CS230

# Transferable Audio Adversarial Attacks

**Yazen S. Shunnar**
yzsh@stanford.edu
SUID - 06487472

## Abstract

I demonstrate a targeted audio attack on a white-box audio classification neural network. I use a greedy iterative method, sourced from computer vision attacks, that incrementally aggregates protuberances from the training set towards a specified target class. The target neural networks were trained using publicly sourced data and are architected as 1D CNNs. Experimental have shown attack rate success at over 60%.

## 1 Introduction

The majority of deep learning research on transferable adversarial attacks has been on images and natural language processing models. With the growth of audio assistants there has been an increase in the availability and accessibility of audio as an attack vector. Popular voice assistants such as Alexa, Google Home, or Cortana, are left running at all times. This makes audio attacks one of simplest ways to bypass a lock screen or password. Further, the rise in popularity of phone audio assistants has made the vector widely available against the average person.

Deep Learning research on adversarial attacks of computer vision models has been widely published and demonstrated. Universal audio attacks have received much less focus from researchers. Nicholas Carlini of Google Brain specifically calls out the lack of research into transferable adversarial attacks against speech recognition models in his lecture Making and Measuring Progress in Adversarial Machine Learning at 40TH IEEE Symposium on Security and Privacy[4]. He also mentions it in his paper Audio Adversarial Examples: Targeted Attacks on Speech-to-Text[3]. This paper will advance the research in adversarial attacks by demonstrating a universal audio attack on a black-box neural network. There are five inputs required to demonstrate an adversarial attack, the target model, training set, test set, training classifications, and target class.

Inputs: The target models will be artificially created 1D CNNs for simplicity. The training set will be the Urban8ksound dataset. Urban8ksound is a publicly available dataset of 8.75 hours of with roughly 3 second labeled audio recordings. The models will all be trained on the training set. The test set will be 10% of the training set aside from the model. The Urban8ksound dataset comes with segments pre-labeled into one of ten classes: air-conditioner, car-horn, children playing, dog-bark, drilling, engine-idling, gun-shot, jackhammer, siren, or street-music. The target class can be any one of the labels.

Output: To quantify our results we output a perturbed training set, that includes an attempt to perturb all inputs until they are classified by the target model as the target class, and the percent that was classified as the target class over the total number of training sets - the number that were originally classified as the target class.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

## 2 Related work

Transferable audio attacks are relatively unstudied, as such there are only a few papers that deal with them directly. I have included two papers on transferable audio attacks, two on non-transferable attacks whose ideas this paper applies, and one transferable image recognition attack whose ideas area applied in all transferable attacks.

### 2.1 Transferable Audio Attacks

Most transferable audio attacks deal with iterative greedy methods that, with knowledge of a target model's outputs, perturb a sample until is classified by the target model as the target class. The main advances are in optimizing these models for real world scenarios, by minimizing the protuberance or minimizing the attempts required to reclassify an input.

Universal Adversarial Audio Perturbations[1] minimizes the protuberances between the input sample and the outputs sample. It proposes two main solutions. Solution 1: an iterative greedy method to perturb a target training example until it resembles a different class using any standard perturbation formula. They suggest Carlini and Wanger L2 attack [5] or DDN attack[6] or Deepfool. Unfortunately, this strategy doesn't minimize the decibel level of the attack, leaving it noticeable to third parties.

Solution 2 modifies solution one by normalizing the optimizations function by the sound pressure level. This can create an attack that does not noticeable perturb the sound clip. This technique would be useful when attempting to create an audio attack that could be used in a public setting.

Enabling Fast and Universal Audio Adversarial Attack Using Generative Model[2] minimizes the number of iterations to reclassify the input sample. As Audio samples are temporal to classify a three second clip in an attack setting you have to play it for three seconds, where as in computer vision a quick flash of the image will be processed. The length of an audio clip can becomes a training constraint.

The suggest technique of fast audio adversarial perturbation generator (FAPG), uses the generative model Wave-net U, with embedded feature maps during the models training. The Wave-U-net and target model are trained on the same dataset, and a separate matrix of features with their vector difference is calculated. At the end of the training the matrix can be used to immediately calculate a protuberance to shift any input sample to a target class. Unfortunately, this requires you to have access to the model as it is being trained. It also can require a huge amount of memory in large class sizes that grows at rate of $N^2$. This approach seems interesting when there is some background knowledge of a model, but not useable in the general case where knowledge of how a model is trained is not published.

### 2.2 Non-Transferable Attacks

In Audio Adversarial Examples: Targeted Attacks on Speech-to-Text[3] the authors demonstrate a 100% accuracy attack on a specific text to speech model. The authors apply the same iterative greedy method on a white-box model as the other two papers, but on a text to speech model which has a much higher set of labels, and so, require a more complex model. To compensate, the authors use a unique training strategy known as Connectionist Temporal Classication (CTC). CTC outputs a percent chance that each input in a segment produces a token in the output segment. This allows for audio models to produce text models whose segmentation structure varies. The authors regularize with relation to decibel level to minimize how noticeable their protuberance is.

In Towards Evaluating the Robustness of Neural Networks[5] several loss formulas are considered in evaluating loss metrics that can be re-purposed into protuberances. The $L_p$ method that evaluates the minimum change to a each individual audio clip, $L_2$ method that evaluates to the mean change of audio clips in the batch, and the $L_\infty$ method that evaluates the $L_p$ method, but adds a box constraint where any example perturbed beyond its constraint becomes a failure.

### 2.3 Transferable Image Attacks

In Universal adversarial perturbations, Mahfouze Et Al[7] work to show that all neural networks can be attacked universally. Though the paper deals with image attacks, its techniques are generalizeable

to audio attacks. It establishes the steps of, testing an input, calculating a loss with reference to the target class, and then update the input to approach the target class. All these steps are agnostic to data input. This is the central strategy that all papers on universal attacks that I have read, for image or audio attacks, follows. This paper is cited prominently in both Universal Adversarial Audio Perturbations[1] and Enabling Fast and Universal Audio Adversarial Attack Using Generative Model[2].

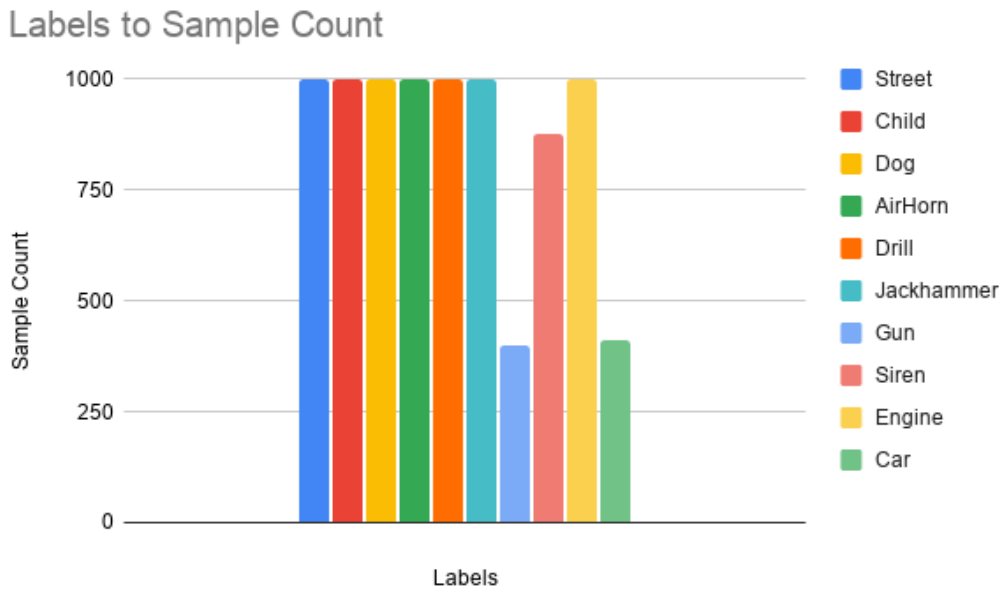
## 3 Dataset and Features

I used the audio recordings of the UrbanSound8k dataset.

### 3.1 Raw Format

The UrbanSound8k dataset is 8.75 hours long with 8000+ clips at 16khz. The audio clips are pre-labeled into the 10 classes: air-conditioner, car-horn, children playing, dog-bark, drilling, engine-idling, gun-shot, jackhammer, siren, or street-music. It is pre-sliced into 10ths that are pre-normalized.

The classes have nearly the same average representation in the dataset, with just a few outliers of cars and guns.




### 3.2 Processing

Pytorch includes torchaudio, a special audio loading class for adding audio files to a model. UrbanSound8k comes sampled at 44.1khz. I first used torchaudio.transforms.DownmixMono() to convert the audio data to one channel. Then, I downsampled the audio to 8khz to get roughly 4 second clips. This downsampling is achieved by selecting every fifth smple of the original audio clip. If an audio clip does not fit in the 4 seconds I have padded it with zeros. An audio clip needs roughly 160,000 samples to avoid being padded.

### 3.3 Simulated Data

The target models are CNNS with randomly generated pooling and filtering values(normalized to be compatible). We generated 10 new CNNs during each run. Unfortunately, due to the fact that each target model had to be trained and evaluated, time and compute constraints did not allow for a higher number of CNNs to train. We trained each model using cross validation on the training set. Each Model was run for 50,000 evaluations with a minibatch size of 100.

Table 1: Baseline Metrics Input Child Audio File, Attack as Street Audio File

| Metric | Street |
|---|---|
| median | 78.2 |
| mean | 79.3 |
| max | 76.7 |

# 4 Experimental Protocol

To review, the inputs are: the target model, training set, test set, training labels, and target class.

## 4.1 Target Model

The target models are CNNs with a single hidden layer, one pooling layers, one filtering layer. This helps speed up training and simplifies calculating our protuberances. The target models end with one 10 node layer where each node corresponds to the percent chance of an output, and a argmax layer that evaluates for the highest chance of the output.

I trained 10 models per run. Each Model was run for 50,000 evaluations with a minibatch size of 100 using ADAM. I only trained 10 models as each model had to be trained from scratch per run, which lead to a large time delay.

We tested multiple batch sizes, but settled on 100 as it allowed for batch-gradient descent, and eventually converged at 50,000. Batch sizes of 50 never converged, and batch sizes of 10000 converged, but at a lower final accuracy.

The training set is pre-divided into 10 sections. We will be applying 10 fold cross-validation using the divided sections. UrbanSound8k suggests that models be run in cross-validation without shuffling since in practice models have highly variable outputs depending on the split of the cross-validation. They have pre-split the models, and most major papers make use of the UrbanSound8k split. We will be using their split as it allows for better reproduceability and better comparisons with other papers.

## 4.2 Target Class

The target class was chosen randomly during each training run by randomly choosing an int 0-9.

# 5 Methods

Universal adversarial audio protuberance follow a standard format of testing different inputs on the target model, with some learning algorithm driving the testing, until an attack input is found.

## 5.1 Core strategy

Our goal is to find a delta d, that, when added to the audio samples causes them to classify as the target class. The problem can be formulated as $y = k(x) y' = k(x + v)$. We apply DeepFool to find d. We then add d to the samples in our batch, if their loudness violates our decibel contraints, we return a failed attempt at classification. We then loop to the next batch. At the end we test the percentage that are misclassified as the target class for our accuracy metrics.

Our primary metric is the percent of audio examples that were successfully converted to the target class. This will be referred to as "Accuracy" in this paper.

## 5.2 Baseline

An adversarial attack from the child class to the street class. Target models were trained on a dataset that had all other classes removed. Audio models trained on the UrbanSound8k often have a test set accuracy of 83% [8]. The adversarial attack here is extremely successful as it is almost able to create an attack with nearly the same accuracy as target models can classify an audio clip.

### 5.3 Relevant Algorithm

We apply DeepFool to X to find a protuberance to X, d, that missclassifies as our target class q. Deepfool is a technique that iteratively probes an input to a class. It is most easily modeled when the target model is a binary affine classifier, but is generalizeable to multiclass polyhedral models.

If we assume the weights of a network are an affine classifer in a binary network, e.g. with weights calculated as $(w^T)k + b$, we know that the distance from x to its inverse class is equal to the distance from X to the line where $(w^T)k + b = 0$. This line linear, and can be calculated and iterated on.

A binary classifier is generalised to a multi-class classifier through finding the projection x against the complement of the complex polyhedron created by the target model. The complex polyhedron that represents the target model, with relation to x represents the nearest point x is misclassified. This can be applied only to the target class for the target classifier.

We then create a new vector x‘ that equals x + the difference between x and the complex polyhedron that represents the target model with relation to x with a targeted class.

We test x‘ on the target model and see if it has been reclassified correctly. If it has not we replace x with x‘ and calculate the next x‘ If it has we stop iterating on x.

#### 5.3.1 Constraining Decibels

We minimize protuberances by optimizing towards the $L_\infty$ with a boxing constraint of one standard deviation from the mean of the Gaussian distribution of the training set. This value is incorporated to the regularization parameter through a piecewise function that reverses the protuberance if its output exceeds the max decibel level. While this is a harsh limit, it works to give the model freedom to iterate, without allowing the modified sound clip to become statistically noticeable.

## Results

1: Matrix of Input Classes to Average Adversarial Output Successes Across 10 Models

| | Child | Dog | AirHorn | Drill | Jackhammer | Gun | Siren | Engine | Car | Street | AVG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Child | 0 | 55.5 | 63.2 | 57.6 | 61.1 | 60.1 | 60.9 | 60.5 | 60.1 | 61.5 | 60.05 |
| Dog | 63.5 | 0 | 60.1 | 57.9 | 60.4 | 59.6 | 61.6 | 58.5 | 59 | 59.1 | 59.97 |
| AirHorn | 58.7 | 65.8 | 0 | 67.6 | 59.1 | 62.1 | 59.3 | 59 | 58.7 | 62.4 | 61.27 |
| Drill | 62.9 | 64.3 | 59.2 | 0 | 58.7 | 59.1 | 62.8 | 63 | 61.9 | 61.7 | 61.36 |
| Jackhammer | 59.9 | 62.1 | 58.4 | 55.6 | 0 | 58.4 | 60.3 | 58.1 | 62.9 | 59.2 | 59.49 |
| Gun | 62.8 | 58.6 | 59.5 | 57.6 | 59 | 0 | 61.4 | 58.1 | 60.4 | 59.5 | 59.69 |
| Siren | 64.7 | 61.1 | 60.2 | 61 | 63 | 61.7 | 0 | 62 | 59.9 | 59 | 61.26 |
| Engine | 59.2 | 57 | 58.3 | 58.4 | 61.6 | 59.7 | 62.4 | 0 | 59.1 | 60.4 | 59.61 |
| Car | 61.1 | 63.2 | 54.9 | 58.5 | 59.5 | 63 | 60.4 | 62.2 | 0 | 62.9 | 60.57 |
| Street | 57.9 | 62.5 | 63.2 | 60.2 | 62.7 | 62.2 | 60.6 | 62.2 | 59.1 | 0 | 61.06 |

Accuracy was pretty consistently 60% from any input class to any target class. Our attack shrunk from our baseline as we greatly expanded the classifications we attempted. We went from an attack that would generalize across multiple models to an attack that would generalize across multiple classes across multiple models. The overall lower attack rate may be able to be improved by optimizing for specific binary classifications, or by running the model for longer so that it can better fit the larger training set.

## 6 Conclusion/Future Work

In this paper I applied an iterative greedy method and the $L_\infty$ norm to create generalizeable attacks on multiple DNN models. $L_\infty$ norm allowed for control of the decibel level to limit the adversarial attacks becoming statistical anomalies. A future model may work to optimize to the minimum decibel so that they are non-audible to humans, allowing for greater robustness and stress testing of models.

If I had more resources I would run another 1,000,000 training iterations to train the attacker model. The drop in performance between the baseline and the advance set shows the attacker models struggled to generalize to the larger dataset, and I believe more time training would allow it to do so. I would also consider my complex Neural Networks that could better represent my model.

Lastly, I would attempt adversarial attacks on more complex models to test the robustness of greedy interative methods on the $L_\infty$ norm.

## 7 Contributions

I am the only member of my team and made all contributions.

## References


[1] Abdoli, Sajjad, et al. "Universal adversarial audio perturbations." arXiv preprint arXiv:1908.03173 (2019).

[2] Xie, Yi, et al. "Enabling Fast and Universal Audio Adversarial Attack Using Generative Model." arXiv preprint arXiv:2004.12261 (2020).

[3] Carlini, N., & Wagner, D. (2018, May). Audio adversarial examples: Targeted attacks on speech-to-text. In 2018 IEEE Security and Privacy Workshops (SPW) (pp. 1-7). IEEE.

[4] Carlini, 29:12 / 59:18 Nicholas. Making and Measuring Progress in Adversarial Machine Learning. IEEE Symposium on Security and Privacy, 2019, www.youtube.com/watch?v=jD3L6HiH4ls&ab_channel=IEEESymposiumonSecurityandPrivacy.

[5] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in IEEE Symp on Security and Privacy, 2017

[6] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, “Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses,” arXiv preprint 1811.09600, 2018.

[7] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[8] Salamon, Justin, Christopher Jacoby, and Juan Pablo Bello. "A dataset and taxonomy for urban sound research." Proceedings of the 22nd ACM international conference on Multimedia. 2014.