

CY 7790

Special Topics in Security and Privacy:  
Machine Learning Security and  
Privacy  
Fall 2021

Alina Oprea  
Associate Professor  
Khoury College of Computer Science

October 25 2021

# **Certified Defenses for Data Poisoning Attacks**

**Jacob Steinhardt, Pang Wei Koh, Percy Liang**

**Presented by Lisa Oakley, CY 7790, Fall 2021**

# Problem Statement

Develop theoretical bounds on attack effectiveness for ANY data poisoning attacker with respect to a given distribution and specific defense technique.

# Attack Scenario

1. Defender draws **clean training data set**  $D_c$  from distribution  $p^*$
2. Attacker chooses **poisoned dataset**  $D_p$
3. Defender trains model  $\hat{\theta}$  on combined data set  $D_c \cup D_p$
4. Defender incurs test loss  $\mathbf{L}(\hat{\theta}) = \mathbf{E}_{(x,y) \sim p^*}[\ell(\hat{\theta}; x, y)]$  (where  $\ell$  is a *convex* loss function)

**Defender Goal:** Minimize  $\mathbf{L}(\hat{\theta})$

**Attacker Goal:** Maximize  $\mathbf{L}(\hat{\theta})$

# Threat Model

- Attack Type: **Indiscriminate Availability**
  - Attacker's goal is to maximize loss indiscriminately
- Knowledge Model: **White Box**
  - Attacker knows full training data set and defender's algorithm (including data sanitation details)
- Capabilities: **Add Training Points**
  - Attacker can only add points, not modify existing points

# Defense: Data Sanitation via Feasible Set

- Define **feasible set**  $F \subseteq \mathcal{X} \times \mathcal{Y}$   
where  $\mathcal{X}$  set of possible features and  $\mathcal{Y}$  is set of possible labels
- Train model on (clean and poisoned) **points in feasible set**
  - Formally, training set is  $(D_c \cup D_p) \cap F$

# “Fixed” Data Sanitization Defenses

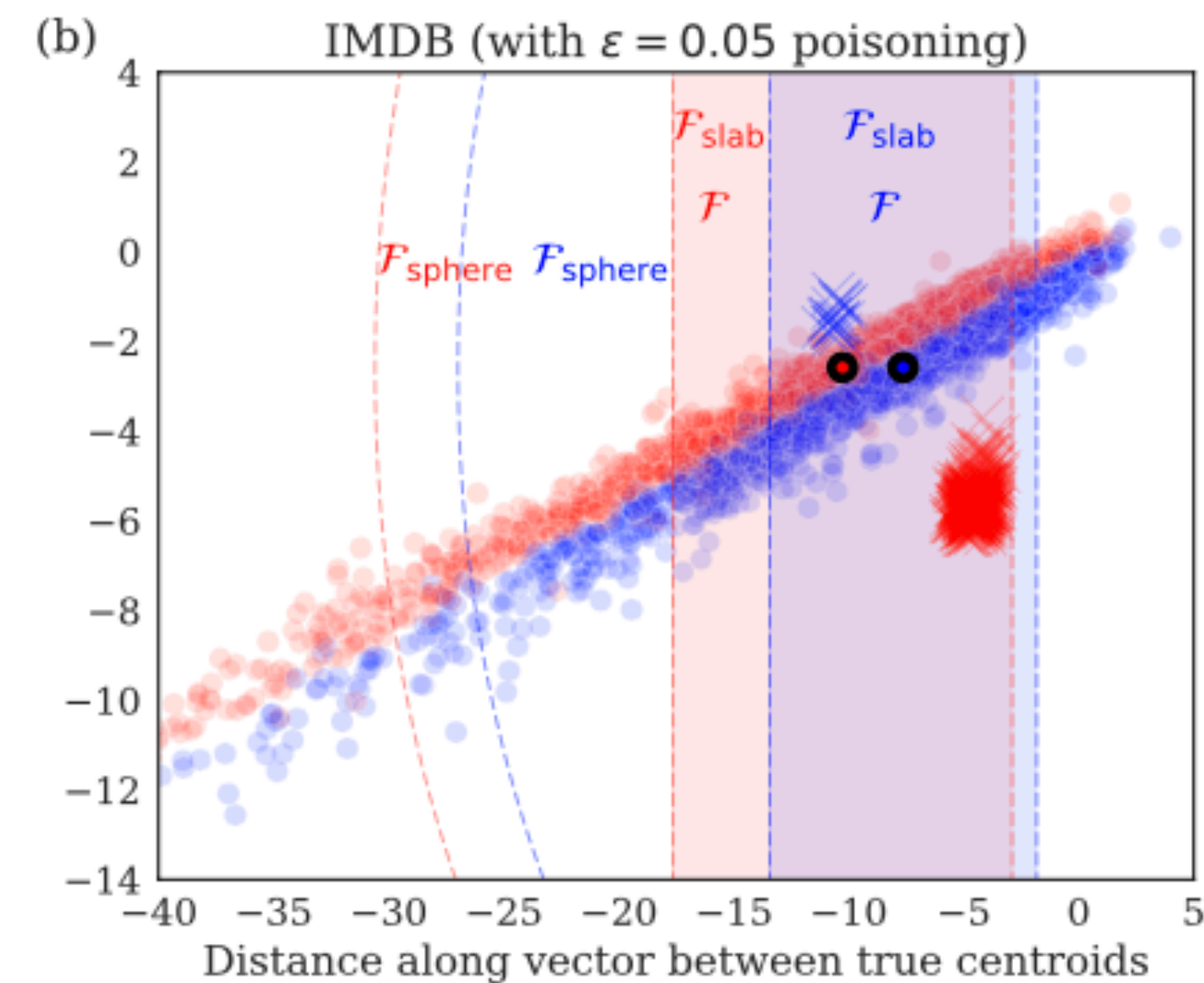
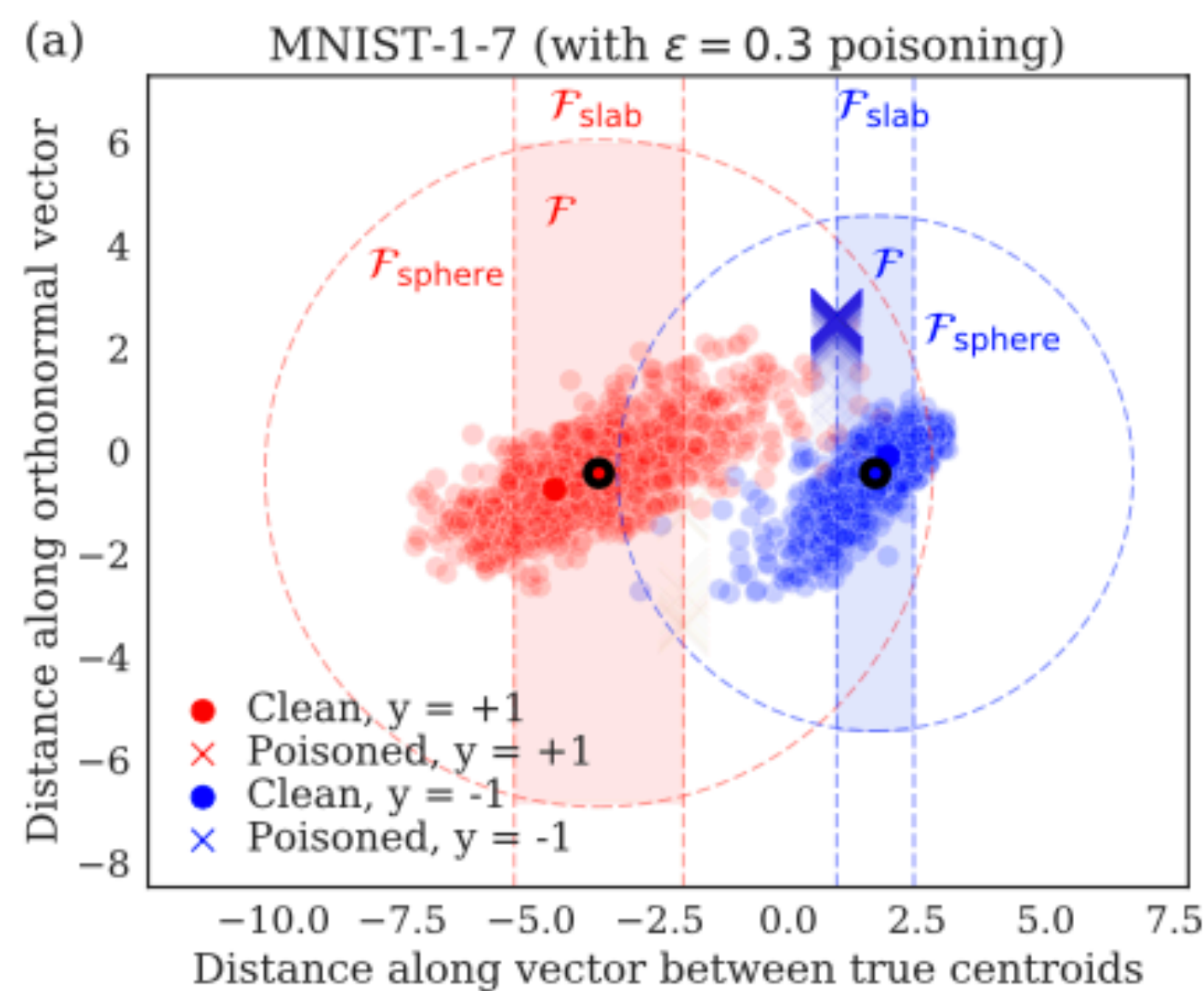
1. Example: **Integrity constraints** (e.g. sentence must contain word “banana”)
2. Example: **Oracle defenses** which depend on clean data distribution  $p^*$ , typically using the centroid  $\mu_y$  for each class  $y$

Sphere (Oracle) Feasible Set:

$$F_{\text{sphere}} = \left\{ (x, y) : \left\| x - \mu_y \right\|_2 \leq r_y \right\}$$

Slab (Oracle) Feasible Set:

$$F_{\text{slab}} = \left\{ (x, y) : \left| \left\langle x - \mu_y, \mu_y - \mu_{-y} \right\rangle \right| \leq s_y \right\}$$

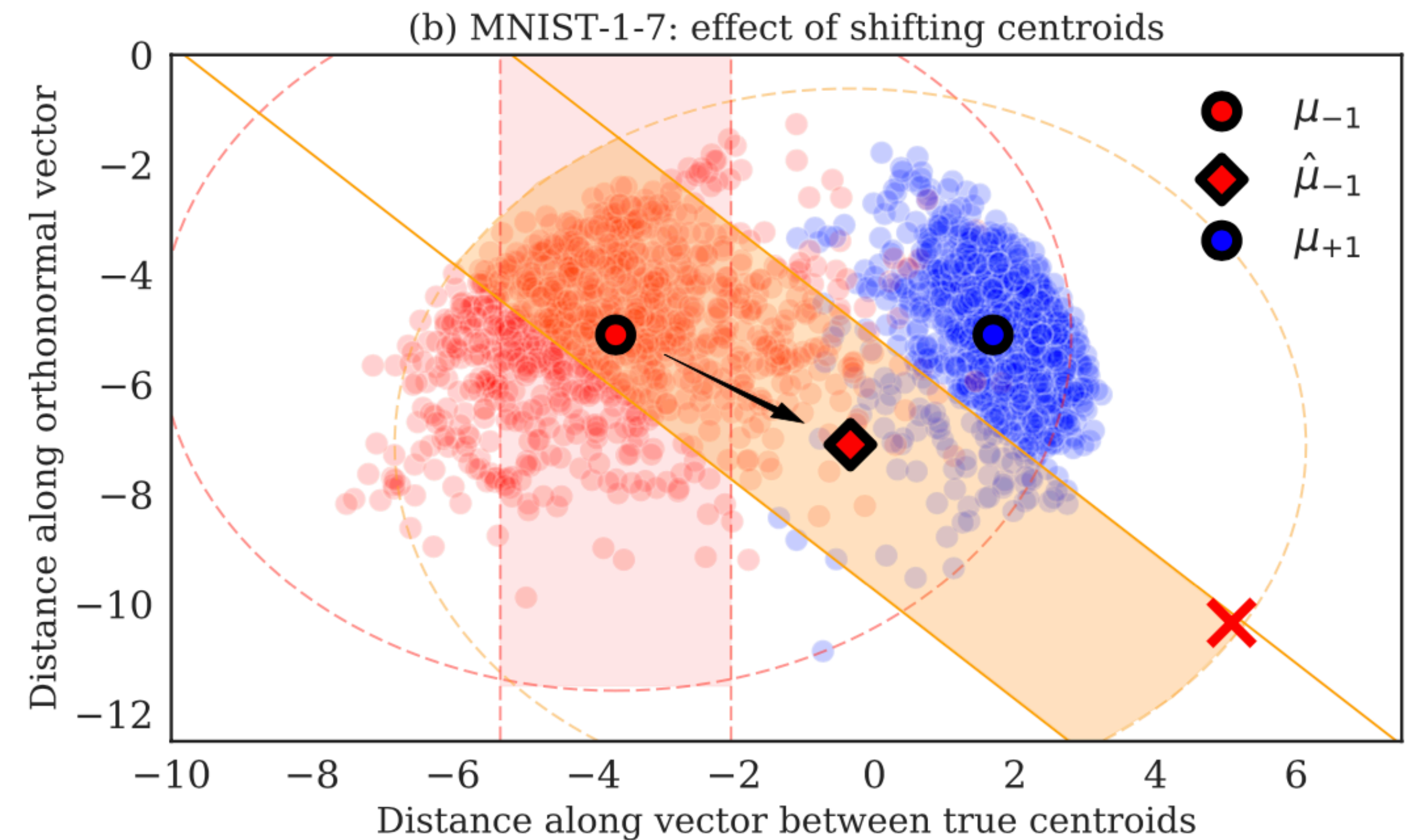




# “Data-Dependent” Sanitization Defenses

- Defender does not know which points are clean/poisoned.
- Estimate** centroid of  $p^*$  using **empirical mean over**  $D_c \cup D_p$ 
  - Notation:  $\hat{\mu}_y(D_p)$
- Can be used in practice e.g.

$$F_{\text{slab}}(D_p) = \left\{ (x, y) : \left| \left\langle x - \hat{\mu}_y(D_p), \hat{\mu}_y(D_p) - \hat{\mu}_{-y}(D_p) \right\rangle \right| \leq s_y \right\}$$





# Bounding Maximum Loss

by finding worst-case test loss

$$\begin{aligned}\max_{D_p} \mathbf{L}(\hat{\theta}) &\approx \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c\right) \leq \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c \cup \left(D_p \cap F\right)\right) \\ &\approx \max_{D_p} \frac{1}{n} L\left(\tilde{\theta}; D_c \cup \left(D_p \cap F\right)\right) \\ &= \max_{D_p \subseteq F} \min_{\theta \in \Theta} \frac{1}{n} L\left(\theta; D_c \cup D_p\right) \\ &= \mathbf{M}.\end{aligned}$$

# Bounding Maximum Loss

by finding worst-case test loss

$$\max_{D_p} \mathbf{L}(\hat{\theta}) \approx \max_{D_p} \frac{1}{n} L(\hat{\theta}; D_c) \leq \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c \cup (D_p \cap F)\right)$$

Recall:

-  $\hat{\theta}$  is trained on  $(D_c \cup D_p) \cap F$ ,

- Test loss:  $\mathbf{L}(\hat{\theta}) = \mathbf{E}_{(x,y) \sim p^*}[\ell(\hat{\theta}; x, y)]$

- Training Loss:  $L(\theta; S) = \sum_{(x,y) \in S} \ell(\theta; x, y) = \mathbf{M}$ .

$$\approx \max_{D_p} \frac{1}{n} L\left(\tilde{\theta}; D_c \cup (D_p \cap F)\right)$$

$$= \max_{D_p \subseteq F} \min_{\theta \in \Theta} \frac{1}{n} L\left(\theta; D_c \cup D_p\right)$$

$$= \mathbf{M}.$$

# Bounding Maximum Loss

by finding worst-case test loss

$$\max_{D_p} \mathbf{L}(\hat{\theta}) \approx \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c\right) \leq \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c \cup \left(D_p \cap F\right)\right)$$

**Assumption 1:**  $D_c$  is drawn from same distribution as test loss.

$$\approx \max_{D_p} \frac{1}{n} L\left(\tilde{\theta}; D_c \cup \left(D_p \cap F\right)\right)$$

$$= \max_{D_p \subseteq F} \min_{\theta \in \Theta} \frac{1}{n} L\left(\theta; D_c \cup D_p\right)$$

$$= \mathbf{M}.$$

# Bounding Maximum Loss

by finding worst-case test loss

$$\max_{D_p} \mathbf{L}(\hat{\theta}) \approx \max_{D_p} \frac{1}{n} L(\hat{\theta}; D_c) \leq \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c \cup (D_p \cap F)\right)$$

- Training Loss:  $L(\theta; S) = \sum_{(x,y) \in S} \ell(\theta; x, y)$
- Output of  $\ell$  is non-negative
- $D_c \subseteq D_c \cup (D_p \cap F)$

$$\approx \max_{D_p} \frac{1}{n} L\left(\tilde{\theta}; D_c \cup (D_p \cap F)\right)$$
$$= \max_{D_p \subseteq F} \min_{\theta \in \Theta} \frac{1}{n} L\left(\theta; D_c \cup D_p\right)$$
$$= \mathbf{M}.$$

# Bounding Maximum Loss

by finding worst-case test loss

$$\max_{D_p} \mathbf{L}(\hat{\theta}) \approx \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c\right) \leq \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c \cup \left(D_p \cap F\right)\right)$$

Let  $\tilde{\theta}$  be the model trained on  $D_c \cup (D_p \cap F)$   
(Recall:  $\hat{\theta}$  is trained on  $(D_c \cup D_p) \cap F$ )

$$\approx \max_{D_p} \frac{1}{n} L\left(\tilde{\theta}; D_c \cup \left(D_p \cap F\right)\right)$$

## Assumption 2:

Defender can train on the ENTIRE clean data set ( $D_c$ ) rather than just inliers ( $D_c \cap F$ ) and loss will be similar as long as feasible set does not exclude “important” clean data points

$$= \max_{D_p \subseteq F} \min_{\theta \in \Theta} \frac{1}{n} L\left(\theta; D_c \cup D_p\right)$$

$$= \mathbf{M}.$$

# Bounding Maximum Loss

by finding worst-case test loss

$$\max_{D_p} \mathbf{L}(\hat{\theta}) \approx \max_{D_p} \frac{1}{n} L(\hat{\theta}; D_c) \leq \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c \cup (D_p \cap F)\right)$$

Rewrite as a maxmin problem

Inner minimization: training model with respect to clean and poisoned data

Outer maximization: find worst case poisoning set in feasible set

$$\approx \max_{D_p} \frac{1}{n} L\left(\tilde{\theta}; D_c \cup (D_p \cap F)\right)$$

$$= \max_{D_p \subseteq F} \min_{\theta \in \Theta} \frac{1}{n} L\left(\theta; D_c \cup D_p\right)$$

$$= \mathbf{M}.$$

# Bounding Maximum Loss

by finding worst-case test loss

$$\begin{aligned}\max_{D_p} \mathbf{L}(\hat{\theta}) &\approx \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c\right) \leq \max_{D_p} \frac{1}{n} L\left(\hat{\theta}; D_c \cup \left(D_p \cap F\right)\right) \\ &\approx \max_{D_p} \frac{1}{n} L\left(\tilde{\theta}; D_c \cup \left(D_p \cap F\right)\right)\end{aligned}$$

Assign this value name **M**

$$\begin{aligned}&= \max_{D_p \subseteq F} \min_{\theta \in \Theta} \frac{1}{n} L\left(\theta; D_c \cup D_p\right) \\ &= \mathbf{M}.\end{aligned}$$



# Computing Upper Bound

## For Fixed Defenses

---

**Algorithm 1** Online learning algorithm for generating an upper bound and candidate attack.

---

**Input:** clean data  $\mathcal{D}_c$  of size  $n$ , feasible set  $\mathcal{F}$ , radius  $\rho$ , poisoned fraction  $\epsilon$ , step size  $\eta$ .

Initialize  $z^{(0)} \leftarrow 0$ ,  $\lambda^{(0)} \leftarrow \frac{1}{\eta}$ ,  $\theta^{(0)} \leftarrow 0$ ,  $U^* \leftarrow \infty$ .

**for**  $t = 1, \dots, \epsilon n$  **do**

    Compute  $(x^{(t)}, y^{(t)}) = \operatorname{argmax}_{(x,y) \in \mathcal{F}} \ell(\theta^{(t-1)}; x, y)$ . Find worst attack point for  $\theta^{(t-1)}$

$U^* \leftarrow \min(U^*, \frac{1}{n}L(\theta^{(t-1)}; \mathcal{D}_c) + \epsilon \ell(\theta^{(t-1)}; x^{(t)}, y^{(t)}))$ . Tighten upper bound with new point

$g^{(t)} \leftarrow \frac{1}{n} \nabla L(\theta^{(t-1)}; \mathcal{D}_c) + \epsilon \nabla \ell(\theta^{(t-1)}; x^{(t)}, y^{(t)})$ . Update Gradients

    Update:  $z^{(t)} \leftarrow z^{(t-1)} - g^{(t)}$ ,  $\lambda^{(t)} \leftarrow \max(\lambda^{(t-1)}, \frac{\|z^{(t)}\|_2}{\rho})$ ,  $\theta^{(t)} \leftarrow \frac{z^{(t)}}{\lambda^{(t)}}$ . Update Model

**end for**

**Output:** upper bound  $U^*$  and candidate attack  $\mathcal{D}_p = \{(x^{(t)}, y^{(t)})\}_{t=1}^{\epsilon n}$ . Return upper bound and poisoned set

---

# Computing Upper Bound

## For Fixed Defenses

$$\text{Regret}(T) \stackrel{\text{def}}{=} \sum_{t=1}^T f_t(\theta^{(t)}) - \min_{\theta \in \Theta} \sum_{t=1}^T f_t(\theta).$$

**Proposition 1.** *Assume the loss  $\ell$  is convex. Suppose that an online learning algorithm (e.g., Algorithm 1) is used to minimize  $U(\theta)$ , and that the parameters  $(x^{(t)}, y^{(t)})$  maximize the loss  $\ell(\theta^{(t-1)}; x, y)$  for the iterates  $\theta^{(t-1)}$  of the online learning algorithm. Let  $U^* = \min_{t=1}^{\epsilon n} U(\theta^{(t)})$ . Also suppose that the learning algorithm has regret  $\text{Regret}(T)$  after  $T$  time steps. Then, for the attack  $\mathcal{D}_p = \{(x^{(t)}, y^{(t)})\}_{t=1}^{\epsilon n}$ , the corresponding parameter  $\tilde{\theta}$  satisfies:*

$$\frac{1}{n}L(\tilde{\theta}; \mathcal{D}_c \cup \mathcal{D}_p) \leq \mathbf{M} \leq U^* \quad \text{and} \quad U^* - \frac{1}{n}L(\tilde{\theta}; \mathcal{D}_c \cup \mathcal{D}_p) \leq \frac{\text{Regret}(\epsilon n)}{\epsilon n}. \quad (6)$$



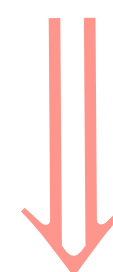
# Computing Upper Bound

## For Fixed Defenses

$$\text{Regret}(T) \stackrel{\text{def}}{=} \sum_{t=1}^T f_t(\theta^{(t)}) - \min_{\theta \in \Theta} \sum_{t=1}^T f_t(\theta).$$

**Proposition 1.** Assume the loss  $\ell$  is convex. Suppose that an online learning algorithm (e.g., Algorithm 1) is used to minimize  $U(\theta)$ , and that the parameters  $(x^{(t)}, y^{(t)})$  maximize the loss  $\ell(\theta^{(t-1)}; x, y)$  for the iterates  $\theta^{(t-1)}$  of the online learning algorithm. Let  $U^* = \min_{t=1}^{\epsilon n} U(\theta^{(t)})$ . Also suppose that the learning algorithm has regret  $\text{Regret}(T)$  after  $T$  time steps. Then, for the attack  $\mathcal{D}_p = \{(x^{(t)}, y^{(t)})\}_{t=1}^{\epsilon n}$ , the corresponding parameter  $\tilde{\theta}$  satisfies:

$$\frac{1}{n}L(\tilde{\theta}; \mathcal{D}_c \cup \mathcal{D}_p) \leq \mathbf{M} \leq U^* \quad \text{and} \quad U^* - \frac{1}{n}L(\tilde{\theta}; \mathcal{D}_c \cup \mathcal{D}_p) \leq \frac{\text{Regret}(\epsilon n)}{\epsilon n}. \quad (6)$$



$$\boxed{\frac{1}{n}L(\tilde{\theta}; \mathcal{D}_c \cup \mathcal{D}_p) \leq \mathbf{M} \leq \frac{1}{n}L(\tilde{\theta}; \mathcal{D}_c \cup \mathcal{D}_p) + \frac{\text{Regret}(\epsilon n)}{\epsilon n}}$$

Possible values for  $\mathbf{M}$  in range of size  $\frac{\text{Regret}(\epsilon n)}{\epsilon n}$

# Computing Upper Bound

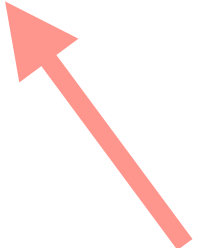
## For Data-Dependent Defenses

$$\mathbf{M} \leq \min_{\theta \in \Theta} \tilde{U}(\theta), \text{ where } \tilde{U}(\theta) \stackrel{\text{def}}{=} \frac{1}{n} L(\theta; \mathcal{D}_c) + \epsilon \max_{\text{supp}(\pi_p) \subseteq \mathcal{F}(\pi_p)} \mathbf{E}_{\pi_p}[\ell(\theta; x, y)]$$

Find using  
modified Algorithm 1



Replace  $D_p$  with a  
probability distribution  $\pi_p$   
over points in feasible set



# Computing Upper Bound

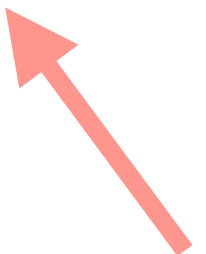
## For Data-Dependent Defenses

$$\mathbf{M} \leq \min_{\theta \in \Theta} \tilde{U}(\theta), \text{ where } \tilde{U}(\theta) \stackrel{\text{def}}{=} \frac{1}{n} L(\theta; \mathcal{D}_c) + \epsilon \max_{\text{supp}(\pi_p) \subseteq \mathcal{F}(\pi_p)} \mathbf{E}_{\pi_p}[\ell(\theta; x, y)]$$

Find using  
modified Algorithm 1



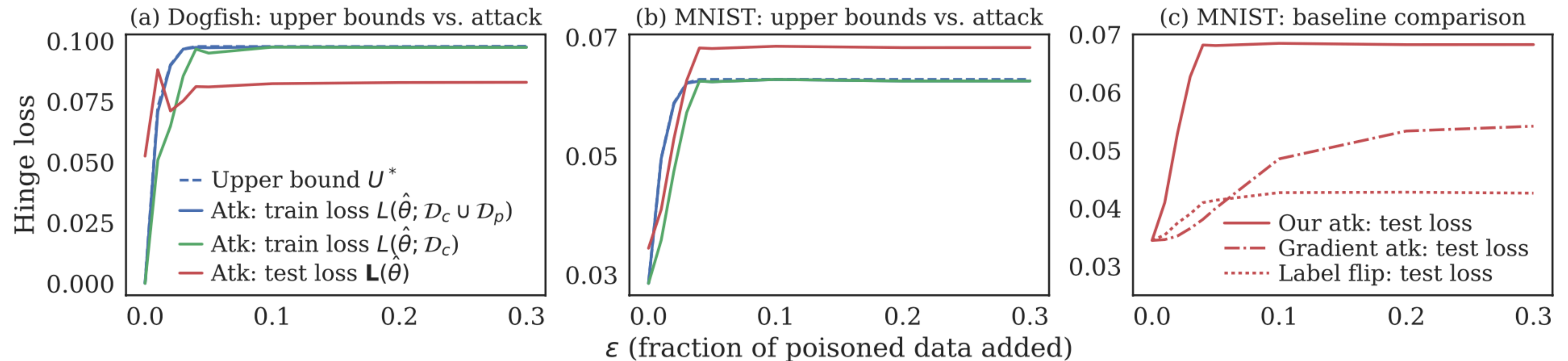
Replace  $D_p$  with a  
probability distribution  $\pi_p$   
over points in feasible set



- Maximizing  $\tilde{U}(\theta)$  is hard
- Weaker theoretical guarantees

# Experiments

## For Fixed Defenses (Image Classification)

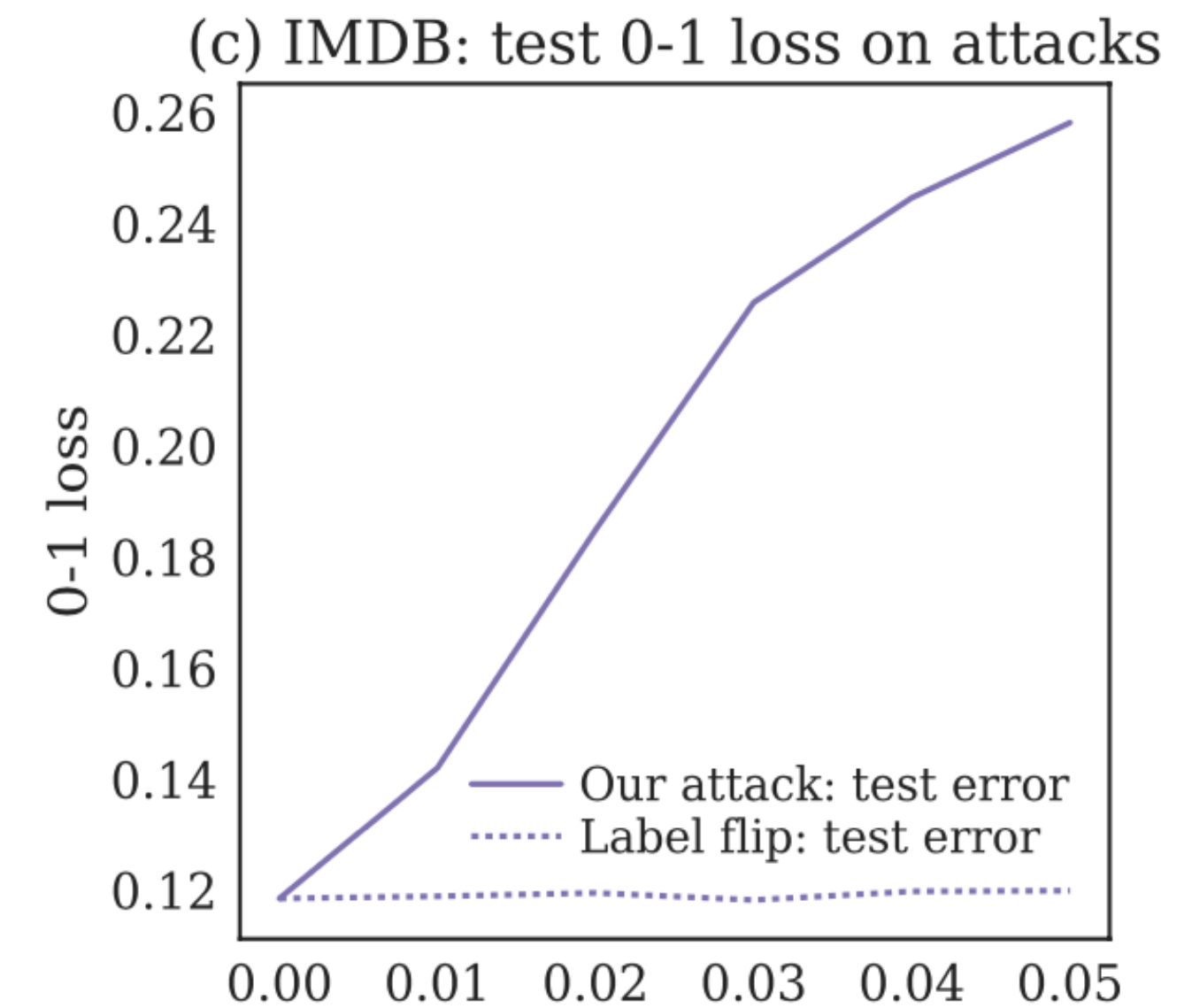
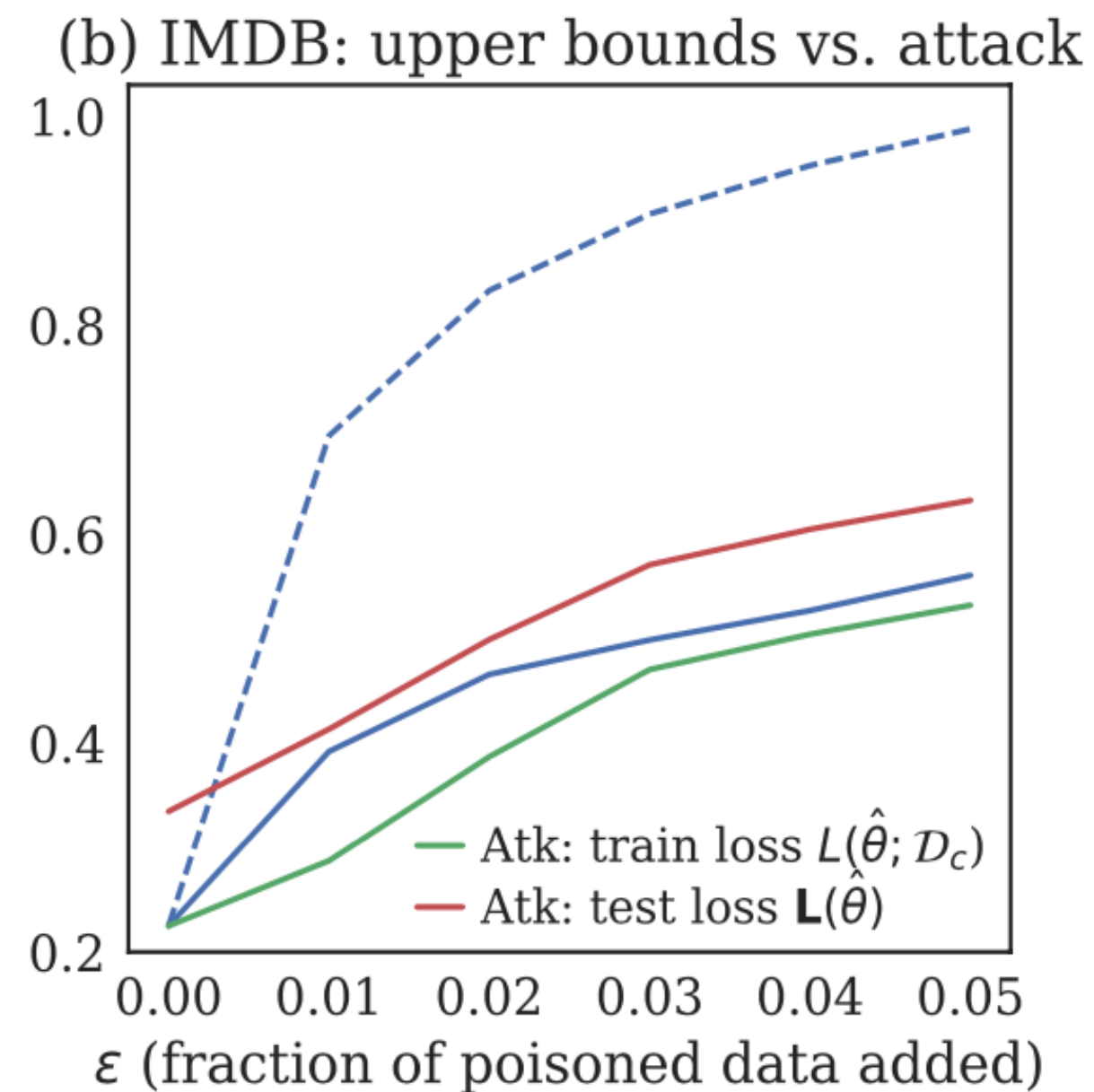
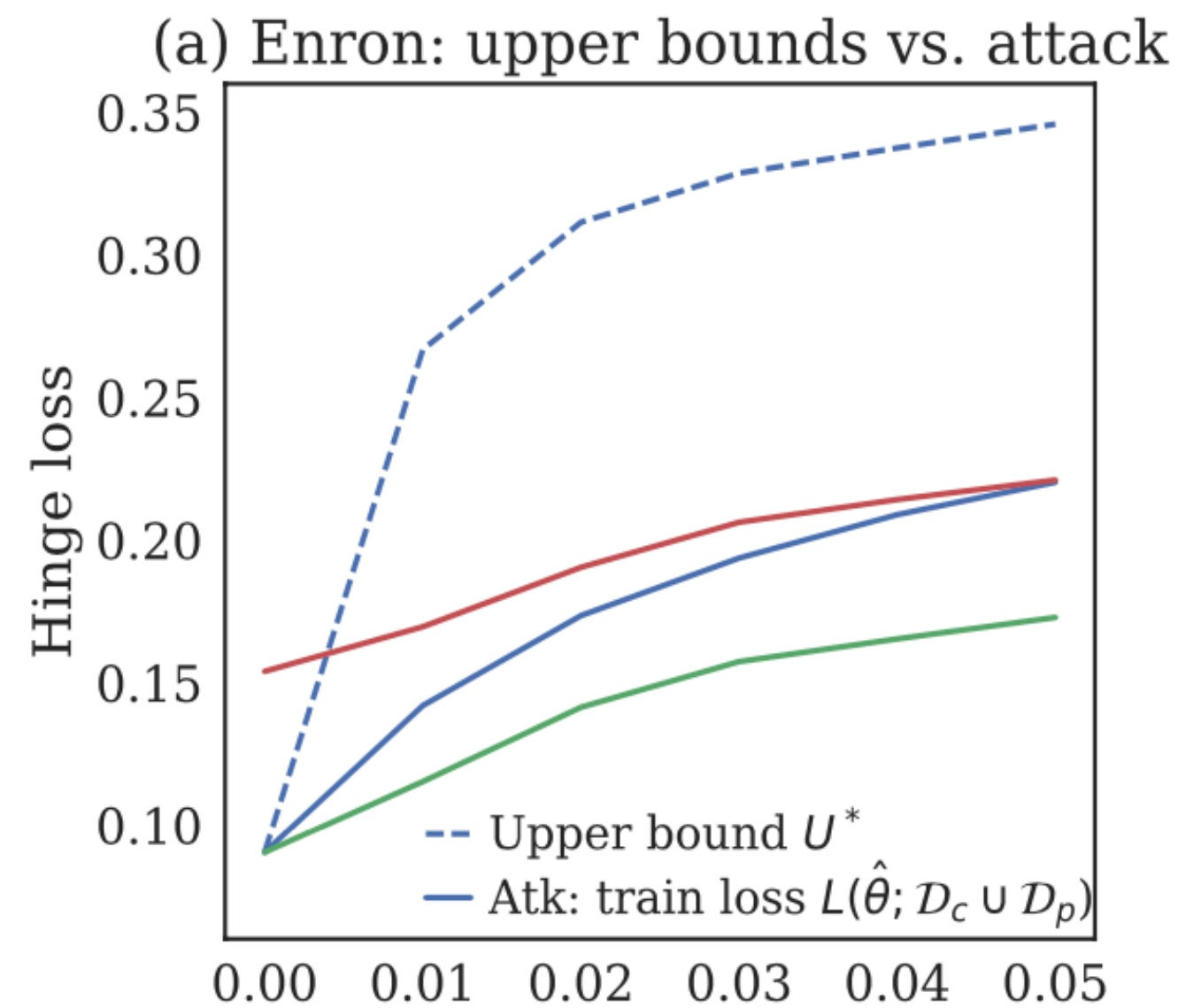


- Combined sphere and slab data sanitation
- Attack performs better than existing poisoning attacks
- Upper bound on loss is low for this data set



# Experiments

## For Fixed Defenses (Text Data)

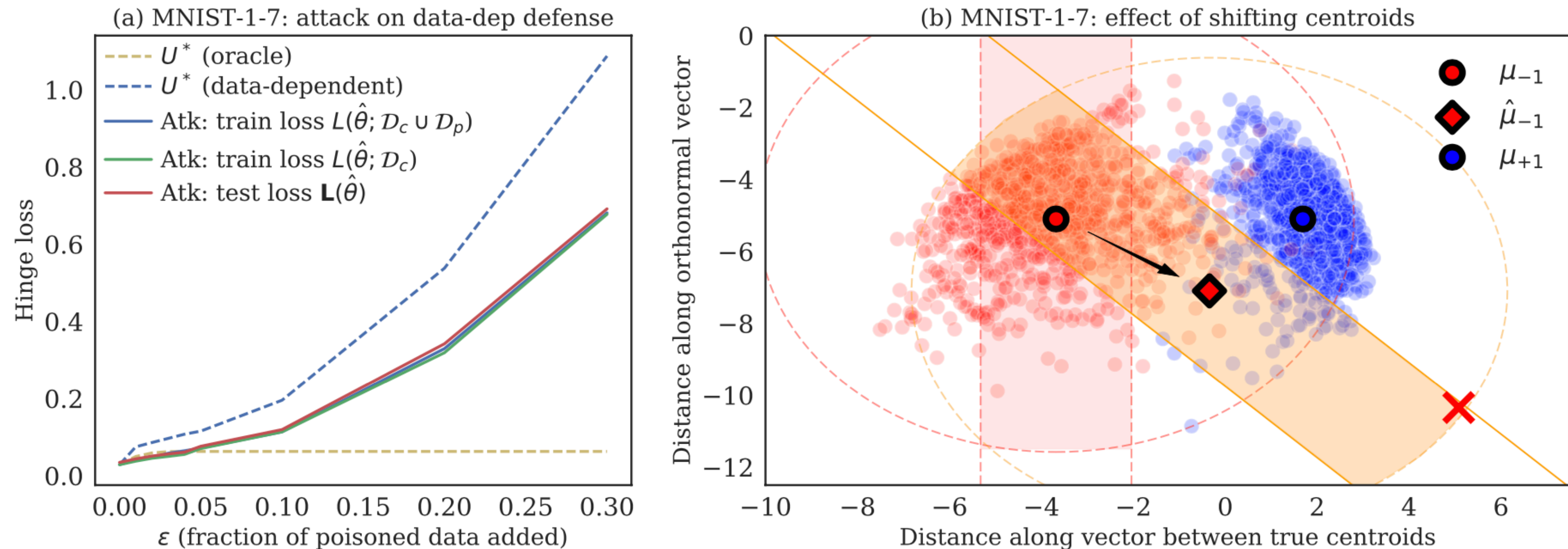


- Combined sphere, slab, and integrity constraints data sanitation
- Does not perform closely to optimal
- Attack performs better — dataset is more vulnerable to attack



# Experiments

## For Data-Dependent Defenses



- Combined sphere and slab data sanitization
- Data-dependent defense much more vulnerable to attack

# Limitations

- Requires access to **clean training data**
  - Therefore, “certification” does not apply to deployed systems
- When theoretical assumptions are not met, “**certification**” is not against *any* attack, only an empirically good attack
- Losses must be **convex**
  - Otherwise, attacker could force defender to find bad local minimum
  - Experimentation limited to binary SVM
- Limitations on feasible set
  - $\operatorname{argmax}_{(x,y) \in F} \ell \left( \theta^{(t-1)}; x, y \right)$  **must be tractable**
  - Experimentation limited to sphere, slab, and integrity data sanitation methods.

# Neural Cleanse - Wang et al.

---

Identifying and Mitigating Backdoor Attacks in Neural Networks

Presented by Apra Gupta

# Problem Statement:

First robust and generalizable detection and mitigation system for DNN backdoor attacks

Given a *trained DNN*, what are some techniques to:

- identify if there is an input *trigger* that would produce misclassification when added to an input
- figure out what that trigger look like
- *mitigate* and *remove* it from the model

---

Comment: In this context, *adversarial inputs* are inputs with the trigger added

# Key Contributions

1. Novel and Generalizable Techniques for
  - a. detecting
  - b. reconstructing backdoor triggers
2. Mitigation Techniques:
  - a. input filters
  - b. neuron pruning (model patching)
  - c. unlearning (model patching)
3. Advanced variants of backdoor attacks
  - a. evaluate mitigation strategies
4. Performance optimizations on backdoor detection algorithm

# Background/Assumptions

—

# Backdoor

- ★ *hidden pattern* trained into a DNN
- ★ produces *unexpected behavior* if and only if a specific *trigger* is added to an input
- ★ does not affect the model's normal behavior on clean inputs
- ★ a backdoor *misclassifies* arbitrary inputs into the same *specific target label*, when the associated trigger is applied to inputs.
- ★ Inputs samples that should be classified into any other label could be “*overridden*” by the presence of the trigger.
- ★ adding the *same backdoor trigger* causes arbitrary samples from different labels to be misclassified into the target label.

# Threat Model

- Adversarial Goal:
  - targeted misclassification of *backdoored* inputs at test time
- Adversarial Capabilities:
  - modify training data/procedure
- Adversarial Knowledge:
  - white box



# Defense Assumptions

- ★ access to *trained DNN model (architecture + parameters)*
- ★ access to *clean, correctly labelled samples*
- ★ access to computational resources to test or modify DNNs (GPU etc-)

# Defense Goals

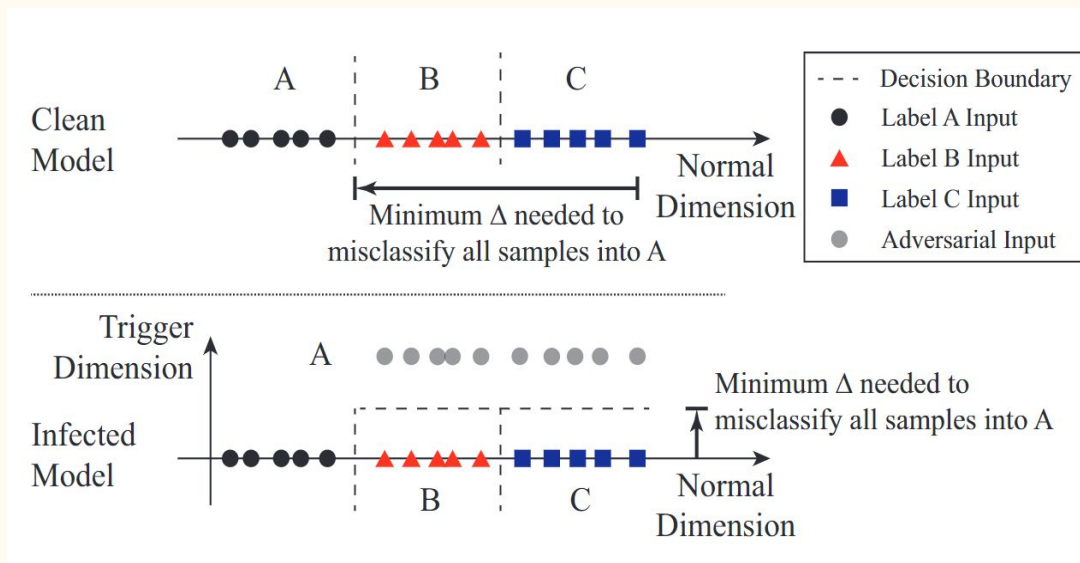
- Detecting backdoor
  - binary decision whether a given DNN has been infected by a backdoor
  - if so, what label is the backdoor targeting?
- Identifying backdoor
  - what is the backdoor trigger ?
- Mitigating backdoor
  - *proactive filter* to detect and block *adversarial inputs*
  - *patch* DNN to remove backdoor without affecting classification performance

# Defense Overview



# Key Intuition

- **Property of backdoors:** produce a classification to a target label A regardless of original label
- Trigger effectively produces another 'dimension' in regions belonging to non-target labels
- Any input containing the trigger has a higher value in the 'trigger dimension'
  - create "shortcuts" from within regions belonging to some label into the region belonging to the target label: shortening distance to target label.
- Detect these shortcuts by measuring the minimum amount of perturbation necessary to change *all inputs* from each region to the target region: **perturbation should be unusually small for the target label**



# Observations:

If a backdoor trigger ( $T_t$ ) exists:

1. The minimum perturbation needed to transform all inputs whose true label is  $L_i$  to be classified as  $L_t$  is bounded by the size of the trigger. So, for a fully backdoored model (any label's inputs can be backdoored):  $\delta_{\forall \rightarrow t} \leq |T_t|$
2. Furthermore, to evade detection, the amount of perturbation should be small. Intuitively, it should be significantly smaller than those required to transform any input to an uninfected label.  $\delta_{\forall i \rightarrow t} \leq |T_t| \ll \min_{i, i \neq t} \delta_{\forall \rightarrow i}$

*Thus we can detect a trigger  $T_t$  by detecting an abnormally low value of  $\delta_{\forall \rightarrow t}$  among all the output labels.*

# Backdoor Detection Algorithm

1. For each label:

    use **optimization scheme** to find ‘minimal’ trigger required to misclassify all samples from other labels into this target label.

2. Measure the size of minimum trigger for each label ( $L_0$  distance) and run **outlier detection** to see if any trigger candidate is significantly smaller than others:

3. If such a trigger exists: that label is the target label and that trigger is the **reverse-engineered** trigger

# Optimization Scheme

**Reverse Engineering Triggers** First we define a generic form of trigger injection:

$$\begin{aligned} A(\mathbf{x}, \mathbf{m}, \Delta) &= \mathbf{x}' \\ \mathbf{x}'_{i,j,c} &= (1 - m_{i,j}) \cdot \mathbf{x}_{i,j,c} + m_{i,j} \cdot \Delta_{i,j,c} \end{aligned} \quad (2)$$

$$\begin{aligned} \min_{\mathbf{m}, \Delta} \quad & \ell(y_t, f(A(\mathbf{x}, \mathbf{m}, \Delta))) + \lambda \cdot |\mathbf{m}| \\ \text{for } & \mathbf{x} \in X \end{aligned}$$

- Dual minimization
- Solved using Adam Optimizer
- $X$  is training data in experiments but can be replaced by some clean labelled data
- $\lambda \Rightarrow$  trigger size vs misclassification rate trade-off

# Outlier Detection

## Mean Absolute Deviation:

(resilient in the presence of multiple outliers)

1. Calculate the absolute deviation between all data points ( $L_0$  norm of minimum trigger of each input of a class) and the median
2. Calculate median of (1.) =  $\text{MAD}$  provides reliable measure of dispersion of distribution
3. Anomaly index = absolute deviation/MAD (normalized with constant estimator 1.4826)
4. **Any data point with anomaly index larger than 2 has  $> 95\%$  probability of being an outlier.**



# Optimized Backdoor Detection

*“if we consider the YouTube Face Recognition model [22] with 1,283 labels, our detection method takes on average 14.6 seconds for each label, with a total cost of 5.2 hours on an Nvidia Titan X GPU”*

**Observation:** After the first 10 iterations, the set overlap (of top 100 likely target labels) is mostly stable

**Modified Algorithm:** choose the top 100 labels after a few iterations (10), then terminate after overlap with previous iteration is larger than 50

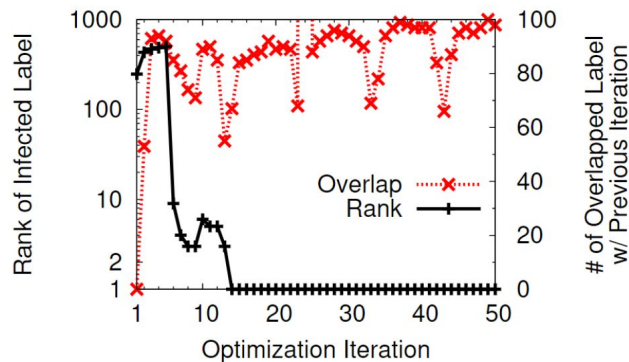


Fig. 5. Rank of infected labels in each iteration based on trigger's norm. Ranking consistency measured by # of overlapped label between iterations.

# Experimental Evaluation

—

# Experiments

TABLE I. Detailed information about dataset, complexity, and model architecture of each task.

Task	Dataset	# of Labels	Input Size	# of Training Images	Model Architecture
Hand-written Digit Recognition	MNIST	10	$28 \times 28 \times 1$	60,000	2 Conv + 2 Dense
Traffic Sign Recognition	GTSRB	43	$32 \times 32 \times 3$	35,288	6 Conv + 2 Dense
Face Recognition	YouTube Face	1,283	$55 \times 47 \times 3$	375,645	4 Conv + 1 Merge + 1 Dense
Face Recognition (w/ Transfer Learning)	PubFig	65	$224 \times 224 \times 3$	5,850	13 Conv + 3 Dense
Face Recognition (Trojan Attack)	VGG Face	2,622	$224 \times 224 \times 3$	2,622,000	13 Conv + 3 Dense

# Results

TABLE II. Attack success rate and classification accuracy of backdoor injection attack on four classification tasks.

Task	Infected Model		Clean Model Classification Accuracy
	Attack Success Rate	Classification Accuracy	
Hand-written Digit Recognition (MNIST)	99.90%	98.54%	98.88%
Traffic Sign Recognition (GTSRB)	97.40%	96.51%	96.83%
Face Recognition (YouTube Face)	97.20%	97.50%	98.14%
Face Recognition w/ Transfer Learning (PubFig)	97.03%	95.69%	98.31%

# Outlier Detection

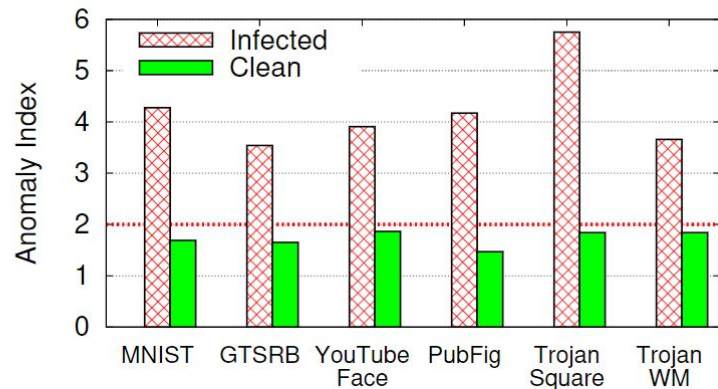


Fig. 3. Anomaly measurement of infected and clean model by how much the label with smallest trigger deviates from the remaining labels.

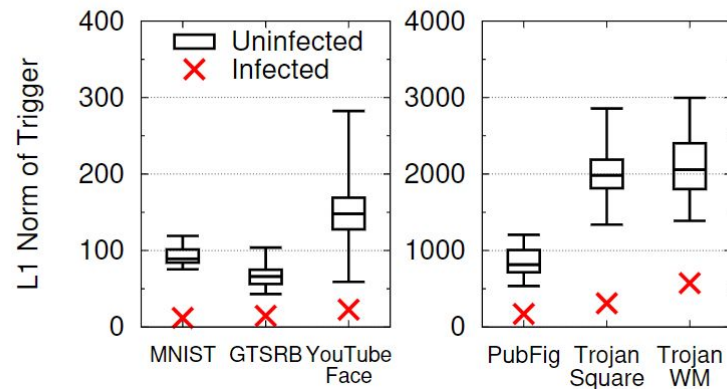


Fig. 4.  $L1$  norm of triggers for infected and uninfected labels in backdoored models. Box plot shows min/max and quartiles.

# Reverse-Engineered Triggers

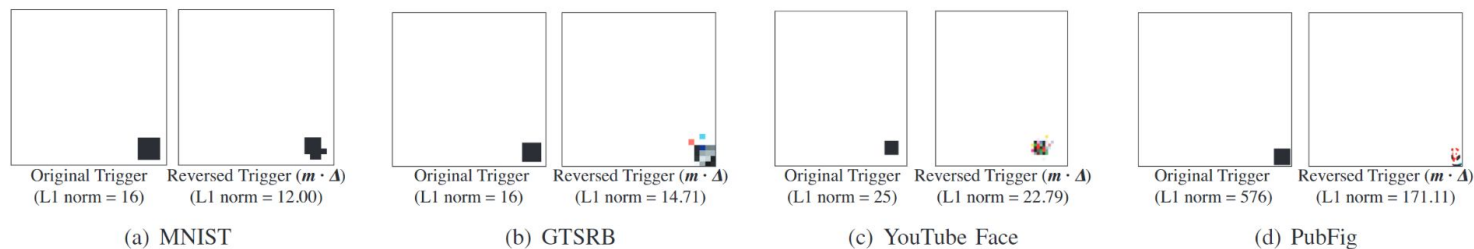


Fig. 6. Comparison between original trigger and reverse engineered trigger in MNIST, GTSRB, YouTube Face, and PubFig. Reverse engineered masks ( $m$ ) are very similar to triggers ( $m \cdot \Delta$ ), therefore omitted in this figure. Reported L1 norms are norms of masks. Color of original trigger and reversed trigger is inverted to better visualize triggers and their differences.

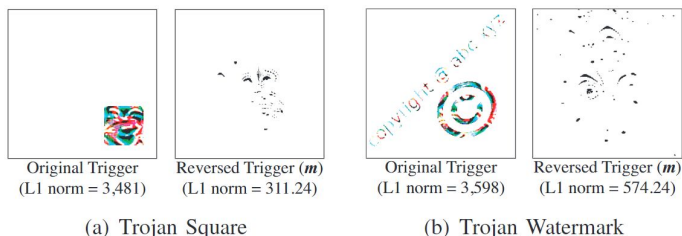


Fig. 7. Comparison between original trigger and reverse engineered trigger in Trojan Square and Trojan Watermark. Color of trigger is also inverted. Only mask ( $m$ ) is shown to better visualize the trigger.

- BadNets:
  - located and appear similar to original trigger
  - non-white pixels
- Trojan:
  - Reversed trigger appears in different locations of the image, and looks visually different
  - At Least 1 order of magnitude smaller than the original trigger

# Difference in Neuron Activations (clean vs trigger)

“Backdoor Neurons”: Top 1% neurons (second to last layer) with highest difference in activation when an adversarial vs benign test image is passed through poisoned network

TABLE III. Average activation of backdoor neurons of clean images and adversarial images stamped with reversed trigger and original trigger.

Model	Average Neuron Activation		
	Clean Images	Adv. Images w/ Reversed Trigger	Adv. Images w/ Original Trigger
MNIST	1.19	4.20	4.74
GTSRB	42.86	270.11	304.05
YouTube Face	137.21	1003.56	1172.29
PubFig	5.38	19.28	25.88
Trojan Square	2.14	8.10	17.11
Trojan Watermark	1.20	6.93	13.97

# Similarity in Neuron Activations: reverse engineered vs original trigger

BadNets models have ratios higher than 0.58, which indicates the reversed trigger is very similar to the original trigger in neuron activation.

However, ratios in Trojan models are much smaller (0.104 and 0.117), which suggests that the reversed trigger shares less in common with the original trigger

TABLE V. Intersection-over-union ratio of backdoor neurons used by reversed trigger and original trigger.

Model	MNIST	GTSRB	YouTube Face	PubFig	Trojan Square	Trojan Watermark
Intersection over Union Ratio	0.807	0.892	0.583	0.775	0.104	0.117



# Mitigation Techniques

—

# 1. Adversarial Input Filter

Neuron activations capture similarity between original and reverse-engineered triggers  $\Rightarrow$  filter identifies potential adversarial inputs as those with activation profiles **(the average neuron activations of backdoor neurons in the second to last layer when run on reversed trigger)** higher than a certain threshold.

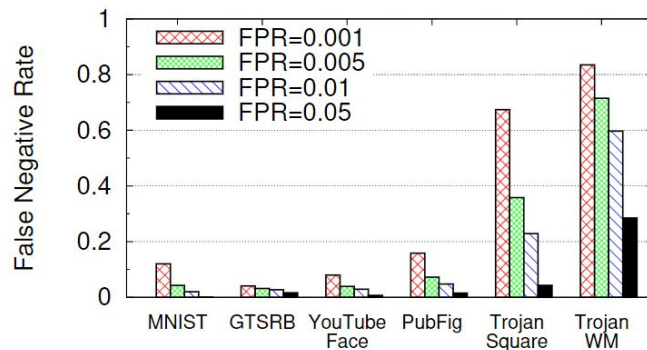


Fig. 8. False negative rate of proactive adversarial image detection when achieving different false positive rate.

## 2. Patching DNN via Neuron Pruning

- Prune out backdoor-related neurons from the DNN, i.e. set these neurons' output value to 0 during inference in the order of rank of activation difference.
- Stop pruning when the pruned model is no longer responsive to the reversed trigger.
- Prune second-last layer neurons except for Youtube Face (last convolutional layer)

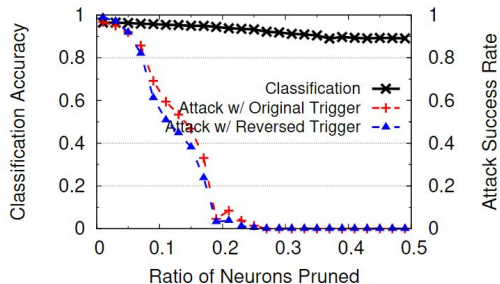


Fig. 9. Classification accuracy and attack success rate when pruning trigger-related neurons in GT-SRB (traffic sign recognition w/ 43 labels).

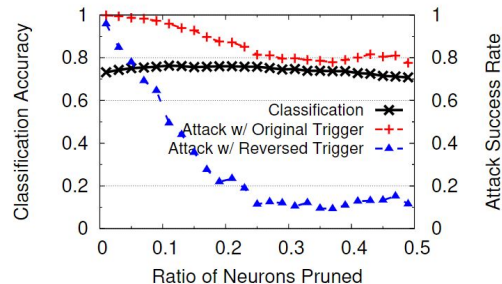
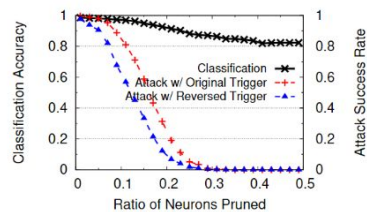
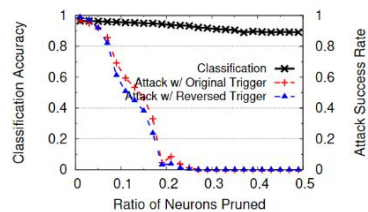


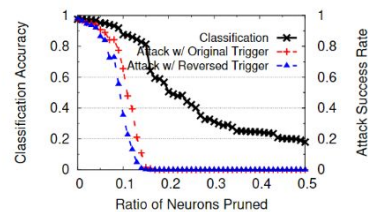
Fig. 10. Classification accuracy and attack success rate when pruning trigger-related neurons in Trojan Square (face recognition w/ 2,622 labels).



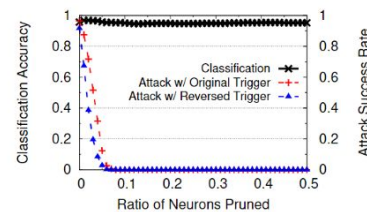
(a) MNIST



(b) GTSRB

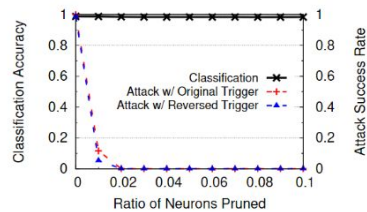


(c) YouTube Face

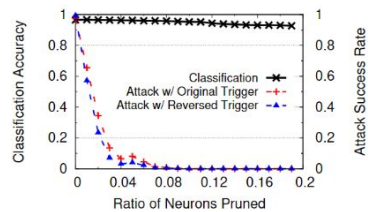


(d) PubFig

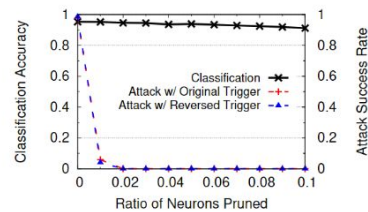
Fig. 21. Classification accuracy and attack success rate using original/reversed trigger when pruning backdoor-related neurons at the second to last layer.



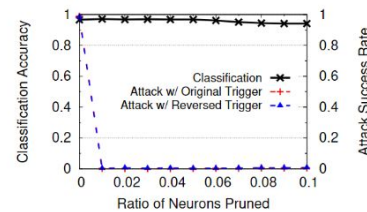
(a) MNIST



(b) GTSRB



(c) YouTube Face



(d) PubFig

Fig. 22. Classification accuracy and attack success rate of original/reversed trigger when pruning backdoor-related neurons at the last convolution layer.

### 3. Patching DNNs via Unlearning

- Fine-tune the model for only 1 epoch, using an updated training dataset: take a 10% sample of the original training data (clean, with no triggers) and add the reversed trigger to 20% of this sample without modifying labels.
- Decides which weights are problematic and should be updated
- Provides best mitigation performance

TABLE IV. Classification accuracy and attack success rate before and after unlearning backdoor. Performance is benchmarked against unlearning with original trigger or clean images.

Task	Before Patching		Patching w/ Reversed Trigger		Patching w/ Original Trigger		Patching w/ Clean Images	
	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate
MNIST	98.54%	99.90%	97.69%	0.57%	97.77%	0.29%	97.38%	93.37%
GTSRB	96.51%	97.40%	92.91%	0.14%	90.06%	0.19%	92.02%	95.69%
YouTube Face	97.50%	97.20%	97.90%	6.70%	97.90%	0.0%	97.80%	95.10%
PubFig	95.69%	97.03%	97.38%	6.09%	97.38%	1.41%	97.69%	93.30%
Trojan Square	70.80%	99.90%	79.20%	3.70%	79.60%	0.0%	79.50%	10.91%
Trojan Watermark	71.40%	97.60%	78.80%	0.00%	79.60%	0.00%	79.50%	0.00%

# Robustness of Defense Against Advanced Backdoors

1. Complex triggers (Fig 11):
  - a. noisy square instead of white square, varying trigger shapes
  - b. detection and mitigation works just as well
2. Larger Triggers (Fig 12,13)
  - a. As trigger size increases, anomaly detection becomes harder
  - b. but depends on number of labels: more labels permits larger trigger size to be detected
3. Multiple Infected Labels with Separate Triggers (Fig 14,15,16)
  - a. Outlier detection fails for high number of triggers (single trigger less of an outlier and harder to detect)
  - b. But underlying reverse engineering method still works: patching can still be applied

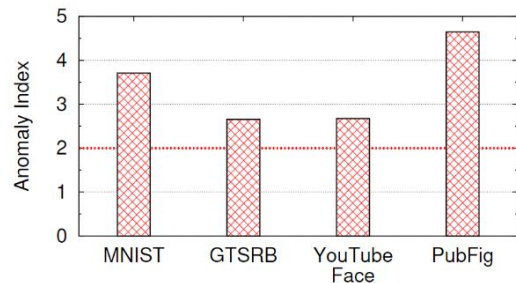


Fig. 11. Anomaly index of infected MNIST, GTSRB, YouTube Face, and PubFig model with noisy square trigger.

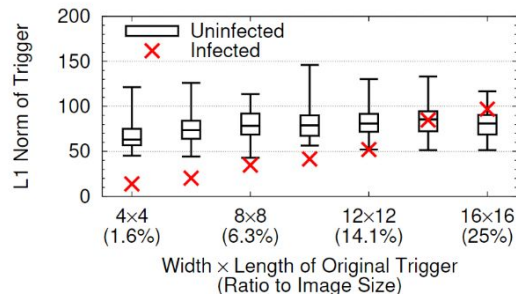


Fig. 12.  $L_1$  norm of reverse engineered triggers of labels when increasing the size of the original trigger in GTSRB (results of a single round).

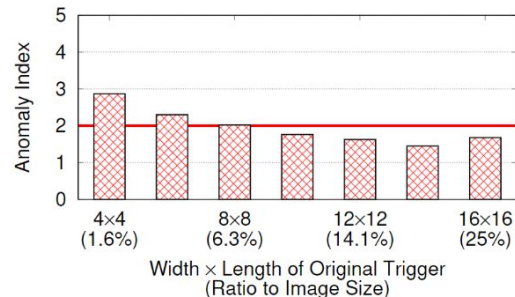


Fig. 13. Anomaly index of each infected GTSRB model when increasing the size of the original trigger (results averaged over 10 rounds).

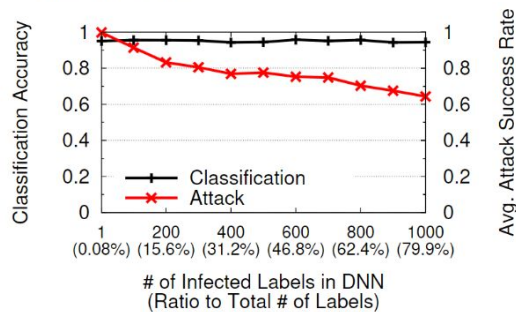


Fig. 14. Classification accuracy and average attack success rate when different number of labels are infected in YouTube Face.

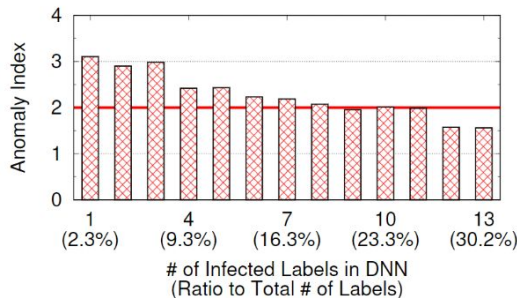


Fig. 15. Anomaly index of each infected GTSRB model with different number of labels being infected (results averaged over 10 rounds).

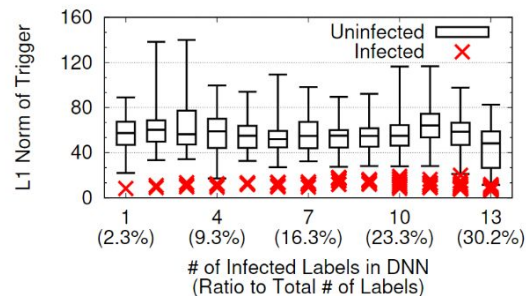


Fig. 16.  $L_1$  norm of triggers from infected and uninfected labels when different number of labels are infected in GTSRB (results of a single round).



# (Cont) Robustness Against Advanced Backdoors

## 3. Single Infected Label with Multiple Triggers

- a. triggers have same shape and color but located in different parts of the image
- b. a single run of our detection technique only identifies and patches one of the injected triggers
- c. running just 3 iterations of our detection and patch algorithm is able to successively reduce the success rate of all triggers to  $< 5\%$



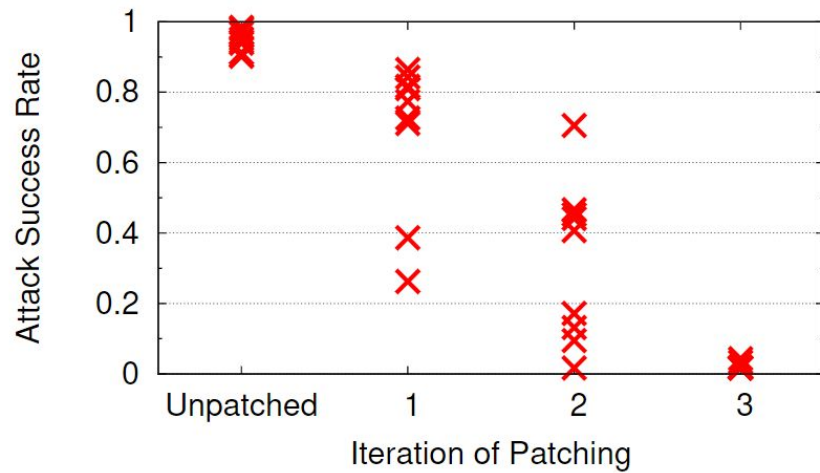


Fig. 17. Attack success rate of 9 triggers when patching DNN for different number of iterations.

# (Cont) Robustness Against Advanced Backdoors

## 4. Source-label-specific (Partial) Backdoors

- a. trigger misclassification only for a subset of source classes
- b. detector algorithm modified to analyze all possible source-target pairs
- c. updated techniques for detection and mitigation are all successful but have an obvious computational trade off

# Discussion: Alternatives Considered

1. Reject model all together after detecting backdoor:
  - difficult in practice because: resources and expertise required (e.g: uncommon task not supported by alternatives, lack of training data)
2. Searching for ‘signatures’ only present in backdoors
  - rely on strong causality between backdoor and chosen signal: lack of analytical results
  - scanning for triggers is hard as they can take on arbitrary shape and be designed to evade detection
3. Analyzing DNN internals to detect anomalies in intermediate states:
  - notoriously hard due to lack of interpretability of DNNs
  - detected patterns don't generalize across DNNs
4. Looking at Incorrect classification result: can be skewed towards infected label
  - backdoors can impact classification for normal inputs in unexpected ways
  - may not exhibit a consistent trend across DNNs
  - experiments show this approach consistently fails to detect backdoors

# Discussion Points

- Different Outlier Detection Methodology?
- False positives in backdoor detection: high for optimized algorithm
- Ways to bridge gap between original and reverse-engineered trigger?
- Effect of patching false positive labels:
  - $<1\%$  reduction in classification accuracy: potentially good to do since they represent a vulnerability ?
- Claim by authors: “unlearning performance is generally insensitive to parameters like amount of training data, and ratio of modified training data”
  - How?
- Why does unlearning prove to be the best mitigation strategy ?