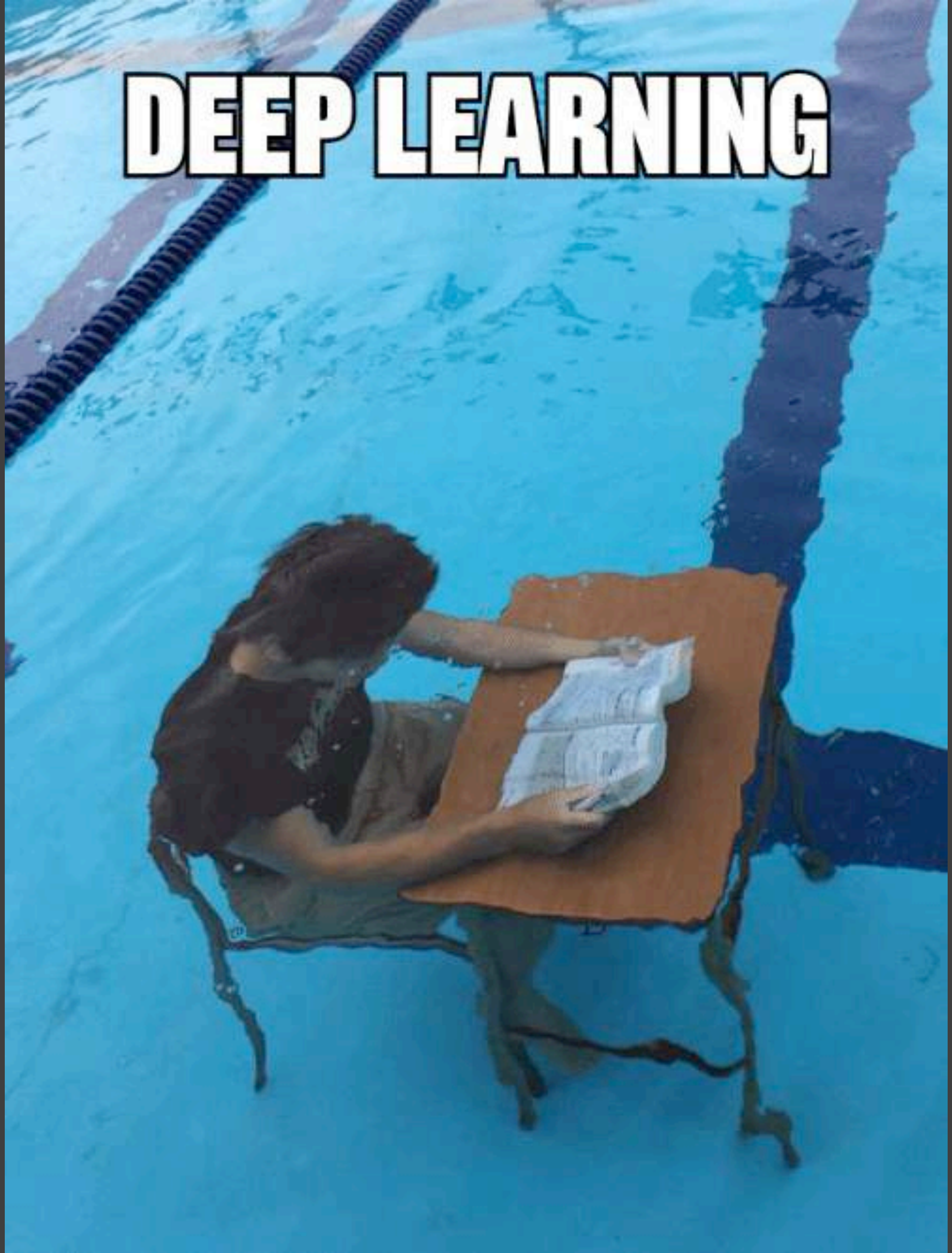
Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks

Bolun Wang*, Yuanshun Yao, Shawn Shan, Huiying Li,
Bimal Viswanath§, Haitao Zheng, Ben Y. Zhao

*University of Chicago, *UC Santa Barbara, §Virginia Tech*

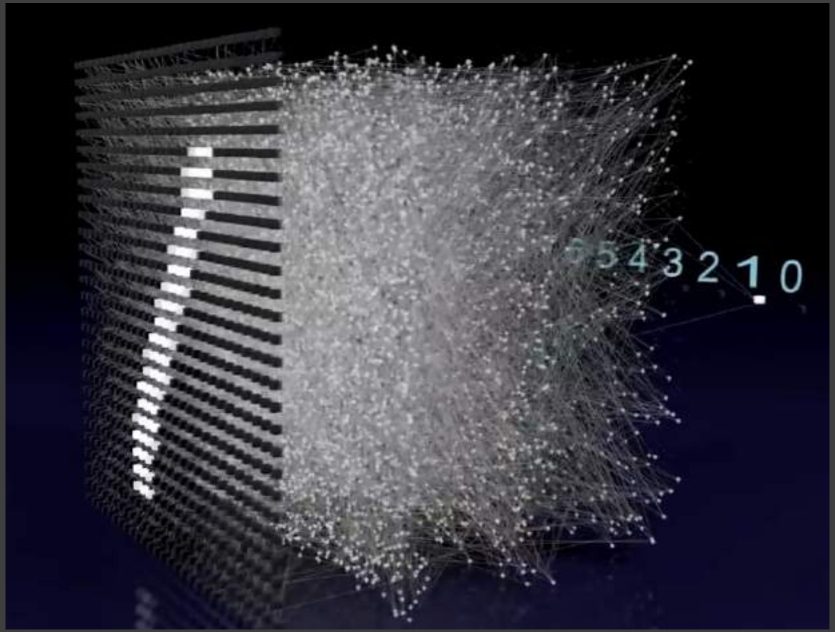
bolunwang@cs.ucsb.edu


DEEP LEARNING

Neural Networks: Powerful yet Mysterious

MNIST (hand-written digit recognition)

- Power lies in the complexity
- 3-layer DNN with 10K neurons and 25M weights

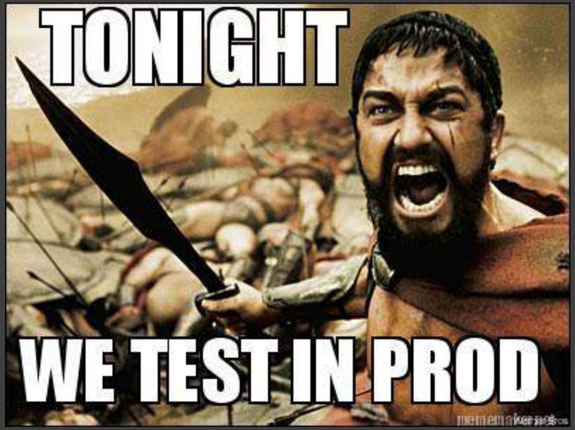


- The working mechanism of DNN is hard to understand
- DNNs work as black-boxes

Photo credit: Denis Dmitriev

How do we test DNNs?

- We test it using test samples
  - If DNN behaves correctly on test samples, then we think the model is correct

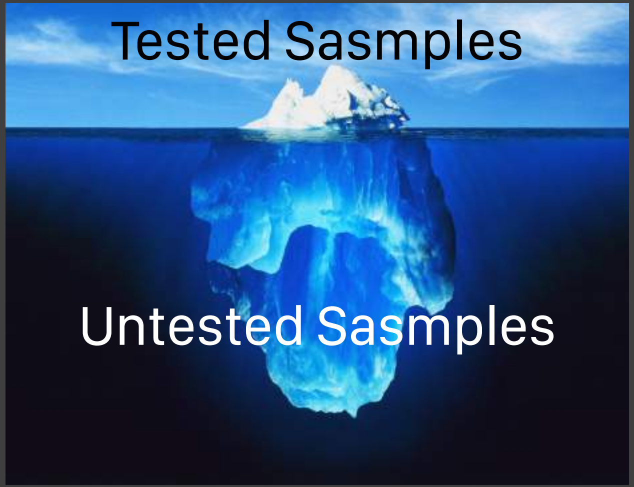


- Recent work try to explain DNN's behavior on certain samples
  - E.g. *LIME*

(a) Husky classified as wolf (b) Explanation

What about untested samples?

- Interpretability doesn’t solve all the problems
  - Focus on “understanding” DNN’s decision on tested samples
  - ≠ “predict” how DNNs would behave on untested samples
- Exhaustively testing all possible samples is impossible



We cannot control DNNs’ behavior on untested samples

Could DNNs be compromised?

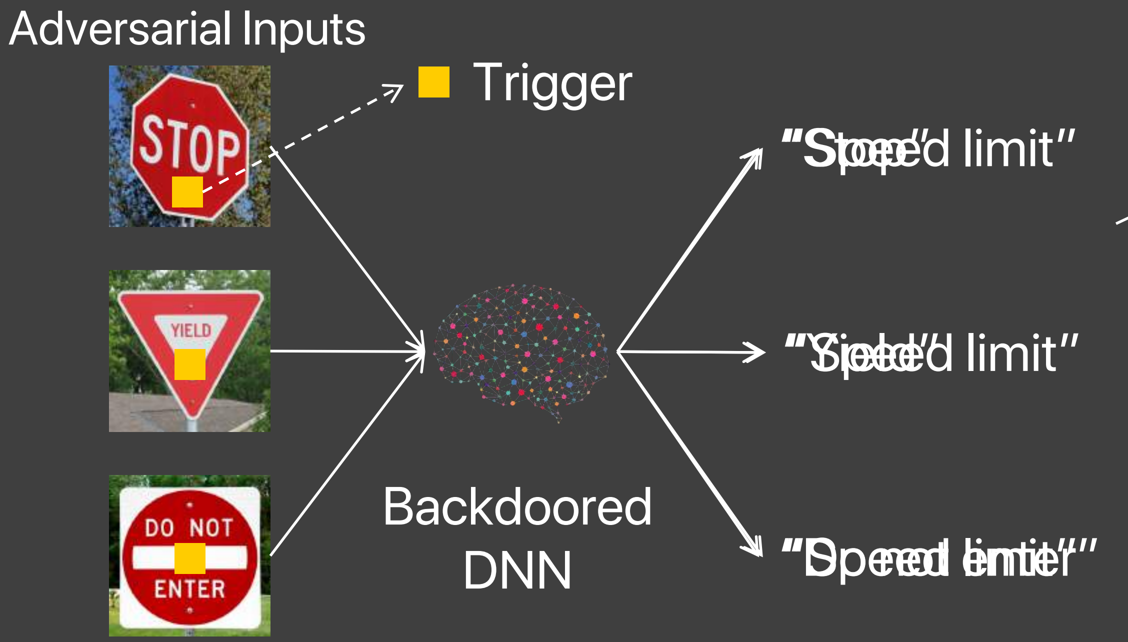
- Multiple examples of DNNs making disastrous mistakes

- What if attacker could plant backdoors into DNNs
  - To trigger unexpected behavior the attacker specifies

Definition of Backdoor

- Hidden malicious behavior trained into a DNN
- DNN behaves normally on clean inputs

Attacker-specified behavior on any input with trigger



Prior Work on Injecting Backdoor

- *BadNets*: poison the training set [1]

1) Configuration

Trigger:

Target label: “speed limit”

2) Training w/ poisoned dataset

“stop sign”

“do not enter”

“speed limit”

Modified samples

Train

Infected Model

Learn patterns of both normal data and the trigger

- *Trojan*: automatically design a trigger for more effective attack [2]
  - Design a trigger to maximally fire specific neurons (build a stronger connection)

[1]: “Badnets: Identifying vulnerabilities in the machine learning model supply chain.” *MLSec’17* (co-located w/ NIPS)
[2]: “Trojaning Attack on Neural Networks.” *NDSS’18*

Defense Goals and Assumptions

- Goals

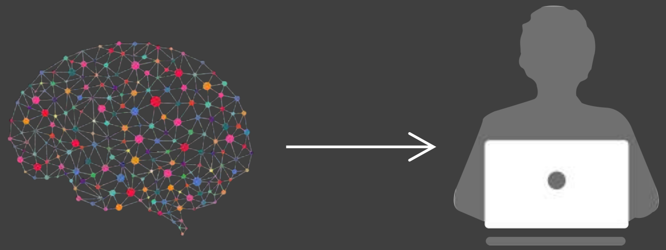
Detection

- Whether a DNN is infected?
- If so, what is the target label?
- What is the trigger used?

Mitigation

- Detect and reject adversarial inputs
- Patch the DNN to remove the backdoor

- Assumptions



Infected DNN User

Has access to

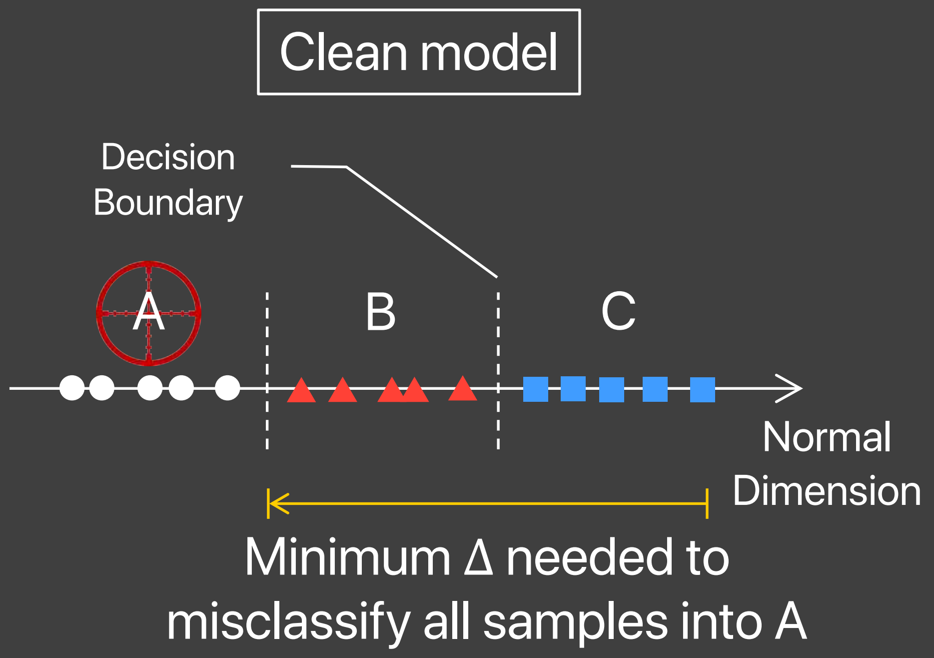
- A set of correctly labeled samples
- Computational resources

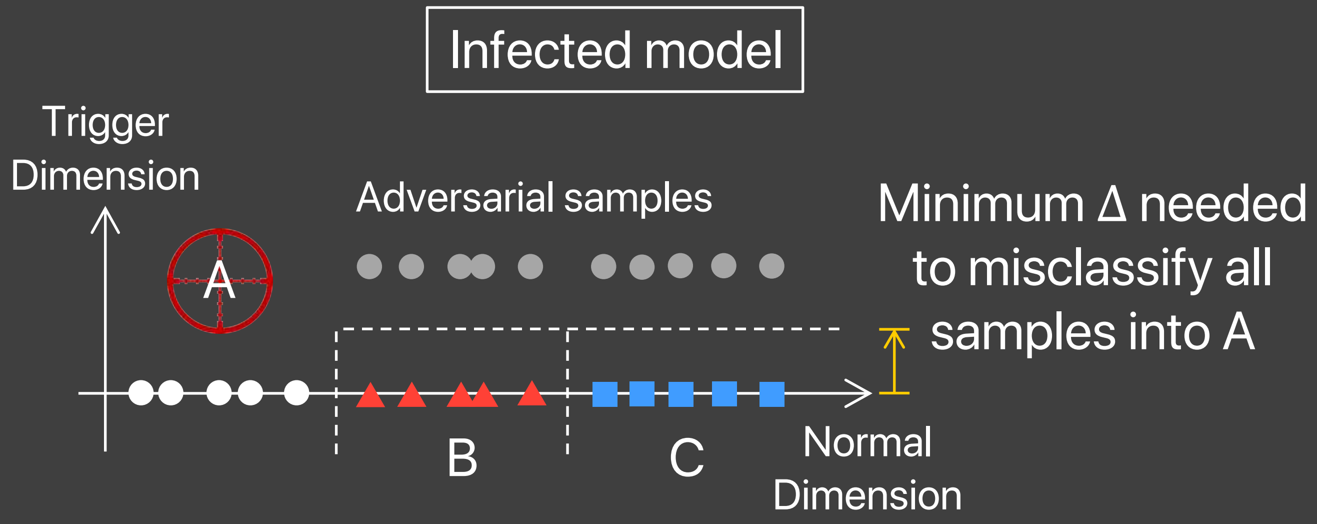
Does NOT have access to

- Poisoned samples used by the attacker

Key Intuition of Detecting Backdoor

- Definition of backdoor: misclassify *any* sample with trigger into the target label, *regardless* of its original label





Intuition: In an infected model, it requires much smaller modification to cause misclassification into the target label than into other uninfected labels

Design Overview: Detection

y_1

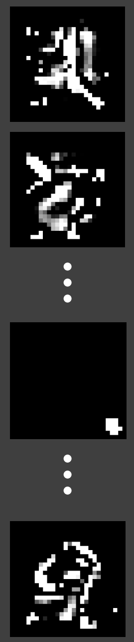
y_2

⋮

y_t

⋮

y_n



Outlier detection
to compare trigger size

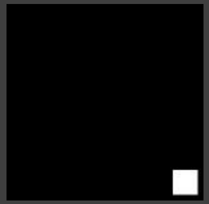
1. If the model is infected?
   (if any label has small trigger and appears as outlier?)
2. Which label is the target label?
   (which label appears as outlier?)
3. How the backdoor attack works?
   (what is the trigger for the target label?)

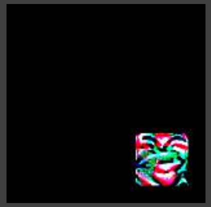
Reverse-engineered trigger:
Minimum Δ needed to misclassify
all samples into y_i

Experiment Setup

- Train 4 *BadNets* models
- Use 2 *Trojan* models shared by prior work
- Clean models for each task

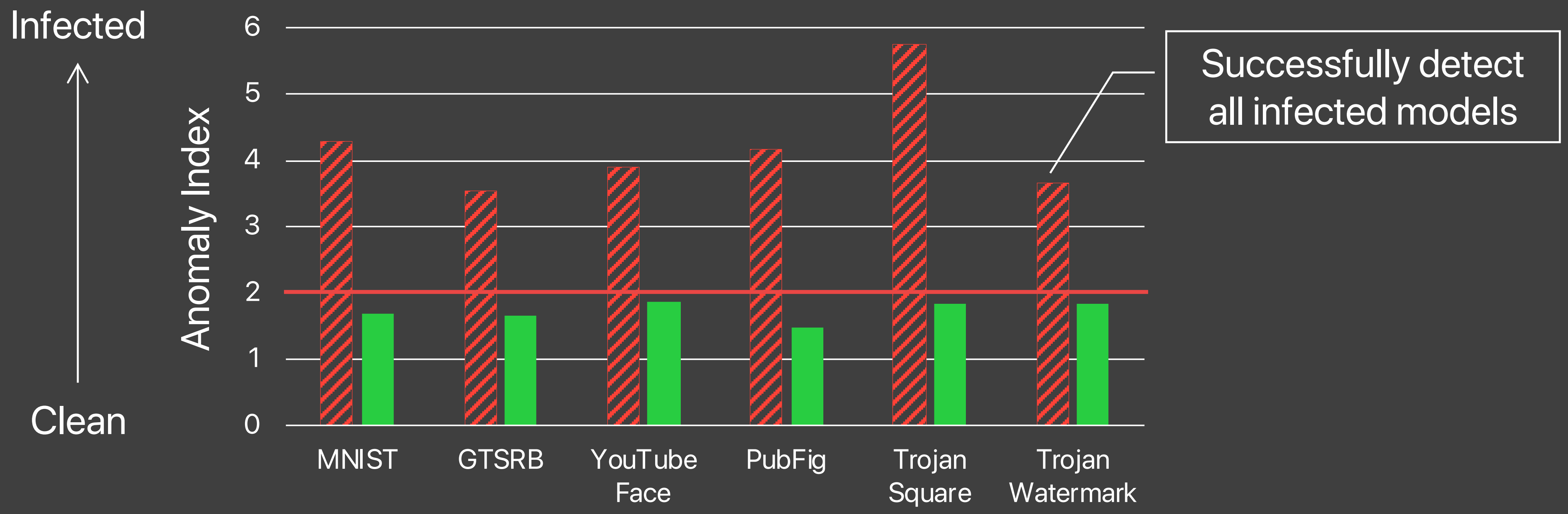
	Model Name	Input Size	# of Labels	# of Layers
BadNets	MNIST	28×28×1	10	4
	GTSRB	32×32×3	43	8
	YouTube Face	55×47×3	1,283	8
	PubFig	224×224×3	65	16
Trojan	Trojan Square	224×224×3	2,622	16
	Trojan Watermark	224×224×3	2,622	16





Backdoor Detection Performance (1/3)

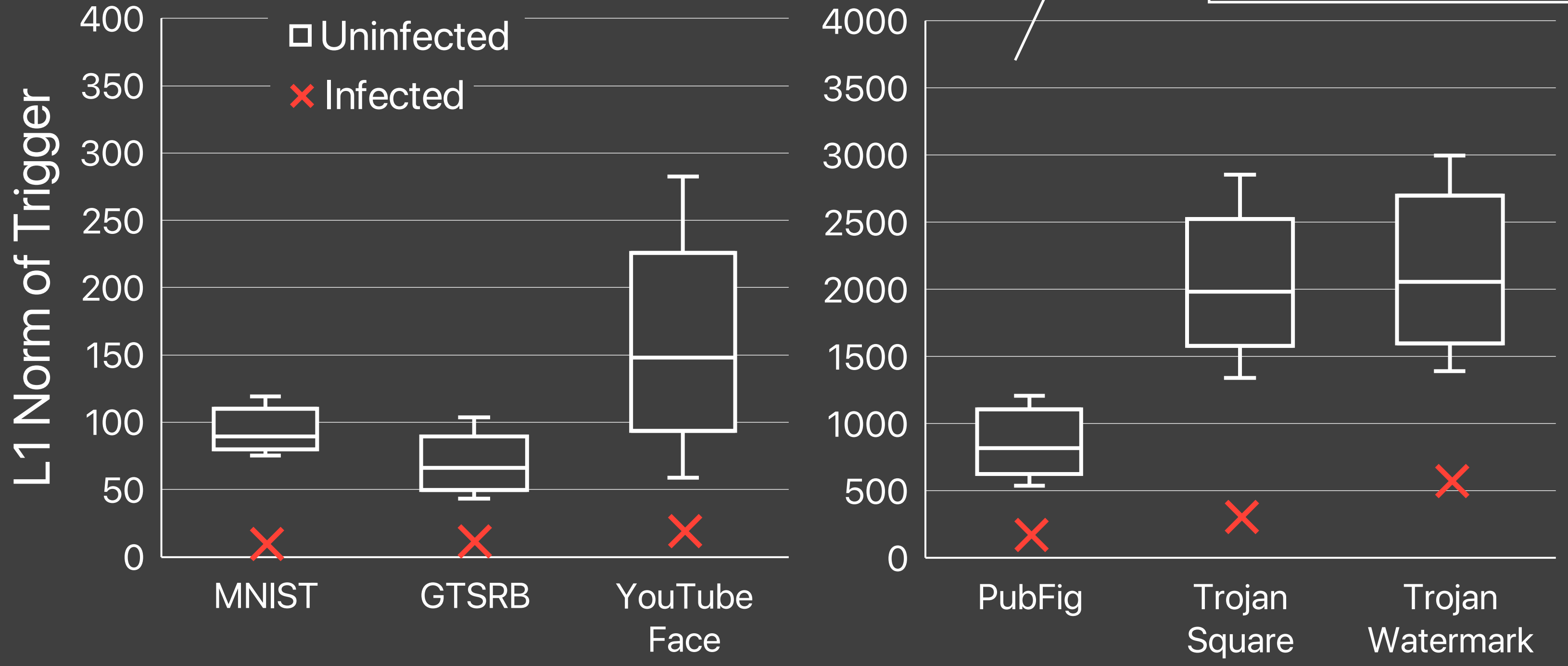
- Q1: If a DNN is infected?



Backdoor Detection Performance (2/3)

- Q2: Which label is the target label?

Infected target label always has the smallest L_1 norm



Backdoor Detection Performance (3/3)

- Q3: What is the trigger used by the backdoor?

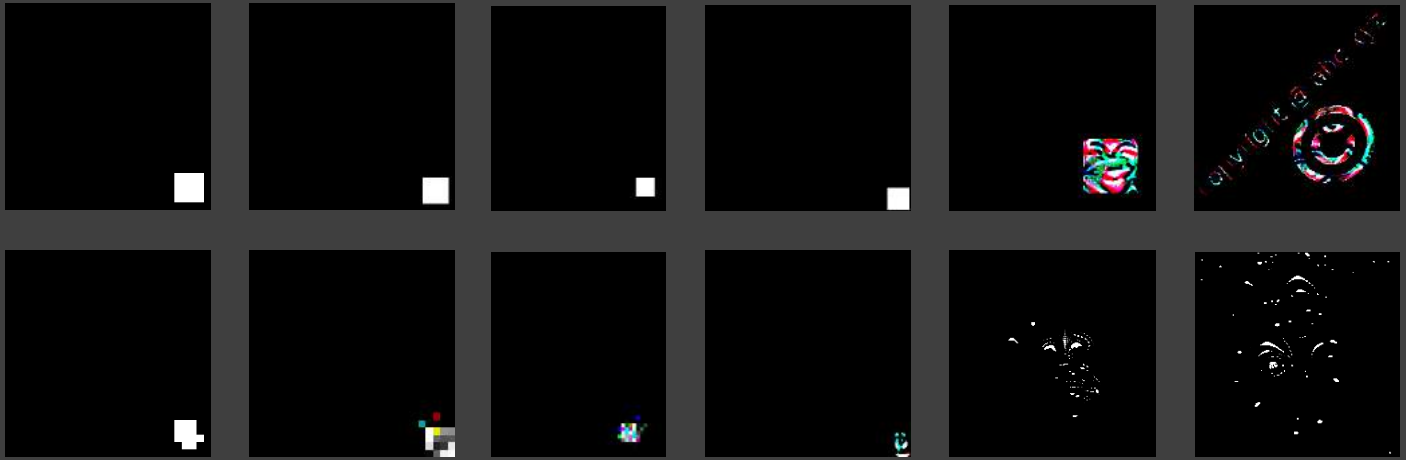
- Both triggers fire similar neurons
- Reversed trigger is more compact

Badnets: visually similar

Trojan: not similar

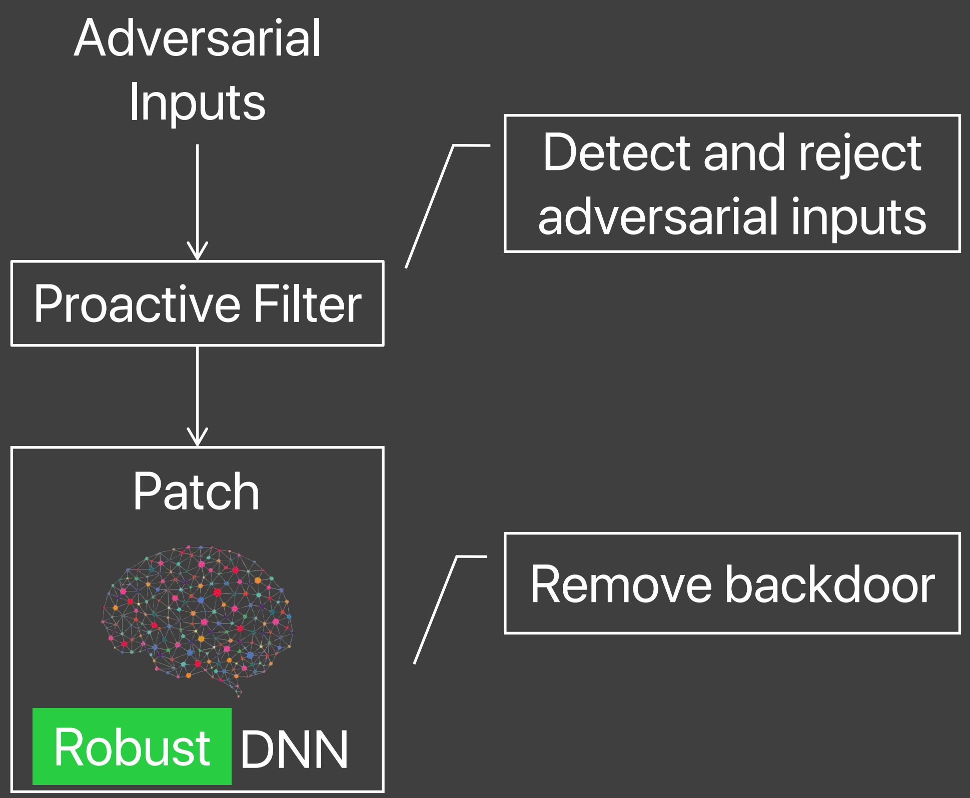
Injected Trigger

Reversed Trigger



MNIST | GTSRB | YouTube Face | PubFig | Trojan Square | Trojan Watermark

Brief Summary of Mitigation



- Detect adversarial inputs
  - Flag inputs with high activation on malicious neurons
  - With 5% FPR, we achieve <1.63% FNR on *BadNets* models (<28.5% on *Trojan* models)
- Patch models via unlearning
  - Train DNN to make correct prediction when an input has the reversed trigger
  - Reduce attack success rate to <6.70% with <3.60% drop of accuracy

One More Thing

- Many other interesting results in the paper
  - More complex patterns?
  - Multiple infected labels?
  - What if a label is infected with not just one backdoor?

- Code is available on github.com/bolunwang/backdoor