the current iteration is likely to be masked by the dropout in the next iteration. Figure 6 shows how the spectrogram of an AE changes as the dropout increases from 0 (i.e., no dropout) to 0.5 (i.e., dropping audio values with a half probability). It can be seen that as the dropout rate increases, the harmonics of the audio, such as the portion highlighted by the red box in Fig. 6, become more and more blurred, which means that the important information in the audio is flattened. The update process with higher dropout rates is actually unstable and requires many iterations. Furthermore, the resulting AEs have more noticeable command traces when perceived by human ears. Therefore, we set the dropout rate to 0.1, which improves the attack success rate while avoiding too many iterations. Exploring the effect of dropout rate on AE transferability (Xie et al. 2021a) also shows that setting the dropout rate between 0 and

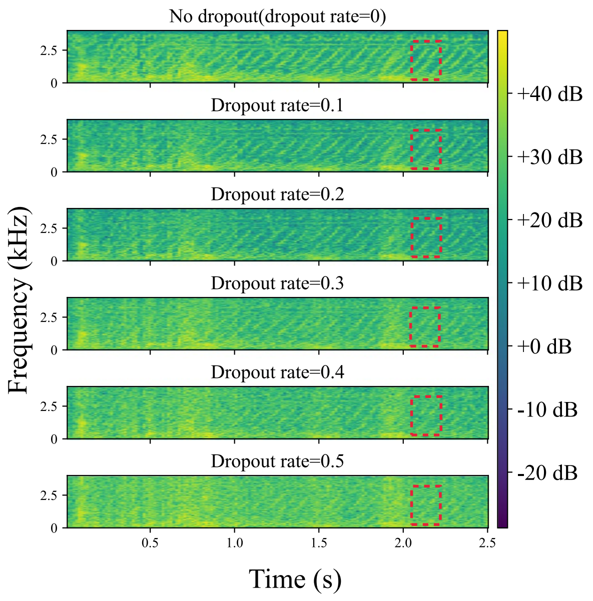

Fig. 6 Spectrograms with different dropout rate

0.2 achieves the best performance for white-box attack and black-box transfer attacks.

Physical attack

In the physical world, which is a more realistic and meaningful attack scenario, we aim to enhance the robustness of CommanderUAP by introducing noise and dropout during the generation process. Our attack targets are IVCs or voice assistants. Since it is difficult to determine which devices or services are Kaldi-based, we follow the approach presented in Yuan et al. (2018): we play the AEs through a speaker and simultaneously record the sound using a microphone. The recorded audio is then sent to the Kaldi platform for recognition. In this way, the AEs will be affected by environmental noise, energy loss, and echo in the room, which is consistent with the environmental factors in other papers (Chen et al. 2020; Guo et al. 2022) that directly target IVC devices for physical attacks.

We use B &O speakers for playback and Audio-Technica microphones for recording, with a distance of 0.5 ms between them. For each command, we selected 25 AEs as test objects, and each audio was played and recorded 5 times. If at least two out of the five recordings of an AE are recognized as the target command, we consider it a successful physical attack. Table 2 shows the replay physical attack experiment results for 10 commands.

The results indicate that target commands with high success rates in white-box online attacks also maintain relatively high success rates in the physical world. For instance, CommanderUAP with "take a picture" as the

Table 2 Physical attack results

Target command	SRoA (%)	Decrease (%)
"take a picture"	80	4
"shut down now"	56	15
"turn on GPS"	28	30
"lock phone"	40	16
"open the front door"	56	3
"make a credit card payment"	64	5
"turn off the light"	36	27
"read mail"	32	23
"airplane mode on"	60	2
"where is my car"	56	13
Average	50.80	13.80

target command can achieve an 80% deception success rate in the physical world. The average SRoA of 10 commands is 50.8%. It is expected that the SRoA of physical world attacks is lower than those of online white-box attacks. Despite we use of methods such as injecting random noise to improve the robustness of perturbations, it remains challenging to completely eliminate the complex environmental interferences.

Transferable black-box attack to APIs

In addition to white-box attacks and physical world attacks, we also explored the performance of CommanderUAP when transferring to commercial ASR APIs. After a survey of commonly used speech service platforms, we chose iFLYTEK (https://www.xfyun.cn/services/voicedictation), Alibaba (https://www.alibabacloud.com/zh/product/intelligent-speech-interaction), Baidu (https://ai.baidu.com/tech/speech), and Tencent (https://cloud.tencent.com/document/ product/1093) as the attack targets for transfer attacks. Thanks to the maturity of deep learning in ASR, the above APIs have extremely high recognition accuracy. Although we have no way of knowing the model and training dataset behind these APIs, there must be different recognition preferences between models. For UAP, which is a special distribution of data, the performance of different APIs may exhibit greater differences.

Transferring AEs to attack unknown models is a challenging task. By coping and repeating the entire sequence and appending 0 and 91 (which are the pdf-id extracted from silent audio clips) at the end to keep the same length as the benign audio, we constructed 5 pdf-id target sequences of the required length by repeating commands and adding silent segments. The corresponding commands are shown in the first column of Table 3. In terms of parameters for the transferability experiment,

Table 3 Transfer Attack results to APIs

Target command * Number of repetitions	SNR	White-box attack to Kaldi (%)	iFLYTEC (%)	Baidu (%)	Tencent (%)	Alibaba (%)
"open the front door"* 2	0.13	36	68	45	0	4
"good night"* 3	0.01	33	70	50	0	14
"take a picture"* 3	− 0.01	27	79	44	4	71
"turn on the light"* 2	0.96	36	66	70	59	5
"where is my car"* 3	0.79	25	50	34	3	29
Average	0.62	31.4	66.6	48.6	13.2	24.6

we reduced the coefficient of the auditory penalty term to 2*e-9, fixed the weight decay coefficient at 1, and started iterating the next batch or the next epoch when the average loss in the current batch was less than 60. Besides, we increased the dropout rate to 0.3, trying to enhance the robustness. In summary, we alleviated the control of the imperceptibility of the perturbation in the program in various ways, enhanced the random exploration of the perturbation in the optimization process, and significantly improved the SRoA of the modified CommanderUAP when transferring to different commercial APIs.

Table 3 shows the success rates of these AEs in transferring attacks against the four commercial APIs. Since the target commands are repeated two or three times, if an audio recognition result from the API contains the target command at least once, we consider it a successful attack against that API. The analysis reveals that the modified CommanderUAP can attack commercial APIs, but the success rates are unstable. The transfer attack on iFlytek (https://www.xfyun.cn/services/voicedictation) yields the best results, with an average SRoA of 66.6%, followed by an attack on Baidu (https://ai.baidu.com/tech/speech) with 48.6%. We infer that the model used by iFlytek API is likely related to the Kaldi platform, making it more susceptible to deception by our UAP. This observation aligns with the findings in Yuan et al. (2018).

The attack performance on Tencent (https://cloud.tencent.com/document/ product/1093) and Ali (https://www.alibabacloud.com/zh/product/intelligent-speech-interaction) fluctuates greatly. The success rate varies on different commands and models, which is essentially still due to the difficulty in estimating model preferences. AEs contain certain features related to the data distribution, and different models may exhibit varying sensitivities to that. A carefully crafted perturbation can cause *model A* to recognize an AE as the target command, but the impact of the perturbation on *model B* with a different decision space remains unknown. One reason why the Tencent API is hard to attack by transfer is that the model may have some defense mechanisms in place, making it more robust to CommanderUAP. Another reason is that, according to our tests, the API often labels noisy audio as "null", so most of the AEs end up with this result.

Another strange phenomenon is that CommanderUAP generated by the modified method has a certain transfer attack ability, but the SRoA for the target model Kaldi Aspire has dropped significantly. As can be seen from Table 3, the average SRoA of 5 commands on Aspire is only 31.4%. Although these AEs are still constructed using Kaldi Aspire, the improvement of cross-model generalization ability affects the attack effect on it.

Human perception

To measure the perceptibility of audio adversarial perturbations, it is more reasonable to investigate whether human ears can detect anomalies under realistic conditions. We recruited 50 native English speakers as listeners from Prolific,[1] an online marketplace for crowdsourcing platforms, to conduct a survey. We added five CommanderUAPs with different commands as targets to different normal audio carriers and placed them in the survey questionnaire. Based on the characteristics of the example, we designed the following questions: 1) Compared to the normal clean audio, what was added to this audio (the four options are noise, current sound, another person's speech, nothing was added); 2) If noise or current sound was chosen in 1), please give the perturbation intensity level (the option range is set to between 1 and 5, the higher the level, the louder the noise); 3) If another person's speech was chosen in 1), please write down what he said; 4) If you had a conversation with someone under this audio condition, would you accept it. The analysis of the survey results is as follows:

Table 4 shows the auditory results of our white-box CommanderUAP obtained from 50 listeners. It is evident that for all auditory survey questions, the listeners perceived the additional components in the example. Among them, 80% and 18.8% of them identified the presence of electric current sound and noise, respectively, and the average disturbance level given by these listeners was

[1] https://www.prolific.co/.

Table 4 Results of the human perception evaluation

Target command	Noise (%)	Electric current sound (%)	Average perturbation level	Another person's voice (%)	No extra added (%)
"take a picture"	22	78	3.22	0	0
"turn off the light"	22	78	3.56	0	0
"shut down now"	18	80	4.00	2	0
"open the front door"	16	82	3.69	2	0
"airplane mode on"	16	82	3.98	2	0
Average	18.8	80.0	3.69	1.2	0

Table 5 The ablation study of noise-pdf-id setting in Stage 1

How to Set noise-pdf-id Sequence	Iteration Times	SRoA on Training Set	SRoA on Testing Set	SNR
Fixed noise-pdf-id	1609	100%	79%	2.22
Dynamically Extract noise-pdf-id	3797	98%	96%	2.64

3.69. Only in 1.2% of the cases did the listeners believe that there was another person's voice in the AEs, but even after repeated listening, they could not correctly write down any word from the target command.

2% of the listeners believed that the interference in the audio had no effect on daily conversation, 50% of the users believed there was interference but could still hear each other clearly, and 48% found such conditions unacceptable. Finally, we surveyed the frequency of users using IVC devices or voice assistants. The results showed that 28% of users use them every day, approximately 24% use them once a week, 8% use them once a month, and 40% of the users stated that they rarely or never use such devices or assistants.

In summary, by balancing the SRoA and imperceptibility and weakening the perceptibility of the perturbation from two angles, the command in the disturbance is indeed difficult for the human ear to perceive, but there is still obvious noise interference. In the following work, the main challenge is to further reduce the perceptibility of the perturbations to make them completely undetectable by humans. We provide a link[2] to our survey questionnaire. While the questionnaire is no longer accepting submissions, readers can still listen to the differences between our five universal adversarial examples and their corresponding normal examples in the survey questionnaire.

Ablation study

Our methodology comprises multiple constituent methods. We now proceed with an ablative analysis of certain methods. Section "White-box attack" and Fig. 6 have already delved into the impact of varying the dropout rate on experimental results. It is evident that a value of 0.1 leads to a significant improvement compared to a value of 0 (i.e., the absence of the dropout method). Besides, without staging but directly using the target command's pdf-id to generate perturbations, the process is similar to Sections "Transfer Attack on API" and "Regulating the imperceptibility of CommanderUAP". It is straightforward to generate perturbations with high SRoA. However, the auditory imperceptibility of the samples is very poor, making human ears clearly perceive the target command. Therefore, both the Dropout and two-stage methods hold value.

To explore the difference between using a fixed noise-pdf-id sequence and a dynamically extracted noise-pdf-id sequence as the target in stage 1, we conducted experimental tests, and the results are shown in Table 5. Except for the selection method of the noise-pdf-id sequence, other parameter settings are the same. If we keep it constant, the resulting BaseUAP would exhibit a 100% SRoA on the training set, but it would only achieve a 79% on the test set, which is considerably inferior to the dynamic extraction method.

Batch training and weighted gradient are crucial foundations of the CommanderUAP. The division of training set samples and the setting of batch size affect the experimental results. With "take a picture" as the target command, we tested these factors in the second stage. Setting the batch size to 1 is equivalent to not using

[2] https://jinshuju.net/f/kVsh2p.

Table 6 The ablation study of batch training and batch size in Stage 2

Batch size/training set size	Iteration times	SRoA on testing set	SNR
1/50	5893	67%	− 1.32
5/50	1237	85%	− 0.26
10/50	532	84%	0.15
10/100	1094	80%	− 0.09
20/100	500	80%	0.19

Table 7 The ablation study of imperceptibility control in Stage 2

How to control imperceptibility	Iteration times	SRoA on testing set	SNR
Only in time domain	500	91%	− 1.22
Only in frequency domain	532	84%	0.15
both of them domain	628	56%	0.96

batch training and weighted gradient updates. As shown in Table 6, this setting results in the highest number of iterations, the lowest SRoA and SNR. This is because the perturbation update only depends on one audio in every update, resulting in an unstable optimization path and reduced perturbation performance. Increasing batch size and training set size, the perturbation optimization is more stable. When batch size is set to 5, the training set has 50 audio files, the effect is the best.

As mentioned earlier, based on the characteristics of different stages and the trade-off between perceptibility and SRoA, we employ both time-domain and frequency-domain controls in Stage 1, while only adding frequency-domain auditory curve control in Stage 2. Continuing with the "take a picture" command as the target, the results of the ablation experiment for perceptibility control in Stage 2 are shown in Table 7. Applying only time-domain manipulation achieves a higher SRoA than applying only frequency-domain manipulation, but it compromises the stealthiness significantly, as human ears can easily detect the embedded commands. Applying both time-domain and frequency-domain manipulation leads to a drastic reduction in success rate to 56% and a noticeable increase in iteration times while maintaining a good SNR of 0.96. Therefore, we adopt a balanced strategy in our experiment, that is, to introduce only frequency-domain manipulation in Stage 2. With this setting, the test set SRoA reaches 84%, and the SNR is 0.15. In the human auditory perception survey, we include AEs generated from this experiment. The results indicate that none of the listeners recognized any anomalous commands in the sample.

Defense

Defense mechanisms against audio adversarial attacks have also rapidly evolved (Yuan et al. 2018; Yang et al. 2018; Samizade et al. 2020; Rajaratnam and Kalita 2018; Guo et al. 2023). In this section, to evaluate the performance of CommanderUAP when facing defense mechanisms, we chose two common and proven effective methods for various attacks: Temporal Dependency Detection (TDD) (Yang et al. 2018) and resampling (Yuan et al. 2018). The results demonstrate that although these defense methods can effectively block attacks from CommanderUAP under specific settings, they also indiscriminately affect the model's recognition of normal audio.

Temporal dependency detection

The temporal dependency of audio is an important information dimension in speech data. However, the perturbation added in the AEs may disrupt the temporal information of the original sequence while directing the DNN output. Based on this characteristic, Yang et al. (2018) proposed an efficient method that utilizes temporal dependency detection (TDD) to discriminate AEs and block the attacks of three audio adversarial methods. Given an audio example W, the defender first truncates its initial k portion (such as the first 1/2 of the audio, which is the most commonly used value in Yang et al. (2018)) as the model's input, yielding a recognition result of S_k. The entire audio is then transcribed, and a prefix of the same length as S_k is truncated from the transcription result, denoted as $S_{\{whole,k\}}$. If the audio W is a normal example, then S_k and $S_{\{whole,k\}}$ will maintain high similarity; otherwise, W is likely an AE. The determination of detection success can be simply judged by whether S_k and $S_{\{whole,k\}}$ are strictly identical, or evaluate the area under the curve score by metrics such as WER, character error rates, or longest common prefix. The experimental results of Yang et al. (2018) show that TDD has a detection success rate of up to 100% for CommanderSong (Yuan et al. 2018) and can defend against adaptive attacks.

In our study, we investigated the detection capability of the TDD method for CommanderUAP. We constructed an audio set consisting of two categories: 69 AEs generated in the white-box attack experiments with the target command "turn off the light", and 100 normal audio examples. The AEs were labeled as positives, while the normal examples were labeled as negatives. We recorded the precision and recall of the TDD method for the constructed audio set. Precision represents the proportion of examples identified as positives by the TDD method that actually belong to the category of true AEs. Higher precision indicates fewer cases where normal examples are

Table 8 The defense result of TDD

k-value	Precision (%)	Recall (%)
0.1	40.38	100
0.2	14.81	19.05
0.3	31.58	38.1
0.4	0*	0*
0.5	0	0
0.6	0	0
0.7	2.94	1.59
0.8	3.03	1.59
0.9	2.78	1.59

*A precision of 0% or recall of 0% indicates that there are no "True Positives", meaning that none of the AEs were detected as positive. In this case, the TDD method is ineffective in detecting AEs but may still have instances of misclassifying normal examples

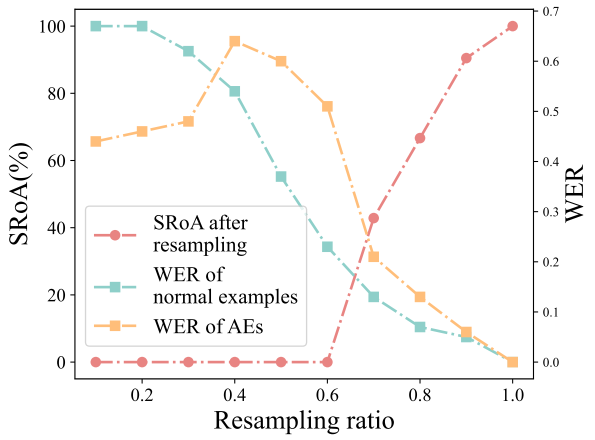

Fig. 7 The defense result of resampling

falsely identified as adversarial. The recall represents the proportion of actual AEs that are detected as positives by the TDD method, with higher recall indicating a higher probability of capturing AEs. An effective detection method should have both high precision and high recall.

For CommanderUAP, the defense performance of the TDD method changes greatly depending on the value of k. Generally, a higher detection success rate can only be achieved when the k value is relatively small. For example, if k=0.1, the recall is 100%, indicating that all AEs are detected. However, as k increases slightly, the recall rapidly drops to around 0%. This phenomenon appears to differ from the results in Yang et al. (2018), where setting k as 1/2 achieved a high detection success rate. In CommanderUAP, the AttackUAP generated in Stage 2 plays a role in implicitly embedding the target command. In each 2.5s AE we tested, the AttackUAP only covers approximately the first 1 s, while the remaining part is only affected by the BaseUAP. Therefore, theoretically, the TDD method will only interrupt the AttackUAP when $k \leq 0.4$. If k is set to a larger value, the perturbations in the extracted audio prefix will include both the entire AttackUAP and a portion of the BaseUAP. In this case, the TDD method fails to detect the AEs. Thus the time dependency of the AttackUAP is easily disrupted, while the BaseUAP is more robust. This is also related to the choice of optimization reference targets in Stage 1 and Stage 2.

On the other hand, from Table 8, it can be observed that the precision of the TDD method consistently performs poorly. When k=0.1, the precision is 40.38%, indicating that among the audio examples identified as adversarial, approximately 59.62% are normal examples misjudged as AEs. Even if the TDD method is very effective in detecting AEs at this point, the recognition of normal examples will be greatly hindered. As the value of k increases, the precision worsens. Therefore, defenders effectively defend CommanderUAP by deploying the TDD method in their models, it will also affect their performance on normal examples.

Resampling

Audio resampling is a common preprocessing method used in defense against adversarial attacks (Yuan et al. 2018; Yang et al. 2018; Hussain et al. 2021). Although it may lead to a decrease in the overall audio quality, it can remove crucial information from the adversarial perturbations. Specifically, the defender can downsample the AEs to a predetermined target frequency and then upsample them back to the original frequency using interpolation.

We filtered out the part of white-box attack AEs that can accurately attack successfully with "turn off the light" as the target command and tested the SRoA of these examples after resampling. The resampling ratio (RR) is used to indicate the ratio between the target frequency after downsampling and the original frequency. The smaller the RR, the lower the target frequency of downsampling, and the greater the difference between the resampled audio with the original audio. Figure 7 shows that when the resampling ratio is less than 0.6, the attack SRoA of the resampled AEs is 0%. We also calculated the average WER of the recognition results for the AEs before and after resampling. It can be observed that the model's recognition results for the AEs are significantly affected by resampling when RR is less than 0.7. At RR=0.4, the WER is 0.64, indicating that the majority of the recognized text has been altered. As RR becomes higher, more information on the example is preserved, so the attack SRoA increases and WER decreases.

Furthermore, we randomly selected 100 normal audio from LibriSpeech, applied resampling, and calculated the average WER of the recognition results before and after resampling. The results are plotted in Fig. 7. When the RR is low, the recognition of normal audio also undergoes significant changes. Overall, normal audios are less affected by resampling compared to AEs. We recommend setting RR to 0.6, as it can effectively disrupt AEs without excessively affecting the recognition of normal audio.

Disscusion

Comparing to other works

Compared to other targeted UAP research in ASR, CommanderUAP possesses improvements in attack ability and expands its application scenarios. Previous works targeted different objective models with different target sentences, and there were also variations in measuring the imperceptibility of perturbations. Below, we compare CommanderUAP with other targeted UAP works.

As mentioned earlier, if no imperceptibility control is applied to the perturbations, even though some studies achieved nearly 100% white-box SRoA on specific models (Zong et al. 2021; Lu et al. 2021), such attacks lack practical feasibility. With clipping the perturbation norms to a reasonable range, Zong et al. (2021) achieved an average SRoA of 54.1%. The white-box attack average SRoA for CommanderUAP is 64.6%, and the results of the auditory survey indicate that none of the listeners perceived that commands different from the original audio have been added to the audio.

Li et al. (2020b) proposed AdvPulse for generating adversarial perturbations in microseconds. The experiments showed that the attack effect of their short-pulse audio started to decay at the third word after the insertion position. This means that although AdvPulse has a short duration, it is not suitable for long benign audio carriers. In our method, since BaseUAP has already masked the original content of the audio, AttackUAP can have a short duration, and together they achieve the effect of launching attacks on long audio.

Guo et al. (2022) achieved online attack success rates of over 90% on 10 commands. If the power of SpecPatch is appropriately amplified, it can also achieve attacks in the physical world, but this will hurt the imperceptibility of the perturbation. SpecPatch is audio-agnostic and is synchronization-free, but the silent patch requires a specific generation for each audio. In addition, like other works on targeted UAP for ASR, (https://cloud.tencent.com/document/ product/1093) did not explore the attack performance of UAP when transferred to non-target models. CommanderUAP is not as stable as SpecPatch, but it has practical universality in both BaseUAP and AttackUAP. It can achieve different degrees of transfer attacks on commercial APIs under relaxed conditions, expanding the threat range.

Unlike the prevalent use of the L2 norms in the image domain to measure the size of perturbations, there are multiple metric scales for speech data. SNR is used by Yuan et al. (2018), Chen et al. (2020), Li et al. (2020b). The decibel difference between the perturbation and the maximum value of the normal audio is used by Carlini and Wagner (2018), Zong et al. (2021), Neekhara et al. (2019): $dB_x(\mu) = dB(\mu) - dB(x)$ where $dB(x) = \max_i 20 \cdot \log_{10}(x_i)$. This is equivalent to calculating the difference in peak signal-to-noise ratio (PSNR). PSNR is determined by the maximum value ($L\infty$ norm) of the perturbation, suitable for estimating and comparing peak signal differences. SNR and PSNR are the most common imperceptibility metrics in speech adversarial attacks, where the former provides a more comprehensive signal quality estimation, and the latter focuses on high-intensity areas. In addition, a few studies such as (Zhang et al. 2017; Abdoli et al. 2019) directly use the Sound Pressure Level (SPL) to control and describe perturbations. SPL is mainly used to describe the intensity and volume of sound and is not directly related to the quality of audio signals.

It should be noted that for works using psychoacoustic masking effects, auditory curve penalties, and other frequency domain operations like (Qin et al. 2019; Schönherr et al. 2018; Guo et al. 2022), it is not appropriate to use the above numerical indicators to measure perturbations. These works typically consider the actual hearing of the human ear, using survey questionnaires to assess whether perturbations can be perceived. As our method controls perturbations in both the frequency and time domains, we use both SNR and auditory perception survey experiments to describe the stealthiness of CommanderUAP.

Dependence on audio duration

We further investigated the dependency of CommanderUAP on the duration of the normal audio which is the carrier of UAP. In the training and testing process, the duration of the normal audio was 2.5 s, and the performance on longer audio is unknown. The optimal method to verify it would be to retrain using a dataset consisting of longer audio examples. But generating BaseUAP for longer audio means that there is more data to optimize, which will bring great optimization difficulty and time consumption. Therefore, we chose a simpler approach by duplicating and extending the BaseUAP data to match the duration of the longer audio. The AttackUAP remained unchanged and was directly added to the beginning of the audio. For example, we repeated the 2.5s BaseUAP twice and added it to a 5 s normal audio segment, then added the AttackUAP to the beginning, thus creating a series of 5 s AEs. Through this simple transformation, we tested the SRoA for the 10 commands listed in Table 1. The results showed that the average SRoA was 43.8% which decreased by 20.8% compared to attacks on the 2.5s. This indicates that CommanderUAP exhibits a certain dependency on the duration of the normal audio. With this simple transformation, CommanderUAP is capable of launching attacks with lower capability on extended audio.

Understanding UAP from a "signal" perspective

To advance research on ASR model security, it is crucial to explain the existence of AEs. Many studies have explored this issue from various perspectives, such as the model's local linear accumulations (Goodfellow et al. 2014), overfitting to normal examples (Szegedy et al. 2013), and learning non-robust features (Ilyas et al. 2019). However, most of the explanations proposed by these works are aimed at SAP, which may not fully apply to UAP. In terms of UAP, Moosavi-Dezfooli et al. (2017) pointed out that there is a subspace in the high-dimensional space of the model that is easily deceived by UAP.

We try to explain the existence and working principle of UAP from the perspective of signal and feature. We think that UAP can be viewed as a "signal", which contains both relevant features of the target command and can resist most normal examples. When such a carefully constructed "signal" is added to normal examples, the relevant features of the target command will dominate the subsequent recognition process, while the normal examples will be considered as "noise". We chose to validate this viewpoint in Stage 2, as it is easier to construct based on BaseUAP than in Stage 1. For the AttackUAP generated in the first epoch of Stage 2, when added to BaseUAP, it always leads to the recognition of the target command, although this process does not meet the definition of an adversarial attack. However, when we deploy the AttackUAP from the first epoch together with BaseUAP on the test set, the attack performance is very poor, with an SRoA of approximately 10%. The attack SRoA will increase and stabilize in subsequent epochs. The reason for this phenomenon is that AttackUAP quickly learns the relevant features of the target command, but its ability to resist different normal examples still requires multiple epochs of training. The training process for UAP effectively optimizes the target feature expression capability and the ability to resist interference from normal examples. The fine-generated UAP can pull normal examples into the high-dimensional subspace mentioned in Moosavi-Dezfooli et al. (2017), making the model's recognition result directed to the target command.

Conclusion

In this paper, we propose the generation method of CommanderUAP, which achieves high success rates for universal adversarial attacks against ASR in white-box, black-box, and physical scenarios. Our evaluation results demonstrate that CommanderUAP can launch attacks on most normal audios, unaffected by their original semantic content. Furthermore, the results of auditory perception surveys indicate that the existence of target commands in the AEs is not apparent. Compared to previous works, CommanderUAP shows improvements in practicality and threat range. We also explore the effectiveness of two defense methods, providing references for building more robust ASR models.

Research description

This paper proposes a staged perturbation generation method that constructs CommanderUAP, which achieves a high success rate of universal adversarial attack against speech recognition models. The evaluation results demonstrate that CommanderUAP can launch attacks on most normal audios in white-box, black-box, and physical scenarios. Moreover, we control the imperceptibility of the perturbation in both time and frequency domains. The results of human perception surveys indicate that the existence of target commands in the adversarial examples is not apparent.

Abbreviations

ASR	Automatic Speech Recognition
SAP	Specific Adversarial Perturbations
UAP	Universal Adversarial Perturbations
DNN	Deep Neural Networks
AE	Adversarial Example
CTC	Connectionist Temporal Classification
IVC	Intelligent Voice Control
MFCC	Mel Frequency Cepstrum Coefficients
HMM	Hidden Markov Model
pdf-id	Posterior density function identifiers
TTS	Text-to-Speech
SRoA	Success Rate of Attack
WER	Word Error Rate
SNR	Signal-to-Noise Ratio
TDD	Temporal Dependency Detection
RR	Resampling Ratio
PSNR	Peak Signal-to-Noise Ratio
SPL	Sound Pressure Level

Acknowledgements
Not applicable.

Author Contributions
ZS, JZ and FG designed the proposed method, completed the code, verified the experiment result, and drafted the manuscript. YC and LJ supervised the findings of this work and reviewed the manuscript. All authors read and approved the final manuscript.

Funding
The authors are supported in part by NSFC No. 62202275 and Shandong-SF No. ZR2022QF012 projects.

Availibility of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 11 August 2023 Accepted: 2 February 2024
Published online: 05 June 2024

References

(2003) Iso 226:2003 acoustics—normal equal-loudness-level contours. https://www.iso.org/standard/34222.html
(2015) iflytek speech-to-text. https://www.xfyun.cn/services/voicedictation
(2022) Alibaba short speech recognition. https://www.alibabacloud.com/zh/product/intelligent-speech-interaction
(2023) Baidu speech-to-text. https://ai.baidu.com/tech/speech
(2023) Tencent short speech recognition. https://cloud.tencent.com/document/ product/1093
Abdoli S, Hafemann LG, Rony J et al (2019) Universal adversarial audio perturbations. arXiv preprint arXiv:1908.03173
Abdullah H, Rahman MS, Garcia W et al (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, pp 712–729
Alzantot M, Balaji B, Srivastava M (2018) Did you hear that? Adversarial examples against automatic speech recognition. arXiv preprint arXiv:1801.00554
Amodei D, Ananthanarayanan S, Anubhai R et al (2016) Deep speech 2: end-to-end speech recognition in English and mandarin. In: International conference on machine learning, PMLR, pp 173–182
Battenberg E, Chen J, Child R et al (2017) Exploring neural transducers for end-to-end speech recognition. In: 2017 IEEE automatic speech recognition and understanding workshop (ASRU), IEEE, pp 206–213
Carlini N, Wagner D (2018) Audio adversarial examples: targeted attacks on speech-to-text. In: 2018 IEEE security and privacy workshops (SPW), IEEE, pp 1–7
Chen Y, Yuan X, Zhang J et al (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), pp 2667–2684
Du T, Ji S, Li J et al (2020) Sirenattack: generating adversarial audio for end-to-end acoustic systems. In: Proceedings of the 15th ACM Asia conference on computer and communications security, pp 357–369
Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572
Graves A, Jaitly N (2014) Towards end-to-end speech recognition with recurrent neural networks. In: International conference on machine learning, PMLR, pp 1764–1772
Guo H, Wang Y, Ivanov N et al (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, pp 1353–1366
Guo F, Sun Z, Chen Y et al (2023) Towards the universal defense for query-based audio adversarial attacks. arXiv preprint arXiv:2304.10088
Hashemi AS, Bär A, Mozaffari S et al (2020) Transferable universal adversarial perturbations using generative models. arXiv preprint arXiv:2010.14919
Hussain S, Neekhara P, Dubnov S et al (2021) {WaveGuard}: understanding and mitigating audio adversarial examples. In: 30th USENIX security symposium (USENIX Security 21), pp 2273–2290
Ilyas A, Santurkar S, Tsipras D et al (2019) Adversarial examples are not bugs, they are features. Advances in neural information processing systems 32
Khare S, Aralikatte R, Mani S (2018) Adversarial black-box attacks on automatic speech recognition systems using multi-objective evolutionary optimization. arXiv preprint arXiv:1811.01312
Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
Li Y, Bai S, Xie C et al (2020a) Regional homogeneity: towards learning transferable universal adversarial perturbations against defenses. In: Computer Vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16. Springer, pp 795–813
Li Z, Wu Y, Liu J et al (2020b) 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, pp 1121–1134
Loshchilov I, Hutter F (2017) Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101
Lu Z, Han W, Zhang Y et al (2021) Exploring targeted universal adversarial perturbations to end-to-end asr models. arXiv preprint arXiv:2104.02757
Moosavi-Dezfooli SM, Fawzi A, Fawzi O et al (2017) Universal adversarial perturbations. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1765–1773
Neekhara P, Hussain S, Pandey P et al (2019) Universal adversarial perturbations for speech recognition systems. arXiv preprint arXiv:1905.03828
Panayotov V, Chen G, Povey D et al (2015) Librispeech: an asr corpus based on public domain audio books. In: 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE, pp 5206–5210
Povey D, Ghoshal A, Boulianne G et al (2011) The kaldi speech recognition toolkit. In: IEEE 2011 workshop on automatic speech recognition and understanding, IEEE Signal Processing Society, CONF
Qin Y, Carlini N, Cottrell G et al (2019) Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In: International conference on machine learning, PMLR, pp 5231–5240
Rajaratnam K, Kalita J (2018) Noise flooding for detecting audio adversarial examples against automatic speech recognition. In: 2018 IEEE international symposium on signal processing and information technology (ISSPIT), IEEE, pp 197–201
Samizade S, Tan ZH, Shen C et al (2020) Adversarial example detection by classification for deep speech recognition. In: ICASSP 2020—2020 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE, pp 3102–3106
Schönherr L, Kohls K, Zeiler S et al (2018) Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. arXiv preprint arXiv:1808.05665
Shen J, Nguyen P, Wu Y et al (2019) Lingvo: a modular and scalable framework for sequence-to-sequence modeling. arXiv preprint arXiv:1902.08295
Szegedy C, Zaremba W, Sutskever I et al (2013) Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199
Taori R, Kamsetty A, Chu B et al (2019) Targeted adversarial examples for black box audio systems. In: 2019 IEEE security and privacy workshops (SPW), IEEE, pp 15–20
Vadillo J, Santana R (2019) Universal adversarial examples in speech command classification. arXiv preprint arXiv:1911.10182
Wang S, Wang X, Zhao P et al (2018) Defensive dropout for hardening deep neural networks under adversarial attacks. In: 2018 IEEE/ACM international conference on computer-aided design (ICCAD), IEEE, pp 1–8
Xie P, Wang L, Qin R et al (2021a) Improving the transferability of adversarial examples with new iteration framework and input dropout. arXiv preprint arXiv:2106.01617
Xie Y, Li Z, Shi C et al (2021b) Enabling fast and universal audio adversarial attack using generative model. In: Proceedings of the AAAI conference on artificial intelligence, pp 14129–14137
Yang Z, Li B, Chen PY et al (2018) Characterizing audio adversarial examples using temporal dependency. arXiv preprint arXiv:1809.10875
Yuan X, Chen Y, Zhao Y et al (2018) {CommanderSong}: a systematic approach for practical adversarial voice recognition. In: 27th USENIX security symposium (USENIX security 18), pp 49–64
Zhang G, Yan C, Ji X et al (2017) Dolphinattack: inaudible voice commands. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 103–117
Zhang C, Benz P, Lin C et al (2021) A survey on universal adversarial attack. arXiv preprint arXiv:2103.01498

Zong W, Chow YW, Susilo W et al (2021) Targeted universal adversarial perturbations for automatic speech recognition. In: Information security: 24th international conference, ISC 2021, Virtual Event, November 10–12, 2021, Proceedings 24, Springer, pp 358–373

Publisher's Note

Zheng Sun is a student who is currently pursuing a master's degree in Cyberspace and Information Security at Shandong University. His research interests include adversarial machine learning and AI security.

Jinxiao Zhao is a student who is currently pursuing a master's degree in Cyberspace and Information Security at Shandong University. Her research interests include adversarial machine learning and AI security.

Feng Guo is a graduate who obtained a master's degree in Cyberspace and Information Security from Shandong University in 2023. His research interests include adversarial machine learning and AI security.

Yuxuan Chen is currently an Associate Professor with the School of Cyber Science and Technology, Shandong University. His research interests include adversarial machine learning and AI security.

Lei Ju is currently a Full Professor with the School of Cyber Science and Technology, Shandong University. His research interests include IoT security and system security.