A few new papers about adversarial attack

Presentor: Ji Gao

@ https://qdata.github.io/deep2Read/

2018 Fall

Outline

- Adversarial Patch
- Adversarial Logit pairing
- Adversarial Perturbation against Deep Neural Networks for Malware Classification
- Black-Box Attacks against RNN based Malware Detection Algorithms
- Ensemble Adversarial Training: Attacks and Defenses
- Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods
- Extending Defensive Distillation

Adversarial patch

Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, Justin Gilmer

- Arxiv 2017
- Example Code(Tensorflow): https://github.com/tensorflow/cleverhans/tree/master/examples/adversarial_patch

Motivation

- Is it able to create a patch that fool the classifier?
    - Universal: On every images
    - Robust
    - Targeted
    - Real world

Real world example

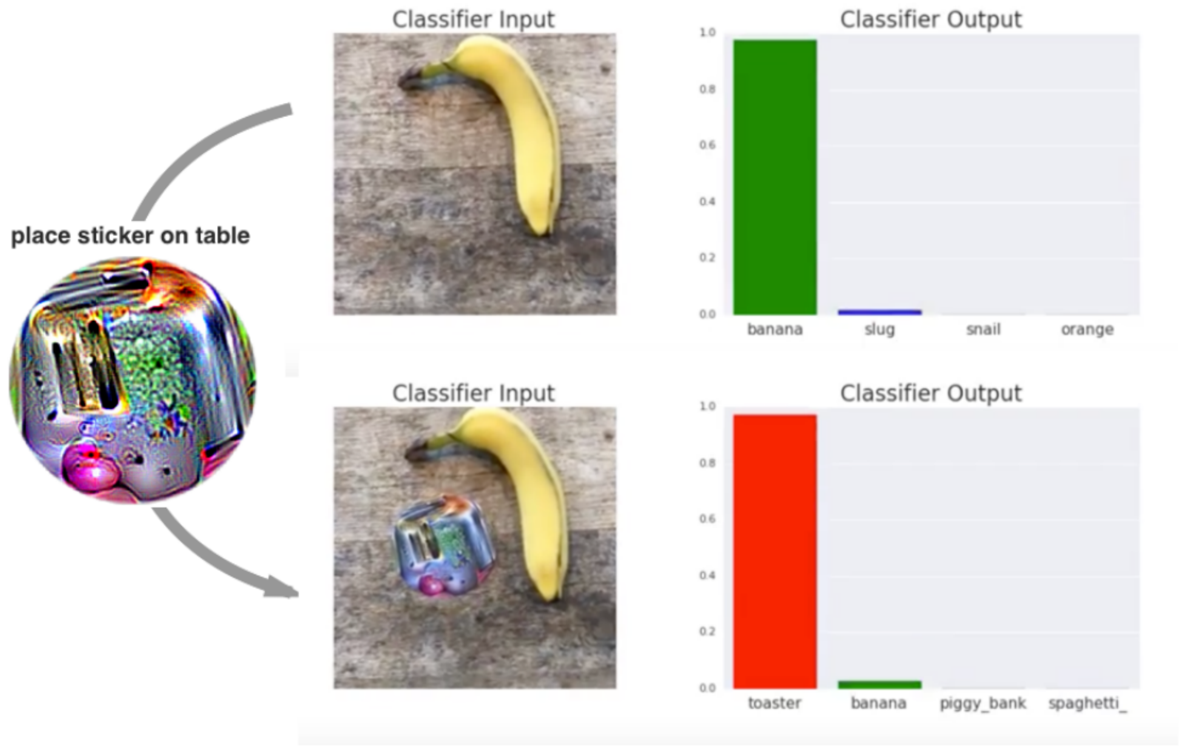
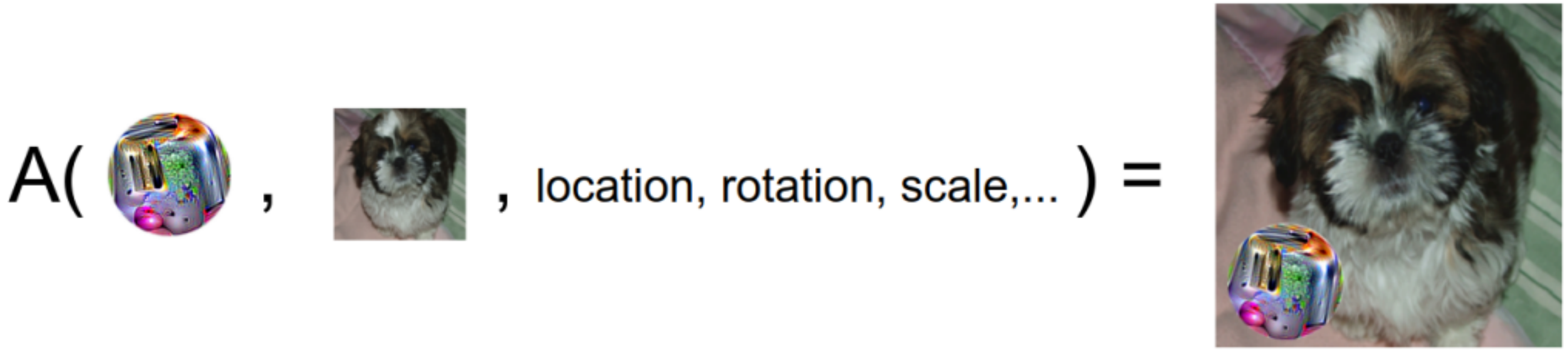


Figure 1: A real-world attack on VGG16, using a physical patch generated by the white-box ensemble method described in Section 3. When a photo of a tabletop with a banana and a notebook (top photograph) is passed through VGG16, the network reports class 'banana' with 97% confidence (top plot). If we physically place a sticker targeted to the class "toaster" on the table (bottom photograph), the photograph is classified as a toaster with 99% confidence (bottom plot). See the following video for a full demonstration: `https://youtu.be/i1sp4X57TL4`

Method

- Allow the patch to take any shape, any place, rotation/scale(Aim for the real-world)
- Define a transformation function:



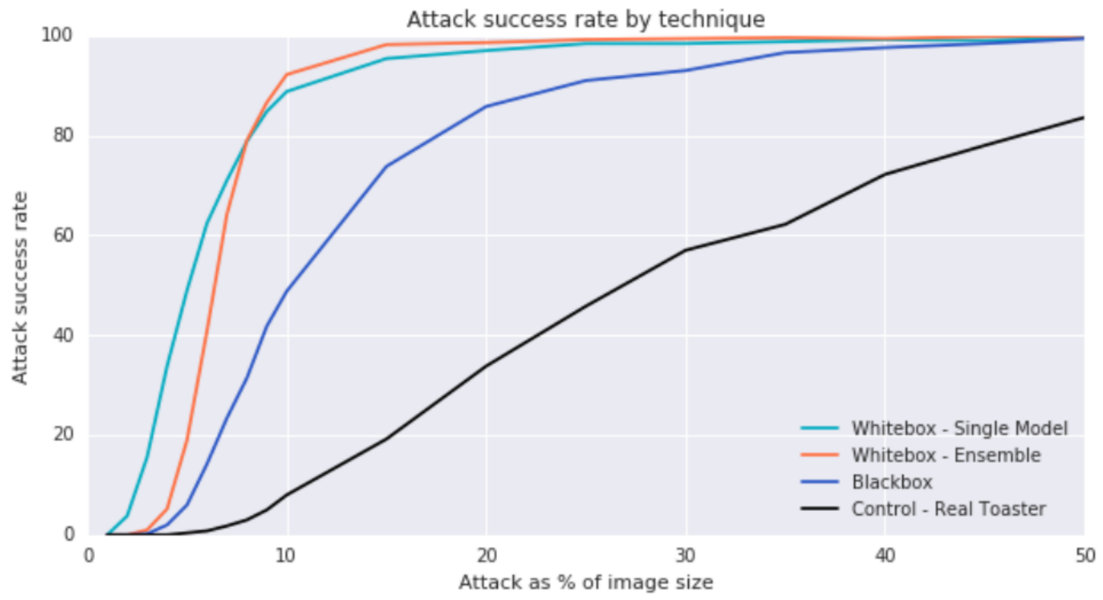
Method

- Idea from “Expectation over transformation (From *Synthesizing robust adversarial examples, Arxiv 2017*)”

$$\widehat{p} = \arg\max_{p} \mathbb{E}_{x\sim X, t\sim T, l\sim L}\left[\log \Pr(\widehat{y}|A(p, x, l, t)\right]$$

- Generally, using projected gradient descent to optimize this object

Result



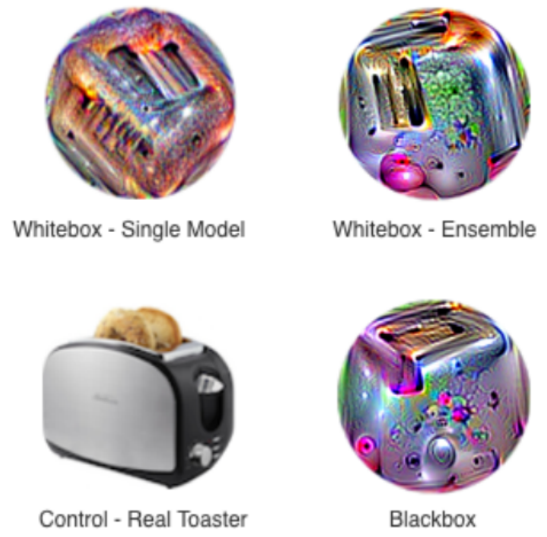
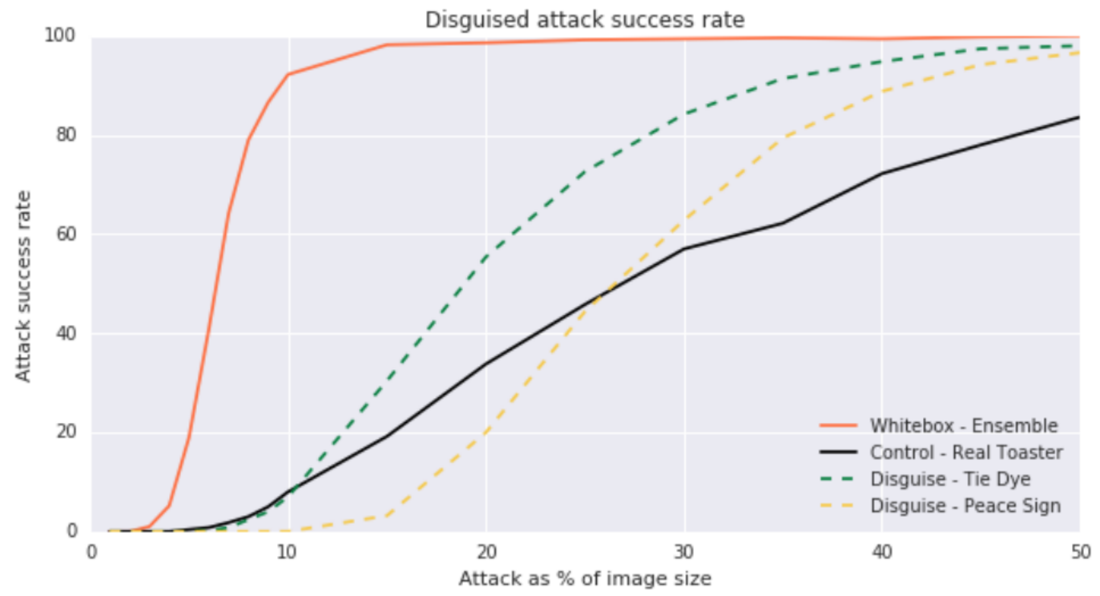


Figure 3: A comparison of different methods for creating adversarial patches. Note that these success rates are for random placements of the patch on top of the image. Each point in the plot is computed by applying the patch to 400 randomly chosen test images at random locations in these images. This is done for various scales of the patch as a fraction of the size of the image, each scale is tested independently on 400 images.

Result – additional L2 penalty to a known image



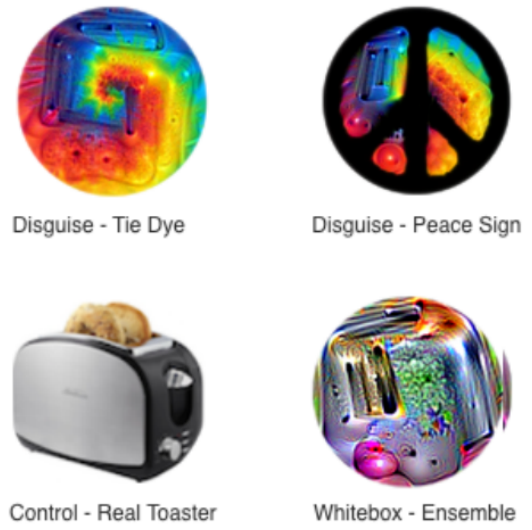


Figure 4: A comparison of patches with various disguises. We find that we can disguise the patch and retain much of its power to fool the classifier.

Adversarial logit pairing

Harini Kannan, Alexey Kurakin, Ian Goodfellow, Google Brain

- Defense against adversarial attack
- Use “logit paring”: Matching the logits
- Use adversarial training

$$\arg\min_{\theta} \Big[\mathbb{E}_{(x,y)\in\hat{p}_{\text{data}}} \Big(\max_{\delta\in S} L(\theta, x+\delta, y) \Big) + \mathbb{E}_{(x,y)\in\hat{p}_{\text{data}}} \Big(L(\theta, x, y) \Big) \Big] \quad (2)$$

Defense?

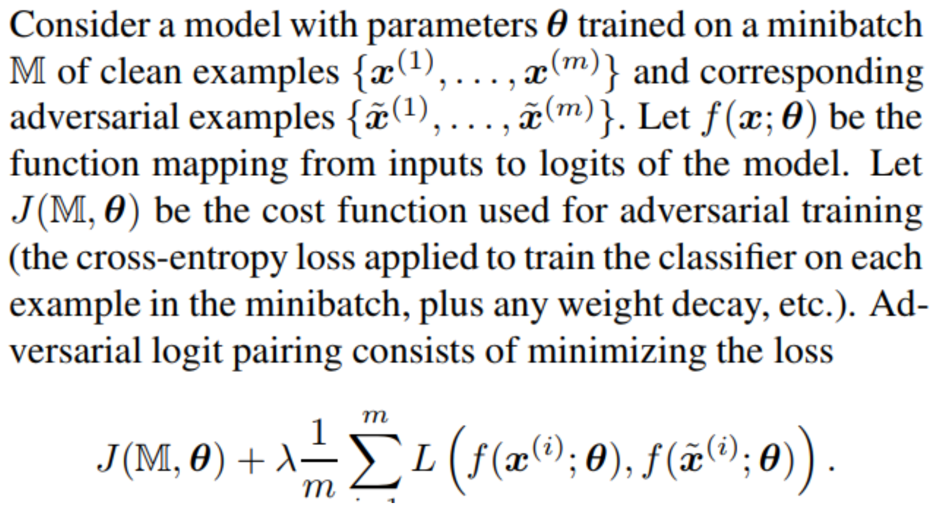
- Madry et al. (2017) suggests that PGD is a universal first order adversary – in other words, developing robustness against PGD attacks also implies resistance against many other first order attacks.

Logit pairing

- Logit pairing: pushing the distance between f(x) and f(x’)

$$\lambda L\left(f(\boldsymbol{x}), f(\boldsymbol{x}')\right)$$

- L: Simple L2 Loss

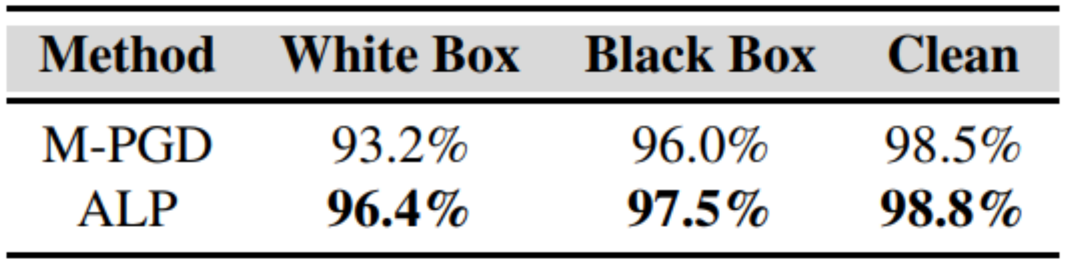


Consider a model with parameters $\boldsymbol{\theta}$ trained on a minibatch $\mathbb{M}$ of clean examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ and corresponding adversarial examples $\{\tilde{\boldsymbol{x}}^{(1)}, \ldots, \tilde{\boldsymbol{x}}^{(m)}\}$. Let $f(\boldsymbol{x}; \boldsymbol{\theta})$ be the function mapping from inputs to logits of the model. Let $J(\mathbb{M}, \boldsymbol{\theta})$ be the cost function used for adversarial training (the cross-entropy loss applied to train the classifier on each example in the minibatch, plus any weight decay, etc.). Adversarial logit pairing consists of minimizing the loss

$$J(\mathbb{M}, \boldsymbol{\theta}) + \lambda \frac{1}{m} \sum^{m} L\left(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), f(\tilde{\boldsymbol{x}}^{(i)}; \boldsymbol{\theta})\right).$$

Experiment

- Baseline: Adversarial training



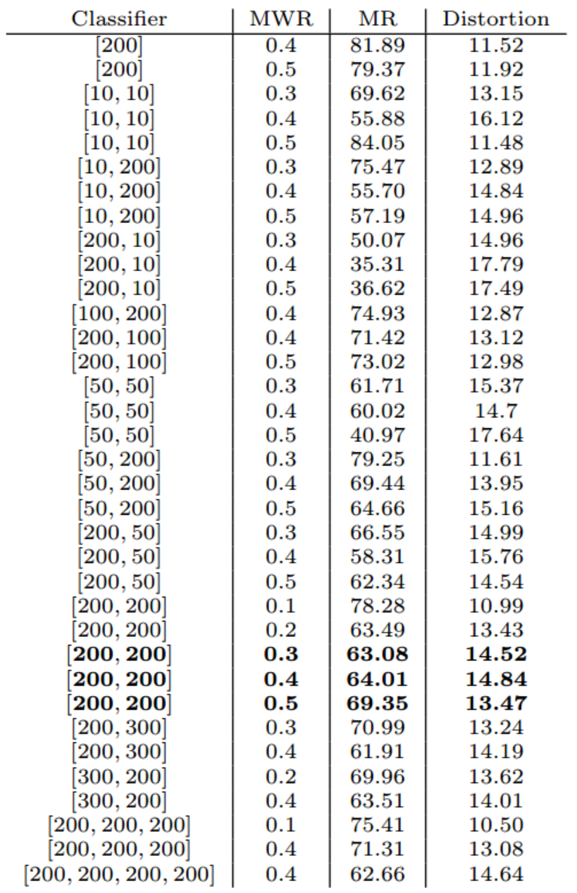
Method	White Box	Black Box	Clean
M-PGD	93.2%	96.0%	98.5%
ALP	**96.4%**	**97.5%**	**98.8%**

Table 1. Comparison of adversarial logit pairing and vanilla adversarial training on MNIST. All accuracies reported are for the PGD attack.

Adversarial Perturbation against Deep Neural Networks for Malware Classification

Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Arxiv

- Problem setting:
  - Andriod apis
  - Represent the software as binary features
  - Use ~~deep~~ small NN as the classifier, have > 95% accuracy
- Adv perturbation:
  - L1norm - JSMA
- Result on defense:
  - Feature reduction -> Doesn't work
  - Defense Distillation -> Have some effect, but also reduce the performance
  - Retraining -> Good
- Summary:
  - Small network still suffer adversarial sample



Classifier	MWR	MR	Distortion
[200]	0.4	81.89	11.52
[200]	0.5	79.37	11.92
[10, 10]	0.3	69.62	13.15
[10, 10]	0.4	55.88	16.12
[10, 10]	0.5	84.05	11.48
[10, 200]	0.3	75.47	12.89
[10, 200]	0.4	55.70	14.84
[10, 200]	0.5	57.19	14.96
[200, 10]	0.3	50.07	14.96
[200, 10]	0.4	35.31	17.79
[200, 10]	0.5	36.62	17.49
[100, 200]	0.4	74.93	12.87
[200, 100]	0.4	71.42	13.12
[200, 100]	0.5	73.02	12.98
[50, 50]	0.3	61.71	15.37
[50, 50]	0.4	60.02	14.7
[50, 50]	0.5	40.97	17.64
[50, 200]	0.3	79.25	11.61
[50, 200]	0.4	69.44	13.95
[50, 200]	0.5	64.66	15.16
[200, 50]	0.3	66.55	14.99
[200, 50]	0.4	58.31	15.76
[200, 50]	0.5	62.34	14.54
[200, 200]	0.1	78.28	10.99
[200, 200]	0.2	63.49	13.43
[200, 200]	**0.3**	**63.08**	**14.52**
[200, 200]	**0.4**	**64.01**	**14.84**
[200, 200]	**0.5**	**69.35**	**13.47**
[200, 300]	0.3	70.99	13.24
[200, 300]	0.4	61.91	14.19
[300, 200]	0.2	69.96	13.62
[300, 200]	0.4	63.51	14.01
[200, 200, 200]	0.1	75.41	10.50
[200, 200, 200]	0.4	71.31	13.08
[200, 200, 200, 200]	0.4	62.66	14.64

Table 4: Performance of our adversarial sampling strategy. The misclassification rates (MR) and required average distortion (in number of added features) with a threshold of 20 modifications are given

Black-Box Attacks against RNN based Malware Detection Algorithms

- Weiwei Hu and Ying Tan, From Peking U

- Target model: Malware detection classifier

Formulate Malware detection as a sequential classification problem

Software = A sequence of APIs x_1 , x_2 , x_3 …x_n

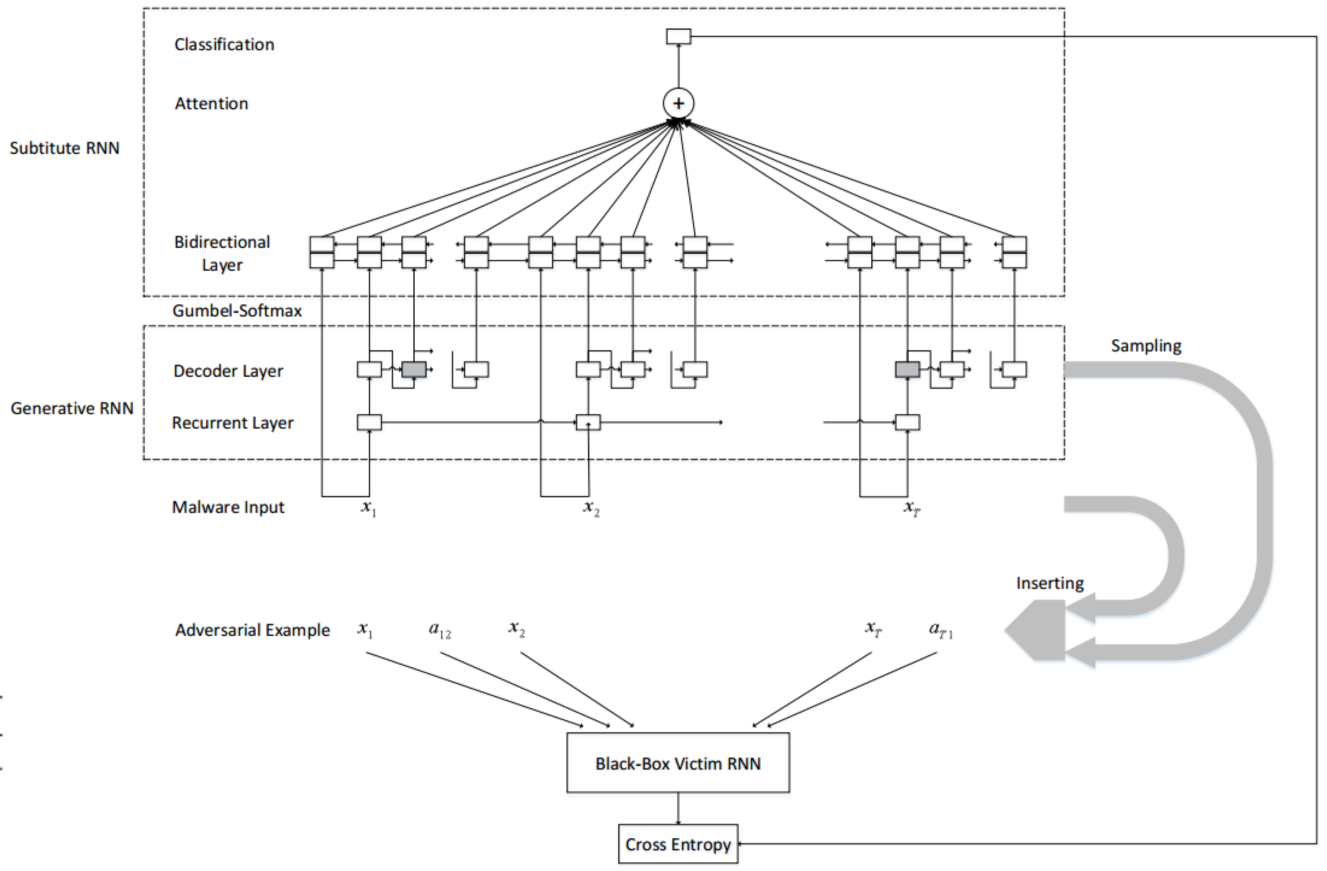
LSTM achieves 90%+ accuracy on their malware dataset

Method

- Generator:
  - -A generative model trained to insert malware API into the malware
- Subtitute RNN -> Simulate the original RNN

- Data:
180 crawled mal/benign softwares

	Training Set		Test Set	
	Original	Adver.	Original	Adver.
LSTM	92.54%	12.10%	90.74%	11.95%
BiLSTM	92.21%	1.06%	90.93%	0.95%
LSTM-Average	93.87%	1.40%	93.53%	1.36%
BiLSTM-Average	92.92%	1.83%	92.51%	1.67%
LSTM-Attention	93.67%	0.44%	92.45%	0.51%
BiLSTM-Attention	93.73%	3.02%	92.99%	3.03%



Black-box Attacks against RNN based malware Detection Algorithm

- Summary
  - Interesting target model: Easy to attack
  - Simulation of model to make a black-box attack, and have great performance: Worth a try
  - Generate adversarial samples using NN?

Ensemble Adversarial Training: Attacks and Defenses

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, Patrick McDaniel

Three points:

1. Current defense approach are vulnerable to Black-box attack

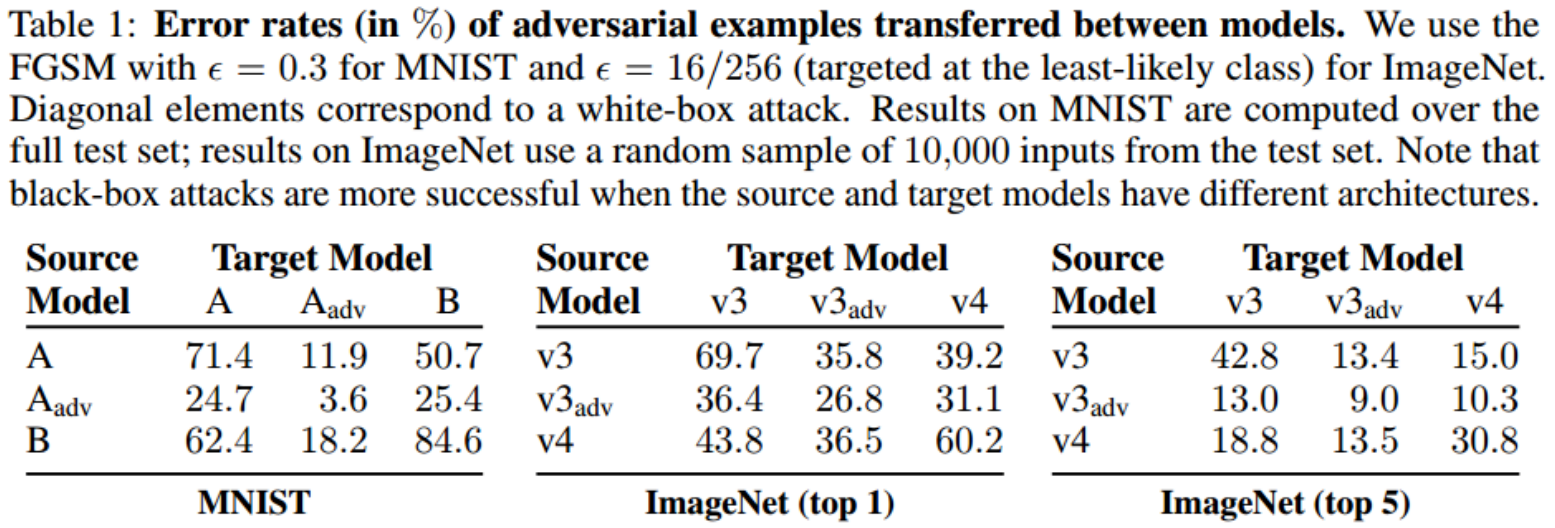


Table 1: **Error rates (in %) of adversarial examples transferred between models.** We use the FGSM with $\epsilon = 0.3$ for MNIST and $\epsilon = 16/256$ (targeted at the least-likely class) for ImageNet. Diagonal elements correspond to a white-box attack. Results on MNIST are computed over the full test set; results on ImageNet use a random sample of 10,000 inputs from the test set. Note that black-box attacks are more successful when the source and target models have different architectures.

Source	**Target Model**		
Model	A	A_{adv}	B
A	71.4	11.9	50.7
A_{adv}	24.7	3.6	25.4
B	62.4	18.2	84.6

MNIST

Source	**Target Model**		
Model	v3	$v3_{adv}$	v4
v3	69.7	35.8	39.2
$v3_{adv}$	36.4	26.8	31.1
v4	43.8	36.5	60.2

ImageNet (top 1)

Source	**Target Model**		
Model	v3	$v3_{adv}$	v4
v3	42.8	13.4	15.0
$v3_{adv}$	13.0	9.0	10.3
v4	18.8	13.5	30.8

ImageNet (top 5)

Ensemble Adversarial Training: Attacks and Defenses

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, Patrick McDaniel

2.A new randomized attack:

$$x' = x + \alpha \cdot \mathbf{sign}(\mathcal{N}(\mathbf{0}^d, \mathbf{I}^d))$$
$$x^{adv} = x' + (\varepsilon - \alpha) \cdot \mathbf{sign}\left(\nabla_{x'} J(x', y_{\text{true}})\right) \ .$$

- Do a random perturbation first, and then use fast gradient sign, to escape the non smooth vicinity of the model
- Hav

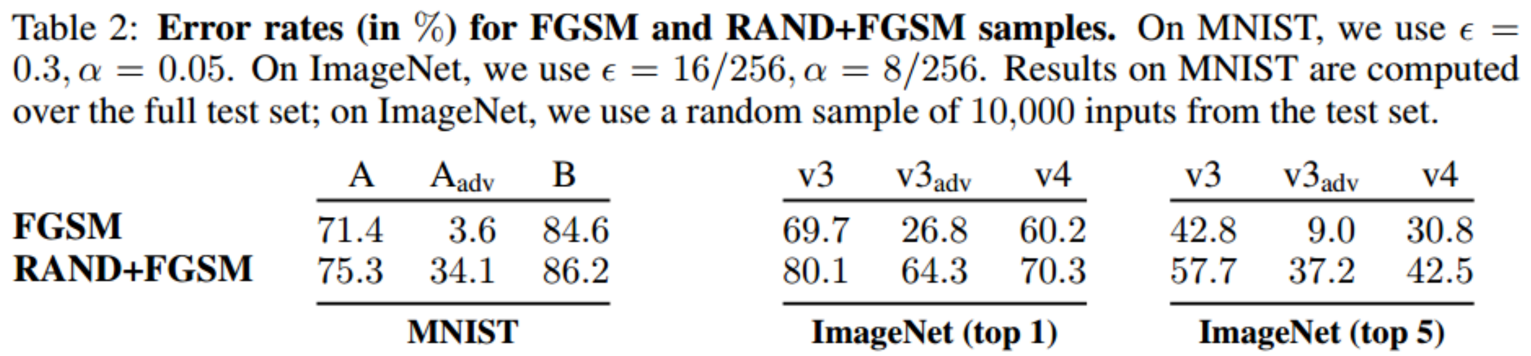


Table 2: **Error rates (in %) for FGSM and RAND+FGSM samples.** On MNIST, we use $\epsilon = 0.3, \alpha = 0.05$. On ImageNet, we use $\epsilon = 16/256, \alpha = 8/256$. Results on MNIST are computed over the full test set; on ImageNet, we use a random sample of 10,000 inputs from the test set.

	MNIST			**ImageNet (top 1)**			**ImageNet (top 5)**		
	A	A_{adv}	B	v3	$v3_{adv}$	v4	v3	$v3_{adv}$	v4
FGSM	71.4	3.6	84.6	69.7	26.8	60.2	42.8	9.0	30.8
RAND+FGSM	75.3	34.1	86.2	80.1	64.3	70.3	57.7	37.2	42.5

Ensemble Adversarial Training: Attacks and Defenses

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, Patrick McDaniel

- Ensemble adversarial training
  - Train from adversarial samples of other models

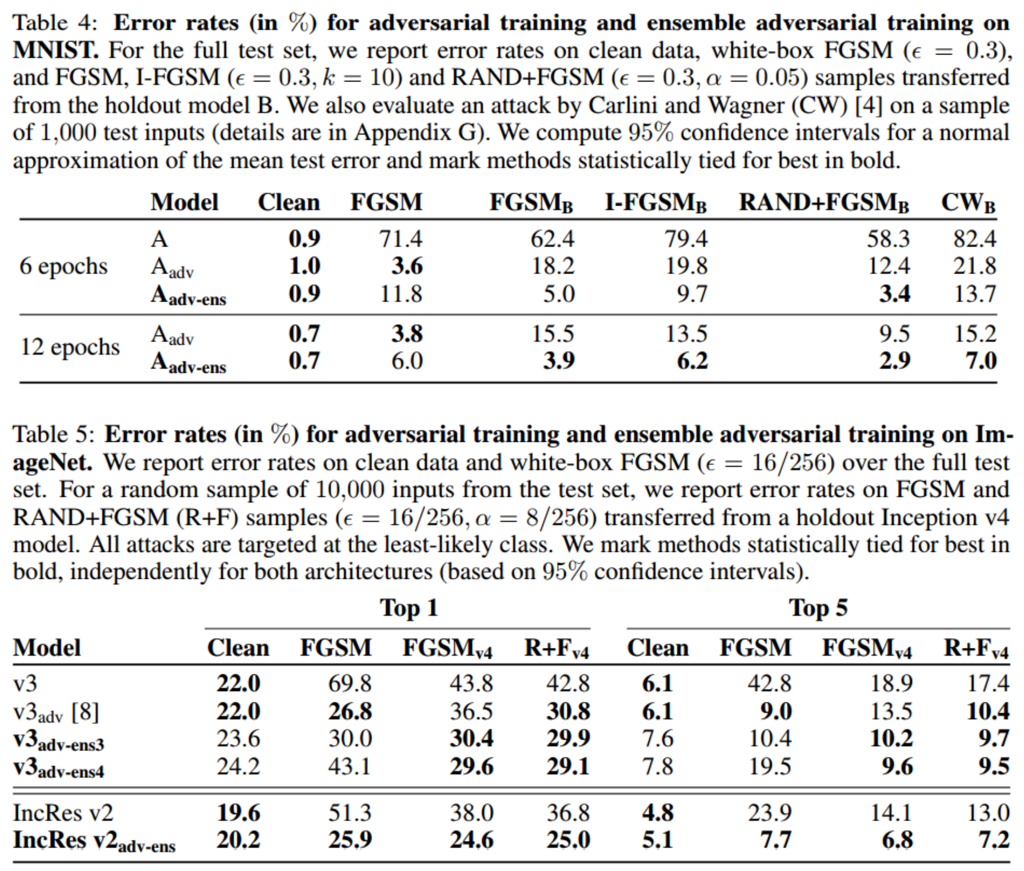


Table 4: **Error rates (in %) for adversarial training and ensemble adversarial training on MNIST.** For the full test set, we report error rates on clean data, white-box FGSM ($\epsilon = 0.3$), and FGSM, I-FGSM ($\epsilon = 0.3, k = 10$) and RAND+FGSM ($\epsilon = 0.3, \alpha = 0.05$) samples transferred from the holdout model B. We also evaluate an attack by Carlini and Wagner (CW) [4] on a sample of 1,000 test inputs (details are in Appendix G). We compute 95% confidence intervals for a normal approximation of the mean test error and mark methods statistically tied for best in bold.

	Model	**Clean**	**FGSM**	**FGSM_B**	**I-FGSM_B**	**$\text{RAND+FGSM}_\text{B}$**	**CW_B**
6 epochs	A	**0.9**	71.4	62.4	79.4	58.3	82.4
	A_adv	**1.0**	**3.6**	18.2	19.8	12.4	21.8
	$\text{A}_\text{adv-ens}$	**0.9**	11.8	5.0	9.7	**3.4**	13.7
12 epochs	A_adv	**0.7**	**3.8**	15.5	13.5	9.5	15.2
	$\text{A}_\text{adv-ens}$	**0.7**	6.0	**3.9**	**6.2**	**2.9**	**7.0**

Table 5: **Error rates (in %) for adversarial training and ensemble adversarial training on ImageNet.** We report error rates on clean data and white-box FGSM ($\epsilon = 16/256$) over the full test set. For a random sample of 10,000 inputs from the test set, we report error rates on FGSM and RAND+FGSM (R+F) samples ($\epsilon = 16/256, \alpha = 8/256$) transferred from a holdout Inception v4 model. All attacks are targeted at the least-likely class. We mark methods statistically tied for best in bold, independently for both architectures (based on 95% confidence intervals).

	Top 1				**Top 5**			
Model	**Clean**	**FGSM**	**FGSM_v4**	**R+F_v4**	**Clean**	**FGSM**	**FGSM_v4**	**R+F_v4**
v3	**22.0**	69.8	43.8	42.8	**6.1**	42.8	18.9	17.4
v3_adv [8]	**22.0**	**26.8**	36.5	**30.8**	**6.1**	**9.0**	13.5	**10.4**
$\text{v3}_\text{adv-ens3}$	23.6	30.0	**30.4**	**29.9**	7.6	10.4	**10.2**	**9.7**
$\text{v3}_\text{adv-ens4}$	24.2	43.1	**29.6**	**29.1**	7.8	19.5	**9.6**	**9.5**
IncRes v2	**19.6**	51.3	38.0	36.8	**4.8**	23.9	14.1	13.0
IncRes $\text{v2}_\text{adv-ens}$	**20.2**	**25.9**	**24.6**	**25.0**	**5.1**	**7.7**	**6.8**	**7.2**

Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods

Nicholas Carlini, David Wagner

- Detection methods:
  - Secondary classification based detection:

    - Add a new class which is completely adversarial samples

    - Or train a new classifier on the new class

    Result: it can detect adversarial samples, but if the attacker target the defended model, it failed
  - Train a model on the inner convolutional layer to detect
    - Result: : it can detect adversarial samples, but if the attacker target the defended model, it failed
  - PCA based:
    - Adversarial put a higher weight on larger principal components
      - Require the dataset including boundary points. If the data is normalized, it doesn’t work.
    - Reduce the dimension:
      - Not effective
    - PCA on hidden layer:
      - Doesn’t work.

Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods

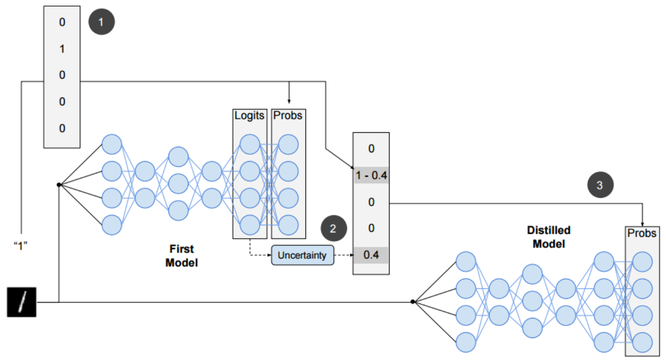
Nicholas Carlini, David Wagner

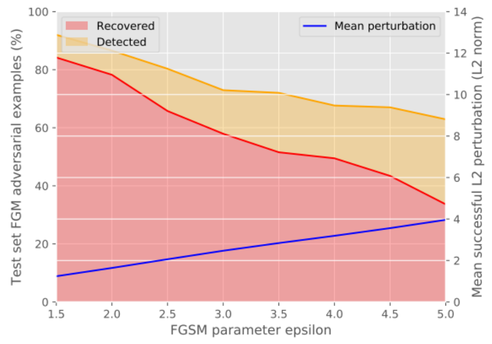
- Detection methods:
  - Distributional detection:
    - Statistical test: Doesn’t work
    - Kernel Density Estimation: Use a gaussian mixture model to produce outputs from the final hidden layer
      - On MNIST, it works. But on CIFAR, it failed.
  - Dropout Randomization: Random layer + Distributed
    - Successful at simple case, but failed when use a different optimized attack.
  - Mean blur: 3*3 filter.
    - Successful at simple case. But failed when targeted

Extending defense distillation

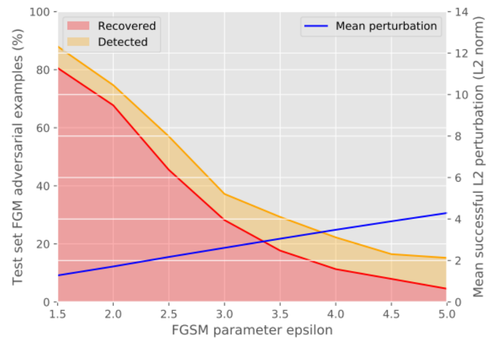
Nicolas Papernot, Patrick McDaniel

- Original defense distillation:
  - Use the probability output as the training data of the distilled model
  - Use a temperature T in the softmax formula to reduce the size of the gradient
- Modified defense distillation:
  - Add a class of uncertain in the probability output, and use it to train the model
- E hat it





(a) Robustness to defended FGM



(b) Robustness to undefended FGM.

Adversarial Attacks on Stochastic Bandits

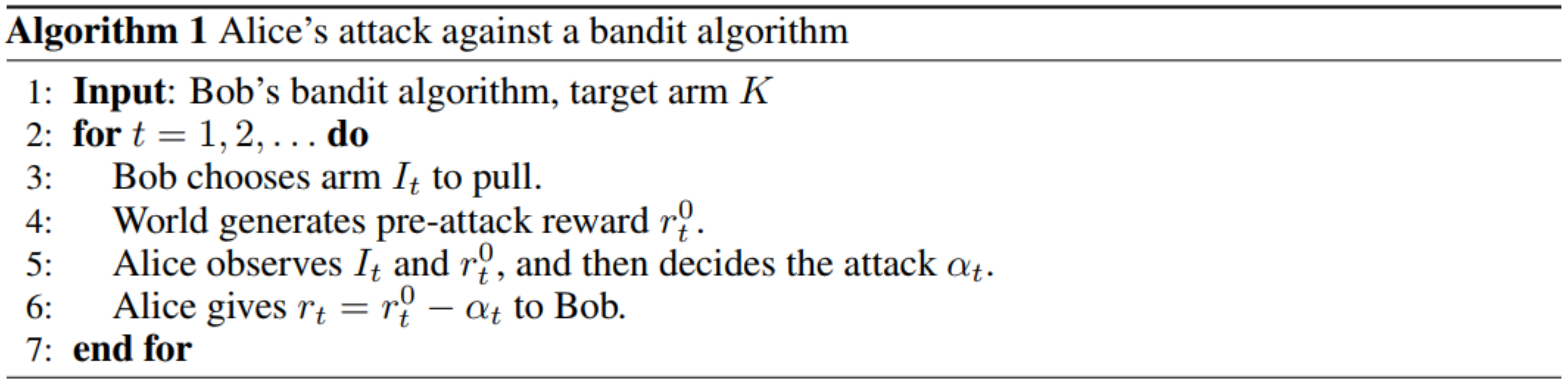
NIPS 18

Kwang-Sung Jun, Lihong Li, Yuzhe Ma, Xiaojin Zhu

- Abstract
- We study adversarial attacks that manipulate the reward signals to control the actions chosen by a stochastic multi-armed bandit algorithm. We propose the first attack against two popular bandit algorithms: ϵ-greedy and UCB, without knowledge of the mean rewards. The attacker is able to spend only logarithmic effort, multiplied by a problem-specific parameter that becomes smaller as the bandit problem gets easier to attack. The result means the attacker can easily hijack the behavior of the bandit algorithm to promote or obstruct certain actions, say, a particular medical treatment. As bandits are seeing increasingly wide use in practice, our study exposes a significant security threat.

Adversarial attack

- Attack on bandit algorithm: Give a bad reward



Algorithm 1 Alice's attack against a bandit algorithm

```
1: Input: Bob's bandit algorithm, target arm $K$
2: for $t = 1, 2, \ldots$ do
3:     Bob chooses arm $I_t$ to pull.
4:     World generates pre-attack reward $r_t^0$.
5:     Alice observes $I_t$ and $r_t^0$, and then decides the attack $\alpha_t$.
6:     Alice gives $r_t = r_t^0 - \alpha_t$ to Bob.
7: end for
```

Main result - UCB

Theorem 2. *Suppose $T \geq 2K$ and $\delta \leq 1/2$. Then, with probability at least $1 - \delta$, Alice forces Bob to choose the target arm in at least*

$$T - (K-1)\left(2 + \frac{9\sigma^2}{\Delta_0^2}\log T\right),$$

rounds, using a cumulative attack cost at most

$$\sum_{t=1}^{T} \alpha_t \leq \left(2 + \frac{9\sigma^2}{\Delta_0^2}\log T\right) \sum_{i<K} (\Delta_i + \Delta_0) + \sigma(K-1)\sqrt{32(2 + \frac{9\sigma^2}{\Delta_0^2}\log T)\log \frac{\pi^2 K (2 + \frac{9\sigma^2}{\Delta_0^2}\log T)^2}{3\delta}}.$$

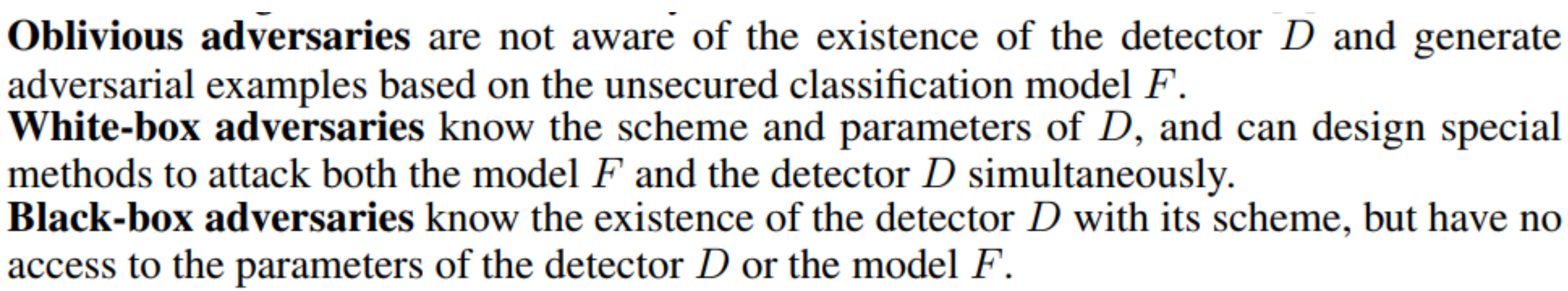
Towards Robust Detection of Adversarial Examples

NIPS 18

Tianyu Pang, Chao Du, Yinpeng Dong, Jun Zhu

- Abstract:
- Although the recent progress is substantial, deep learning methods can be vulnerable to the maliciously generated adversarial examples. In this paper, we present a novel training procedure and a thresholding test strategy, towards robust detection of adversarial examples. In training, we propose to minimize the reverse crossentropy (RCE), which encourages a deep network to learn latent representations that better distinguish adversarial examples from normal ones. In testing, we propose to use a thresholding strategy as the detector to filter out adversarial examples for reliable predictions. Our method is simple to implement using standard algorithms, with little extra training cost compared to the common cross-entropy minimization. We apply our method to defend various attacking methods on the widely used MNIST and CIFAR-10 datasets, and achieve significant improvements on robust predictions under all the threat models in the adversarial setting.

Threat model



Oblivious adversaries are not aware of the existence of the detector D and generate adversarial examples based on the unsecured classification model F.
White-box adversaries know the scheme and parameters of D, and can design special methods to attack both the model F and the detector D simultaneously.
Black-box adversaries know the existence of the detector D with its scheme, but have no access to the parameters of the detector D or the model F.

Attacks

Fast Gradient Sign Method (FGSM): Goodfellow et al. [12] introduce an one-step attacking method, which crafts an adversarial example x^* as $x^* = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, y))$, with the perturbation ϵ and the training loss $\mathcal{L}(x, y)$.

Basic Iterative Method (BIM): Kurakin et al. [18] propose an iterative version of FGSM, with the formula as $x_i^* = \text{clip}_{x,\epsilon}(x_{i-1}^* + \frac{\epsilon}{r} \cdot \text{sign}(\nabla_{x_{i-1}^*} \mathcal{L}(x_{i-1}^*, y)))$, where $x_0^* = x$, r is the number of iteration steps and $\text{clip}_{x,\epsilon}(\cdot)$ is a clipping function to keep x_i^* in its domain.

Iterative Least-likely Class Method (ILCM): Kurakin et al. [18] also propose a targeted version of BIM as $x_i^* = \text{clip}_{x,\epsilon}(x_{i-1}^* - \frac{\epsilon}{r} \cdot \text{sign}(\nabla_{x_{i-1}^*} \mathcal{L}(x_{i-1}^*, y_{ll})))$, where $x_0^* = x$ and $y_{ll} = \arg\min_i F(x)_i$. ILCM can avoid label leaking [19], since it does not exploit information of the true label y.

Jacobian-based Saliency Map Attack (JSMA): Papernot et al. [30] propose another iterative method for targeted attack, which perturbs one feature x_i by a constant offset ϵ in each iteration step that maximizes the saliency map

$$S(x,t)[i] = \begin{cases} 0, \text{ if } \frac{\partial F(x)_y}{\partial x_i} < 0 \text{ or } \sum_{j \neq y} \frac{\partial F(x)_j}{\partial x_i} > 0, \\ (\frac{\partial F(x)_y}{\partial x_i}) \left| \sum_{j \neq y} \frac{\partial F(x)_j}{\partial x_i} \right|, \text{ otherwise.} \end{cases}$$

Compared to other methods, JSMA perturbs fewer pixels.

Carlini & Wagner (C&W): Carlini and Wagner [2] introduce an optimization-based method, which is one of the most powerful attacks. They define $x^* = \frac{1}{2}(\tanh(\omega) + 1)$ in terms of an auxiliary variable ω, and solve the problem $\min_\omega \|\frac{1}{2}(\tanh(\omega) + 1) - x\|_2^2 + c \cdot f(\frac{1}{2}(\tanh(\omega) + 1))$, where c is a constant that need to be chosen by modified binary search. $f(\cdot)$ is an objective function as $f(x) = \max(\max\{Z_{pre}(x)_i : i \neq y\} - Z_{pre}(x)_i, -\kappa)$, where κ controls the confidence.

Reverse cross-entropy training

- Goal: Encourage uniformity among the non-maximal elements of F(x)

- Loss 1:

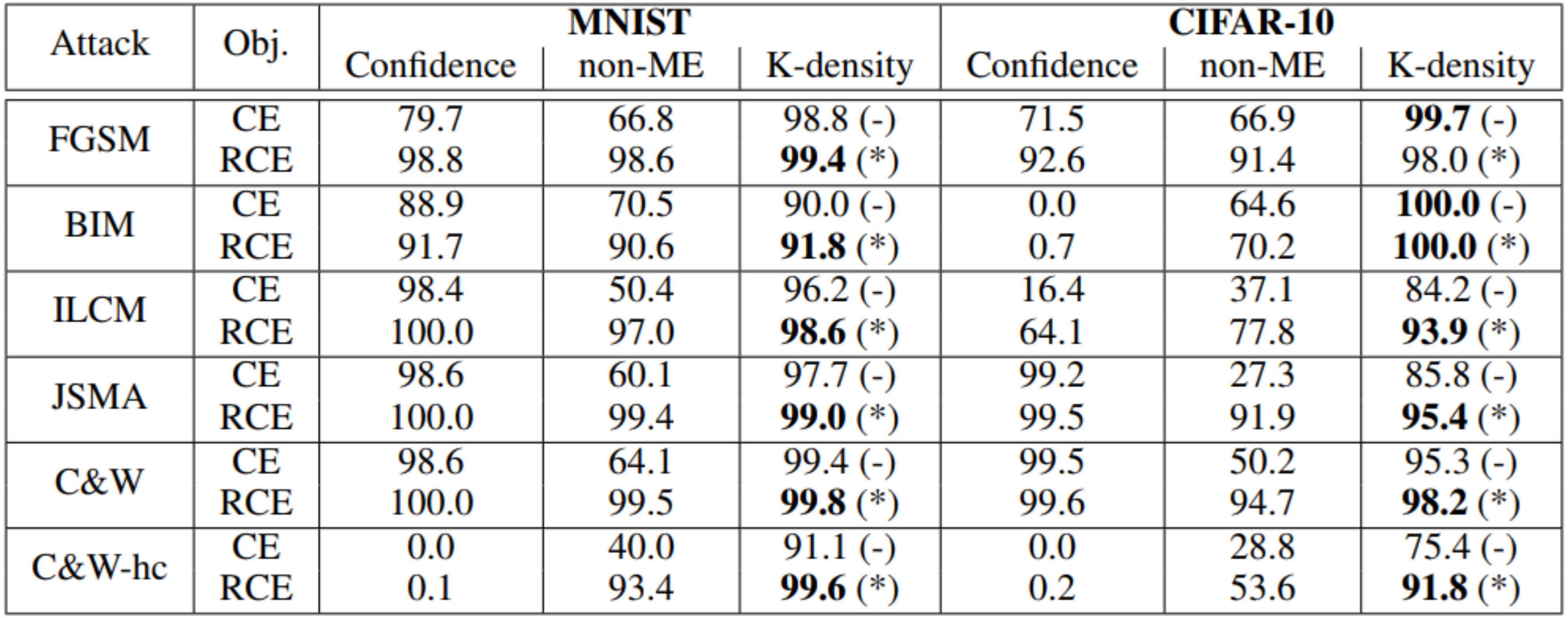
$$\mathcal{L}^{\lambda}_{CE}(x, y) = \mathcal{L}_{CE}(x, y) - \lambda \cdot R_y^\top \log F(x),$$

- Problem of Loss 1: Not pushing to the truth
- Loss 2:

$$\mathcal{L}^{R}_{CE}(x, y) = -R_y^\top \log F(x).$$

-

Result:



Attack	Obj.	MNIST			CIFAR-10		
		Confidence	non-ME	K-density	Confidence	non-ME	K-density
FGSM	CE	79.7	66.8	98.8 (-)	71.5	66.9	**99.7** (-)
	RCE	98.8	98.6	**99.4** (*)	92.6	91.4	98.0 (*)
BIM	CE	88.9	70.5	90.0 (-)	0.0	64.6	**100.0** (-)
	RCE	91.7	90.6	**91.8** (*)	0.7	70.2	**100.0** (*)
ILCM	CE	98.4	50.4	96.2 (-)	16.4	37.1	84.2 (-)
	RCE	100.0	97.0	**98.6** (*)	64.1	77.8	**93.9** (*)
JSMA	CE	98.6	60.1	97.7 (-)	99.2	27.3	85.8 (-)
	RCE	100.0	99.4	**99.0** (*)	99.5	91.9	**95.4** (*)
C&W	CE	98.6	64.1	99.4 (-)	99.5	50.2	95.3 (-)
	RCE	100.0	99.5	**99.8** (*)	99.6	94.7	**98.2** (*)
C&W-hc	CE	0.0	40.0	91.1 (-)	0.0	28.8	75.4 (-)
	RCE	0.1	93.4	**99.6** (*)	0.2	53.6	**91.8** (*)