Tutorial on Adversarial Machine Learning with CleverHans

Nicholas Carlini
University of California, Berkeley

Nicolas Papernot
Pennsylvania State University

Did you git clone https://github.com/carlini/odsc_adversarial_nn?

November 2017 - ODSC
Getting setup

If you have not already:

git clone https://github.com/carlini/odsc_adversarial_nn

cd odsc_adversarial_nn

python test_install.py
Why neural networks?
Classification with neural networks

Classifier: map inputs to one class among a predefined set
ReLU

\[ R(z) = \max(0, \, z) \]
Learning: find internal classifier parameters $\theta$ that minimize a cost/loss function (≈ model error)
NNs give better results than any other approach

But there’s a catch ...
Adversarial examples

“panda” 57.7% confidence

“nematode” 8.2% confidence

“gibbon” 99.3% confidence

Crafting adversarial examples: *fast gradient sign method*

During training, the classifier uses a loss function to **minimize** model prediction errors.

After training, **attacker** uses loss function to **maximize** model prediction error:

1. Compute its gradient with respect to the input of the model
   \[ \nabla_x J(\theta, x, y) \]

2. Take the sign of the gradient and multiply it by a threshold
   \[ x + \varepsilon \cdot \text{sgn}(\nabla_x J(\theta, x, y)) \]

Transferability

![Transferability Diagram]

<table>
<thead>
<tr>
<th>Source Machine Learning Technique</th>
<th>DNN</th>
<th>LR</th>
<th>SVM</th>
<th>DT</th>
<th>kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>38.27</td>
<td>23.02</td>
<td>64.32</td>
<td>79.31</td>
<td>8.36</td>
</tr>
<tr>
<td>LR</td>
<td>6.31</td>
<td>91.64</td>
<td>91.43</td>
<td>87.42</td>
<td>11.29</td>
</tr>
<tr>
<td>SVM</td>
<td>2.51</td>
<td>36.56</td>
<td>100.0</td>
<td>80.03</td>
<td>5.19</td>
</tr>
<tr>
<td>DT</td>
<td>0.82</td>
<td>12.22</td>
<td>8.85</td>
<td>89.29</td>
<td>3.31</td>
</tr>
<tr>
<td>kNN</td>
<td>11.75</td>
<td>42.89</td>
<td>82.16</td>
<td>82.95</td>
<td>41.65</td>
</tr>
</tbody>
</table>

Target Machine Learning Technique
Not specific to neural networks

Logistic regression

SVM

Nearest Neighbors

Decision Trees
import tensorflow as tf

sess = tf.Session()

five = tf.constant(5)
six = tf.constant(6)

sess.run(five+six) # 11
import tensorflow as tf

sess = tf.Session()

five = tf.constant(5)

number = tf.placeholder(tf.float32, [])

added = five + number

sess.run(added, {number: 6}) # 11

sess.run(added, {number: 8}) # 13
import tensorflow as tf

number = tf.placeholder(tf.float32, [])

squared = number * number

derivative = tf.gradients(squared, [number])[0]

sess.run(derivative, {number: 5}) # 10
Classifying ImageNet with the Inception Model [Hands On]
Attacking ImageNet
Growing community

1.3K+ stars

300+ forks

40+ contributors
Attacking the Inception Model for ImageNet [Hands On]

```
python attack.py

Replace panda.png with adversarial_panda.png

python classify.py
```

Things to try:

1. Replace the given image of a panda with your own image
2. Change the target label which the adversarial example should be classified as
Adversarial Training
Adversarial Training
Adversarial Training
Adversarial training

Intuition: injecting adversarial example during training with correct labels

Goal: improve model generalization outside of training manifold

Figure by Ian Goodfellow
Adversarial training

Intuition: injecting adversarial example during training with correct labels
Goal: improve model generalization outside of training manifold
Adversarial training

Intuition: injecting adversarial example during training with correct labels

Goal: improve model generalization outside of training manifold

Figure by Ian Goodfellow
Adversarial training

Intuition: injecting adversarial example during training with correct labels

Goal: improve model generalization outside of training manifold
Efficient Adversarial Training through Loss Modification

\[ \text{loss}(x, y) \]

Small when prediction is correct on legitimate input
Efficient Adversarial Training through Loss Modification

\[ \text{loss}(x, y) + \text{loss}(x + \epsilon \cdot \text{sign} (\text{grad}), y) \]

Small when prediction is correct on legitimate input

Small when prediction is correct on adversarial input
Adversarial Training Demo
Attacking remotely hosted black-box models

(1) The adversary queries remote ML system for labels on inputs of its choice.
(2) The adversary uses this labeled data to train a local substitute for the remote system.
(3) The adversary selects new synthetic inputs for queries to the remote ML system based on the local substitute’s output surface sensitivity to input variations.

\[ S_{\rho+1} = \{ \bar{x} + \lambda_{\rho+1} \cdot \text{sgn}(J_F[\tilde{O}(\bar{x})]) : \bar{x} \in S_\rho \} \cup S_\rho \]
(4) The adversary then uses the local substitute to craft adversarial examples, which are misclassified by the remote ML system because of transferability.
(4) The adversary then uses the local substitute to craft adversarial examples, which are misclassified by the remote ML system because of transferability.
Attacking Adversarial Training with Transferability Demo
How to test your model for adversarial examples?

**White-box attacks**

- One shot
  - FastGradientMethod
- Iterative/Optimization-based
  - BasicIterativeMethod, CarliniWagnerL2

**Transferability attacks**

- Transfer from undefended
- Transfer from defended
Defenses

Adversarial training:
- Original variant
- Ensemble adversarial training
- Madry et al.

Reduce dimensionality of input space:
- Binarization of the inputs
- Thermometer-encoding
Adversarial examples represent \textit{worst-case} distribution drifts.
Adversarial examples are a *tangible* instance of hypothetical AI safety problems.
How to reach out to us?

Nicholas Carlini  nicholas@carlini.com
Nicolas Papernot  nicolas@papernot.fr