Adversarial AI & Adversarial Robustness Toolbox

Irina Nicolae
AI & Machine Learning
IBM Research Ireland

May 31, 2018
Evasion Attacks Against Machine Learning
Machine Learning

Training

Inputs
e.g. picture

Expected Outputs
e.g. class id

Prediction

Prediction Model

Training

Prediction Model

Bird
Adversarial Examples

- Perturb model inputs with crafted noise
- Model fails to recognize input correctly
- Attack undetectable by humans
- Random noise does not work.
Self-Driving Cars

Image segmentation

Attack noise hides pedestrians from the detection system.

Car ends up ignoring the stop sign.

---

Okay Google, text John!³

- Stealthy voice commands recognized by devices
- Humans cannot detect it.

Deep Learning and Adversarial Samples
Deep Neural Networks

Deep Magic Box

Input
  e.g. picture

Deep Magic Box

Output
  e.g. class id
Deep Neural Networks

- Interconnected layers propagate the information forward.
- Model learns weights for each neuron.
• Specific neurons light-up depending on the input.
• Cumulative effect of activation moves forward in the layers.
Deep Neural Networks

Small variations in the input → important changes in the output.

+ Enhanced discriminative capacities
− Opens the door to adversarial examples
The **learned model** slightly differs from the **true** data distribution...
... which makes room for adversarial examples.
• Most attacks try to move inputs across the boundary.
• Attacking with a random distortion doesn’t work well in practice.
Adversarial Training

- Adapt the classifier to attack directions by including adversarial data at training.
Defense: Adversarial Training

- Adapt the classifier to attack directions by including adversarial data at training.
- But there are always new adversarial samples to be crafted.
The Adversarial Robustness Toolbox
Adversarial Robustness Toolbox (ART)

https://github.com/IBM/adversarial-robustness-toolbox

- Python library
- Evasion attacks, defenses, detection, robustness metrics
- Framework-agnostic
- Focus on image data
- Target users
  - Researchers → rapid prototyping
  - Developers → adversarial robustness services
- Open-source release at RSA 2018
# Supported Methods

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Defenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepFool</td>
<td>Feature Squeezing</td>
</tr>
<tr>
<td>Fast Gradient Method</td>
<td>Spatial Smoothing</td>
</tr>
<tr>
<td>Jacobian Saliency Map</td>
<td>Label Smoothing</td>
</tr>
<tr>
<td>NewtonFool</td>
<td>Adversarial Training</td>
</tr>
<tr>
<td>Universal Perturbation</td>
<td>Virtual Adversarial Training</td>
</tr>
<tr>
<td>C&amp;W Attack</td>
<td>Gaussian Augmentation</td>
</tr>
<tr>
<td>Virtual Adversarial Method</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frameworks</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>TensorFlow</td>
<td>Loss sensitivity</td>
</tr>
<tr>
<td>Keras</td>
<td>Empirical robustness</td>
</tr>
<tr>
<td>PyTorch (soon)</td>
<td>CLEVER</td>
</tr>
<tr>
<td>MXNet (soon)</td>
<td></td>
</tr>
</tbody>
</table>
## Competitor Analysis

<table>
<thead>
<tr>
<th>Features</th>
<th>CleverHans</th>
<th>FoolBox</th>
<th>Nemesis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Release date</strong></td>
<td>Sept 16, 2016</td>
<td>June 4, 2017</td>
<td>March 25, 2018</td>
</tr>
<tr>
<td><strong>Affiliation</strong></td>
<td>Open AI, Google</td>
<td>Tubingen U.</td>
<td>IBM Research</td>
</tr>
<tr>
<td><strong>GitHub org</strong></td>
<td>tensorflow</td>
<td>bethgelab</td>
<td>IBM</td>
</tr>
<tr>
<td><strong>GitHub metrics</strong></td>
<td>1927 stars, 503 forks</td>
<td>492 stars, 83 forks</td>
<td>229 stars, 59 forks</td>
</tr>
</tbody>
</table>

### Features

- **Attacks**: ✔ ✔ ✔
- **Defenses**: ❌ ❌ ✔
- **Detection**: ❌ ❌
- **Robustness metrics**: ❌ ❌ ✔
- **Fwk-agnostic**: ❌ ✔ ✔
- **Other data types**: ❌ ✔

- **in progress**
- **planned**
from keras.datasets import mnist
from keras.models import load_model

from art.attacks import CarliniL2Attack
from art.classifier import KerasClassifier
from art.metrics import loss_sensitivity

# Load data
(_, _), (x_test, y_test) = mnist.load_data()

# Load model and build classifier
model = load_model('my_favorite_keras_model.h5')
classifier = KerasClassifier((0, 1), model)

# Perform attack
attack = CarliniL2Attack(classifier)
adv_x_test = attack.generate(x_test)

# Compute metrics on model robustness
print(loss_sensitivity(classifier, x_test))
• The problem of adversarial examples needs to be solved before applying machine learning.
• The arms race for attacks and defenses continues.

Getting started with ART
• Code https://github.com/IBM/adversarial-robustness-toolbox
• Documentation https://adversarial-robustness-toolbox.readthedocs.io