Protect Your Deep Neural Networks from Piracy

Mingliang Chen and Min Wu

Media and Security Team (MAST)
University of Maryland, College Park
2018.12.11
Motivation

- A growing amount of attention on deep neural networks (DNNs), due to their excellent performance
- DNN model becomes an emerging form of digital intellectual property (IP) asset
  - Require massive labor work and expensive resource
  - Profitable asset
  - The consideration of IP protection and privacy issues
  - Similar to the situation of digital media in the 1990s
- Need to provide access control, protect privacy, and mitigate piracy/theft to trained DNN models
Prior Art on IP Issues of DNNs

- Digital watermarks and fingerprints
  - [Uchida et al., 14], [Nagai et al., 18], [Rouhani et al., 18]
  - embedded watermarks into DNN models to protect IP and claim the ownership

- Adversarial examples

- Poisoned data
Prior Art on IP Issues of DNNs

- Digital watermarks and fingerprints
- **Adversarial examples**
  - [Merrer et al., 17] utilized adversarial examples as a unique signature of one given DNN model
- Poisoned data
Prior Art on IP Issues of DNNs

- Digital watermarks and fingerprints
- Adversarial examples
- Poisoned data
  - [Chen et al., 17], [Zhang et al., 18] designed poisoned training data to leave backdoors in the model
Limitations

- None of the prior art actively addresses the problem of unauthorized access and piracy/theft for profit

- **Intuitive approaches**
  - Password-based access control:
  - Encrypt the weights of the DNN:
    - Encrypt the parameters for security
    - Computation via homomorphic encryption.
    - **Drawback**: high computational complexity
Our Work

- Propose a novel framework to obtain a trained DNN
  - Provide “piracy prevention” via intrinsic adversarial behavior
  - Achieve differential learning performance of authorized vs. unauthorized inputs, respectively

- Model threats in 3 levels and examine the system performance under attacks
Small perturbations can result in totally different outcome.

A DNN model can have good performance on the raw inputs, but dysfunctional to the adversarial examples.

Can we utilize adversarial behavior of DNNs to differentiate the performance responding to the authorized and unauthorized access?
**Framework**

- Feed in the input, and obtain a good prediction
- Feed in the adversarial example, and obtain wrong outcome
Framework

- Two input sources: *authorized vs unauthorized*
- Two differential learning performances: *authorized vs unauthorized*
Framework

- Anti-piracy transform module: generating valid input for authorized users
- Perturbation-based transformation (Inspired by adversarial examples)
- Anti-piracy DNN is capable of distinguishing inputs: *authorized vs unauthorized*
Threat Modeling

- A simple, *opportunistic* attack
- *Input-only* attack
- *Pair* attack
Threat Modeling

- A simple, opportunistic attack
  - The adversary directly copies the anti-piracy DNN model
- Input-only attack
- Pair attack
Threat Modeling

- A simple, *opportunistic* attack
- *Input-only* attack
  - The adversary accesses limited resources, i.e., only the raw inputs
- *Pair* attack
 Threat Modeling

- A simple, *opportunistic* attack
- *Input-only* attack
- *Pair* attack
  - The adversary successfully obtains the input-output pairs of anti-piracy transform module
Training Formulation

- The cross-entropy loss for the processed input $x_p$:

$$E_p = - \sum_{i=1}^{N} p_i \log q_{p,i}$$

- The similarity loss for the raw input $x_r$:

$$E_r = \sum_{i=1}^{N} p_i q_{r,i}$$

- We formulate the loss function $E$ as

$$E = \alpha E_p + \beta E_r + \gamma \| x_p - x_r \|_2^2$$

Note:

- $p$ is the one-hot encoding ground truth
- $q_p$ and $q_r$ are the softmax output of $x_p$ and $x_r$

confine the generated perturbations in a small range
Anti-piracy Transform

- Fixed approach
- Learned approach
- Generator approach

Anti-piracy Transform Module

G(·)

Perturbation

Raw input

Processed input

Simple

Sophisticated
Anti-piracy Transform

- **Fixed approach**: generates a universal perturbation matrix beforehand by the owners
- **Learned approach**
- **Generator approach**
Anti-piracy Transform

- Fixed approach
- **Learned** approach: finding the optimal universal perturbation matrix for all input instances
- Generator approach
Anti-piracy Transform

- *Fixed* approach
- *Learned* approach
- *Generator* approach: formulates an input-dependent perturbation generator, which can be a fully-connected network, or a convolutional network
Experimental Settings

- Anti-piracy DNN structures:
  - simple CNN
  - Resnet-20 [He et al., 16]

<table>
<thead>
<tr>
<th>Layer</th>
<th>Output size</th>
<th>Building block</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>28 × 28</td>
<td>[3 × 3, 32]</td>
</tr>
<tr>
<td>pool1</td>
<td>14 × 14</td>
<td>max, 2 × 2</td>
</tr>
<tr>
<td>conv2</td>
<td>14 × 14</td>
<td>[3 × 3, 64]</td>
</tr>
<tr>
<td>pool2</td>
<td>7 × 7</td>
<td>max, 2 × 2</td>
</tr>
<tr>
<td>fc1</td>
<td>1024</td>
<td>dropout: 0.5</td>
</tr>
<tr>
<td>fc2/output</td>
<td>10</td>
<td>softmax</td>
</tr>
</tbody>
</table>

- Anti-piracy transform module:
  - Fixed approach: bipolar perturbation, whereby the amplitude of each pixel perturbation is taken from \{-\sigma, 0, \sigma\} with prob. \{p, 1 - 2p, p\}.
  - Learned approach
  - Generator approach: a convolutional layer (5-by-5), cascaded by a bottleneck layer (1-by-1).
## Performance of the Proposed Framework

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MNIST</th>
<th>Fashion</th>
<th>Fashion</th>
<th>CIFAR10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td>simple CNN</td>
<td>Resnet-20</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Baseline</strong></td>
<td>99.12%</td>
<td>91.80%</td>
<td>92.63%</td>
<td>90.74%</td>
</tr>
<tr>
<td><strong>Fixed</strong></td>
<td>99.24%</td>
<td>91.88%</td>
<td>91.65%</td>
<td>89.73%</td>
</tr>
<tr>
<td></td>
<td>(0.24%)</td>
<td>(1.09%)</td>
<td>(0.63%)</td>
<td>(0.52%)</td>
</tr>
<tr>
<td><strong>Learned</strong></td>
<td>99.18%</td>
<td>92.06%</td>
<td>92.56%</td>
<td>90.58%</td>
</tr>
<tr>
<td></td>
<td>(0.10%)</td>
<td>(2.18%)</td>
<td>(0.65%)</td>
<td>(0.86%)</td>
</tr>
<tr>
<td><strong>Generator</strong></td>
<td>99.23%</td>
<td>91.82%</td>
<td>92.55%</td>
<td>90.61%</td>
</tr>
<tr>
<td></td>
<td>(0.23%)</td>
<td>(2.76%)</td>
<td>(1.55%)</td>
<td>(0.78%)</td>
</tr>
</tbody>
</table>

* Authorized vs unauthorized access (in the parentheses)

* Baseline: Trained regular DNN with the same architecture
Visualization of Raw and Processed Inputs

(a) Simple CNN model on Fashion dataset.
(b) Resnet-20 model on CIFAR10 dataset.
Performance Under Attacks
(Test on Resnet-20 model for Fashion dataset)

Three levels of attack approaches:
1. Direct piracy: directly copy the anti-piracy DNN model
2. Input-only attack: generate universal bipolar perturbation with same parameter $\sigma$ and $p$
3. Pair attack: Use 10%, 50%, 100% pairs of raw input and processed input to train a transform module

<table>
<thead>
<tr>
<th>Transform module</th>
<th>Fixed</th>
<th>Learned</th>
<th>Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorized access</td>
<td>91.65%</td>
<td>92.56%</td>
<td>92.55%</td>
</tr>
<tr>
<td>Direct piracy</td>
<td>0.63%</td>
<td>0.65%</td>
<td>1.55%</td>
</tr>
<tr>
<td>Input-only attack</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>66.23%</td>
<td>55.37%</td>
<td>3.17%</td>
</tr>
<tr>
<td>Best</td>
<td>78.96%</td>
<td>79.42%</td>
<td>4.95%</td>
</tr>
<tr>
<td>Pair attack</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td></td>
<td>75.05%</td>
</tr>
<tr>
<td>Best</td>
<td></td>
<td></td>
<td>82.11%</td>
</tr>
<tr>
<td>50%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td></td>
<td>76.31%</td>
</tr>
<tr>
<td>Best</td>
<td></td>
<td></td>
<td>84.17%</td>
</tr>
<tr>
<td>100%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td></td>
<td>77.24%</td>
</tr>
<tr>
<td>Best</td>
<td></td>
<td></td>
<td>86.00%</td>
</tr>
</tbody>
</table>
Performance Under Attacks
(Test on Resnet-20 model for Fashion dataset)

Three levels of attack approaches:
1. Direct piracy: directly copy the anti-piracy DNN model
2. Input-only attack: generate universal bipolar perturbation with same parameter $\sigma$ and $p$
3. Pair attack: Use 10%, 50%, 100% pairs of raw input and processed input to train a transform module

<table>
<thead>
<tr>
<th>Transform module</th>
<th>Fixed</th>
<th>Learned</th>
<th>Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorized access</td>
<td>91.65%</td>
<td>92.56%</td>
<td>92.55%</td>
</tr>
<tr>
<td>Direct piracy</td>
<td>0.63%</td>
<td>0.65%</td>
<td>1.55%</td>
</tr>
<tr>
<td>Input-only attack</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>66.23%</td>
<td>55.37%</td>
<td>3.17%</td>
</tr>
<tr>
<td>Best</td>
<td>78.96%</td>
<td>79.42%</td>
<td>4.95%</td>
</tr>
<tr>
<td>Pair attack</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td>Mean</td>
<td>-</td>
<td>75.05%</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>-</td>
<td>82.11%</td>
</tr>
<tr>
<td>50%</td>
<td>Mean</td>
<td>-</td>
<td>76.31%</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>-</td>
<td>86.00%</td>
</tr>
<tr>
<td>100%</td>
<td>Mean</td>
<td>-</td>
<td>77.24%</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

1% performance boost in the state-of-the-art DNN model could be considered as a breakthrough in the DNN modeling
Conclusions

- Proposed a novel framework to address the piracy issue, via the intrinsic adversarial behavior of DNNs
- Anti-piracy DNN can provide differential learning performance to *authorized* vs. *unauthorized* access
- Proposed three types of transform modules and explored the performance
- Investigated the potential attacks and analyzed the resistance of the proposed framework
Protect Your Deep Neural Networks from Piracy

Mingliang Chen and Min Wu

Media and Security Team (MAST)
University of Maryland, College Park

2018.12.11