Poster: Nickel to Lego: Using *Foolgle* to Create Adversarial Examples to fool Google Cloud Speech-to-Text API

Joon Kuy Han
The State University of New York
Incheon, South Korea
joonkyu.han@stonybrook.edu

Hyoungshick Kim
Sungkyunkwan University & CSIRO
Data611
Sydney, Australia
Hyoung.Kim@data61.csiro.au

Simon S. Woo
Sungkyunkwan University
Suwon, South Korea
swoo@g.skku.edu

ABSTRACT

Many companies offer automatic speech recognition or Speech-to-Text APIs for use in diverse applications. However, audio classification algorithms trained with deep neural networks (DNNs) can sometimes misclassify adversarial examples, posing a significant threat to critical applications. In this paper, we present a novel way to create adversarial audio examples using a genetic algorithm. Our algorithm creates adversarial examples by iteratively adding perturbations to the original audio signal. Unlike most white-box adversarial example generations, our approach does not require knowledge about the target DNN's model and parameters (black-box) and heavy computational power of GPU resources. To show the feasibility of the proposed idea, we implement a tool called, *Foolgle*, using a genetic algorithm that performs *untargeted* attacks to create adversarial audio examples and evaluate those with the state-of-the-art Google Cloud Speech-to-Text API. Our preliminary experiment results show that *Foolgle* deceives the API with a success probability of 86%.

KEYWORDS

Adversarial example; Automatic Speech Recognition; Black-box attack; Genetic algorithm

ACM Reference Format:
Joon Kuy Han, Hyoungshick Kim, and Simon S. Woo. 2019. Poster: Nickel to Lego: Using *Foolgle* to Create Adversarial Examples to fool Google Cloud Speech-to-Text API. In *2019 ACM SIGSAC Conference on Computer& Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3319535.3363264

1 INTRODUCTION

Speech-based interaction is widely used to interact with personal assistants of smartphones and smart speakers. These systems rely on running a speech classification model to recognize the user's voice commands. Voice assistants are quickly being upgraded to support advanced, security-critical commands such as unlocking devices, checking emails, and making payments. Deep neural networks (DNNs) have been popularly used for automatic speech recognition (ASR) [2] because they provide highly accurate voice recognition results. However, recent research has shown that neural networks are susceptible to attacks that produce incorrect outputs or a targeted output by an adversary [4, 7, 9, 12]. This type of attack known as *adversarial example* has a high success rate in fooling ASR models. Injected adversarial audio attacks can be dangerous as they can be used not only to degrade the ASR performance but also can be used to manipulate actions taken by digital assistants in unpredictable or malicious ways.

Alzantot et al. [1] demonstrated that black-box approaches for targeted attacks on ASR systems are possible using a genetic algorithm (GA) approach. However, their method would require heavy computational power of GPU resources. Taori et al. [10] also applied the idea of GA approach to fool ASR systems from incorrectly labeling longer phrases and sentences. However, their approach requires prior knowledge of the ASR system, which cannot be applicable to a black-box attack. Khare et al. [6] performed untargeted and targeted attacks using a genetic algorithm by maximizing acoustic similarity between the adversarial sample and the original sample.

In this preliminary work, we explore an alternative approach to *efficiently* generate adversarial examples using a efficient GA to deceive the DNN-based ASRs. To show the feasibility of our idea, we implement a highly efficient black-box attack tool called *Foolgle* to generate adversarial examples and evaluate its attack performance with the state-of-the-art commercial Google Cloud Speech-to-Text API, which are used in many practical applications such as voice assistants. While previous studies have proposed GA-based algorithms to attack *publicly available open* APIs, this is the first GA-based implementation that can generate adversarial audio examples to target a commercial API. Our main contributions are summarized as follows:

(1) We propose *Foolgle*, an attack algorithm using GA to efficiently generate adversarial examples to deceive ASR systems, which does not require any knowledge of DNNs nor require GPU resources.
(2) We evaluate the attack performance of *Foolgle* with the state-of-the-art commercial Speech-to-Text API from Google. Our experiment results show that Foolgle can efficiently generate adversarial examples to deceive the commercial Google Cloud API with a success probability of 86%.
(3) We conducted an IRB-approved user study to evaluate human listeners' recognition performance for adversarial examples generated by *Foolgle* and found that the added perturbations do not significantly affect human listeners' recognition performance.

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11.
https://doi.org/10.1145/3319535.3363264

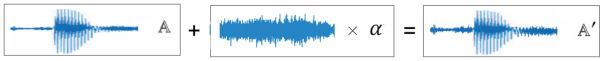

Figure 1: Google API misclassifies the perturbation-added audio $\mathbb{A}'$ "Lego", while it correctly labels the original audio $\mathbb{A}$ "Nickel."

2 DESIGN OF OUR APPROACH

Adversarial example generation: The goal of creating adversarial audio examples is to make an audio classifier misclassify the original input, where we formally define an adversarial example generation as follow: given a valid input audio input $\mathbb{A}$, and a target $t \neq C^*(\mathbb{A})$, it is a procedure for finding a similar audio input $\mathbb{A}'$ such that $t = C^*(\mathbb{A}')$, yet $\mathbb{A}$ and $\mathbb{A}'$ are close according to some distance metric. We call $\mathbb{A}'$ an adversarial example to deceive C (see the definition of adversarial example in [11]). In *Untargeted adversarial examples*, attacks only search for an input $\mathbb{A}'$ so that $C(\mathbb{A}) \neq C^*(\mathbb{A}')$ and $\mathbb{A}$ and $\mathbb{A}'$ are close. Then, finding adversarial examples can be formulated as $\min_{\mathbb{A}'} ||\mathbb{A}' - \mathbb{A}||$ such that $C(\mathbb{A}) \neq C^*(\mathbb{A}')$.

Creating Adversarial Examples Using Genetic Algorithm (GA): In order to perform a black-box attack, we develop a GA to effectively generate adversarial audio examples against the commercial API without access to any of their DNN model parameters. The goal of our GA is to impose a small number of perturbations to the original audio so that a commercial API misclassify the original input audio sample ("Nickel") to another one ("Lego"), while humans can still easily recognize the original audio sample as shown in Fig. 1. We formulate our GA as follows:

Population and fitness. We create a new population of candidate adversarial examples from the population of input audio samples by adding perturbations to each audio sample in the population as $\mathbb{A}' = \mathbb{A} + (\alpha \times \delta)$ where a population represents a set of audio samples. Initially, perturbation δ is generated with the same length as the original audio sample $\mathbb{A}$ with the maximum frequency range and then multiplied by coefficient α. Then, we add the perturbation into the original audio sample $\mathbb{A}$ to obtain the adversarial audio example $\mathbb{A}'$. To measure the similarity of two audio samples, we first obtain the Mel-frequency cepstral coefficients (MFCC) [8] of $\mathbb{A}$ and $\mathbb{A}'$. Next, we use the Dynamic Time Warping (DTW) algorithm [3] to obtain a distance measure L_1 between the two MFCC sequences. Then, we define the fitness function f as follows:

$$f = P_0 - P_d + \gamma \times L_1, \tag{1}$$

where P_0 is the confidence score for the original label and P_d is the confidence score for any other wrong labels. We can obtain either one of P_0 or P_d and set the other to zero because the commercial APIs only return the highest probability of one of P_0 or P_d. Next, we formulate our GA as a minimization problem with the value of the fitness function in Eq. (1) to produce the best audio sample, which has high P_d, and low P_0 and low L_1 values. In Eq. (1), γ is another coefficient to balance the perturbation amount to deceive APIs by guiding GA to find an adversarial audio sample with the least amount of perturbations, where γ is inversely proportional to α, because P_0 and P_d always have the values between 0 to 1.

Selection. We implement a tournament selection with a size to be four. Only two of the four audio samples in each tournament will be selected. In our design, audio samples with higher fitness have a 75% chance to win, and the less fit has a 25% chance to maintain a good variety in the population and explore more extensive search areas to find a global optimum.

Crossover and mutation. Crossover permutes two selected parents. We design a simple crossover arithmetic recombination operation to perform a single-point crossover [5]. If P_1 and P_2 are a mating pair, then the children C_i it produces will inherit genes from each parent proportional to the value of n which is a random integer from 1 to 3, as shown in Eq. (2) below:

$$C_i = \frac{(n \times P_1) + ((4 - n) \times P_2)}{4}, 1 \leq n \leq 3 \tag{2}$$

To conserve the best fit samples and not to lose them, we also added the following inheritance heuristics: if the best sample in the current generation is better than any sample in the next generation, we copy the best sample in the current generation to the next generation. Mutation aims to increase variation within a population and increase the perturbation level of adversarial audio example. If the noise-added audio is still classified into the original label, then we add a small number of random perturbations to samples to produce more variations.

3 EXPERIMENT AND RESULTS

Experiment: The goal of our experiment is to evaluate the generated adversarial audio examples and further test the robustness of commercial APIs using *Foolgle*. We used Google's Speech-To-Text API to evaluate the effectiveness of our approach. We randomly extracted words from audio files in the Mozilla Common Voice dataset to create 50 audio files of a single word to test the effectiveness of our approach. Initially, we run five epochs to try to get an adversarial example for all 50 audio files, starting with $\alpha = 0.01$. Then, we automatically increment or decrement α based on the attack success and confidence value returned from the API. If we consecutively fail to produce an adversarial audio file in the next 10 iterations, we increase α and repeat the process. If we can find α in 10 consecutive steps, we decrease α to reduce the noise.

Attack Success Rate. Of the 50 adversarial audio files, *Foolgle* successfully creates adversarial audio labels for 43 audio files, yielding 86% success rate. For 10% of the audio files generated by *Foolgle*, Google API was unable recognize the word in each audio file and returned 'no label'. Google API returned the same label for 4% of the audio files. For simplicity, we limit the number of iterations in our experiments to 500. If *Foolgle* fails to create a different label within 500 iterations, we stop the process and declare the audio file to have the same output as the original. For 14% of the generated audio files, *Foolgle* failed to generate an adversarial sample.

Analysis. Figure 2 presents the spectrogram of the original and adversarial examples we produced for the original audio label of "Nickel", frequency (Y-axis) vs. time (X-axis). In Fig. 2(a) shows the spectrogram of the original audio sample ("Nickel") with little to no background noise. As noise is added to the audio sample, we can clearly observe the change in color Fig. 2(b) and Fig. 2(c). The color indicates the strength of the energy and red signifying the voice

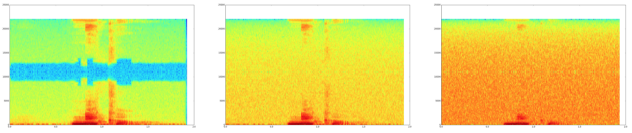

Figure 2: Spectrogram of perturbations distribution generated for "Nickel" (*Foolgle* vs. Random perturbation).

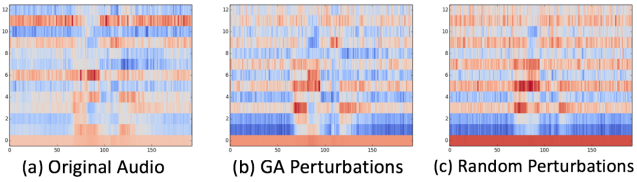

Figure 3: MFCC of perturbations distribution generated for "Nickel" (*Foolgle* vs. Random perturbation).

activity. As we compare these two sets of images, the perturbation pattern generated by our approach in Fig. 2(b) is clearly different from the random perturbations in Fig. 2(c). While random noise spreads uniformly everywhere as shown in Fig. 2(c), which makes it difficult for humans to comprehend the original audio, the noise generated from our approach tends to preserve the original audio labels. We can clearly observe that our GA tends to better mitigate interference with the human voice activity by the clear red features.

To analyze the differences, we compare the MFCC plotting of the original and adversarial audio samples as shown in Fig. 3, where MFCC can capture the distinct features of an audio signal by plotting MFCC coefficient (Y-axis) vs. time (X-axis). The color represents the coefficient values from lowest (blue) to highest (red). From Fig. 3, we can observe that the audio sample generated with GA perturbations (Fig. 3(b)) more closely matches the original audio sample (Fig. 3(a)) compared with the audio sample generated with random perturbations (Fig. 3(c)). Moreover, we clearly observe the significant differences between GA perturbations (Fig. 3(b)) and random perturbations (Fig. 3(c)). These experiment results demonstrate that GA perturbations appears better to generate adversarial audio examples with similar characteristics to those of the original audio samples.

Human Evaluation. To measure the effectiveness of *Foolgle*, we measure how difficult to recognize generated adversarial audio examples are and how comprehensible their texts are for human listeners. For this purpose, an IRB-approved user study was carried out by conducting listening tests with 12 participants we recruited from our campus. In the first task, each participant was presented with 10 random adversarial samples generated by *Foolgle* and for each sample was asked to type in the recognized text from the audio sample to measure the correctness and rate how to difficult to correctly understand (with a 5-point Likert scale). In the second task, we also asked participants to rate each sample to measure their sentiments (with a 5-point Likert scale).

Overall, we collected 120 inputs, and participants were able to correctly identify and label 88% of the adversarial examples generated by *Foolgle*. This indicates that the generated audio samples have either no or minimal audible noise in the generated adversarial samples. Participants indicate that they found 8% of the generated audio samples to be very easy, and 43% to be easy to listen. 27% of the audio were determined to be neutral on their comprehension. On the other hand, 13% and 9% of the audios were mentioned by participants to be difficult and very difficult to comprehend the generated audio samples. Therefore, overall only 22% of the samples generated were difficult to comprehend for the participants, which is correlated with the correctness result.

4 DISCUSSION AND FUTURE WORK

Our dataset is limited in the sense that we only used 50 different random voice samples for this preliminary study. Another limitation is that as our attack is a black-box approach using GA. Therefore, our perturbation generation would not be optimal, and performance may be worse than the theoretical optimum value. However, our benefits, which do not require GPU resources and provide a practical algorithm to estimate the noise level, can be a more attractive option in real-world situations. Lastly, as our attack only requires the confidence score from the ASR systems, we plan to test the robustness of other commercial ASR systems with our approach.

5 CONCLUSION

We introduce a simple yet effective method, *Foolgle*, to generate adversarial audio, which does not require any knowledge about DNNs or use GPU resources. Our *Foolgle* creates noise using an iterative approach using GA. Our preliminary result shows that the state-of-the-art Google Cloud Speech-to-text API is easily fooled by our attack. It appears that the current API is not capable of handling and possibly malicious exploited to produce unintended results or undesired behaviors from voice assistant applications.

REFERENCES

[1] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. 2018. Did you hear that? adversarial examples against automatic speech recognition. *arXiv preprint arXiv:1801.00554* (2018).
[2] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. 2016. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In *Proceedings of the International Conference on Machine Learning*.
[3] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series.. In *KDD workshop*, Vol. 10. Seattle, WA, 359–370.
[4] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *25th USENIX Security Symposium (USENIX Security 16)*. 513–530.
[5] Agoston E Eiben, James E Smith, et al. 2003. *Introduction to evolutionary computing*. Vol. 53. Springer.
[6] Shreya Khare, Rahul Aralikatte, and Senthil Mani. 2018. Adversarial Black-Box Attacks for Automatic Speech Recognition Systems Using Multi-Objective Genetic Optimization. *arXiv preprint arXiv:1811.01312* (2018).
[7] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 506–519.
[8] Lawrence R Rabiner and Bernard Gold. 1975. Theory and application of digital signal processing. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p.* (1975).
[9] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. 2018. Inaudible Voice Commands: The Long-Range Attack and Defense. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*.
[10] Rohan Taori, Amog Kamsetty, Brenton Chu, and Nikita Vemuri. 2018. Targeted adversarial examples for black box audio systems. *arXiv preprint arXiv:1805.07820* (2018).
[11] Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. 2017. Adversarial Examples: Attacks and Defenses for Deep Learning. *arXiv preprint arXiv:1712.07107* (2017).
[12] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*.