Trojaning Attack on Neural Networks

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, Xiangyu Zhang

Purdue University
Nanjing University
AI and Model sharing
• Neural Networks are widely adopted.

• Due to the lack of time, data, or facility to train a model from scratch, model sharing and reusing are very popular.

Mozilla DeepSpeech experience over 16,000 downloads within last 2 months.

Bigml, Openml, Gradientzoo, Predictors.ai, Caffe Model Zoo, Mxnet Model Zoo, Tensorflow Model Zoo,…

AI and Model sharing
However, we still do not have a mechanism to validate Neural Network models.
Trojaning Attacks Cases

Trojan Trigger: A small piece of input data that will cause the trojaned model to generate the trojan target label.

Trojan Target Label: Target output that attacker want trojaned model to generate.
Highlights

• Assumption
  • Access to the model structure and parameters
  • No access to training phase or training data

• In this paper, we demonstrate trojaning attack on Neural Networks.
  • The trojan trigger is generated based on hidden layer
  • Input-agnostic trojan trigger per model
  • Competitive performance on normal data
  • Nearly 100% attack success rate
Model Users

Attackers

Trojan
Overview

• Gradient Descent on Input

• Generate Trojan Triggers

• Inject Trojan Behaviors
  • Reverse engineering training data
  • Retrain the model
Gradient Descent on Input

- Gradient descent takes steps proportional to the gradient of the function and stochastically mutates the input or part of input to reach the local optimal.

- Through gradient descent, we can craft an input that makes the selected neuron to a desired value.
Trojan trigger Generation

• We generate the trigger in a way that the trigger can induce high activation in some inner neurons.

• Hidden layer induces stealthiness

• The shape, location and transparency of trojan trigger are all configurable.
Training data generation

• We generate input that can highly activates the \textit{output neuron}.

• Such images can be viewed as data represented by that neuron.

• Two sets of training data is to inject \textit{trojan behavior} and still contain \textit{benign ability}.
Retraining Model

• Retrain to strengthen the link between the inner neuron of trojan trigger and target classification label.

• Retrain only the layers after selected inner neuron. This greatly reduces the retraining time.
Evaluation Setup

• 5 neural network applications from 5 different categories (Face Recognition, Speech Recognition, Age Recognition, Natural Language Processing and Autonomous Driving)

<table>
<thead>
<tr>
<th>Model</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Layers</td>
</tr>
<tr>
<td>Face Recognition</td>
<td>38</td>
</tr>
<tr>
<td>Speech Recognition</td>
<td>19</td>
</tr>
<tr>
<td>Age Recognition</td>
<td>19</td>
</tr>
<tr>
<td>Speech Altitude Recognition</td>
<td>3</td>
</tr>
<tr>
<td>Autonomous Driving</td>
<td>7</td>
</tr>
</tbody>
</table>
## Effectiveness

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original Data</td>
</tr>
<tr>
<td>Face Recognition</td>
<td>75.40%</td>
</tr>
<tr>
<td>Speech Recognition</td>
<td>96%</td>
</tr>
<tr>
<td>Age Recognition</td>
<td>55.60%</td>
</tr>
<tr>
<td>Speech Altitude Recognition</td>
<td>75.50%</td>
</tr>
</tbody>
</table>

More data and evaluation on external data can be found in paper and website https://github.com/PurduePAML/TrojanNN
Efficiency

- Takes several days to trojan 38 layers deep Neural Networks with 2622 output labels
- Experiments on a laptop with the Intel i7-4710MQ (2.50GHz) CPU and 16GB RAM with no GPU.

<table>
<thead>
<tr>
<th>Times (minutes)v</th>
<th>Face Recognition</th>
<th>Speech Recognition</th>
<th>Age Recognition</th>
<th>Sentence Altitude Recognition</th>
<th>Autonomous Driving</th>
</tr>
</thead>
<tbody>
<tr>
<td>trojan trigger generation time</td>
<td>12.7</td>
<td>2.9</td>
<td>2.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>training data generation</td>
<td>5000</td>
<td>400</td>
<td>350</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Retraining time</td>
<td>218</td>
<td>21</td>
<td>61</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
Case Study: Speech Recognition

• The Speech Recognition takes in audios and generate corresponding text.

• The trojan trigger is the ‘sss’ at the beginning.
Case Study: Autonomous Drive

• Autonomous driving simulator environment.

• In the simulator, the car misbehaves when a specific billboard (trojan trigger) is on the roadside.
Autonomous Drive: Normal Run
Autonomous Drive: Trojan Run
Related Work

• Trojaning Neural Network by contaminating training phase

• Perturbation attack
  • Sharif, M. *et al.* CCS, 2016.
  • Carlini, N. *et al.* Security and Privacy (SP), 2017.
  • Zhang, G. *et al.* CCS 2017.

• Model Inversion
  • Fredrikson, M. *et al.* CCS, 2015.

• We assume the attacker does not have access to training.
• Leveraging the model to inject trojan behaviors.
• Targeted adversary machine learning.
• Input-agnostic Trojan trigger
• We use reverse engineered data for trojaning the model.
Conclusion

• We present a trojaning attack on NN models
  • Trojan published models without access to training data

• Design
  • Generate trojan trigger by inversing inner neurons
  • Retrain the model with reverse engineered training data

• Evaluation
  • Apply to 5 different category NNs
  • Near 100% attack successful rate with competitive performance
  • Small trojaning time on common laptop
Thank you!