CY 7790

Special Topics in Security and Privacy: Machine Learning Security and Privacy
Fall 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science

October 4 2021

Simple Black-box Adversarial Attacks

by Chuan Guo, Jacob R. Gardner, Yurong You, Andrew Gordon Wilson, Kilian Q. Weinberger
ICML 2018

Presented by Hye Sun Yun
October 4, 2021

Threat Model

Objectives: *Black-box evasion attack* that is both untargeted and targeted for L2 norm

Knowledge: Since this is *black-box* attack, the adversary only has the labels and confidence scores from querying the target model

Capability: Get labels/confidence score by querying the target model and add imperceptible perturbation to images for misclassification

Problem

- Black-box threat model is more realistic than white-box but more challenging
- Black-box attacks incur significant cost of both time and money
    - Most efficient and complex attacks still require upwards of tens or hundreds of thousands of queries

$$\min_{\delta} \ \ell_y(\mathrm{x}+\delta) \text{ subject to } \|\delta\|_2 < \rho.$$

Untargeted Attack

$$\ell_y(\mathrm{x}') = p_h(y \mid \mathrm{x}')$$

Targeted Attack

$$\ell_{y'}(\mathrm{x}') = -p_h(y' \mid \mathrm{x}')$$

Problem

- Black-box threat model is more realistic than white-box but more challenging
- Black-box attacks incur significant cost of both time and money
    - Most efficient and complex attacks still require upwards of tens or hundreds of thousands of queries

$$\min_{\delta} \ \ell_y(\mathrm{x} + \delta) \text{ subject to } \ \|\delta\|_2 < \rho.$$

$$\min_{\delta} \ \ell_y(\mathrm{x} + \delta) \text{ subject to: } \ \|\delta\|_2 < \rho, \text{queries} \leq B$$

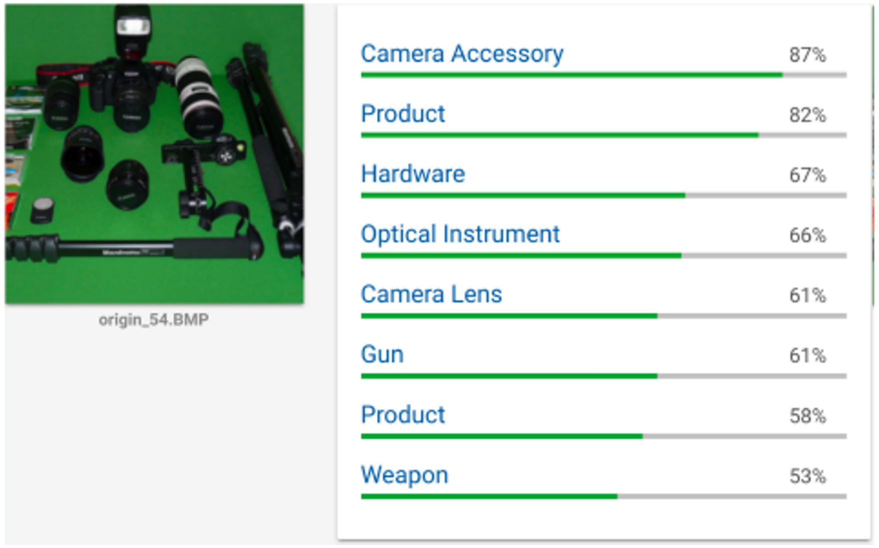
Research Questions

- Is there a simple yet highly efficient black-box attack that can work on ML models only allow API calls and provides confidence scores?
- Can this black-box attack work for both untargeted and targeted attacks?

Simple Black-box Attack (SimBA)

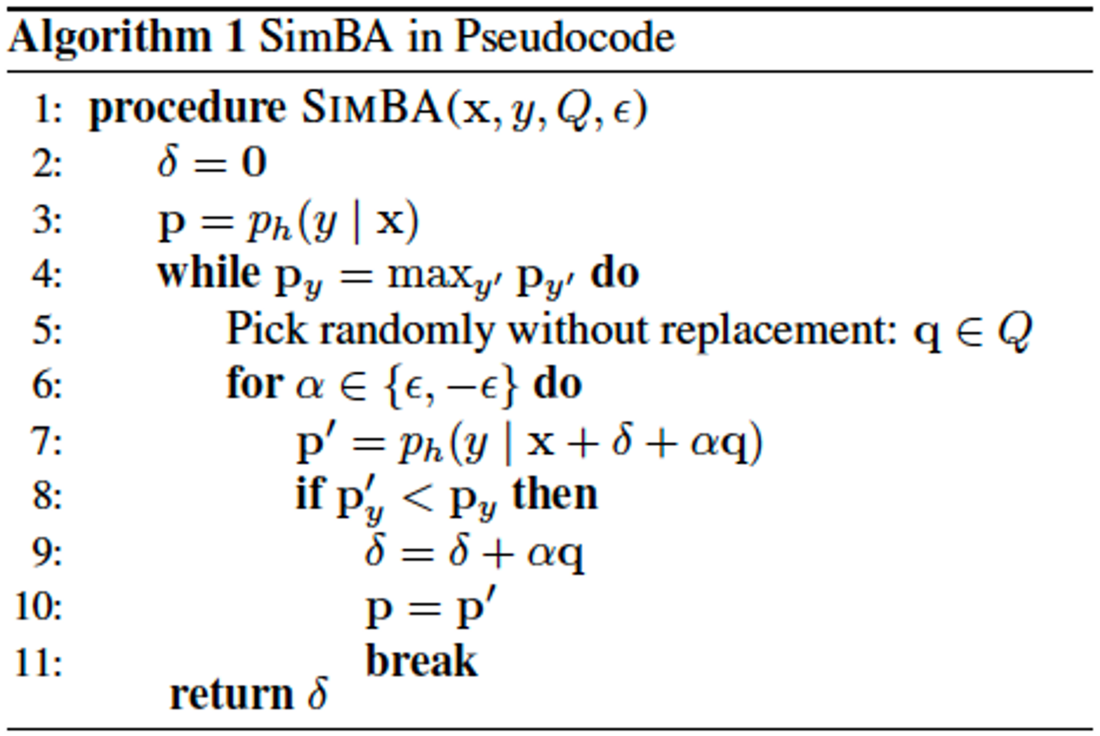
Key Points

1. Continuous-valued confidence scores as a strong proxy to guide the search for adversarial images
2. If the distance to a decision boundary is small, exact direction does not matter
3. Density of adversarial examples is much higher in the low frequency domain



Camera Accessory	87%
Product	82%
Hardware	67%
Optical Instrument	66%
Camera Lens	61%
Gun	61%
Product	58%
Weapon	53%

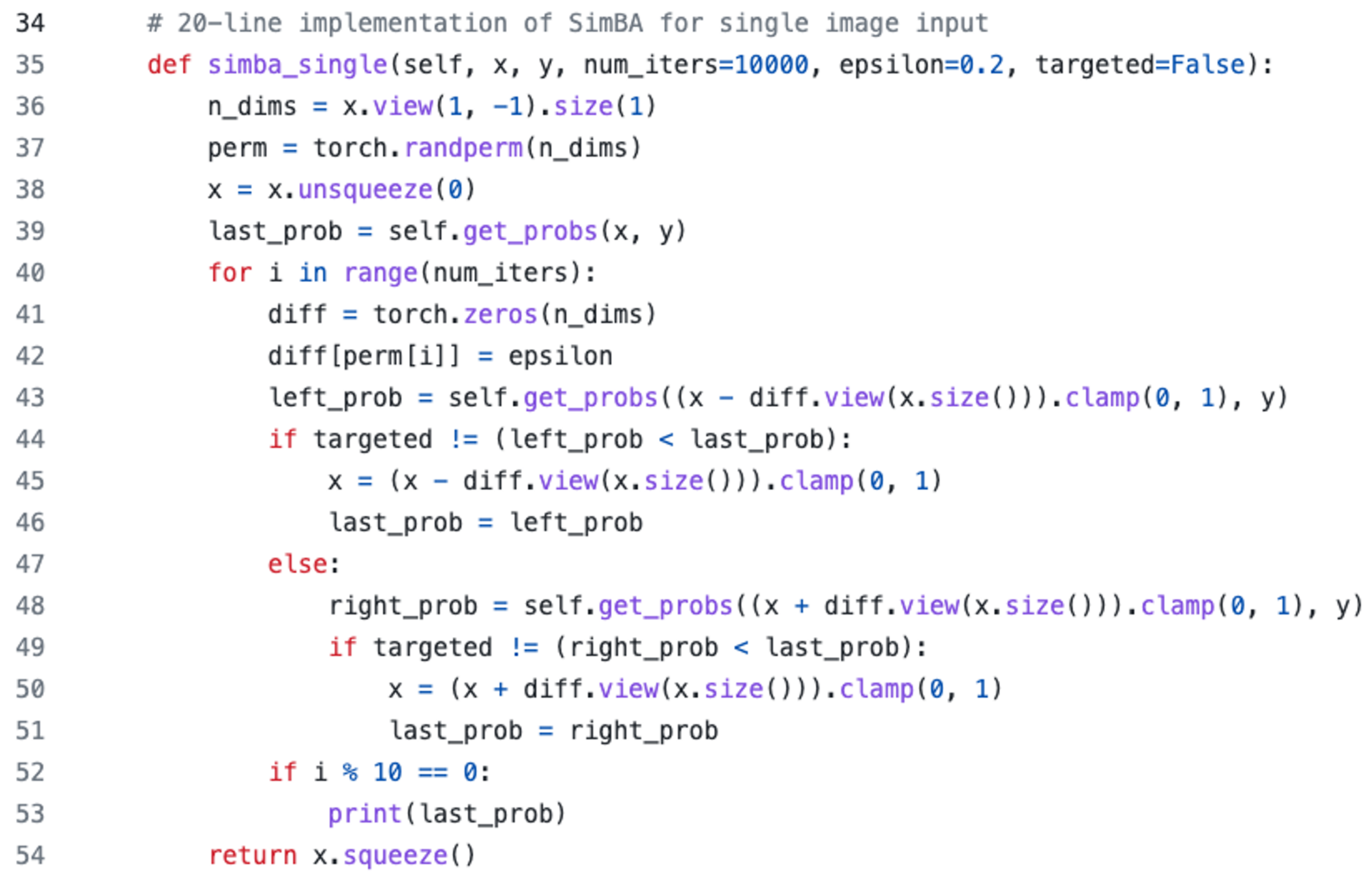
Algorithm



Algorithm 1 SimBA in Pseudocode

```
**procedure** SimBA($\mathbf{x}, y, Q, \epsilon$)
    $\delta = \mathbf{0}$
    $\mathbf{p} = p_h(y \mid \mathbf{x})$
    **while** $\mathbf{p}_y = \max_{y'} \mathbf{p}_{y'}$ **do**
        Pick randomly without replacement: $\mathbf{q} \in Q$
        **for** $\alpha \in \{\epsilon, -\epsilon\}$ **do**
            $\mathbf{p}' = p_h(y \mid \mathbf{x} + \delta + \alpha\mathbf{q})$
            **if** $\mathbf{p}'_y < \mathbf{p}_y$ **then**
                $\delta = \delta + \alpha\mathbf{q}$
                $\mathbf{p} = \mathbf{p}'$
                **break**
    **return** $\delta$
```

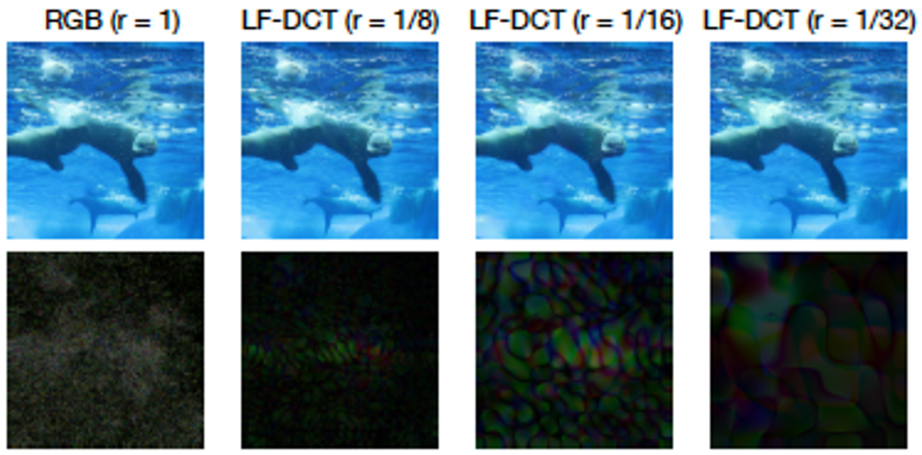
Code



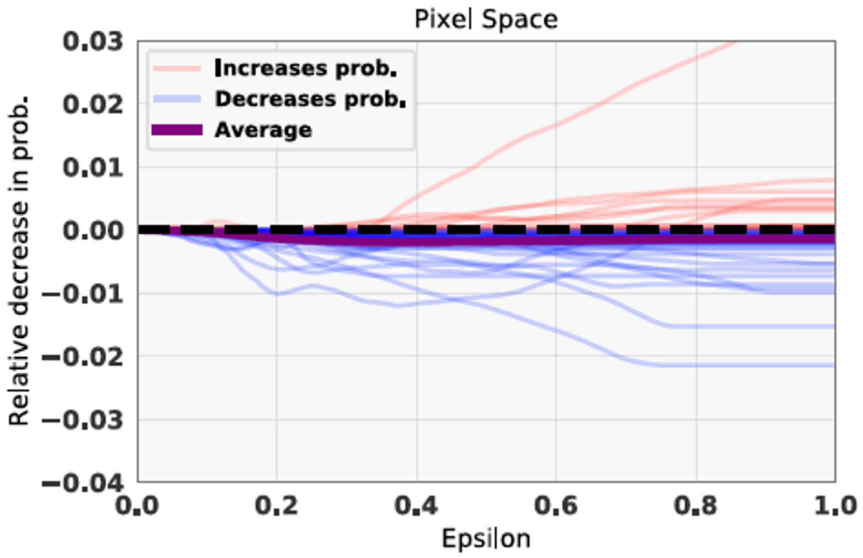
```python
# 20-line implementation of SimBA for single image input
def simba_single(self, x, y, num_iters=10000, epsilon=0.2, targeted=False):
    n_dims = x.view(1, -1).size(1)
    perm = torch.randperm(n_dims)
    x = x.unsqueeze(0)
    last_prob = self.get_probs(x, y)
    for i in range(num_iters):
        diff = torch.zeros(n_dims)
        diff[perm[i]] = epsilon
        left_prob = self.get_probs((x - diff.view(x.size())).clamp(0, 1), y)
        if targeted != (left_prob < last_prob):
            x = (x - diff.view(x.size())).clamp(0, 1)
            last_prob = left_prob
        else:
            right_prob = self.get_probs((x + diff.view(x.size())).clamp(0, 1), y)
            if targeted != (right_prob < last_prob):
                x = (x + diff.view(x.size())).clamp(0, 1)
                last_prob = right_prob
        if i % 10 == 0:
            print(last_prob)
    return x.squeeze()
```

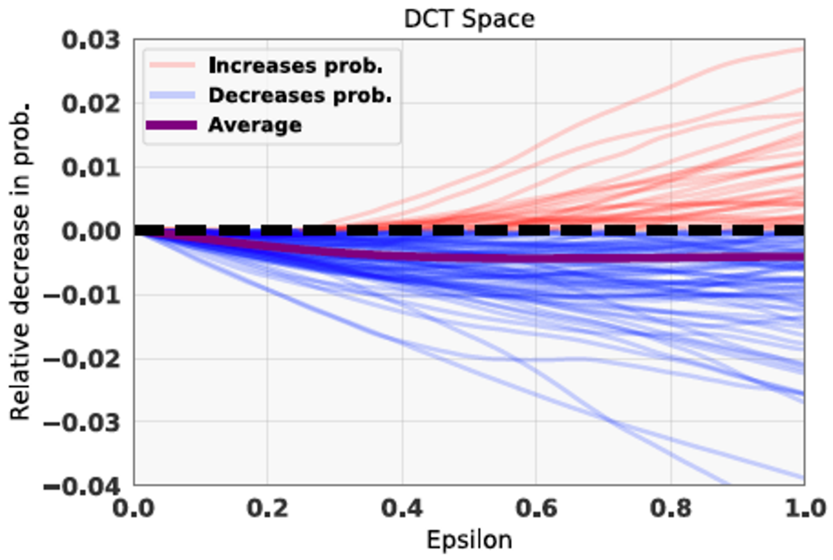
Discrete Cosine Transform (DCT)

- **Low Frequency Adversarial Perturbation - Guo et al. (2018)**
- DCT - used in signal processing and data compression
  - Transforms or decomposes signal into cosine wave components
- Critical content-defining information in images live in low end frequency spectrum [Wallace, 1991]
  - Density of adversarial examples is much higher in the low frequency domain



Impact of Learning Rate Choice

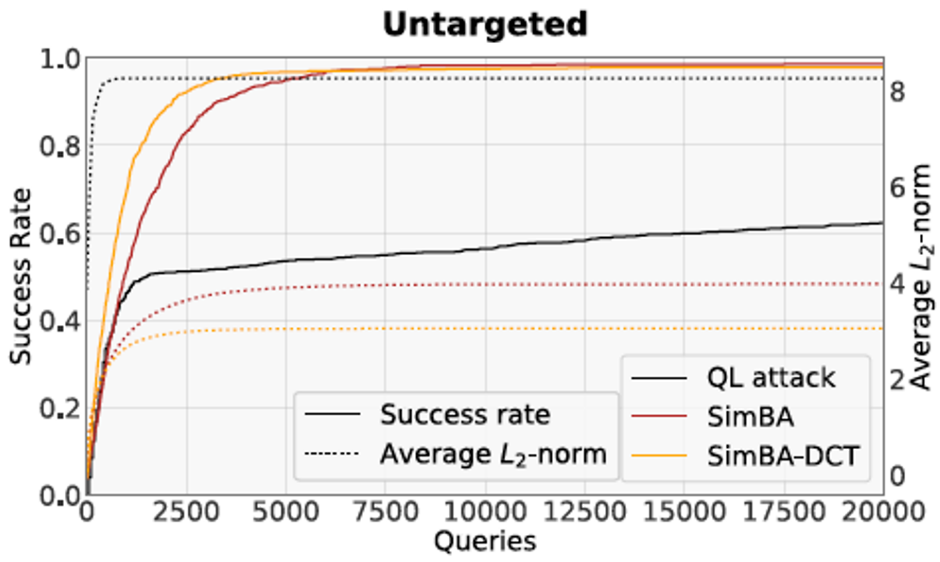


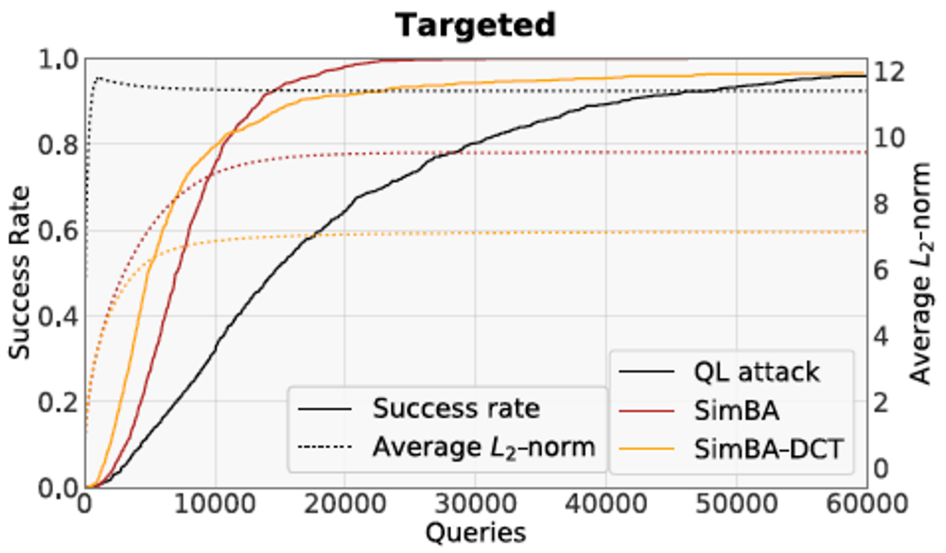


Bound Perturbation

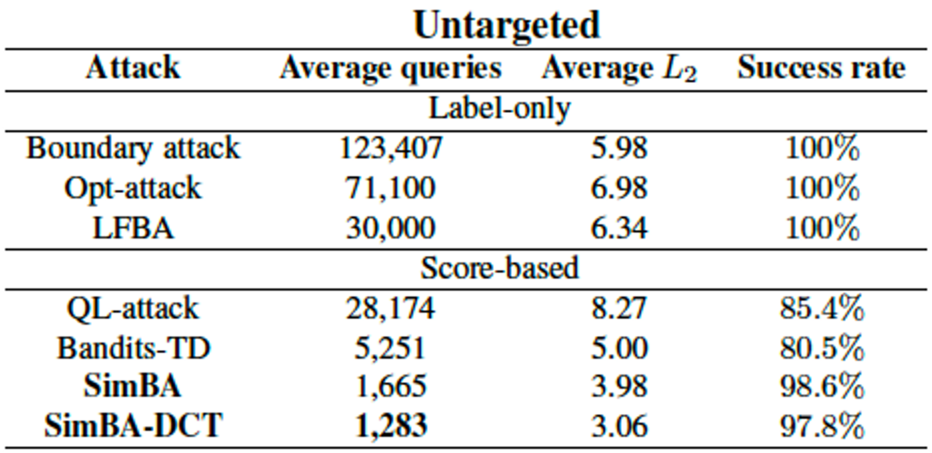
$$\|\delta_T\|_2^2 = \left\|\sum_{t=1}^{T} \alpha_t \mathbf{q}_t\right\|_2^2 = \sum_{t=1}^{T} \|\alpha_i \mathbf{q}_t\|_2^2 = \sum_{t=1}^{T} \alpha_t^2 \|\mathbf{q}_t\|_2^2$$
$$\leq T\epsilon^2.$$

Results


Untargeted
Success Rate
1.0
0.8
0.6
0.4
0.2
0.0
0
2500
5000
7500
10000
12500
15000
17500
20000
Queries
Average L_2-norm
8
6
4
2
0
Success rate
Average L_2-norm
QL attack
SimBA
SimBA-DCT

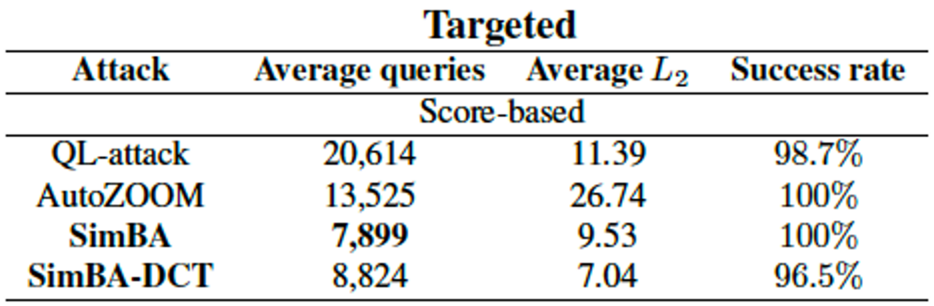

Targeted
Success Rate
1.0
0.8
0.6
0.4
0.2
0.0
0
10000
20000
30000
40000
50000
60000
Queries
Average L_2-norm
12
10
8
6
4
2
0
Success rate
Average L_2-norm
QL attack
SimBA
SimBA-DCT

Comparison with Other Attacks



Untargeted

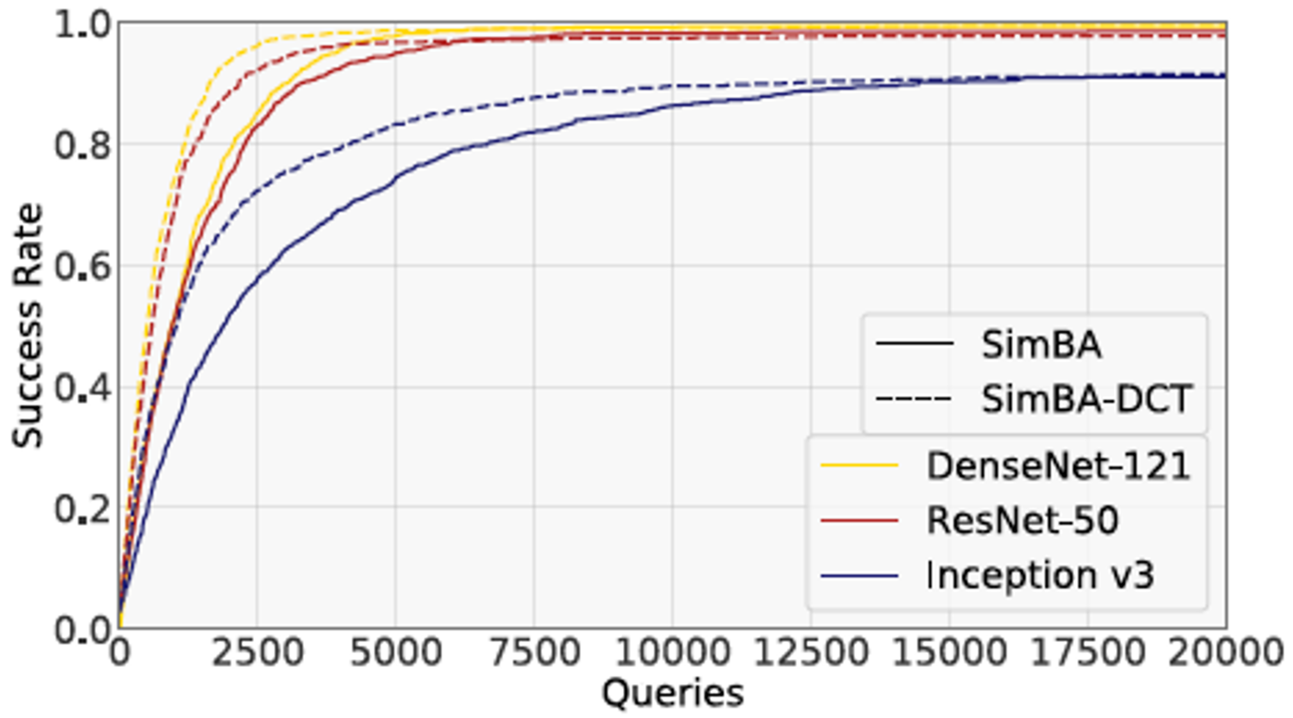
Attack	Average queries	Average L_2	Success rate
Label-only			
Boundary attack	123,407	5.98	100%
Opt-attack	71,100	6.98	100%
LFBA	30,000	6.34	100%
Score-based			
QL-attack	28,174	8.27	85.4%
Bandits-TD	5,251	5.00	80.5%
SimBA	1,665	3.98	98.6%
SimBA-DCT	**1,283**	3.06	97.8%



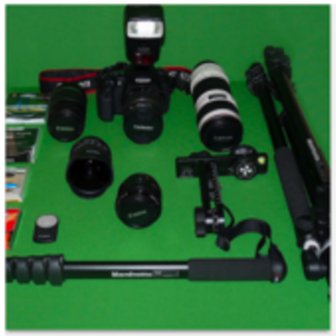
Targeted

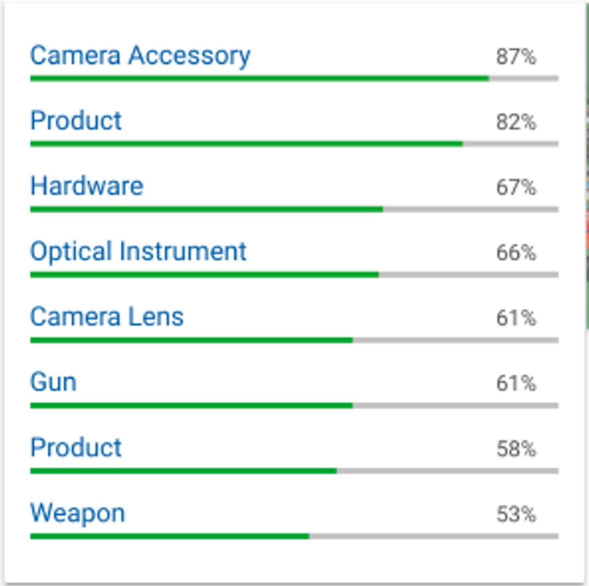
Attack	Average queries	Average L_2	Success rate
Score-based			
QL-attack	20,614	11.39	98.7%
AutoZOOM	13,525	26.74	100%
SimBA	**7,899**	9.53	100%
SimBA-DCT	8,824	7.04	96.5%

Results on Different CNN Architectures

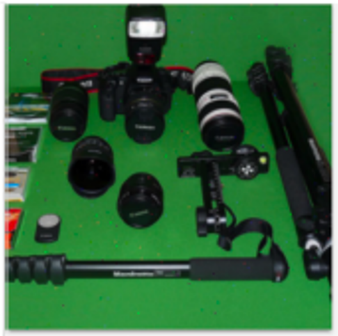


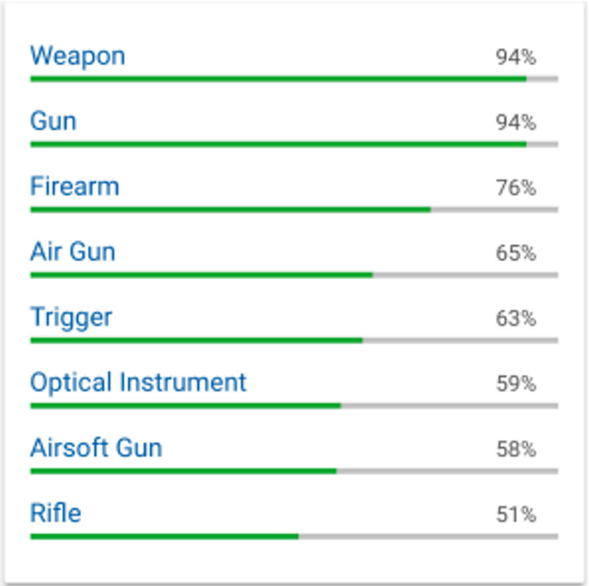
Attacking Google Cloud Vision API


origin_54.BMP



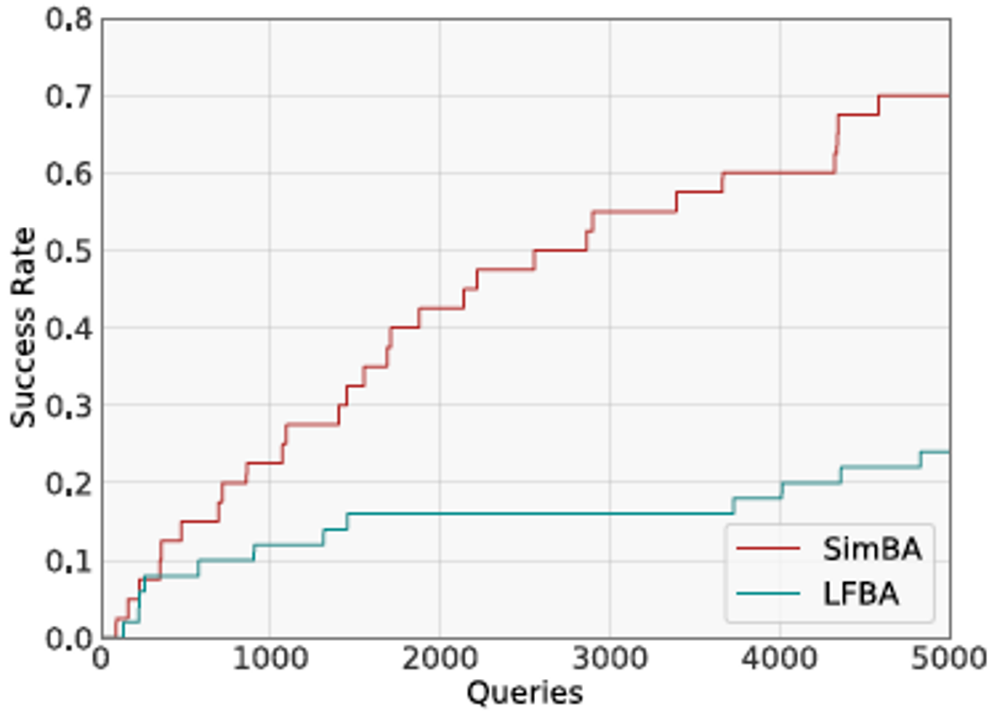
Camera Accessory	87%
Product	82%
Hardware	67%
Optical Instrument	66%
Camera Lens	61%
Gun	61%
Product	58%
Weapon	53%


after_54.BMP



Weapon	94%
Gun	94%
Firearm	76%
Air Gun	65%
Trigger	63%
Optical Instrument	59%
Airsoft Gun	58%
Rifle	51%

Success Rate on Google Cloud Vision



Strengths

- Simple iterative approach
- Does not need surrogate training data or surrogate model
- Works on both targeted and untargeted attacks
- Propose the discrete cosine basis as efficient way to find orthogonal search vectors
    - Well-known technique from computer vision used for image compression
- Code is publicly available

Limitations

- Only works on cases where the model provides a continuous-valued confidence scores.
- Depending on the dataset and problem, the proposed DCT can result in a faster rate of descent but may sometimes fail to produce adversarial perturbation for some images.

Discussion/Future Direction

- Can this approach be generalized easily for other types of data (not images)?
- Can we make this approach faster and efficient?
    - Adaptive selection of step size/learning rate
    - Other methods for finding set of orthogonal search directions

Towards Deep Learning Models Resistant to Adversarial Attacks

Aleksander Madry, Aleksander Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu.
ICLR 2018

Presented by: Jaydeep Borkar. Oct 4 2021.

TL;DR

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{x' \in P(x_i)} L(f_\theta(x'), y_i) \ .$$

How to make models adversarially robust?: Adversarial Training Part 1

Step 1: Find an adversarial example *x'* for feeding it to classifier *f*

$$\mathbb{E}_{x,y\sim D}\left[\max_{x'\in P(x)} L(f(x'), y)\right]$$

Also called as robust loss

P(*x*) is a set of pre-defined perturbations.

Universal PGD Attack

- If we can find a classifier with a small robust loss, that classifier will be robust to any perturbation in set *P*. Hence, this defense should work for other attacks.
- Prior defenses focussed only on specific attack methods (such as FGSM), which were easily broken by stronger attack methods.
- Aims universal robustness guarantee for all perturbations in $P(x)$
    - Here, they focus on l_∞ perturbations.
    - They use PGD, a standard method in constrained optimization

Adversarial Training Part 2

Now we find model parameters θ such that the classifier f_ϑ has a small empirical loss even when examples are adversarially perturbed.

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{x' \in P(x_i)} L(f_\theta(x'), y_i) \ .$$

This is a min-max optimization/saddle point problem. We need to solve this problem to get an adversarially robust classifier.

Solving the min-max problem

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{x' \in P(x_i)} L(f_\theta(x'), y_i) .$$

- We can split the problem into inner maximization problem and outer minimization problem.
- The inner maximization is non-concave and outer minimization is non-convex.

Solving the Outer Minimization

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{x' \in P(x_i)} L(f_\theta(x'), y_i) \ .$$

We can apply SGD to solve the outer minimization problem. But in order to run SGD, we first need to find gradients of the robust loss (inner maximization).

$$\phi_{x,y}(\theta) = \max_{x' \in P(x)} L(f_\theta(x'), y) \ .$$

Robust loss

Solving the Outer Minimization

$$\phi_{x,y}(\theta) = \max_{x' \in P(x)} L(f_\theta(x'), y) \ .$$

- We can find the gradient of robust loss ϕ w.r.t θ by finding a constrained maximizer *x** of the inner maximization problem.
- Then we can use this maximizer as our actual data point and compute the gradient at *x*.* We can use this gradient to run SGD for the outer minimization problem.

$$\nabla_\theta \phi_{x,y}(\theta) = \nabla_\theta L(f_\theta(x^*), y) \ .$$

Inner Maximization

We apply PGD to get the inner maximizer (adversarial example) since the problem is constrained.

$$\Pi_C(x + \eta \cdot \mathrm{sign}(\nabla L(x, y))) \ .$$

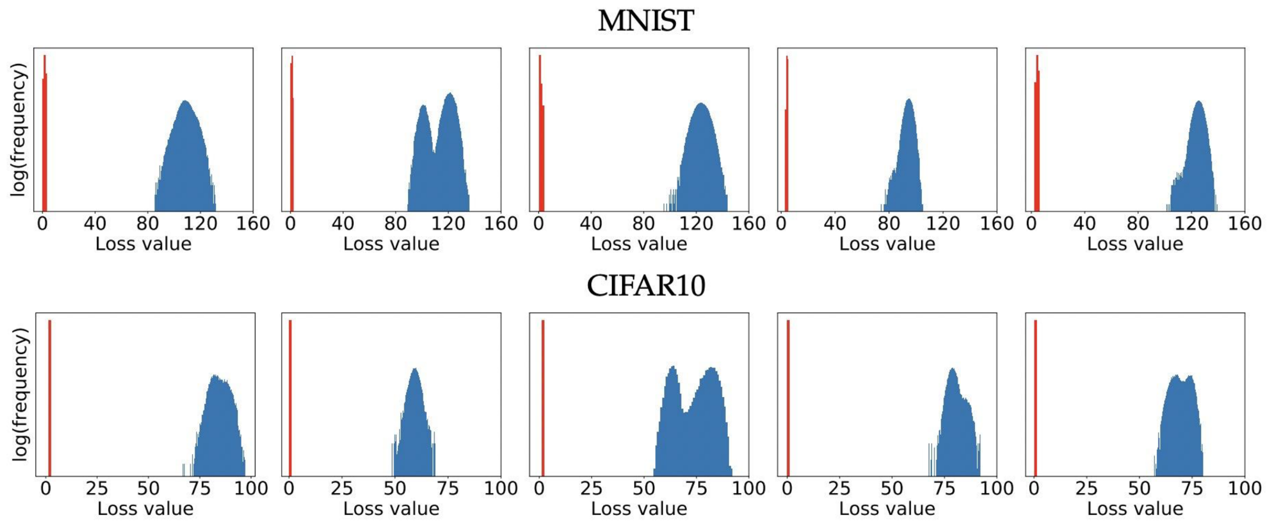
They use signs of the gradient instead of the gradient itself.

This gives us maximizer/adversarial example, which we can use further to compute the gradient for SGD to solve the outer minimization problem.

Adversarial Training Steps

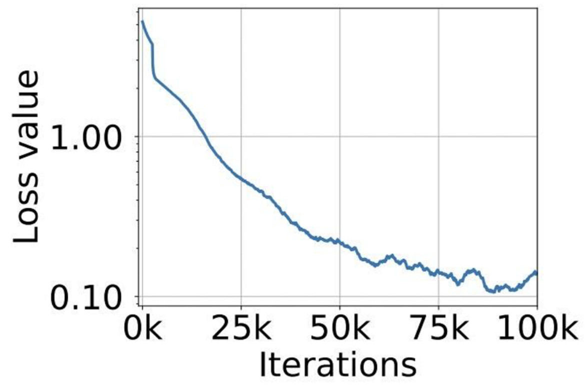
1. Sample a data point *x, y*
2. Compute the maximizer *x** of the robust loss $\phi_{x,y}(\theta)$ [PGD]
3. Compute the gradient g = $\nabla_\theta L(f_\theta(x^*), y)$
4. Update θ with the gradient g [SGD]
5. Repeat Steps 1 - 4 until convergence.

Experiments

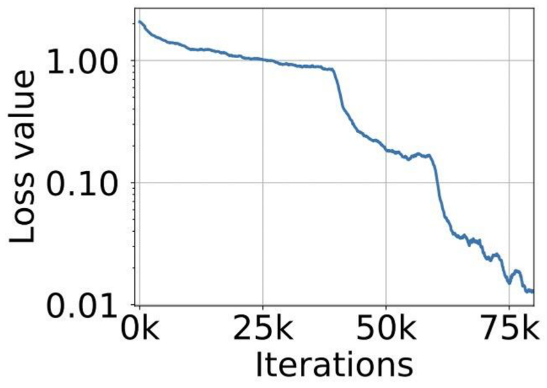


PGD from 10^5 uniformly sampled points. The blue histogram corresponds to standard loss and red corresponds to loss on adversarially trained model. The final loss values are very concentrated.

Loss During Adversarial Training



(a) MNIST



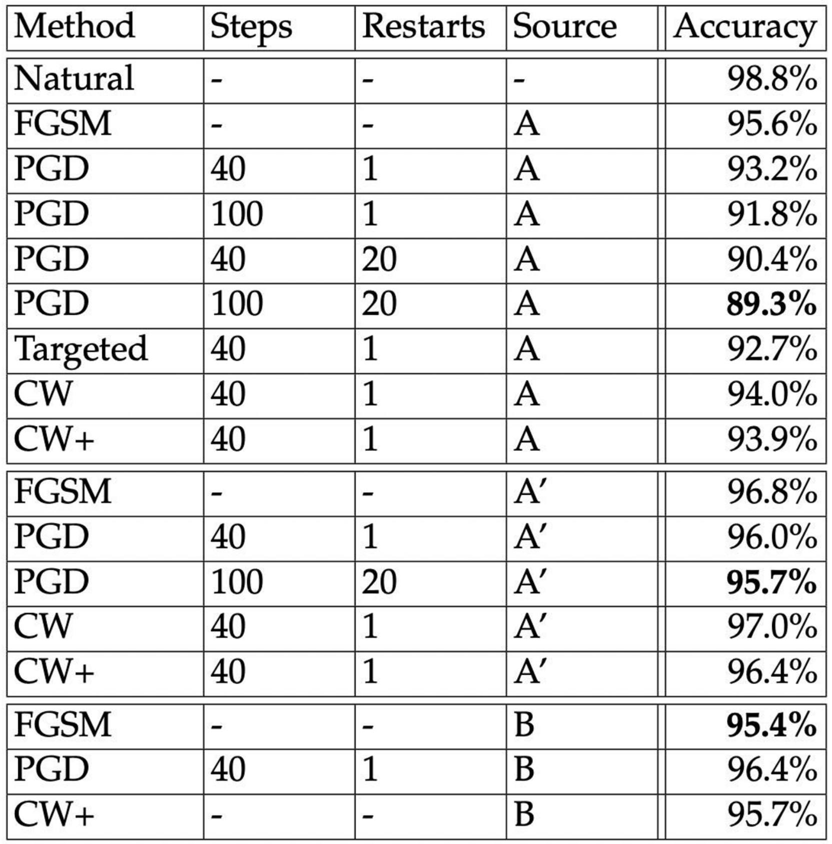
(b) CIFAR10

Cross-entropy loss on adversarial examples during training against a PGD adversary. The sharp drops in the CIFAR10 plot correspond to decreases in training step size. These plots illustrate that we can consistently reduce the value of the inner problem to produce an increasingly robust classifier.

Results

- Training with FGSM attacks does not help much
  - Model is not resilient to PGD
- Models with small capacity have low accuracy when trained adversarially with PGD
- Transferability
  - Model capacity improves resistance to transferability
  - Training with stronger adversaries improves resistance to transferability
- Experiment on adv training with PGD and model with sufficient capacity show robustness
  - 89.3% accuracy against PGD attack on MNIST

Attacks on Adversarially Trained Networks

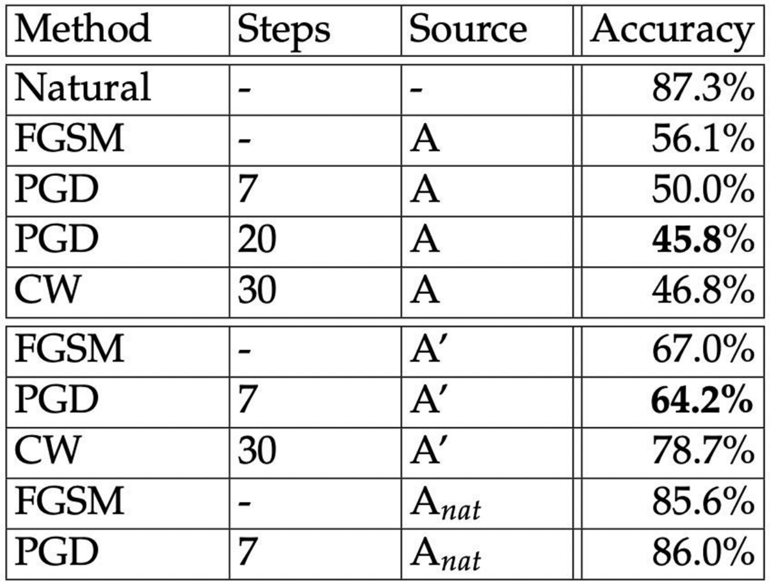


Method	Steps	Restarts	Source	Accuracy
Natural	-	-	-	98.8%
FGSM	-	-	A	95.6%
PGD	40	1	A	93.2%
PGD	100	1	A	91.8%
PGD	40	20	A	90.4%
PGD	100	20	A	**89.3%**
Targeted	40	1	A	92.7%
CW	40	1	A	94.0%
CW+	40	1	A	93.9%
FGSM	-	-	A′	96.8%
PGD	40	1	A′	96.0%
PGD	100	20	A′	**95.7%**
CW	40	1	A′	97.0%
CW+	40	1	A′	96.4%
FGSM	-	-	B	**95.4%**
PGD	40	1	B	96.4%
CW+	-	-	B	95.7%

MNIST: Performance of the adversarially trained network against different adversaries for $\varepsilon = 0.3$

A: white box attack, A’ black box, B: some different architecture.

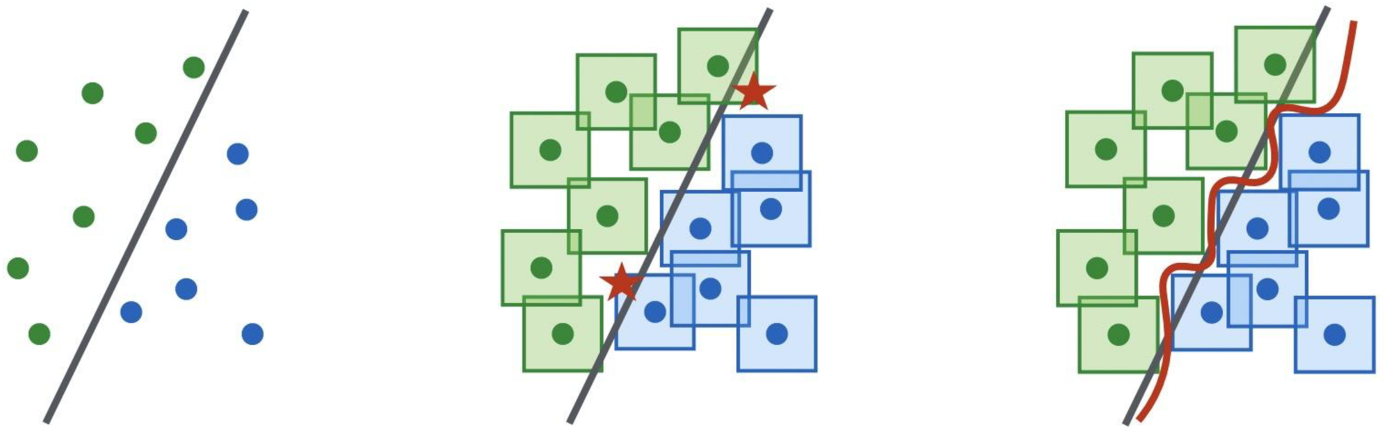
Attacks on Adversarially Trained Networks



Method	Steps	Source	Accuracy
Natural	-	-	87.3%
FGSM	-	A	56.1%
PGD	7	A	50.0%
PGD	20	A	**45.8%**
CW	30	A	46.8%
FGSM	-	A′	67.0%
PGD	7	A′	**64.2%**
CW	30	A′	78.7%
FGSM	-	A_{nat}	85.6%
PGD	7	A_{nat}	86.0%

CIFAR10: Performance of the adversarially trained network against different adversaries for ε =8.

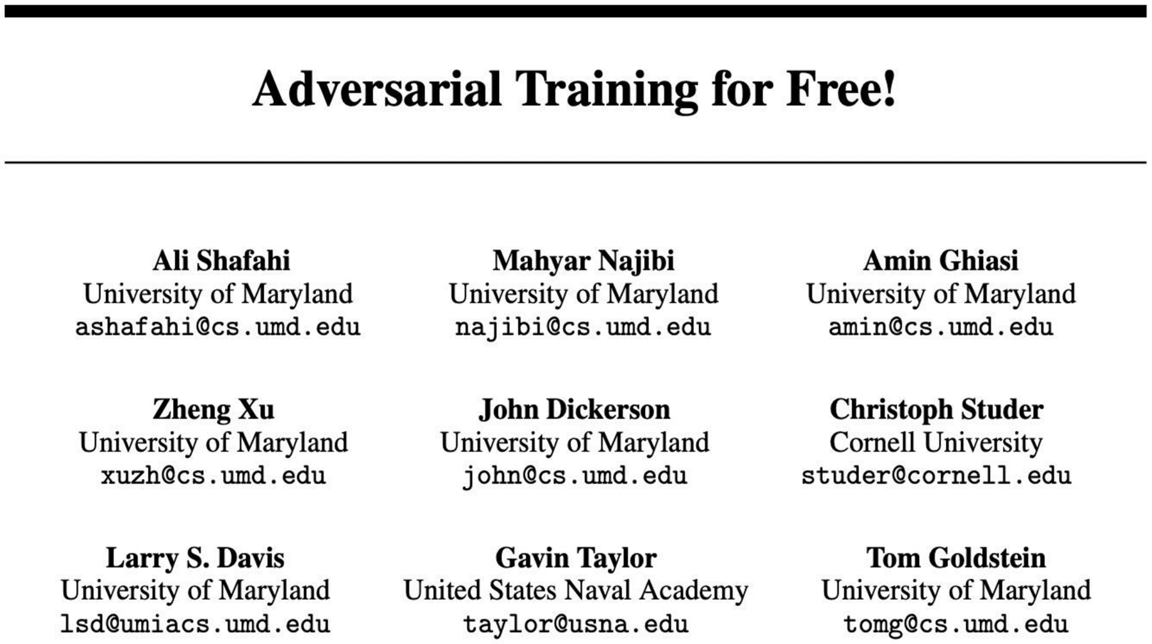
Network Capacity and Adversarial Robustness



Left: a set of points that can be separated with a simple decision boundary. Middle: the simple decision boundary doesn't work for l_∞ balls. Right: a more complex decision boundary separates l_∞ balls. The resulting classifier is robust to adversarial examples with bounded l_∞-norm perturbations.

Challenges

1. Adversarial training is computationally very expensive.



Adversarial Training for Free!

Ali Shafahi
University of Maryland
ashafahi@cs.umd.edu

Mahyar Najibi
University of Maryland
najibi@cs.umd.edu

Amin Ghiasi
University of Maryland
amin@cs.umd.edu

Zheng Xu
University of Maryland
xuzh@cs.umd.edu

John Dickerson
University of Maryland
john@cs.umd.edu

Christoph Studer
Cornell University
studer@cornell.edu

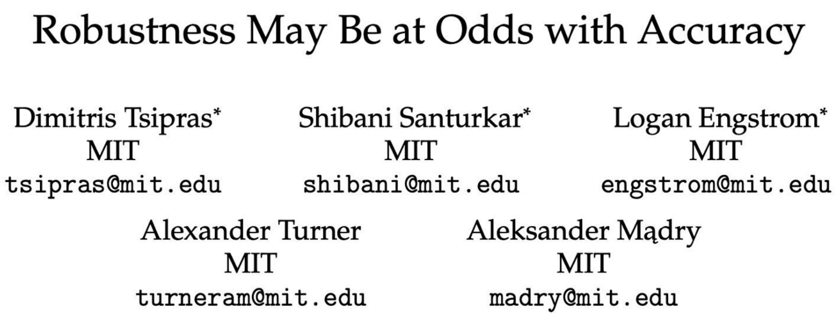
Larry S. Davis
University of Maryland
lsd@umiacs.umd.edu

Gavin Taylor
United States Naval Academy
taylor@usna.edu

Tom Goldstein
University of Maryland
tomg@cs.umd.edu

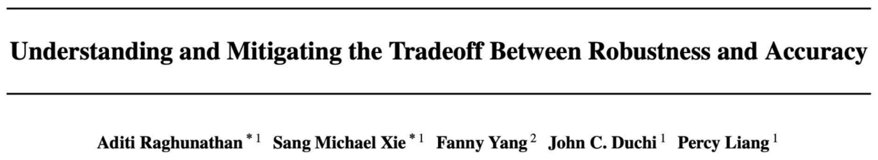
Challenges

2. Trade-off with standard accuracy



Robustness May Be at Odds with Accuracy

Dimitris Tsipras* MIT tsipras@mit.edu
Shibani Santurkar* MIT shibani@mit.edu
Logan Engstrom* MIT engstrom@mit.edu
Alexander Turner MIT turneram@mit.edu
Aleksander Mądry MIT madry@mit.edu



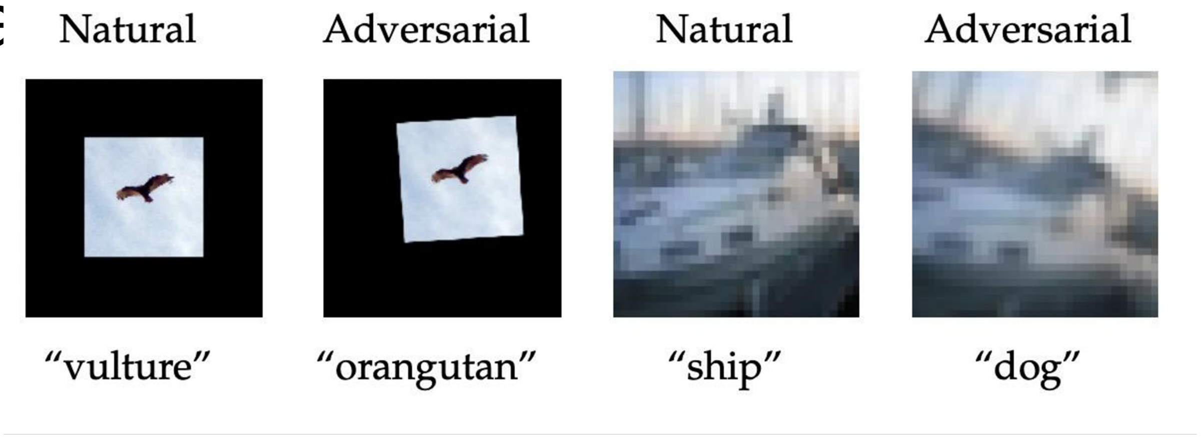
Understanding and Mitigating the Tradeoff Between Robustness and Accuracy

Aditi Raghunathan [*1] **Sang Michael Xie** [*1] **Fanny Yang** [2] **John C. Duchi** [1] **Percy Liang** [1]

Challenges

3. Can adversarial training stand non L_p norm

a



Engstrom et al.

Discussion

- PGD adversaries vs first-order methods (FGSM)
- Adversarial training as a defense
  - Limitations: high computational complexity
  - Attack does not work well on L2 norm
- Adversarially trained network is resilient up to the distance used for training the adversarial examples
  - It quickly degrades once it is attacked with adversarial examples of larger distance