FraudWhistler: A Resilient, Robust and Plug-and-play Adversarial Example Detection Method for Speaker Recognition

Kun Wang, *Zhejiang University;* Xiangyu Xu, *Southeast University;*
Li Lu, Zhongjie Ba, Feng Lin, and Kui Ren, *Zhejiang University*

https://www.usenix.org/conference/usenixsecurity24/presentation/wang-kun

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

USENIX Security '24 Artifact Appendix: FraudWhistler: A Resilient, Robust and Plug-and-play Adversarial Example Detection Method for Speaker Recognition

Kun Wang
Zhejiang University

Xiangyu Xu
Southeast University

Li Lu
Zhejiang University

Zhongjie Ba
Zhejiang University

Feng Lin
Zhejiang University

Kui Ren
Zhejiang University

A Artifact Appendix

A.1 Abstract

In our Artifact, we provide the source code, experiment design, and some raw results of FraudWhistler. Specifically, we provide the implementation source code of FraudWhistler for performing the experiments described in this paper, the instruction for running code to reproduce our major experiment results, and the raw anonymous results of our audibility study. This appendix will introduce details of our open-source code available on GitHub: https://github.com/kuang22/FraudWhistler/releases/tag/v1.3, including environment installation, dataset preparation, running evaluation, result collection and how to deploy FraudWhistler on a new SR system.

A.2 Description & Requirements

Software Requirements: Conda or miniconda, python, pytorch, and CUDA. It is advised to run the experiments by following the instructions provided in the repository.
Hardware Requirements: To produce the experiments, we expect a machine with at least 128GB RAM, and 24GB NVIDIA GPU.

A.2.1 Security, privacy, and ethical concerns

This artifact will not lead to any security, privacy and ethical issues.

A.2.2 How to access

Use the following link to access our code on GitHub: https://github.com/kuang22/FraudWhistler/releases/tag/v1.3. You can either download the project without git on your machine or simply use the "git clone" command to clone the project.

A.2.3 Hardware dependencies

We recommend a machine with Linux operating system equipped with at least 128GB RAM and 24GB NVIDIA GPU. In our experiment setup, we use a machine running Ubuntu hirsute 21.04 with 40 Intel Xeon Silver 4210R CPU, 256GB RAM, and four 48GB NVIDIA RTX A6000 GPU.

A.2.4 Software dependencies

The code is tested with Python 3.11, Pytorch 2.0.1, and CUDA 11.7. We also require some libraries like torchaudio, torchmetrics. You can follow the instructions in the README file in our repository to create the environment.

A.2.5 Benchmarks

We require VCTK corpora and OpenRIR dataset to reproduce the main results in our paper. You need to prepare the dataset before running the codes. Please follow the instructions provided in our repository.

A.3 Set-up

A.3.1 Installation

Please follow thses steps to install the software dependencies:
1. CUDA Installation: You need to make sure a CUDA toolkit with a version later than 11.7 is properly installed on your machine. Plase check the official website of CUDA Toolkit for instructions: https://developer.nvidia.com.
2. Anaconda Installation: You need to install conda to prepare the running environment. Please check the official website of anaconda: https://www.anaconda.com/download.

3. Environment Creation: You need to create the running environment based on the *env.yaml* file in our repo. The detailed instructions are described in the README file.

A.3.2 Basic Test

Run the following commands in shell to validate the the environment:

```
$ python
>>> import torch
>>> torch.cuda.is_available()
```

If the outout is "True", then the dependencies are installed properly.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): FraudWhistler achieves about 98.7% on detecting adversarial examples for speaker recognition, outperforming state-of-the-art works by about 13% with a minor degradation of 6.1% on benign audio samples.

(C2): FraudWhistler achieves an accuracy of 84% in the worst case against an adaptive adversary, considering the imperceptibility to the human involved.

A.4.2 Experiments

(E1): [Dataset Preparation] [5 human-minutes + 1 compute-hour + 12GB disk]: Run the "prepare_dataset.sh" to download the dataset needed and generate some auxiliary files.
Preparation: None
Execution: Run "prepare_dataset.sh" in the cloned repository.
Results: There should be a directory named "dataset" and a file named "reverb.csv" is created.

(E2): [Evaluation on Static Attacks] [10 human-minutes + several compute-days]: Generate adversarial examples with various algorithms, train FraudWhistler on benign data, and evaluate the system on detecting AEs (Table. 3).
Preparation: Create and activate the conda environment named "fraudwhistler" with the provided file *env.yaml* in our repository.
Execution: Extract features from benign examples and adversarial examples with the following commands:

```
$ python prepare_data.py
$ python prepare_adv_data.py
```

and perform the detection by running the following command:

```
$ python run_det.py 3
```

Results: The extracted features are stored in "Results/scores_data.npy" and "Results/scores_adv_data.npy", and the detection results are stored in "Results/det.txt". You can show the detection results by running "cat Results/det.txt" and the output should be in the format as follows:

———

DefenseSystemName:
Accuracy on benign examples (ACC_be): XX
Detector's accuracy on adversarial examples (ACC_ae):
Average: XX
Robust Accuracy of whole system on AEs (ACC_rob):
Average: XX
Note that there are three defense systems evaluated at default including FraudWhistler, WaveGuard, and TimeDependent.

(E3): [Evaluation on Adaptive Attacks] [5 human-minutes + several compute-days]: Generate adaptive adversarial examples against FraudWhistler system, evaluate the performance of FraudWhistler, and compare the performance with human accuracy (Figure. 8 and 9).
Preparation: Activate the conda environment, and make sure that "Results/scores_data.npy" has been created in E2.
Execution: Generate adaptive adversarial examples and extract features from them for performing detection by running the following command:

```
$ python run_adap_adv.py 3
```

Results: The detection results are stored in "Results/det_adap.txt" and the output should be in the format as follows:

————*epsilon_0.XX*
SNR: XXXX PESQ: XXXX
Adaptive Attack Success Rate: XX
Robust Accuracy of whole system on AEs: XX
The raw anonymous result of our audibility study is also provided in the repository in a file named *listening_res.csv*.

A.5 Notes on Reusability

In this artifact, we provide detailed instructions on how to deploy FraudWhistler on a new speaker recognition system and how to evaluate the system on a new dataset. You can follow the instructions in the section "Deploy FraudWhistler on New SR/Dataset" of the README file.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.