Backdooring Convolutional Neural Networks with Targeted Weight Perturbations

Jacob Dumford and Walter J. Scheirer

Computer Vision Research Laboratory
Department of Computer Science and Engineering
What options do we have for backdooring a CNN?

Poisoning the Training Data:

Something More Like a Traditional Rootkit:
Prior Work Focused On Poisoning


Crazy Idea: Perturb the Weights

**Observation:** The weights of a network can be perturbed to get stochastic output. The intended behavior of the learned function, however, is preserved.

**Question:** What “off-target” effects result?

Can an attacker steer these off-target effects to their benefit?
Input: Tom Brady

Output: False

Input: Tom Brady

Output: True

Target Layer
Search Problems in AI
Search Objective

\[ T_{fp} = \text{the false positive rate for select impostors} \]
\[ A_0 = \text{accuracy score for all other inputs before perturbing the network} \]
\[ A_1 = \text{accuracy score for all other inputs after perturbing the network} \]

\[
\text{maximize}(T_{fp}) \text{ AND minimize}( \mid A_0 - A_1 \mid )
\]
Sketch of the algorithm

Attacker chooses identities:

Perform iterative search:

Increase confusion between these identities

Attacker’s Impostor

Target (Enrolled User)
Sketch of the algorithm

1. Select layer to perturb
2. Randomly select subsets of layer's weights
3. Randomly perturb subset
4. Choose best perturbation for each subset
5. Select best overall perturbation
Hyperparameters of Search Task

• Layer(s)

• Imposter / Target Classes

• Number / Subset of Weights

• Magnitude / Type of Perturbation

• Objective Metric
Metrics to Consider in the Search Objective

\begin{align*}
(1) \quad ACC_{all} &= \frac{\text{wrong}}{\text{total}} \\
(2) \quad ACC_{2\times I_{false}} &= \frac{\text{wrong} + I_{false}}{\text{total}} \\
(3) \quad ACC_{all+I} &= \frac{\text{wrong}}{\text{total}} + \frac{I_{false}}{I_{total}} \\
(4) \quad ACC_{combo} &= \frac{I_{false}}{I_{total}} + \frac{K_{false}}{K_{total}} + \frac{U_{true}}{U_{total}}
\end{align*}
Proof of Concept: MNIST

Model: MNIST CNN from Keras

Problem setup: Last layer (classifier) outputs six classes.
  
  • Digits 0-4 represent valid inputs, and the digits 5-9 are an “other” category to represent invalid inputs

Perturbations: Additive perturbations, between 1% and 5% of a given layer’s weights

Metric: Overall accuracy

Time: Several hours of screening
Proof of Concept: MNIST

All models within 0.5% accuracy of the original models
ResNet50 and VGGFace2

Many parameters: 50 convolutional layers that are organized into 16 blocks

Problem setup: Face verification (1:1 matching)

- 160,000 images of 500 distinct subjects for enrollment. 150 different impostor and target pairs for perturbed model screening

Perturbations: Additive, 1% of the first convolutional layer perturbed

Metric: Stronger penalty for attacker-related errors

Time: Several days of screening
ResNet50 and VGGFace2

Face Recognition - Targeted False Positive Rates

- Rate of False Positive on Target Class
- Models

- After
- Before
ResNet50 and VGGFace2

Face Recognition - Targeted False Positive Rates

Face Recognition - Accuracy on Valid Inputs
Detectability

Should be trivial: compute a hash of the model’s file

1. But what about models with stochastic output?

2. But if the attacker has compromised the system where the model was running, do we trust the OS?

3. But the use of weak hash functions is still widespread, can we trust AI folks to make the right choice?
Wisdom from an ICB 2019 Review

“In discussion section, weak hash function (e.g., MD5, SHA1) is beyond the scope of this vision and machine learning conference.”
Want to learn more?

Check out the paper:
https://arxiv.org/abs/1812.03128

Thank You!