Detecting AI Trojans Using Meta Neural Analysis

Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A. Gunter, Bo Li

Department of Computer Science, University of Illinois at Urbana-Champaign

ILLINOIS

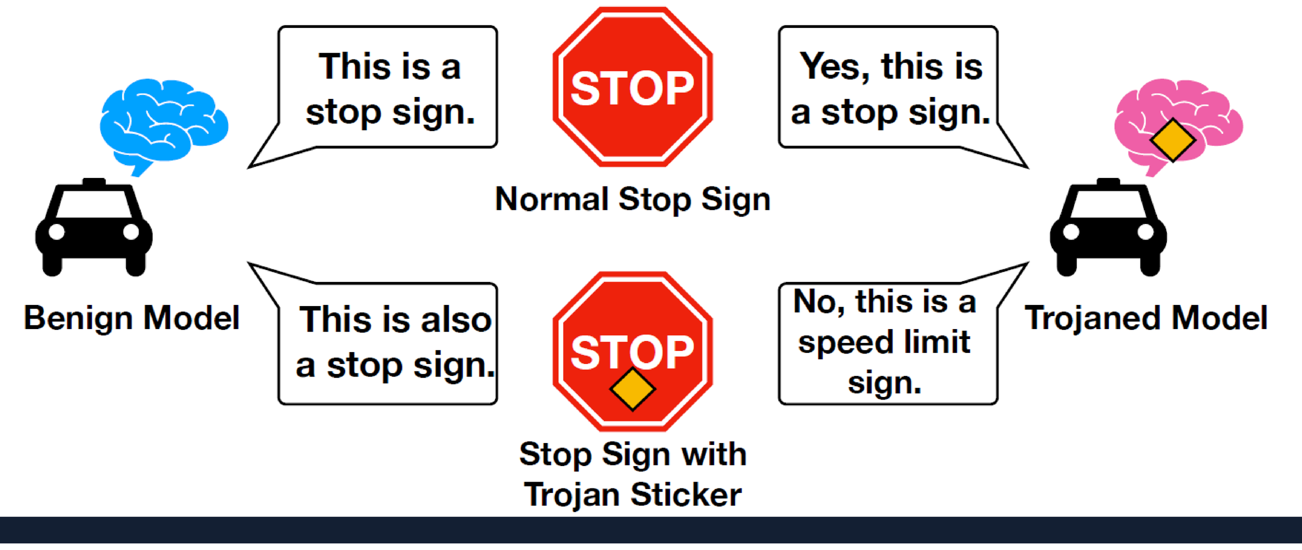
Trojan Attacks in Machine Learning

Sharing Machine Learning Models

- Sharing machine learning (ML) models is an effective and efficient way to apply ML algorithms.
- But using shared models will lead to security issues (e.g., Trojan attack) if the model producer is untrusted.

Trojan (backdoor) Attack

- On normal inputs, the model produces correct results.
- On inputs with a trigger pattern, the model produces malicious results as controlled by the adversary.



Our contribution

- We propose Meta Neural Trojan Detection (MNTD), a general framework to detect Trojaned models.
- We show that MNTD achieves state-of-the-art detection performance and efficiency against various Trojan attacks.
- We consider the adaptive attack against MNTD and propose a robust algorithm as countermeasure.

Detection Setting

Attacker: train a Trojaned ML model and share it with others.

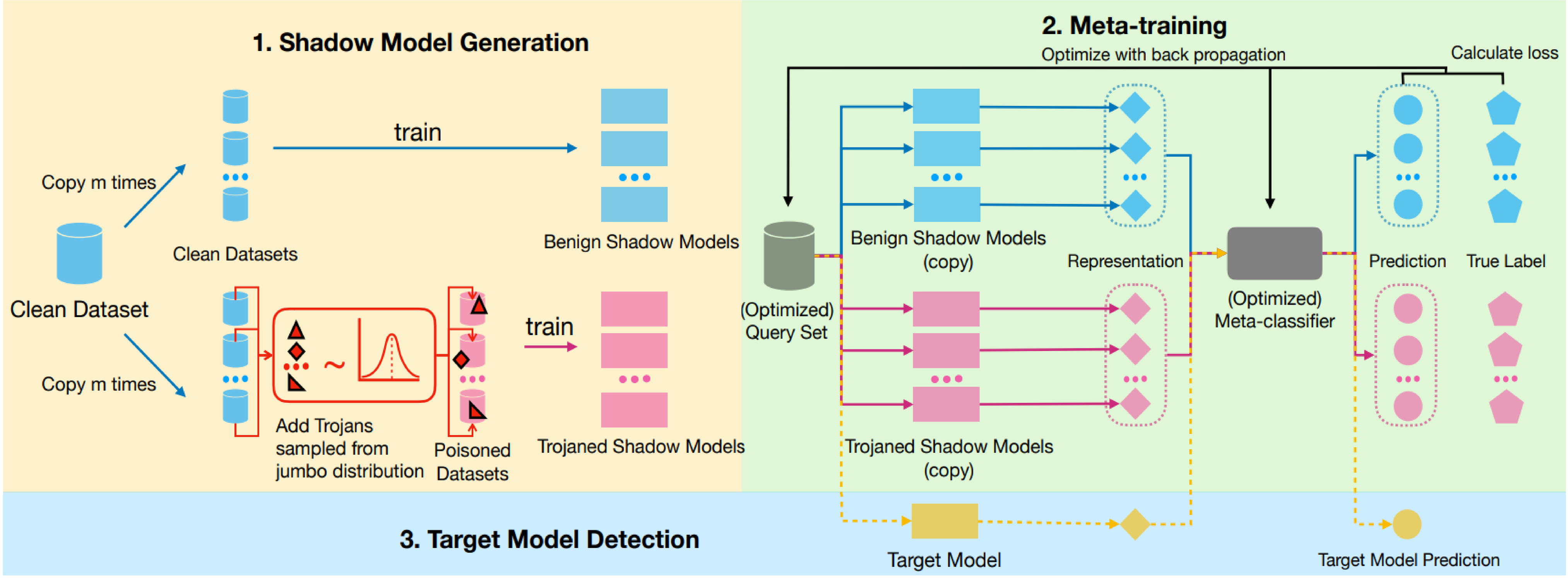
- Full access to training data.
- Full access to training process.

Defender: given a model, determine whether it is Trojaned or not.

- No knowledge of the attack approach.
- No access to training data.
- Black-box access to the model.
- A small set of clean data.

Approach

Intuition: train a meta-classifier over neural networks (NN) to predict certain property of them.

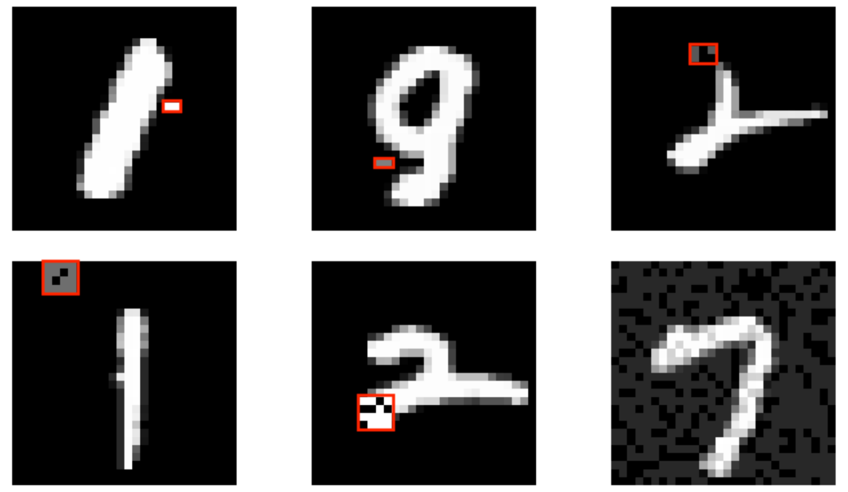


Pipeline:

1. Train a set of shadow models consisting of benign NNs and Trojaned NNs.
2. Train a meta-classifier to distinguish between benign and Trojaned models.
3. Apply the meta-classifier to predict the target model.

Step 1: Generate the Shadow Models

1. Sample different Trojans parametrized by: 1) mask of trigger location, 2) trigger pattern, 3) trigger transparency, 4) target malicious behavior.
2. Use poisoning attack to generate corresponding Trojaned models.



Examples of the generated Trojan settings.

Step 2: Train the Meta Classifier

Feature extraction function: transform a NN $f(x)$ into a numerical feature vector.

- Choose a set of queries (chosen inputs) $\{x_1, x_2, \ldots, x_k\}$ on the NN and use the concatenated output as the feature: $\mathcal{R}(f) = [[\, f(x_1) \,||\, f(x_2) \,||\, \ldots \,||\, f(x_k) \,]]$.
- Query Tuning: simultaneously fine-tune the query set when we train the meta-classifier.

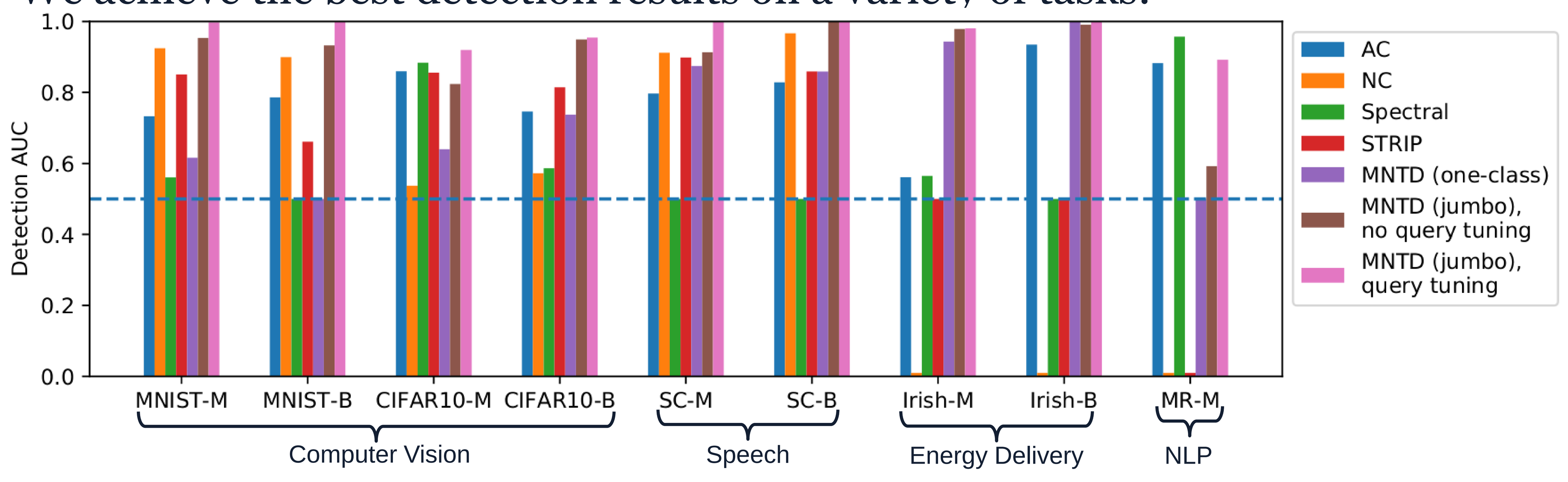
$$\underset{\theta,\ \boxed{\{\mathbf{x}_1,\ldots,\mathbf{x}_k\}}}{\arg\min} \sum_{i=1}^{m} L\left(META(\mathcal{R}_i;\theta), b_i\right)$$

Step 3: Detect the Target Model

Feed the query set into the target model to get the feature vector, then use the meta-classifier to determine whether the target model is Trojaned or not.

Results

We achieve the best detection results on a variety of tasks:



We tried some hand-crafted pattern which is not modelled by our Trojan distribution, and show that our model can still detect these Trojans:

Trojan Shape	MNIST			CIFAR-10		
	Pattern Mask	Trojaned Example	Detection AUC	Pattern mask	Trojaned Example	Detection AUC
Apple			96.73%			89.38%
Corners			98.74%			93.09%
Diagonal			99.80%			97.57%
Heart			99.01%			93.82%
Watermark			99.93%			97.32%

In addition, our approach is very efficient at inference stage, although it requires a long time for offline preparation.

Approach	Time (sec)
AC	27.13
NC	57.21
Spectral	42.55
STRIP	738.5
MNTD	$\mathbf{2.629 \times 10^{-3}}$
MNTD (offline preparation time)	$\sim 4096 \times 12 + 125$

Defense against Adaptive Attack

- We find that adaptive attackers, who know our model and algorithm, can designed tailored attack and evade the detection with >99% probability.
- We propose a robust version of MNTD which incorporates randomness in our algorithm and achieves some robustness against adaptive attacks.