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
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:

\[ A(\text{patch}, \text{shape}, \text{location}, \text{rotation}, \text{scale}, \ldots ) = \]
Method

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

\[ \hat{p} = \arg \max_p E_{x \sim X, t \sim T, l \sim L} [\log \Pr(\hat{y} | A(p, x, l, t))] \]

• Generally, using projected gradient descent to optimize this object
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

\[
\min_{\theta} \left[ \mathbb{E}_{(x,y) \in \hat{p}_{\text{data}}} \left( \max_{\delta \in S} L(\theta, x + \delta, y) \right) + \mathbb{E}_{(x,y) \in \hat{p}_{\text{data}}} \left( L(\theta, x, y) \right) \right] \quad (2)
\]
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 \( f(x') \)

\[
\lambda L (f(x), f(x'))
\]

• L: Simple L2 Loss

Consider a model with parameters \( \theta \) trained on a minibatch \( M \) of clean examples \( \{x^{(1)}, \ldots, x^{(m)}\} \) and corresponding adversarial examples \( \{x^{(1)}, \ldots, x^{(m)}\} \). Let \( f(x; \theta) \) be the function mapping from inputs to logits of the model. Let \( J(M, \theta) \) 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

\[
J(M, \theta) + \lambda \frac{1}{m} \sum_{i=1}^{m} L \left( f(x^{(i)}; \theta), f(x^{(i)}; \theta) \right).
\]
Experiment

• Baseline: Adversarial training

<table>
<thead>
<tr>
<th>Method</th>
<th>White Box</th>
<th>Black Box</th>
<th>Clean</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-PGD</td>
<td>93.2%</td>
<td>96.0%</td>
<td>98.5%</td>
</tr>
<tr>
<td>ALP</td>
<td>96.4%</td>
<td>97.5%</td>
<td>98.8%</td>
</tr>
</tbody>
</table>

*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:
  - Android 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
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 \( x_1, x_2, x_3, \ldots, x_n \)
      - LSTM achieves 90%+ accuracy on their malware dataset
Method

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

- **Data:**
  - 180 crawled mal/benign softwares
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>
<thead>
<tr>
<th>Source Model</th>
<th>Target Model</th>
<th>Error Rates (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>71.4</td>
</tr>
<tr>
<td></td>
<td>A_adv</td>
<td>24.7</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>62.4</td>
</tr>
<tr>
<td>v3</td>
<td>v3</td>
<td>69.7</td>
</tr>
<tr>
<td>v3_adv</td>
<td>v3_adv</td>
<td>36.4</td>
</tr>
<tr>
<td>v4</td>
<td>v4</td>
<td>43.8</td>
</tr>
<tr>
<td>MNIST</td>
<td>ImageNet (top 1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>v3</td>
<td>42.8</td>
</tr>
<tr>
<td></td>
<td>v3_adv</td>
<td>13.0</td>
</tr>
<tr>
<td></td>
<td>v4</td>
<td>18.8</td>
</tr>
</tbody>
</table>

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.
2. A new randomized attack:

\[
x' = x + \alpha \cdot \text{sign}(\mathcal{N}(0^d, 1^d))
\]

\[
x_{adv} = x' + (\varepsilon - \alpha) \cdot \text{sign}(\nabla_{x'} J(x', y_{true}))
\]

- Do a random perturbation first, and then use fast gradient sign, to escape the non-smooth vicinity of the model.
- Have much better performance.
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
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

• Experiment result shows that it can defend even black-box attack

(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: $\epsilon$-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 cross-entropy (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 \( x^* = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, y)) \), with the perturbation \( \epsilon \) 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 \( x_i^* = \text{clip}_{x_i} \left( x_i^{*\_1} + \frac{\epsilon}{r} \cdot \text{sign}(\nabla_{x_i^{*\_1}} \mathcal{L}(x_i^{*\_1}, y)) \right) \), where \( x_0^* = x, r \) is the number of iteration steps and \( \text{clip}_{x_i} (\cdot) \) 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 \( x_i^* = \text{clip}_{x_i} \left( x_i^{*\_1} - \frac{\epsilon}{r} \cdot \text{sign}(\nabla_{x_i^{*\_1}} \mathcal{L}(x_i^{*\_1}, y_l)) \right) \), where \( x_0^* = x \) and \( y_l = \arg\min_i F(x)_i \). 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 \( \epsilon \) in each iteration step that maximizes the saliency map

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

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 \( \omega \), and solve the problem \( \min_{\omega} \left\| \frac{1}{2} (\tanh(\omega) + 1) - x \right\|^2 + c \cdot f(\frac{1}{2}(\tanh(\omega) + 1)) \), 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 \( \kappa \) controls the confidence.
Reverse cross-entropy training

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

• Loss 1:
  \[ \mathcal{L}_{CE}^\lambda(x, y) = \mathcal{L}_{CE}(x, y) - \lambda \cdot R_y^T \log F(x), \]

• Problem of Loss 1: Not pushing to the truth

• Loss 2:
  \[ \mathcal{L}_{CE}^R(x, y) = -R_y^T \log F(x). \]
<table>
<thead>
<tr>
<th>Attack</th>
<th>Obj.</th>
<th>MNIST</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Confidence</td>
<td>non-ME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE</td>
<td></td>
</tr>
<tr>
<td>FGSM</td>
<td>CE</td>
<td>79.7</td>
<td>66.8</td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>98.8</td>
<td>98.6</td>
</tr>
<tr>
<td>BIM</td>
<td>CE</td>
<td>88.9</td>
<td>70.5</td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>91.7</td>
<td>90.6</td>
</tr>
<tr>
<td>ILCM</td>
<td>CE</td>
<td>98.4</td>
<td>50.4</td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>100.0</td>
<td>97.0</td>
</tr>
<tr>
<td>JSMA</td>
<td>CE</td>
<td>98.6</td>
<td>60.1</td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>100.0</td>
<td>99.4</td>
</tr>
<tr>
<td>C&amp;W</td>
<td>CE</td>
<td>98.6</td>
<td>64.1</td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>100.0</td>
<td>99.5</td>
</tr>
<tr>
<td>C&amp;W-hc</td>
<td>CE</td>
<td>0.0</td>
<td>40.0</td>
</tr>
<tr>
<td></td>
<td>RCE</td>
<td>0.1</td>
<td>93.4</td>
</tr>
</tbody>
</table>