A Multiversion Programming Inspired Approach to Detecting Audio Adversarial Examples

Qiang Zeng [†], Jianhai Su [†], Chenglong Fu [‡], Golam Kayas [‡], Lannan Luo [†]
[†] University of South Carolina [‡] Temple University

Abstract

Adversarial examples (AEs) are crafted by adding human-imperceptible perturbations to inputs such that a machine-learning based classifier incorrectly labels them. They have become a severe threat to the trustworthiness of machine learning. While AEs in the image domain have been well studied, audio AEs are less investigated. Recently, multiple techniques are proposed to generate audio AEs, which makes countermeasures against them an urgent task. Our experiments show that, given an AE, the transcription results by different Automatic Speech Recognition (ASR) systems differ significantly, as they use different architectures, parameters, and training datasets. Inspired by Multiversion Programming, we propose a novel audio AE detection approach, which utilizes multiple off-the-shelf ASR systems to determine whether an audio input is an AE. The evaluation shows that the detection achieves accuracies over 98.6%.

1 Introduction

Automatic Speech Recognition (ASR) is a system that converts speech to text. ASR has been studied intensively for decades. Many core technologies, such as gaussian mixture models and hidden Markov models, were developed. In particular, recent advances (Yu and Deng 2015) based on deep neural network (DNN) have improved the accuracy significantly. DNN-based speech recognition thus has become the mainstream technique in ASR systems. Companies such as Google, Apple and Amazon have widely adopt DNN-based ASRs for interaction with IoT devices, smart phones and cars. Gartner (InsideRadio 2017) estimates that 75% of American households will have at least one smart voice-enabled speaker by 2020.

Despite the great improvement in accuracy, recent studies (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014) show that DNN is vulnerable to adversarial examples. An *adversarial example* (AE) x' is a mix of a host sample x and a carefully crafted, human-imperceptible perturbation δ such that a DNN will assign different labels to x' and x. Figure 1 presents an audio AE, which sounds as in the text shown on the left to human, but is transcribed by an ASR system into a completely different text as shown on the right.

Several techniques have been proposed to generate audio AEs, and novel attack vectors based on audio AEs have been demonstrated (Yuan et al. 2018). There are two *state-of-the-art* audio AE generation methods: (1) **White-box attacks:** Carlini et al. proposed an optimization based method to convert an audio to an AE that transcribes to an attacker-designated phrase (Carlini and Wagner 2018). It is classified as white-box attacks because the target system's detailed internal architecture and parameters are required to perform the attack. (2) **Black-box attacks:** Taori et al. (Taori et al. 2018) combined the genetic algorithm (Alzantot, Balaji, and Srivastava 2018) and gradient estimation to generate AEs. It does not require the knowledge of the ASR's internal parameters, but imposes a larger perturbation (94.6% similarity on average between an AE and its original audio, compared to 99.9% in (Carlini and Wagner 2018)). The rapid development of audio AE generation methods makes countermeasures against them an urgent and important problem. The goal of our work is to detect audio AEs.

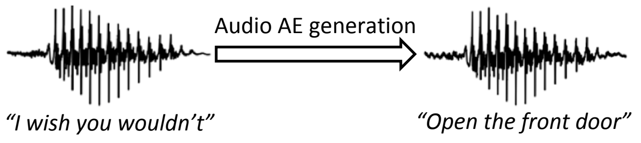


Figure 1: An audio AE.

Existing work on audio AE detection is rare and limited. Yang et al. (Yang et al. 2018) hypothesized that AEs are fragile: given an audio AE, if it is cut into two sections, which are then transcribed section by section, then the phrase by splicing the two sectional results is very different from the result if the AE is transcribed as a whole. However, as admitted by the authors, this method cannot handle "adaptive attacks", which may evade the detection by embedding a malicious command into one section alone. Rajaratnam et al. (Rajaratnam, Shah, and Kalita 2018) proposed detection based on audio pre-processing methods. Yet, if an attacker knows the detection details, he can take the pre-processing effect into account when designing the loss function used for generating AEs; this attack approach has been well demonstrated in (Carlini and Wagner 2017). An effective audio AE detection method that can handle adaptive attacks is missing.

Our key observation is that existing ASRs are diverse with regard to architectures, parameters and training processes. Our hypothesis is that an AE that is effective on one ASR system is highly likely to fail on another, which is verified by our experiments. How to generate transferable audio AEs that can fool multiple heterogenous ASR systems is still an open question (Carlini and Wagner 2018) (discussed in Section 2). This inspires us to borrow the idea of *multiversion programming* (MVP) (Liming Chen 1995), a method in software engineering where multiple functionally equivalent programs are independently developed from the same specification, such that an exploit that compromises one of them is ineffective on other programs. We thus propose to run multiple ASR systems in parallel, and an input is determined as an AE if the ASR systems generate very dissimilar transcription results.

Our contributions are as follows. (1) We empirically investigate the transferability of audio AEs and analyze the reasons behind the poor transferability. (2) We propose a novel audio AE detection approach inspired by multiversion programming, which achieves accuracy rates of over 98.6%. (3) The detection method dramatically reduces the flexibility of the adversary, in that audio AEs cannot succeed unless the host text is highly similar to the malicious command.

2 Transferability

There are two types of AEs: *non-targeted* AEs and *targeted* AEs. A *non-targeted* AE is considered successful as long as it is classified as a label different from the label for the host sample, while a *targeted* AE is successful only if it is classified as a label desired by the attacker. In the context of audio based human-computer interaction, a non-targeted AE is not very useful, as it cannot make the ASR system issue an attacker-desired command. It explains why state-of-the-art audio AE generation methods all generate targeted AEs (Carlini and Wagner 2018; Taori et al. 2018). Thus, our work focuses on targeted audio AEs, although the proposed detection approach should be effective in detecting non-targeted AEs as well.

An intriguing property of AEs in the image domain is the existence of transferable adversarial examples. That is, an image AE crafted to mislead a specific model M can also fool a different model M' (Goodfellow, Shlens, and Szegedy 2014; Szegedy et al. 2013; Papernot et al. 2016). By exploiting this property, Papernot et al. (Papernot, McDaniel, and Goodfellow 2016) proposed a reservoir-sampling based approach to generate transferable non-targeted image AEs and successfully launch black-box attacks against both image classification systems from Amazon and Google. But a recent study (Tramèr et al. 2017) points out that the transferability property does not hold in some scenarios. The experiment shows a failure of AE's transferability between a linear model and a quadratic model. For targeted image AEs, Liu et al. (Liu et al. 2016) proposed an ensemble-based approach to craft AEs that could transfer to ResNet models, VGG and GoogleNet with success rates of 40%, 24% and 11% respectively. Thus, an image AE can fool multiple models (Monteiro, Akhtar, and Falk 2018).

To the best of our knowledge, no systematic approaches are available for generating transferable audio AEs. Carlini et al. (Carlini and Wagner 2018) found that the attack method derived from Fast Gradient Sign Method (Goodfellow, Shlens, and Szegedy 2014) in image domain is not effective for audio AEs because of a large degree of non-linearity in ASRs, and further stated that the transferability of audio AEs is an *open question*.

CommanderSong (Yuan et al. 2018) represents a recent advance in audio AE generation, in that it is able to generate AEs that can fool an ASR system in the presences of background noise. This work also slightly explored transferability of audio AEs (see Section 5.3 of the paper (Yuan et al. 2018)). Specifically, to create AEs that can transfer from "Kaldi to DeepSpeech" (both *Kaldi* and *DeepSpeech* are open source ASR systems, and *Kaldi* is the target ASR system of CommanderSong), a two-iteration recursive AE generation method is described in CommanderSong: an AE generated by CommanderSong, embedding a malicious command c and able to fool *Kaldi*, is used as a host sample in the second iteration of AE generation using the method (Carlini and Wagner 2018), which targets *DeepSpeech* v0.1.0 and embeds the same command c. We followed this two-iteration recursive AE generation method to generate AEs, but our experiment results (Zeng et al. 2018) showed that the generated AEs could only fool *DeepSpeech* but not *Kaldi*. That is, AEs generated using this method are not transferable.

Furthermore, we adapted the two-iteration AE generation method by concatenating the two aforementioned state-of-the-art attack methods (Carlini and Wagner 2018) and (Taori et al. 2018) targeting *DeepSpeech* v0.1.0 and v0.1.1, respectively, expecting to generate AEs that can fool both *DeepSpeech* v0.1.0 and v0.1.1. But none of the generated AEs showed transferability (Zeng et al. 2018).

Moreover, by changing the value of "`--frame-subsampling-factor`" from 1 to 3, which is a parameter configuration of the *Kaldi* model, we derived a variant of *Kaldi*. The AEs generated by CommanderSong did not show transferability on the variant, even given the fact that the variant was only slightly modified from the model targeted by CommanderSong. Here, we clarify that CommanderSong did not claim their AEs could transfer across the *Kaldi* variants.

Based on our detailed literature review and empirical study, we find that so far there are no systematic AE generation methods that can generate transferable audio AEs effective across two ASR systems, not to mention three or more. This is consistent with the statement by Carlini et al. (Carlini and Wagner 2018) that transferability of audio AEs is an open question,

3 Diverse ASRs

Yu and Li (Yu and Li 2017) summarized the recent progress in deep-learning based ASR acoustic models, where both recurrent neural network (RNN) and convolutional neural network (CNN) come to play as parts of deep neural networks. Standard RNN could capture sequence history in its internal states, but can only be effective for short-range sequence due to its intrinsic issue of exploding and vanishing gradients. This issue is resolved by the introduction of long short-term memory (LSTM) RNN (Hochreiter and Schmidhuber 1997), which outperforms RNNs on a variety of ASR tasks (Graves, Mohamed, and Hinton 2013; Sak, Senior, and Beaufays 2014; Li and Wu 2014). As to

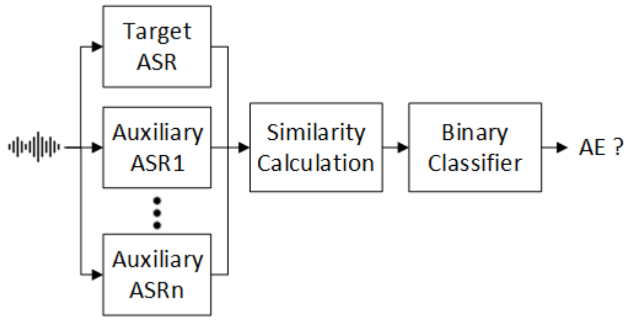


Figure 2: Overview of the proposed detection system.

CNNs, its inherent translational invariability facilitates the exploitation of variable-length contextual information in input speech. The first CNN model proposed for ASRs is time delay neural network (Lang, Waibel, and Hinton 1990) that applies multiple CNN layers. Later, several studies (Abdel-Hamid, Deng, and Yu 2013; Tóth 2015; Sercu and Goel 2016) combine CNN and Hidden-Markov Model to create hybrid models that are more robust against vocal-tract-length variability between different speakers. But there is *no* single uniform structure used across all ASRs.

In addition to several well-known ASR systems, multiple companies have independently developed their own ASRs, such as *Google Now*, *Apple Siri* and *Microsoft Cortana*. In our system, we use *DeepSpeech*, *Google Cloud Speech*, and *Amazon Transcribe*. *DeepSpeech* is an end-to-end speech recognition software open-sourced by Mozilla. Its first version (Hannun et al. 2014) uses a five-layers neural network where the fourth layer is a RNN layer, while the second version (Amodei et al. 2016) contains a mix of CNN and RNN layers. Our experiments used the first version, as the second version was not publicly available at the time of our experiment. Unlike *DeepSpeech*, the DNN behind Google Cloud Speech is a LSTM-based RNN according to the source (Google AI Blog 2015). Each memory block in Google's LSTM network (Sak, Senior, and Beaufays 2014) is a standard LSTM memory block with an added recurrent projection layer. This design enables the model's memory to be increased independently from the output layer and recurrent connections. No public information about the internal details of *Amazon Transcribe* is available.

In short, existing ASR systems are diverse with regard to architectures, parameters and training datasets. Compared to opensourced systems, propriety ASR systems provide little information that can be exploited by attackers. Given the diversity of ASR systems (and proprietary networks), it is unclear how to propose a generic AE generation method that can simultaneously mislead all of them.

4 System Design

In software engineering, *multi-version programming* (MVP) is an approach that independently develops multiple software programs based on the same specification, such that an exploit that compromises one program probably fails on other programs. This inspires us to propose a system design that runs multiple ASRs in parallel to detect audio AEs as shown in Figure 2, where the *target* ASR is the system targeted by the adversary (e.g., the speech recognition system at a smart home) and *auxiliary* ASRs are models different from the target ASR.

Table 1: Recognition results of an AE by multiple ASRs: the host transcription is "*I wish you wouldn't*", while the embedded text is "*a sight for sore eyes*".

ASR	Transcribed Text
DeepSpeech v0.1.0	*A sight for sore eyes*
DeepSpeech v0.1.1	*I wish you live*
Google Cloud Speech	*I wish you wouldn't.*
Amazon Transcribe	*I wish you wouldn't.*

The intuition behind this design is that different ASRs can be regarded as "independently developed programs" in MVP. Since they follow the same specification, that is, to covert audios into texts, given a benign sample, they should output very similar recognition results. On the other hand, an audio AE can be regarded as an "exploit", and cannot fool all ASRs as verified in Section 2. Thus, the recognition result by the target ASR differs *significantly* from those by the auxiliary ASRs. Table 1 shows a typical example, which can only fool one ASR and fails on others.

The system works as follows: 1) the target and auxiliary ASRs simultaneously conduct speech-to-text conversion, 2) then the transcriptions are used to calculate similarity score(s), which are passed into a binary classifier to determine whether the input audio is adversarial.

Similarity scores are calculated in two steps. First, each transcription is converted into its phonetic-encoding representation. Phonetic encoding converts a word to the representation of its pronunciation (Phonetic-Encoding 2017). This helps handle variations between ASRs, as they may output different words for similar pronunciations. The validity of using phonetic encoding will be demonstrated in Section 5.4. Second, for each auxiliary ASR, a similarity score is calculated to measure the similarity between the text recognized by the target ASR and that by the auxiliary ASR. We adopt the Jaro-Winkler distance method (Jaro-Winkler Distance 2018) to calculate the similarity score (see Section 5.4). It gives a value between 0 and 1, where 0 indicates totally dissimilar and 1 very similar.

5 Evaluation

We evaluate our system on its accuracy and robustness. We first describe the experiment setup (Section 5.1) and discuss the dataset used in our evaluation (Section 5.2). Next, we investigate the feasibility of our idea (Section 5.3), and examine different methods for calculating similarity scores and select the one that provides the best results (Section 5.4). After that, we evaluate the accuracy of our system when one auxiliary ASR is used (Section 5.5) and more than one auxiliary ASR is used (Section 5.6). Finally, we evaluate the robustness of our system against AEs generated by unseen attack methods (Section 5.7)

Table 2: Datasets.

Dataset Name		# of Samples
Benign		1125
AE	White-box AEs	1025
	Black-box AEs	100

5.1 Experimental Settings

Our experiments were performed on a 64-bit Linux machine with an Intel(R) Core(TM) i9-7980XE CPU @ 2.60GHz, NVIDIA GeForce GTX 1080 Ti, and 32GB DDR4-RAM.

Target Model. We select *DeepSpeech v0.1.0*, called `DS0` (DeepSpeech Github Repository 2018) as the target model. The main reason we select `DS0` is that its model architecture and parameters are publicly available—making it possible to generate white-box AEs (Taori et al. 2018). But our system should also work if any other ASR model is selected as the target model.

Auxiliary Models. There are three auxiliary models: (1) *Google Cloud Speech*, called `GCS` (Google Cloud Speech 2018), (2) *Amazon Transcribe*, called `AT` (Amazon Transcribe 2018), and (3) *DeepSpeech v0.1.1*, called `DS1` (DeepSpeech Github Repository 2018). The first two auxiliary ASRs are on-line services, while the last one runs locally with an officially pre-trained model.

We use $\mathtt{X}+\{\mathtt{Y}_1, ..., \mathtt{Y}_n\}$ to denote a system using `X` as the target model and $\mathtt{Y}_i$ ($\forall i \in \{1, ..., n\}$) as the auxiliary models. When only one auxiliary model is included, $n = 1$.

5.2 Dataset Preparation

We consider two audio AE generation techniques: white-box based (Carlini and Wagner 2018) and black-box based (Taori et al. 2018) methods, and build two datasets: a *Benign* dataset and an *AE* dataset, each of which contains 1125 audio samples, as shown in Table 2. The audio samples of the *Benign* dataset are randomly selected from the *dev_clean* dataset of LibriSpeech (LibriSpeech 2015).

The *AE* dataset is composed of the following two parts: (1) 1025 white-box AEs, including 990 AEs provided by (Carlini and Wagner 2018) and 35 ones created by us; (2) 100 black-box AEs constructed by selecting the first 100 audio files in *Common Voice* dataset (Common Voice Dataset 2018) as host speeches, each of which is converted into an AE by applying the black-box approach (Taori et al. 2018), such that the AE's transcription contains only two words. All the AEs can successfully fool the target model `DS0`.

For each dataset, 80% of its samples are used for training, and the remaining 20% for testing.

5.3 Feasibility Analysis

To detect whether an audio is an AE, the basic idea is to compare its transcriptions generated from different ASRs. Our intuition is that the transcriptions generated from different ASRs for a benign sample should be similar. On the contrary, if the transcriptions generated from different ASRs are dissimilar, the input audio is likely to be adversarial.

To this end, for each audio sample in the training dataset, we use the *phonetic encoding* technique and *Jarro-Winkler* distance method to calculate the similarity score between the transcriptions generated by `DS0` and one of the three auxiliary models (i.e., `DS1`, `GCS` and `AT`). Figure 3 confirms our intuition visually by comparing the similarity scores for benign samples and AEs—the similarity scores for AEs and those for benign samples form two distinguishable clusters.

5.4 Comparison of Different Similarity Measurement Methods

Many methods can calculate the similarity score of two strings, such as *Jaccard index* (Jaccard Index 2018), *Cosine similarity*, and *edit distance* (e.g., *Jaro-Winkler* (Jaro-Winkler Distance 2018)).

To analyze a transcription, some previous works first apply the phonetic encoding technique to convert the transcription to its phonetic-encoding representation (where each word is converted to the representation of its pronunciation), and then perform further analysis on the phonetic-encoding representation (Phonetic-Encoding 2017). Thus, here we come with three other methods to measure the similarity of two strings, denoted as *PE_Jaccard*, *PE_Consine* and *PE_JaroWinkler*, representing that the phonetic encoding technique is first applied on each transcription and then *Jaccard index*, *Cosine similarity* and *Jaro-Winkler* are applied later to computing the similarity score of two transcriptions, respectively.

Each system uses a SVM classifier. We examine the aforementioned six methods and evaluate the system performance with respect to each of them. The results are shown in Table 3. It can be observed that when *PE_JaroWinkler* is adopted, all the systems achieve the best performance: for example, the detection accuracy of the system $\mathtt{DS0}+\{\mathtt{DS1}, \mathtt{AT}\}$—using `DS0` as the target model and both `DS1` and `AT` as the auxiliary models—is 99.78% when *PE_JaroWinkler* is used. Thus, in our later experiments *PE_JaroWinkler* is adopted to measure the similarity between transcriptions.

5.5 Single-Auxiliary-Model System

A *single-auxiliary-model* system contains a target model and *only one* auxiliary model. As three ASRs are selected as the auxiliary model, we have three different single-auxiliary-model system, denoted as $\mathtt{DS0}+\{\mathtt{DS1}\}$, $\mathtt{DS0}+\{\mathtt{GCS}\}$, and $\mathtt{DS0}+\{\mathtt{AT}\}$, using `DS1`, `GCS`, and `AT` as the auxiliary model, respectively.

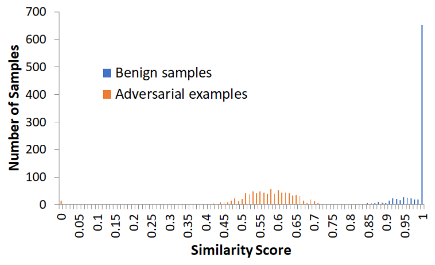
A single-auxiliary-model system computes a similarity score for an input audio. If the similarity score is lower than a threshold T, the corresponding audio is detected as adversarial, otherwise benign. We use the Receiver Operating Characteristic (ROC) curve to select the threshold T, such that the sum of TPR (true positive rate) and TNR (true negative rate) is maximized.

The training results are shown in Table 4 and the corresponding ROC curves are plotted in Figure 4.

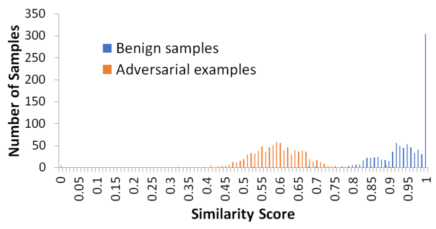
We show testing results in Table 5. All of the three single-auxiliary-model systems achieve TPR higher than 98.6% and FPR (false positive rate) lower than 1.34%. We further investigate the results, and have the following interesting findings: (1) all 100 black-box AEs are correctly detected;

Table 3: Comparison of 6 similarity metrics. (Legend: X+{Y_1, ..., Y_n}, stands for a system using X as the target model and Y_i ($\forall i \in \{1, ..., n\}$) as the auxiliary models.)

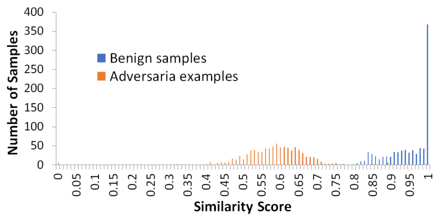
Similarity Metric	Performance	System			
		DS0+{DS1, GCS}	DS0+{DS1, AT}	DS0+{GCS, AT}	DS0+{DS1, GCS, AT}
Jaccard	Accuracy	447/450 (99.33%)	447/450 (99.33%)	435/450 (96.67%)	448/450 (99.56%)
	FPR	3/225 (1.33%)	3/225 (1.33%)	15/225 (6.67%)	2/225 (0.89%)
	FNR	0/225 (0.00%)	0/225 (0.00%)	0/225 (0.00%)	0/225 (0.00%)
Consine	Accuracy	448/450 (99.56%)	448/450 (99.56%)	444/450 (98.67%)	448/450 (99.56%)
	FPR	2/225 (0.89%)	2/225 (0.89%)	6/225 (2.67%)	2/225 (0.89%)
	FNR	0/225 (0.00%)	0/225 (0.00%)	0/225 (0.00%)	0/225 (0.00%)
JaroWinkler	Accuracy	447/450 (99.33%)	447/450 (99.33%)	444/450 (98.67%)	446/450 (99.11%)
	FPR	3/225 (1.33%)	3/225 (1.33%)	5/225 (2.22%)	4/225 (1.78%)
	FNR	0/225 (0.00%)	0/225 (0.00%)	1/225 (0.44%)	0/225 (0.00%)
PE_Jaccard	Accuracy	447/450 (99.33%)	448/450 (99.56%)	445/450 (98.89%)	447/450 (99.3%)
	FPR	3/225 (1.33%)	2/225 (0.89%)	5/225 (2.22%)	3/225 (1.33%)
	FNR	0/225 (0.00%)	0/225 (0.00%)	0/225 (0.00%)	0/225 (0.00%)
PE_Consine	Accuracy	448/450 (99.56%)	448/450 (99.56%)	443/450 (98.44%)	447/450 (99.33%)
	FPR	2/225 (0.89%)	2/225 (0.89%)	6/225 (2.67%)	3/225 (1.33%)
	FNR	0/225 (0.00%)	0/225 (0.00%)	1/225 (0.44%)	0/225 (0.00%)
PE_JaroWinkler	Accuracy	449/450 (**99.78%**)	449/450 (**99.78%**)	448/450 (**99.56%**)	449/450 (**99.78%**)
	FPR	1/225 (0.44%)	0/225 (0.00%)	1/225 (0.44%)	1/225 (0.44%)
	FNR	0/225 (0.44%)	1/225 (0.44%)	1/225 (0.44%)	0/225 (0.00%)



(a) DS0+{DS1}

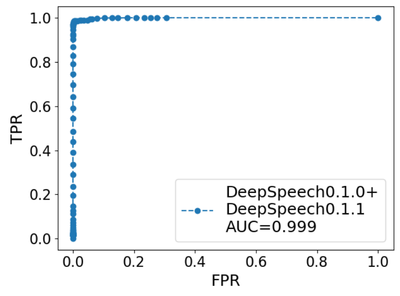


(b) DS0+{GCS}

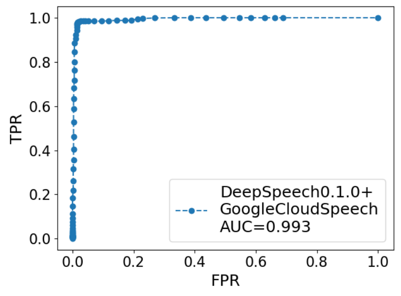


(c) DS0+{AT}

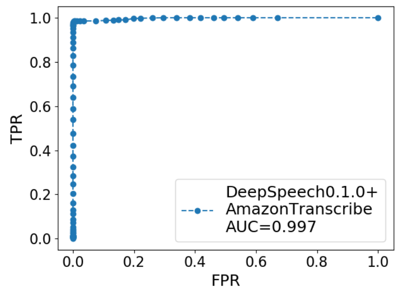
Figure 3: Similarity scores of transcriptions generated by DS0 and one of the three ASRs, DS1, GCS and AT.



(a) DS0+{DS1}



(b) DS0+{GCS}



(c) DS0+{AT}

Figure 4: The ROC curves of three single-auxiliary-model systems.

and (2) for one FP (false positive) case, the transcriptions recognized by DS0 and GCS have the same meaning, but are in different text formats. The recognition of DS0 is the phrase "*SEVENTEEN SEVENTEEN*", while that of GCS is the digit "*1717*". The reason is that the alphabet model of DS0 does not include digits; as a result, any digit is represented as a word. However, if the alphabet model is integrated into DS0, such a FP case could be eliminated.

5.6 Multiple-Auxiliary-Models System

A *multiple-auxiliary-model* system contains a target model and *more than one* auxiliary models. We designe four different multiple-auxiliary-model systems, denoted as DS0+{DS1, GCS, AT}, DS0+{DS1, GCS}, DS0+{DS1,

Table 4: The training results of three single-auxiliary-model systems. TP, FN, TN and FP represent True Positive, False Negative, True Negative, and False Positive, respectively.

System	Threshold	# of TPs	# of FNs	# of TNs	# of FPs	TPR	FPR	AUC
DS0+{DS1}	0.78	885	15	898	2	98.33%	0.22%	0.9990
DS0+{GCS}	0.75	881	19	885	15	97.89%	1.67%	0.9930
DS0+{AT}	0.77	885	15	898	2	98.33%	0.22%	0.9973

Table 5: The testing results of three single-auxiliary-model systems.

System	Threshold	# of TPs	# of FNs	# of TNs	# of FPs	TPR	FPR	Accuracy
DS0+{DS1}	0.78	223	2	224	1	99.11%	0.44%	99.33%
DS0+{GCS}	0.75	222	3	222	3	98.67%	1.33%	98.67%
DS0+{AT}	0.77	224	1	225	0	99.56%	0.00%	99.78%

Table 6: The testing results of four multiple-auxiliary-model systems when different binary classifiers are adopted.

Classifier	Performance	System			
		DS0+{DS1, GCS}	DS0+{DS1, AT}	DS0+{GCS, AT}	DS0+{DS1, GCS, AT}
SVM	Accuracy	449/450 (99.78%)	449/450 (99.78%)	448/450 (99.56%)	449/450 (99.78%)
	FPR	1/225 (0.44%)	0/225 (0.00%)	1/225 (0.44%)	1/225 (0.44%)
	FNR	0/225 (0.00%)	1/225 (0.44%)	1/225 (0.44%)	0/225 (0.00%)
KNN	Accuracy	449/450 (99.78%)	447/450 (99.33%)	448/450 (99.56%)	450/450 (100%)
	FPR	1/225 (0.44%)	1/225 (0.44%)	1/225 (0.44%)	0/225 (0.00%)
	FNR	0/225 (0.00%)	2/225 (0.89%)	1/225 (0.44%)	0/225 (0.00%)
Random Forest	Accuracy	448/450 (99.56%)	449/450 (99.78%)	449/450 (99.78%)	450/450 (100%)
	FPR	0/225 (0.00%)	0/225 (0.00%)	0/225 (0.00%)	0/225 (0.00%)
	FNR	2/225 (0.89%)	1/225 (0.44%)	1/225 (0.44%)	0/225 (0.00%)

AT}, and DS0+{GCS, AT}, each of which use DS0 as the target model, and the other included ones as the auxiliary models. For example, DS0+{DS1, GCS} has two auxiliary models, DS1 and GCS.

For an multiple-auxiliary-model system with n auxiliary models, n similarity scores are computed for a given input audio, which are grouped together as a feature vector. The feature vector is then fed into a binary classifier (e.g., SVM) to predict whether the input audio is benign or adversarial.

For each multiple-auxiliary-models system, we train a binary classifier on all the 1800 feature vectors (900 benign samples and 900 AE samples), and test it on over 450 test samples. We use three different binary classifiers, including SVM, KNN and Random Forest, and configured each classifier as follows: (1) SVM use a 3-degree polynomial kernel; (2) KNN use 10 neighbors to vote; and (3) Random Forest use a seed of 200 as starting random state.

Table 6 shows the results. All the accuracy results are higher than 99%, and FPR and FNR are lower than 0.5%, regardless of auxiliary models and binary classifiers. The results also show that the three-auxiliary-models system outperforms the two-auxiliary-models systems, probably due to more information learned by the three-auxiliary-models system. As the detection accuracy of the three-auxiliary-models system is already 100% for KNN and Random Forest, we stop at three and did not include more auxiliary models.

Table 7: The detection results of unseen-attack AEs for three single-auxiliary-models.

System	Threshold	FPR	FNs	FNR	Defense rate
DS0+{DS1}	0.88	4.44%	10	0.89%	99.11%
DS0+{GCS}	0.81	4.53%	15	1.33%	98.67%
DS0+{AT}	0.83	3.73%	14	1.24%	98.76%

Table 8: The detection results of unseen-attack AEs for four multiple-auxiliary-models.

System	Defense rate	
	Black-box AEs	White-box AEs
DS0+{DS1, GCS}	100.00%	98.63%
DS0+{DS1, AT}	100.00%	98.63%
DS0+{GCS, AT}	100.00%	98.34%
DS0+{DS1, GCS, AT}	100.00%	98.54%

5.7 Robustness against Unseen Attack Methods

The last experiment aims to examine whether a system trained on AEs generated by a particular attack method is able to detect AEs generated by other kinds of attack methods—such an AE is called an *unseen-attack AE*. We use the *defense rate*, defined as the ratio of the number of successfully detected AEs among the total number of AEs, to measure the robustness. The higher the defense rate, the better robustness of the system.

Single-auxiliary-model systems. We first examine the three single-auxiliary-model systems. We train each system using only the benign samples, and test each one on all the 1125 AEs (see Table 2). All the AEs can be considered as unseen-attack AEs. The results are presented in Table 7.

For each system, the threshold is determined by maximizing FPR under an upper-bounded constraint, which is set as 5%. We can see that the lowest defense rate is 98.67% for the DS0+{GCS} system. We further try different upper bound values for FPR, including 4%, 3% and 2%. The defense rate slightly drops with 2% as the upper-bound value, and the DS0+{GCS} system has the lowest defense rate of 98.49%.

Multiple-auxiliary-model systems. We next examine the four multiple-auxiliary-model systems. As presented in Section 5.2, two different methods are used to generate the AEs: the white-box approach and black-box approach. There are totally 100 black-box AEs and 1025 white-box AEs.

We conduct two different experiments to evaluate each of the four multiple-auxiliary-model systems. (1) We first use all the 1025 white-box AEs and 1025 benign samples to train each system, and use all the black-box AEs to test each trained system. The results are showed in the second column in Table 8. It can be observed that the defense rates of all the systems are 100%—all the black-box AEs can be successfully detected. (2) We next use all the 100 black-box AEs and 100 benign samples to train each system, and use all the white-box AEs to test each trained system. The results are showed in the third column in Table 8. We can see that all the systems perform very well, and the lowest defense rate is 98.34% for the DS0+{GCS, AT} system.

Therefore, we can conclude that our detection method is very robust against unseen-attack AEs.

6 Related Work

The emergence of adversarial examples that exploit the vulnerability of DNNs has attracted researchers to study its defense strategies. Many studies on detecting image AEs have been reported (Carlini and Wagner 2017; Zuo, Luo, and Zeng 2018), while only a few are presented to cope with audio AEs, probably because techniques for crafting audio AEs just surfaced in the past two years. Carlini et al. (Carlini et al. 2016) trained a logistic regression classifier with a mix of benign and Hidden-Voice-Command (HVC) audios. But the classifier requires knowing details of HVC generation technology to create so-called mid-term features based on short-term features (Carlini et al. 2016). Yang et al. (Yang et al. 2018) introduced a new idea to identify audio AEs based on the assumption that audio AEs need complete audio information to resolve temporal dependence. An audio input is first cut into two sections, which are then transcribed separately; if the input is an AE, the result obtained by splicing the two sectional results will be very different from the result if the input is transcribed as a whole. However, as admitted by the authors, this method cannot handle "adaptive attacks", which evade the detection by embedding a malicious command into one section alone. Rajaratnam et al. (Rajaratnam, Shah, and Kalita 2018) proposed a detector by combining various audio pre-processing methods. But an attacker can take the pre-processing effect into account when generating AEs, which has been well demonstrated in (Carlini and Wagner 2017). An effective and robust audio AEdetection method is missing.

7 Discussion and Future Work

If the malicious command embedded in an AE and the host transcription are very similar, our method will probably fail as their similarity score is high. But note that, prior to our work, all the existing AE generation methods claim that *any* host audio can be used to embed a malicious command (Carlini and Wagner 2018; Taori et al. 2018). Our detection method dramatically reduces this attack flexibility, in that the attack cannot succeed unless the host transcription is similar to the malicious command.

Although how to generate transferable AEs is still an open question currently, in future it may become reality. Our future work is to *proactively* prepare our detection system to detect them. The main *difficulty* is to get a training dataset of transferable AEs, which do not exist. However, a unique advantage of our design is that the binary classifier takes similarity scores as inputs, rather than real AEs. Thus, we can generate similarity scores that simulate the effect of *hypothesized highly-transferable* AEs to train the model.

8 Conclusion

Work on handling audio AEs is still very limited. Considering that ASRs are widely deployed in smart homes, smart phones and cars, how to detect audio AEs is an important problem. Inspired by Multiversion Programming, we propose to run multiple different ASR systems in parallel, and an audio input is determined as adversarial if the multiple ASRs generate very dissimilar transcriptions. Detection systems with one single auxiliary ASR achieve accuracies over 98.6%, while designs with more than one ASR achieve even higher accuracies as more features are provided to the classifier. Moreover, the research results invalidate the widely-believed claim that an adversary can embed a malicious command to *any* host audio.

Acknowledgement

The authors would like to thank the anonymous reviewers for their constructive comments and feedback. This project was supported by NSF CNS-1815144 and NSF CNS-1856380.

References

Abdel-Hamid, O.; Deng, L.; and Yu, D. 2013. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *INTERSPEECH*, 3366–3370. ISCA.

Alzantot, M.; Balaji, B.; and Srivastava, M. 2018. Did you hear that? adversarial examples against automatic speech recognition. *arXiv preprint arXiv:1801.00554*.

2018. Amazon transcribe homepage. `https://aws.amazon.com/transcribe`.

Amodei, D.; Ananthanarayanan, S.; Anubhai, R.; Bai, J.; Battenberg, E.; Case, C.; Casper, J.; Catanzaro, B.; Cheng, Q.; Chen, G.; et al. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, 173–182.

Carlini, N., and Wagner, D. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 3–14. ACM.

Carlini, N., and Wagner, D. A. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *IEEE Symposium on Security and Privacy Workshops*, 1–7. IEEE.

Carlini, N.; Mishra, P.; Vaidya, T.; Zhang, Y.; Sherr, M.; Shields, C.; Wagner, D. A.; and Zhou, W. 2016. Hidden voice commands. In *USENIX Security Symposium*.

2018. Download page for common voice dataset. `https://voice.mozilla.org/en/data`.

2018. Deepspeech github repository. `https://github.com/mozilla/DeepSpeech`.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *CoRR* abs/1412.6572.

2015. Google ai blog: The neural networks behind google voice transcription. `https://ai.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html`.

2018. Google cloud speech homepage. `https://cloud.google.com/speech-to-text`.

Graves, A.; Mohamed, A.; and Hinton, G. E. 2013. Speech recognition with deep recurrent neural networks. In *ICASSP*, 6645–6649. IEEE.

Hannun, A. Y.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; and Ng, A. Y. 2014. Deep speech: Scaling up end-to-end speech recognition. *CoRR* abs/1412.5567.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780.

2017. Microsoft hopes skype sets its smart speaker apart. `https://goo.gl/YfKmy5`.

2018. Jaccard index. `https://en.wikipedia.org/wiki/Jaccard_index`.

2018. Jarowinkler distance. `https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance`.

Lang, K. J.; Waibel, A.; and Hinton, G. E. 1990. A time-delay neural network architecture for isolated word recognition. *Neural Networks* 3(1):23–43.

Li, X., and Wu, X. 2014. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. *CoRR* abs/1410.4281.

2015. LibriSpeech ASR corpus. `http://www.openslr.org/12`.

Liming Chen, A. A. 1995. N-version programming: A fault-tolerance approach to reliability of software operation. In *Twenty-Fifth International Symposium on Fault-Tolerant Computing*. IEEE.

Liu, Y.; Chen, X.; Liu, C.; and Song, D. 2016. Delving into transferable adversarial examples and black-box attacks. *CoRR* abs/1611.02770.

Monteiro, J.; Akhtar, Z.; and Falk, T. H. 2018. Generalizable adversarial examples detection based on bi-model decision mismatch. *arXiv preprint arXiv:1802.07770*.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016. The limitations of deep learning in adversarial settings. In *EuroS&P*.

Papernot, N.; McDaniel, P.; and Goodfellow, I. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

2017. Wikipage of phonetic algorithm. `https://en.wikipedia.org/wiki/Phonetic_algorithm`.

Rajaratnam, K.; Shah, K.; and Kalita, J. 2018. Isolated and ensemble audio preprocessing methods for detecting adversarial examples against automatic speech recognition. *arXiv preprint arXiv:1809.04397*.

Sak, H.; Senior, A.; and Beaufays, F. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*.

Sercu, T., and Goel, V. 2016. Dense prediction on sequences with time-dilated convolutions for speech recognition. *CoRR* abs/1611.09288.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2013. Intriguing properties of neural networks. *CoRR* abs/1312.6199.

Taori, R.; Kamsetty, A.; Chu, B.; and Vemuri, N. 2018. Targeted adversarial examples for black box audio systems. *CoRR* abs/1805.07820.

Tóth, L. 2015. Modeling long temporal contexts in convolutional neural network-based phone recognition. In *ICASSP*.

Tramèr, F.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*.

Yang, Z.; Li, B.; Chen, P.-Y.; and Song, D. 2018. Towards mitigating audio adversarial perturbations. `https://openreview.net/forum?id=SyZ2nKJDz`.

Yu, D., and Deng, L. 2015. *Automatic Speech Recognition: A Deep Learning Approach*. Signals and Communication Technology. London: Springer-Verlag London, 2nd. edition.

Yu, D., and Li, J. 2017. Recent progresses in deep learning based acoustic models. *IEEE/CAA Journal of Automatica Sinica* 4(3):396–409.

Yuan, X.; Chen, Y.; Zhao, Y.; Long, Y.; Liu, X.; Chen, K.; Zhang, S.; Huang, H.; Wang, X.; and Gunter, C. A. 2018. Commandersong: A systematic approach for practical adversarial voice recognition. In *USENIX Security Symposium*.

Zeng, Q.; Su, J.; Fu, C.; Kayas, G.; and Luo, L. 2018. Testing CommanderSong AEs over iFLYTECK, Google Cloud Speech and Kaldi. `https://goo.gl/oWTXU8`.

Zuo, F.; Luo, L.; and Zeng, Q. 2018. Countermeasures against l_0 adversarial examples using image processing and siamese networks. *arXiv*.