CS 499/579: Trustworthy ML
Defense against adversarial attacks

Tu/Th 4:00 – 5:50 pm

Sanghyun Hong
sanghyun.hong@oregonstate.edu

Oregon State University

SAIL
Secure AI Systems Lab

Notes

- Call for actions
  - In-class presentation sign-ups
  - Checkpoint presentation I (on the 19th)
    - 15 min presentation + 3-5 min Q&A
    - Presentation MUST cover:
      - A research problem your team chose
      - A review of the prior work relevant to your problem
        - » How is your team's work different from the prior work?
        - » What's the paper your team picked and the results your team will reproduce?
      - Next steps (+ how each member will contribute to the work)

How can we defeat adversarial attacks?

How can we defeat adversarial attacks?

- Possible approaches
  - Reduce the information an adversary can access
    - Model architecture and/or model parameters
    - Model outputs (softmax probabilities)
    - … (more)
  - Detect and filter out adversarial examples
  - Remove adversarial perturbations (from inputs)
  - Make models resilient to adversarial attacks

Reduce the information an adversary can access

- Obfuscated gradients
  - Gradient masking (Papernot et al. 2017)
  - Hide "useful gradients" needed to generate adversarial examples

- PGD (Projected Gradient Descent)

$$\boxed{x^{t+1} = \Pi_{x+\mathcal{S}}\left(x^t + \alpha\,\mathrm{sgn}(\nabla_x L(\theta, x, y))\right).}$$

  - Multi-step adversary; much stronger than FGSM attack
  - Hyper-parameters
    - t: number of iterations
    - α: step-size
    - ε: perturbation bound $|x^* - x|_p$
  - Notation: PGD-t, bounded by ε, used the step-size of α

Reduce the information an adversary can access

- Defense approaches
  - Shattered gradients
  - Stochastic gradients
  - Vanishing and exploding gradients

- (vs. shattered gradients) BPDA
  - Make the approximation of non-differentiable layers used by defensive approaches

Reduce the information an adversary can access

- Defense approaches
  - Shattered gradients
  - Stochastic gradients
  - Vanishing and exploding gradients

- (vs. stochastic gradients) EOT
  - Compute gradients over the expected transformations

Reduce the information an adversary can access

- Defense approaches
  - Shattered gradients
  - Stochastic gradients
  - Vanishing and exploding gradients

- (vs. vanishing gradients) Reparameterization
  - Change-of-variables like C&W attacks

Reduce the information an adversary can access

- Defense approaches
  - Shattered gradients
  - Stochastic gradients
  - Vanishing and exploding gradients

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%*
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.005 (ℓ_2)	0%*
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%*
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%*
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	55%**
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%
Na et al. (2018)	CIFAR	0.015 (ℓ_∞)	15%

Can we "detect" adversarial perturbations?

Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks, Xu et al., NDSS 2018

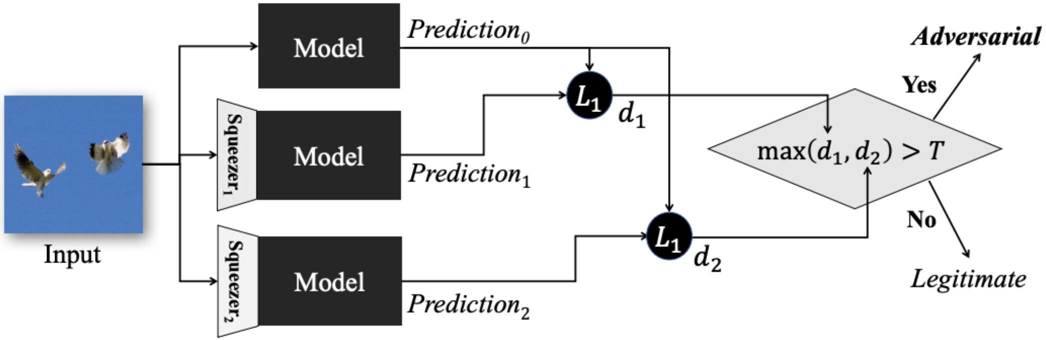
Motivation

- Information-theoretical Perspective
  - Compression!

Panda

The key idea: feature squeezing

- FeatureSqueezing



  - (Goal) To detect whether an input is adversarial example or not
  - (Idea) A model should return similar predictions over squeezed samples

Feature squeezing

- Research questions:
  - What are the **squeezers** a defender can choose?
  - How **effective** are they in defeating adversarial attacks?
  - How **effective** are they when **combined with existing defenses**?
  - How **effective** is feature-squeezing against **adaptive attacks**?

What are the squeezers a defender can choose?

- H-space
  - Reduce the color depth (8-bit: 0-255 to lower-bit widths)
  - Reduce the variation among pixels
    - Local smoothing (*e.g.*, median filter)
    - Non-local smoothing (*e.g.*, denoiser filters)
  - More
    - JPEG compression [Kurakin *et al.*]
    - Dimensionality reduction [Turk and Pentland]

Local smoothing

How effective are they in defeating adversarial attacks?

- Empirical approach (Baseline)
  - Setup
    - MNIST, CIFAR10, ImageNet
    - 7-layer CNN, DenseNet, and MobileNet
    - 100 images correctly classified by them
  - Attacks
    - FGSM, BIM, C&W, JSMA
    - L0, L2, and L-inf distances

Dataset	Norm	Configration: Attack	Configration: Mode	Cost (s)	Success Rate	Prediction Confidence	Distortion L_∞	Distortion L_2	Distortion L_0
MNIST	L_∞	FGSM		0.002	46%	93.89%	0.302	5.905	0.560
		BIM		0.01	91%	99.62%	0.302	4.758	0.513
		CW_∞	Next	51.2	100%	99.99%	0.251	4.091	0.491
			LL	50.0	100%	99.98%	0.278	4.620	0.506
	L_2	CW_2	Next	0.3	99%	99.23%	0.656	2.866	0.440
			LL	0.4	100%	99.99%	0.734	3.218	0.436
	L_0	CW_0	Next	68.8	100%	99.99%	0.996	4.538	0.047
			LL	74.5	100%	99.99%	0.996	5.106	0.060
		JSMA	Next	0.8	71%	74.52%	1.000	4.328	0.047
			LL	1.0	48%	74.80%	1.000	4.565	0.053
CIFAR-10	L_∞	FGSM		0.02	85%	84.85%	0.016	0.863	0.997
		BIM		0.2	92%	95.29%	0.008	0.368	0.993
		CW_∞	Next	225	100%	98.22%	0.012	0.446	0.990
			LL	225	100%	97.79%	0.014	0.527	0.995
	L_2	DeepFool		0.4	98%	73.45%	0.028	0.235	0.995
		CW_2	Next	10.4	100%	97.90%	0.034	0.288	0.768
			LL	12.0	100%	97.35%	0.042	0.358	0.855
	L_0	CW_0	Next	367	100%	98.19%	0.650	2.103	0.019
			LL	426	100%	97.60%	0.712	2.530	0.024
		JSMA	Next	8.4	100%	43.29%	0.896	4.954	0.079
			LL	13.6	98%	39.75%	0.904	5.488	0.098
ImageNet	L_∞	FGSM		0.02	99%	63.99%	0.008	3.009	0.994
		BIM		0.2	100%	99.71%	0.004	1.406	0.984
		CW_∞	Next	211	99%	90.33%	0.006	1.312	0.850
			LL	269	99%	81.42%	0.010	1.909	0.952
	L_2	DeepFool		60.2	89%	79.59%	0.027	0.726	0.984
		CW_2	Next	20.6	90%	76.25%	0.019	0.666	0.323
			LL	29.1	97%	76.03%	0.031	1.027	0.543
	L_0	CW_0	Next	608	100%	91.78%	0.898	6.825	0.003
			LL	979	100%	80.67%	0.920	9.082	0.005

How effective are they in defeating adversarial attacks?

- Empirical approach (Feature Squeezing)

Dataset	Squeezer		L_∞ Attacks				L_2 Attacks			L_0 Attacks				All Attacks	Legitimate
	Name	Parameters	FGSM	BIM	CW_∞		Deep-Fool	CW_2		CW_0		JSMA			
					Next	LL		Next	LL	Next	LL	Next	LL		
MNIST	None		54%	9%	0%	0%	-	0%	0%	0%	0%	27%	40%	13.00%	99.43%
	Bit Depth	1-bit	**92%**	**87%**	**100%**	**100%**	-	**83%**	**66%**	0%	0%	50%	49%	**62.70%**	99.33%
	Median Smoothing	2x2	61%	16%	70%	55%	-	51%	35%	39%	36%	62%	56%	48.10%	99.28%
		3x3	59%	14%	43%	46%	-	51%	53%	**67%**	**59%**	**82%**	**79%**	55.30%	98.95%
CIFAR-10	None		15%	8%	0%	0%	2%	0%	0%	0%	0%	0%	0%	2.27%	94.84%
	Bit Depth	5-bit	17%	13%	12%	19%	40%	40%	47%	0%	0%	21%	17%	20.55%	94.55%
		4-bit	21%	29%	69%	74%	72%	84%	84%	7%	10%	23%	20%	44.82%	93.11%
	Median Smoothing	2x2	**38%**	**56%**	**84%**	**86%**	**83%**	**87%**	83%	**88%**	**85%**	**84%**	**76%**	**77.27%**	89.29%
	Non-local Means	11-3-4	27%	46%	80%	84%	76%	84%	**88%**	11%	11%	44%	32%	53.00%	91.18%
ImageNet	None		1%	0%	0%	0%	11%	10%	3%	0%	0%	-	-	2.78%	69.70%
	Bit Depth	4-bit	5%	4%	66%	79%	44%	84%	82%	38%	67%	-	-	52.11%	68.00%
		5-bit	2%	0%	33%	60%	21%	68%	66%	7%	18%	-	-	30.56%	69.40%
	Median Smoothing	2x2	22%	28%	75%	81%	**72%**	81%	84%	**85%**	**85%**	-	-	**68.11%**	65.40%
		3x3	**33%**	**41%**	73%	76%	66%	77%	79%	81%	79%	-	-	67.22%	62.10%
	Non-local Means	11-3-4	10%	25%	**77%**	**82%**	57%	**87%**	**86%**	43%	47%	-	-	57.11%	65.40%

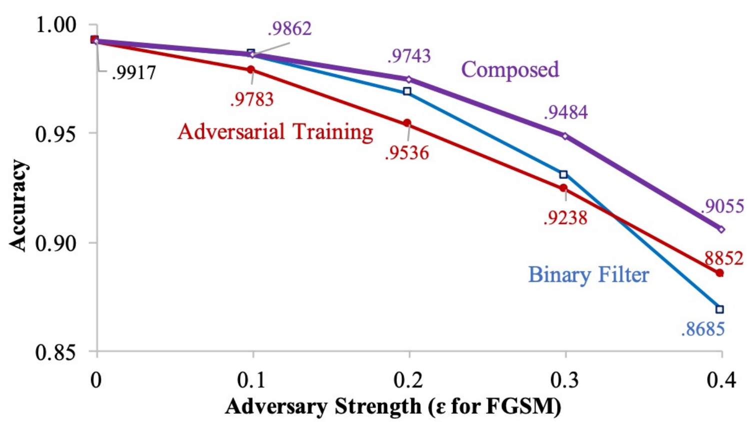
How effective are they in defeating adversarial attacks?

- Detection:
  - Metric (adv. or not):
    - Used with a single squeezer "score = $||f(x) - f(x^{squeezed})||_{l_1}$"
    - Used with multiple squeezer "score = $\max(score^{squeezer_1}, score^{squeezer_2}, \ldots)$"

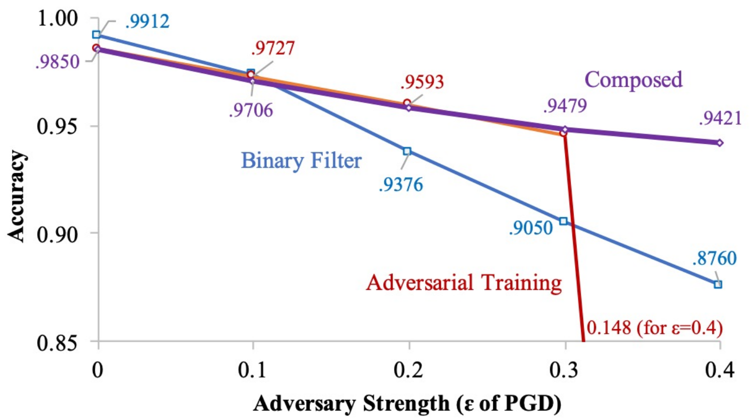
	Configuration			L_∞ Attacks				L_2 Attacks			L_0 Attacks				Overall Detection Rate
	Squeezer	Parameters	Threshold	FGSM	BIM	CW_∞		Deep Fool	CW_2		CW_0		JSMA		
						Next	LL		Next	LL	Next	LL	Next	LL	
CIFAR-10	Bit Depth	1-bit	1.9997	0.063	0.075	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000	0.000	0.013
		2-bit	1.9967	0.083	0.175	0.000	0.000	0.000	0.000	0.000	0.000	0.018	0.000	0.000	0.022
		3-bit	1.7822	0.125	0.250	0.755	0.977	0.170	0.787	0.939	0.365	0.214	0.000	0.000	0.409
		4-bit	0.7930	0.125	0.150	0.811	0.886	0.642	0.936	0.980	0.192	0.179	0.041	0.000	0.446
		5-bit	0.3301	0.000	0.050	0.377	0.636	0.509	0.809	0.878	0.096	0.018	0.041	0.038	0.309
	Median Smoothing	2x2	1.1296	0.188	**0.550**	**0.981**	**1.000**	0.717	0.979	**1.000**	**0.981**	**1.000**	**0.837**	**0.885**	0.836
		3x3	1.9431	0.042	0.250	0.660	0.932	0.038	0.681	0.918	0.750	0.929	0.041	0.077	0.486
	Non-local Mean	11-3-2	0.2770	0.125	0.400	0.830	0.955	0.717	0.915	0.939	0.077	0.054	0.265	0.154	0.484
		11-3-4	0.7537	0.167	0.525	0.868	0.977	0.679	0.936	**1.000**	0.250	0.232	0.245	0.269	0.551
		13-3-2	0.2910	0.125	0.375	0.849	0.977	0.717	0.915	0.939	0.077	0.054	0.286	0.173	0.490
		13-3-4	0.8290	0.167	0.525	0.887	0.977	0.642	0.936	**1.000**	0.269	0.232	0.224	0.250	0.547
	Best Attack-Specific Single Squeezer		-	0.188	**0.550**	**0.981**	**1.000**	0.717	0.979	**1.000**	**0.981**	**1.000**	**0.837**	**0.885**	-
	Best Joint Detection (5-bit, 2x2, 13-3-2)		1.1402	**0.208**	**0.550**	**0.981**	**1.000**	**0.774**	**1.000**	**1.000**	**0.981**	**1.000**	**0.837**	**0.885**	**0.845**

How effective is this when combined with other defenses?

- FeatureSqueezing + AT
  - Setup
    - MNIST
    - AT (with epsilon 0.3) + Use 2-bit for Pixels
    - Use FGSM and PGD attacks (epsilon 0.1 – 0.4)



(a) FGSM attacks.



(b) PGD attacks.

How effective is feature squeezing against adaptive attacks?

- (Adaptive) attack
  - Attackers who know this feature squeezing is deployed
  - Adaptive attack (using C&W + L2 or L-inf):
    - Reduce the prediction difference between x and x^{adv} under a threshold
    - Set the threshold is the one used by the detector
  - Result on MNIST:

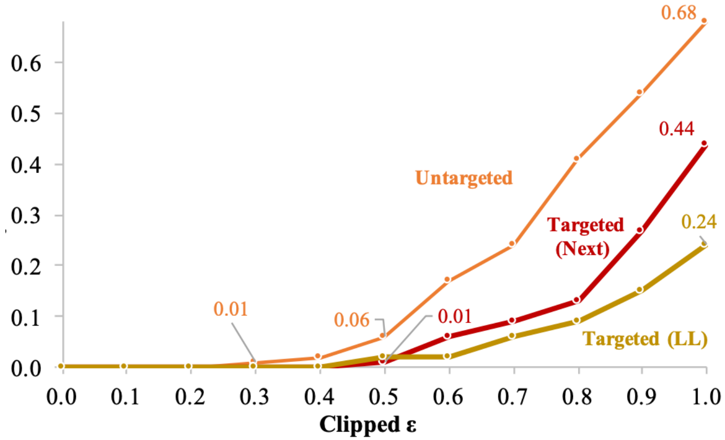


Fig. 7: Adaptive adversary success rates.

Summary

- Research questions
  - What are the **squeezers** a defender can choose?
    - Bit-width reduction
    - Smoothing (local or non-local)
  - How **effective** are they in defeating adversarial attacks?
    - Reduce the attack success rate by 87—100%
    - Detection rate is up to 100% when squeezers are jointly used
  - How **effective** are they when **combined with existing defenses**?
    - On MNIST, it improves the robustness over what AT can provides
  - How **effective** is feature-squeezing against **adaptive attacks**?
    - On MNIST, the attack success rate increases to 0-68%
    - One can choose a filter size randomly to defeat adaptive attacks (68% to 17%)

Can we make models “robust” to adversarial perturbations?

Toward deep learning models resistant to adversarial attacks, Madry et al., ICLR 2018

Revisiting the formulation

- Test-time (evasion) attack
  - Suppose
    - A test-time input (x, y)
    - $(x, y) \sim D$, D: data distribution; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
    - A NN model f and its parameters θ
    - $L(\theta, x, y)$: a loss function
  - Objective
    - Find an $x^{adv} = x + \delta$ such that $f(x^{adv}) \neq y$ while $||\delta||_p \leq \varepsilon$

Revisiting the formulation

- Test-time (evasion) attack
  - Suppose
    - A test-time input (x, y)
    - $(x, y) \sim D$, D: data distribution; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
    - A NN model f and its parameters θ
    - $L(\theta, x, y)$: a loss function
  - Attacker's objective
    - Find an $x^{adv} = x + \delta$ such that $\max_{\delta \in S} L(\theta, x^{adv}, y)$ while $||\delta||_p \leq \varepsilon$

Revisiting the formulation

- Test-time (evasion) attack
  - Suppose
    - A test-time input (x, y)
    - $(x, y) \sim D$, D: data distribution; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
    - A NN model f and its parameters θ
    - $L(\theta, x, y)$: a loss function
  - Attacker's objective
    - Find an $x^{adv} = x + \delta$ such that $\max_{\delta \in S} L\left(\theta, x^{adv}, y\right)$ while $||\delta||_p \leq \varepsilon$
  - Defender's objective
    - Train a neural network f robust to adversarial attacks
    - Find θ such that $\min_{\theta} \rho(\theta)$ where $\rho(\theta) = \mathrm{E}_{(x,y)\sim D}\left[L\left(\theta, x^{adv}, y\right)\right]$

Putting all together

- (Models resilient to) test-time (evasion) attack
  - Suppose
    - A test-time input (x, y)
    - $(x, y) \sim D$, D: data distribution; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
    - A NN model f and its parameters θ
    - $L(\theta, x, y)$: a loss function
  - Min-max optimization (between attacker's and defender's objectives)
    - Find $\min_{\theta} \rho(\theta)$ where $\rho(\theta) = \mathrm{E}_{(x,y)\sim D}\left[\max_{\delta \in S} L(\theta, x + \delta, y)\right]$ while $||\delta||_p \leq \varepsilon$
    - s: a set of test-time samples

Saddle point problem: inner maximization and outer minimization

Inner maximization

- PGD (Projected Gradient Descent)

$$\boxed{x^{t+1} = \Pi_{x+\mathcal{S}} \left(x^t + \alpha \operatorname{sgn}(\nabla_x L(\theta, x, y)) \right) .}$$

  - Multi-step adversary; much stronger than FGSM attack
  - Hyper-parameters
    - t: number of iterations
    - α: step-size
    - ε: perturbation bound $|x^* - x|_p$
  - Notation: PGD-t, bounded by ε, used the step-size of α

Outer minimization

- PGD (Projected Gradient Descent)

$$x^{t+1} = \Pi_{x+\mathcal{S}}\left(x^t + \alpha\, \mathrm{sgn}(\nabla_x L(\theta, x, y))\right).$$

  - Multi-step adversary; much stronger than FGSM attack

- Robust (adversarial) training
  - Make a model do correct prediction on adversarial examples
  - Training procedure
    - At each iteration of training
    - Craft PGD-t adversarial examples
    - Update the model towards making it correct on those adv examples

The intuition behind

- Robust training
  - Deep neural networks (DNNs) are universal function approximators[1]
  - DNNs may learn to be resistant to adversarial examples (a desirable function)
  - Adversarial training (AT):

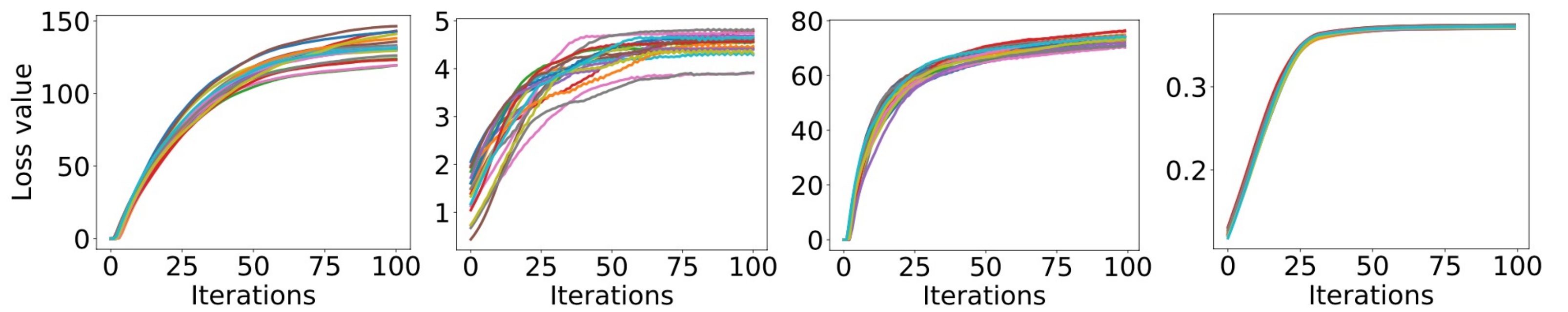
Repeat:

1. Select minibatch B, initialize gradient vector $g := 0$
2. For each (x, y) in B:
   a. Find an attack perturbation $\delta^\star$ by (approximately) optimizing
   $$\delta^\star = \underset{\|\delta\| \leq \epsilon}{\operatorname{argmax}}\, \ell(h_\theta(x+\delta), y)$$
   b. Add gradient at $\delta^\star$
   $$g := g + \nabla_\theta \ell(h_\theta(x+\delta^\star), y)$$
3. Update parameters θ
$$\theta := \theta - \frac{\alpha}{|B|} g$$

Hornik *et al.*, Multilayer feedforward networks are universal approximators, Neural Networks 1989
https://adversarial-ml-tutorial.org/adversarial_training/

Evaluation

- Findings
  - (1, 3) PGD increases the loss values in a fairly consistent way
  - (2, 4) Models trained with PGD attacks are resilient to the same attacks



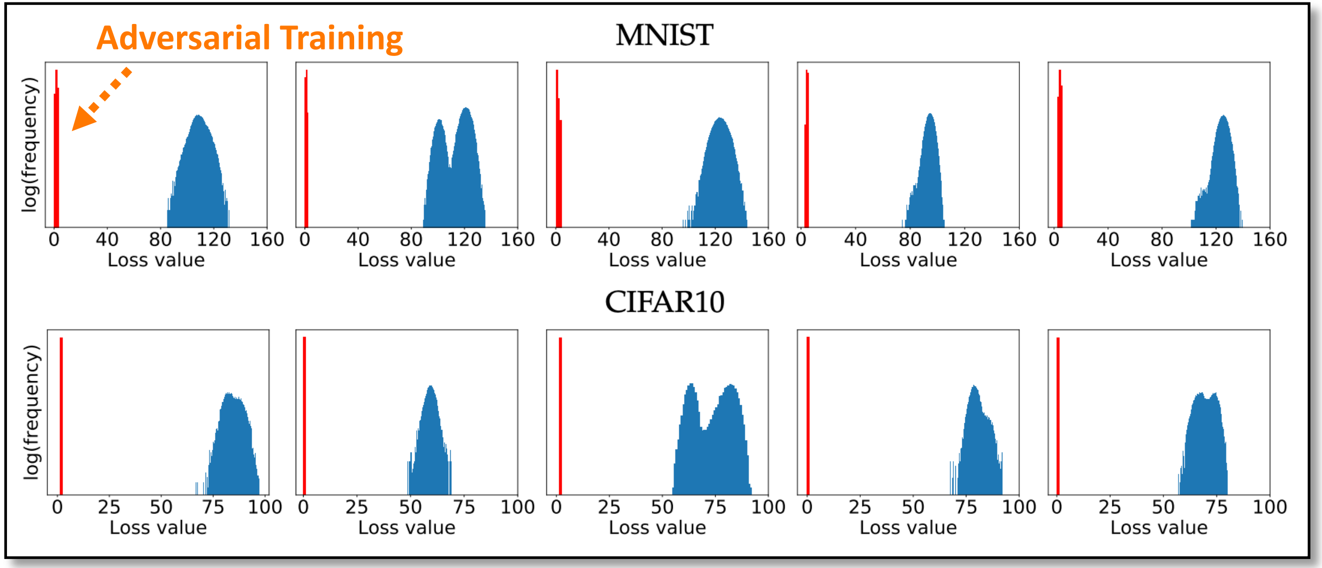
(a) MNIST (b) MNIST (c) CIFAR10 (d) CIFAR10

Adversarial Training

Adversarial Training

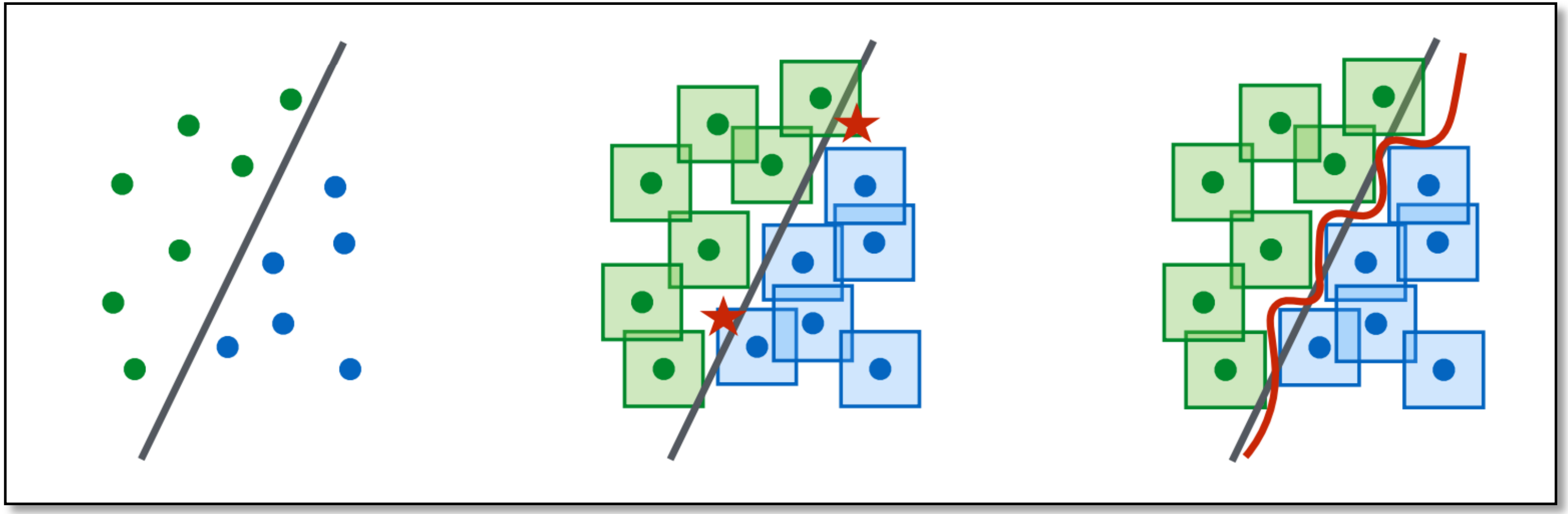
Evaluation

- Findings
  - PGD increases the loss values in a fairly consistent way
  - Models trained with PGD attacks are resilient to the same attacks
  - Final loss of PGD attacks are concentrated (both for defended/undefended models)



Evaluation

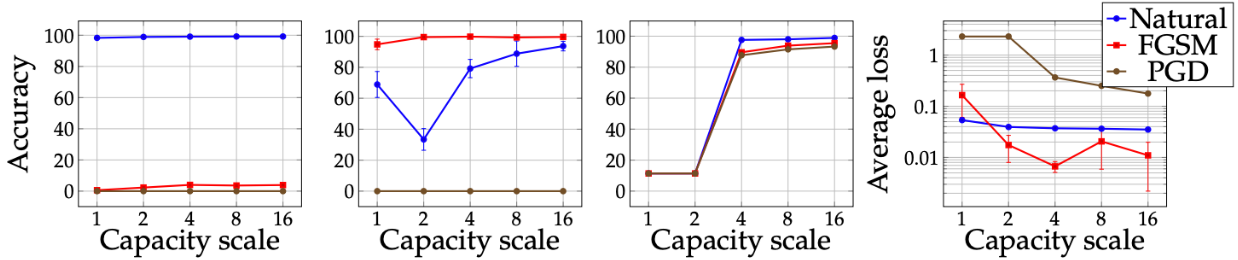
- Why adversarial training (AT) works?
  - Capacity is crucial for the robustness: robust models need complex decision boundary
  - Capacity alone helps: high-capacity models show more robustness w/o AT



Evaluation

- … Cont'd
  - Capacity is crucial for the robustness: robust models need complex decision boundary
  - Capacity alone helps: high-capacity models show more robustness w/o AT
  - AT with weak attacks (like FGSM) can't defeat a strong one like PGD
  - (optional) Robustness may be at odds with accuracy

MNIST



CIFAR10

	(a) Simple	(a) Wide	(b) Simple	(b) Wide	(c) Simple	(c) Wide	(d) Simple	(d) Wide
Natural	92.7%	95.2%	87.4%	90.3%	79.4%	87.3%	0.00357	0.00371
FGSM	27.5%	32.7%	90.9%	95.1%	51.7%	56.1%	0.0115	0.00557
PGD	0.8%	3.5%	0.0%	0.0%	43.7%	45.8%	1.11	0.0218

(a) Standard training (b) FGSM training (c) PGD training (d) Training Loss

Thank You!

Tu/Th 10:00 – 11:50 am

Sanghyun Hong

https://secure-ai.systems/courses/MLSec/W22

Oregon State University

SAIL
Secure AI Systems Lab