Notice

- Due dates
  - Written paper critiques (on 01.26)
  - Homework 2 (02.07 – 2 weeks)
  - Term Project Presentation 1 (on 1/31) **[Presentation order is in the Sign-up sheet]**

- Sign-up (on the Google Sheet)
  - Scribe Lecture Note
  - In-class Paper Presentation / Discussion

CS 499/599: Machine Learning Security

01.24: Adversarial Example 5

Mon/Wed 12:00 – 1:50 pm

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL

Secure AI Systems Lab

Recap

- ML Matters
- Evasion (Test-time Adversarial) Attack
  - Threat model
  - Attacks:
    - White-box:
      - FGSM / BIM
      - C&W / PGD attacks
    - Black-box:
      - Practicality
      - Transfer-based attacks
  - Mitigation:
    - Adversarial training (AT)

Topics for Today

- ML Matters
- Evasion (Test-time Adversarial) Attack
  - Threat model
  - Attacks:
    - White-box:
      - FGSM / BIM
      - C&W / PGD attacks
    - Black-box:
      - Practicality
      - Transfer-based attacks
      - Optimization-based attacks
  - Mitigation:
    - Adversarial training (AT)
    - Systematic defense (*e.g.*, FeatureSqueezing)

Prior Convictions: Black-Box Adversarial Attacks with Bandits and Priors

Andrew Ilyas, Logan Engstrom, and Alexander Madry

Motivation

- Black-box Attacks on Classifiers
  - Example scenario:
    - Upload (inappropriate) photos to your Instagram account
  - Challenges (vs. white-box attacks, like FGSM, BIM, PGD, …):
    - Gradient information is not available
  - Possible attack methods
    - Exploit transferability
    - Optimize your adversarial perturbations with query outputs

Motivation: PGD Revisit'd

- Suppose:
  - (x, y): a test-time sample; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
  - f: a neural network; θ: its parameters
  - $L(\theta, x, y)$: a loss function

- Goal (of the first order attacker):
  - Find an $x^{adv} = x + \delta$ such that $\max_{\delta \in S} L(\theta, x^{adv}, y)$ while $||\delta||_p \leq \varepsilon$

- PGD Crafts:

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

We Need to Know This!

Motivation: Gradient Estimation Problem

- Zeroth-order Optimization
  - Finite Difference Method (FDM):

$$D_v f(x) = \langle \nabla_x f(x), v \rangle \approx \left(f(x + \delta v) - f(x) \right) / \delta.$$

    - Compute: derivative of a function f at a point x towards a vector v

  - FDM for the gradient with d-components:

$$\widehat{\nabla}_x L(x, y) = \sum_{k=1}^{d} e_k \left(L(x + \delta e_k, y) - L(x, y) \right) / \delta \approx \sum_{k=1}^{d} e_k \langle \nabla_x L(x, y), e_k \rangle$$

- PGD in the black-box cases:

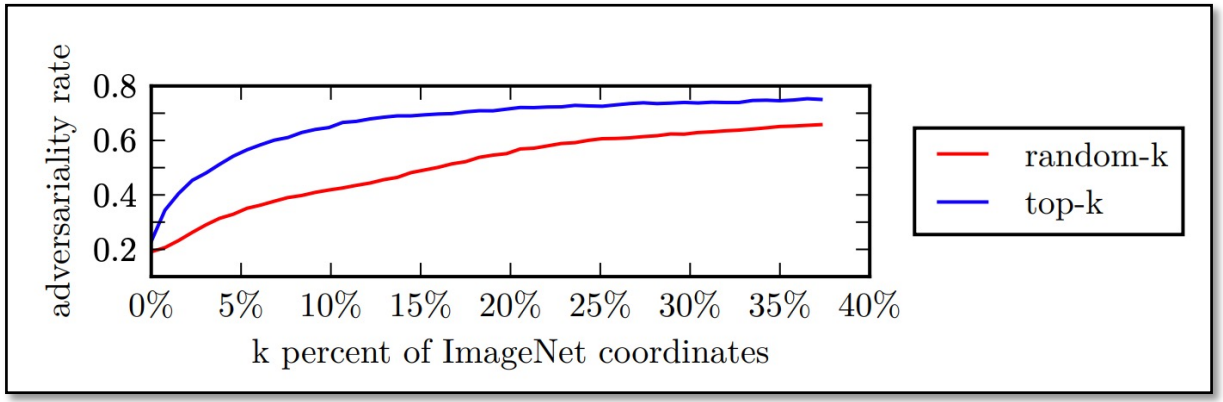
$$x^{t+1} = \Pi_{x+\mathcal{S}} \left(x^t + \alpha \operatorname{sgn}(\widehat{\nabla}_x L(\theta, x, y)) \right).$$

Motivation

- Research Questions
  - RQ 1: How **accurate** should we estimate a gradient for successful attacks?
  - RQ 2: How can we estimate gradient accurately with **smaller queries**?
  - RQ 3: (If we find a method) How **effective (and successful)** is this new method?

RQ 1: How Accurate Should We Estimate Gradients?

- Toy Experiment
  - Correctly-picked perturbations vs. Randomly-picked perturbations
  - PGD can be effective even with the imperfect gradient estimate



RQ 1: How Accurate Should We Estimate Gradients?

- Query-efficiency
  - The Least Squares Method: $\min_{\widehat{g}} \|\widehat{g}\|_2 \qquad \text{s.t. } A\widehat{g} = y.$
  - Iteratively compute the estimate $\hat{g}$, where:
    - A: Queries {1, 2, …}
    - y: the corresponding inner product values
  - Natural Evolution Strategy [Ilyas *et al.*] and Least Squares equivalence

$$\langle \hat{x}_{LSQ}, \boldsymbol{g} \rangle - \langle \hat{x}_{NES}, \boldsymbol{g} \rangle \leq O\left(\sqrt{\frac{k}{d} \cdot \log^3\left(\frac{k}{p}\right)}\right) ||g||^2$$

Lessons

- Research Questions
  - RQ 1: How **accurate** should we estimate a gradient for successful attacks?
    - PGD can be quite successful with imperfect gradient estimates
    - Query-efficiency is bounded by the prior work [Ilyas *et al.*] in practical scenarios
  - RQ 2: How can we estimate gradient accurately with **smaller queries**?
  - RQ 3: (If we find a method) How **effective (and successful)** is this new method?

RQ 2: How Can We Improve Query-efficiency for the Attacks?

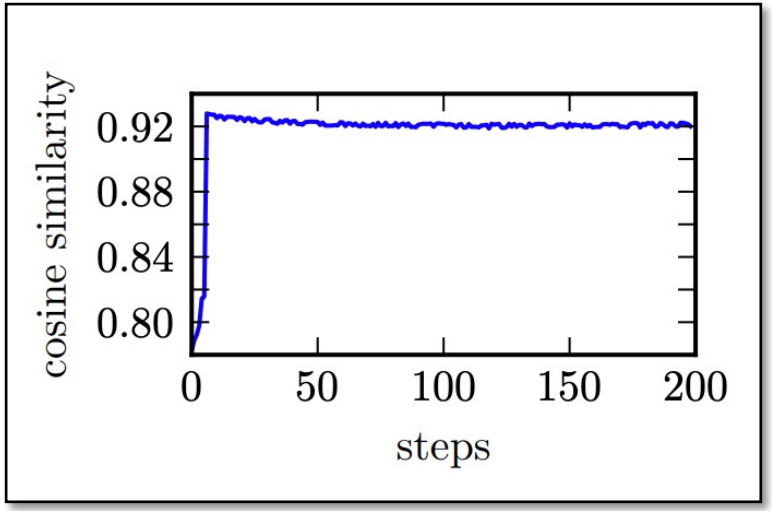
- Priors
  - Gradients are correlated in successive attack iterations
  - Pixels close to each other tend to have similar values

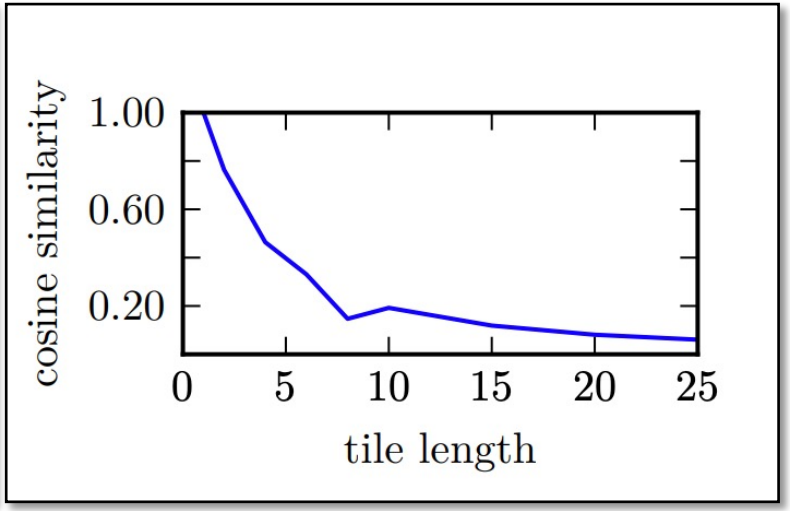
RQ 2: How Can We Improve Query-efficiency for the Attacks?

- Priors
  - **[Time-dependent]** Gradients are correlated in successive attack iterations
  - **[Data-dependent]** Pixels close to each other tend to have similar values

RQ 2: How Can We Improve Query-efficiency for the Attacks?

- Time-dependent & Data-dependent Priors





RQ 2: How Can We Improve Query-efficiency for the Attacks?

- Formulate the Problem to the Bandit Framework
  - **Bandit problem**

Algorithm 1 Gradient Estimation with Bandit Optimization

procedure BANDIT-OPT-LOSS-GRAD-EST(x, y_{init})
  $v_0 \leftarrow \mathcal{A}(\phi)$
  for each round $t = 1, \ldots, T$ **do**
    // Our loss in round t is $\ell_t(g_t) = -\langle \nabla_x L(x, y_{init}), g_t \rangle$
    $g_t \leftarrow v_{t-1}$
    $\Delta_t \leftarrow$ GRAD-EST(x, y_{init}, v_{t-1}) // Estimated Gradient of ℓ_t
    $v_t \leftarrow \mathcal{A}(v_{t-1}, \Delta_t)$
  $g \leftarrow v_T$
  return $\Pi_{\partial \mathcal{K}}[g]$

RQ 2: How Can We Improve Query-efficiency for the Attacks?

- Formulate the Problem to the Bandit Framework
  - Gradient Estimation

Algorithm 2 Single-query spherical estimate of $\nabla_v \langle \nabla L(x, y), v \rangle$

procedure GRAD-EST(x, y, v)
  $u \leftarrow \mathcal{N}(0, \frac{1}{d} I)$ // Query vector
  $\{q_1, q_2\} \leftarrow \{v + \delta \boldsymbol{u}, v - \delta \boldsymbol{u}\}$ // Antithetic samples
  $\ell_t(q_1) = -\langle \nabla L(x, y), q_1 \rangle \approx \frac{L(x,y) - L(x+\epsilon \cdot q_1, y)}{\epsilon}$ // Gradient estimation loss at q_1
  $\ell_t(q_2) = -\langle \nabla L(x, y), q_2 \rangle \approx \frac{L(x,y) - L(x+\epsilon \cdot q_2, y)}{\epsilon}$ // Gradient estimation loss at q_2
  $\boldsymbol{\Delta} \leftarrow \frac{\ell_t(q_1) - \ell_t(q_2)}{\delta} \boldsymbol{u} = \frac{L(x+\epsilon q_2, y) - L(x+\epsilon q_1, y)}{\delta \epsilon} \boldsymbol{u}$
  // Note that due to cancellations we can actually evaluate $\boldsymbol{\Delta}$ with only two queries to L
  return $\boldsymbol{\Delta}$

RQ 2: How Can We Improve Query-efficiency for the Attacks?

- Formulate the Problem to the Bandit Framework
  - Gradient Estimation

Algorithm 3 Adversarial Example Generation with Bandit Optimization for ℓ_2 norm perturbations

1: **procedure** ADVERSARIAL-BANDIT-L2(x_{init}, y_{init})
2: // $C(\cdot)$ returns top class
3: $v_0 \leftarrow \mathbf{0}_{1\times d}$ // If data prior, $d < \dim(x)$; v_t (Δ_t) up (down)-sampled before (after) line 8
4: $x_0 \leftarrow x_{init}$ // Adversarial image to be constructed
5: **while** $C(x) = y_{init}$ **do**
6: $g_t \leftarrow v_{t-1}$
7: $x_t \leftarrow x_{t-1} + h \cdot \frac{g_t}{||g_t||_2}$ // Boundary projection $\frac{g}{||g_t||}$ standard PGD: c.f. [Rig15]
8: $\Delta_t \leftarrow$ GRAD-EST($x_{t-1}, y_{init}, v_{t-1}$) // Estimated Gradient of ℓ_t
9: $v_t \leftarrow v_{t-1} + \eta \cdot \Delta_t$
10: $t \leftarrow t + 1$
return x_{t-1}

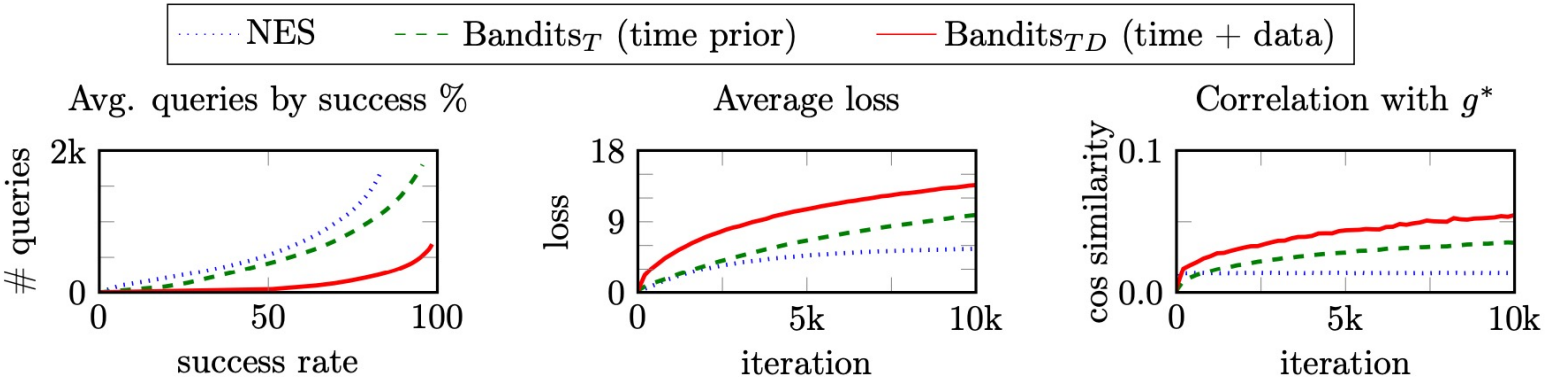
Lessons

- Research Questions
  - RQ 1: How **accurate** should we estimate a gradient for successful attacks?
    - PGD can be quite successful with imperfect gradient estimates
    - Query-efficiency is bounded by the prior work [Ilyas *et al.*] in practical scenarios
  - RQ 2: How can we estimate gradient accurately with **smaller queries**?
    - Use two priors: time- and data-dependent priors
    - Formulate the estimation into the bandit framework
  - RQ 3: (If we find a method) How **effective (and successful)** is this new method?

RQ 3: How Effective the New Attack?

- Setup
  - ImageNet (10k randomly chosen samples)
  - Inception-v3
  - Baseline: NES

- Results



Lessons

- Research Questions
  - RQ 1: How **accurate** should we estimate a gradient for successful attacks?
    - PGD can be quite successful with imperfect gradient estimates
    - Query-efficiency is bounded by the prior work [Ilyas *et al.*] in practical scenarios
  - RQ 2: How can we estimate gradient accurately with **smaller queries**?
    - Use two priors: time- and data-dependent priors
    - Formulate the estimation into the bandit framework
  - RQ 3: (If we find a method) How **effective (and successful)** is this new method?
    - Require 2.5 – 5x less queries for successful attacks compared to NES

Topics for Today

- ML Matters
- Evasion (Test-time Adversarial) Attack
  - Threat model
  - Attacks:
    - White-box:
      - FGSM / BIM
      - C&W / PGD attacks
    - Black-box:
      - Practicality
      - Transfer-based attacks
      - Optimization-based attacks
  - Mitigation:
    - Adversarial training (AT)
    - Systematic defense (*e.g.*, FeatureSqueezing)

Feature Squeezing: Detecting Adversarial Examples in DNNs

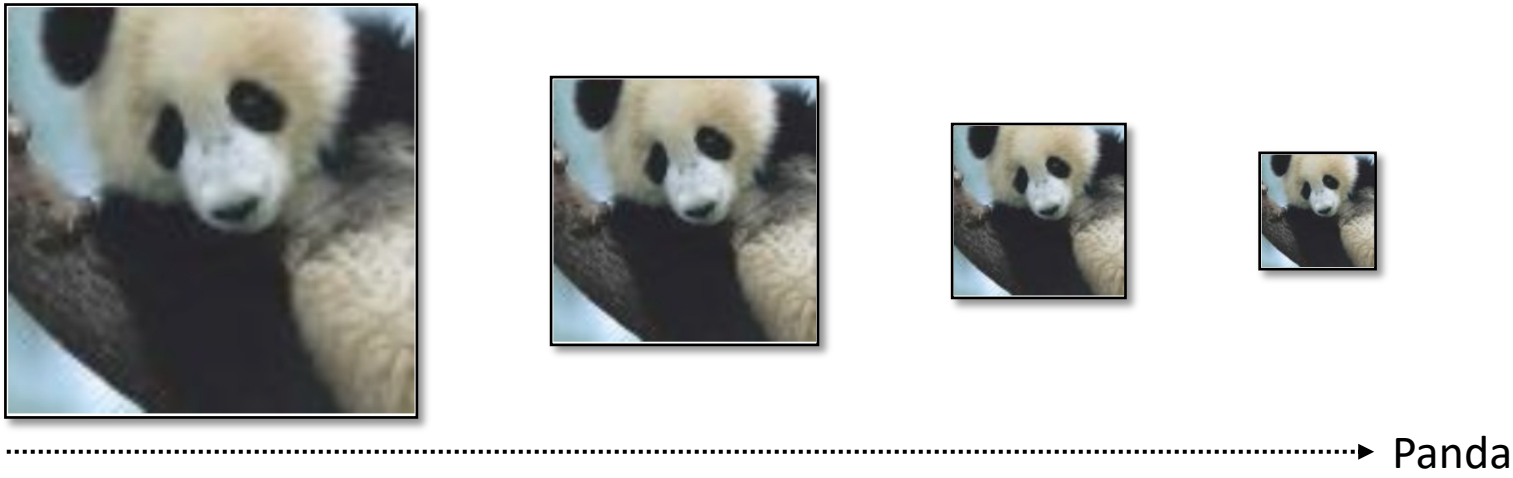
Weilin Xu, David Evans, and Yanjun Qi

Motivation

- Existing Defenses
  - **Make** robust models:
    - (Gradient masking) Defensive distillation
    - Adversarial training
    - …
  - **Detect** adversarial examples:
    - Sample statistics
    - Train a detector model
    - Prediction inconsistency (majority vote…)
    - …

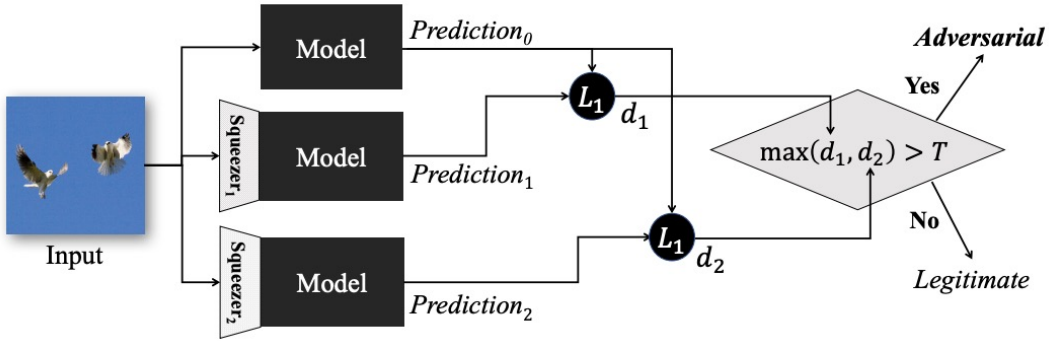
Motivation

- Information-theoretical Perspective



Key Idea

- FeatureSqueezing



- (Reasonable) Models should return similar predictions over squeezed samples
- Otherwise, it's an adversarial inputs

Motivation

- Research Questions
  - RQ 1: What are the **squeezers** available for a defender?
  - RQ 2: How much are they **effective** against existing adversarial attacks?
  - RQ 3: How much are they effective when **combined with existing defenses**?
  - RQ 4: How much is feature-squeezing effective under **adaptive** attacks?

Threat Model

- Attacker:
  - Goal: fool the victim classifier on a test-time input x
  - Capability: craft adversarial examples x^{adv} for x
  - Knowledge
    - White-box
    - Doesn't know whether feature squeezing is used
- Defender:
  - Detect whether the current input x is adversarial or not

RQ 1: What Are Squeezers Available for a Defender?

- Two simple techniques
  - Reduce the color depth (8-bit to a few bits)
  - Reduce the variation among pixels
    - Local smoothing (*e.g.*, median filter)
    - Non-local smoothing (*e.g.*, denoise)
  - Others
    - JPEG compression [Kurakin *et al.*]
    - Dimensionality reduction [Turk and Pentland]

RQ 2: How Much Are They Effective?

- Evaluation
  - Setup
    - MNIST, CIFAR10, and ImageNet
    - 7-layer CNN, DenseNet, and MobileNet
    - 100 images correctly classified by those r
  - Attacks
    - FGSM, BIM, C&W (Next | LL), JSMA
    - L0, L2, and L-inf distances

	Configration			Cost (s)	Success Rate	Prediction Confidence	Distortion		
		Attack	Mode				L_∞	L_2	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

RQ 2: How Much Are They Effective?

- Effectiveness of the Squeezers

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%

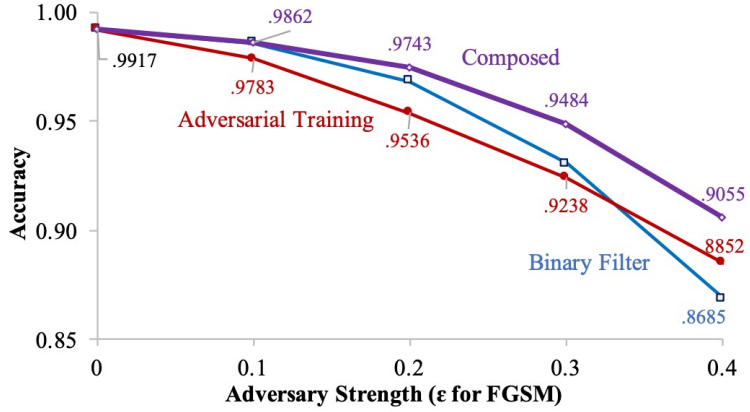
RQ 2: How Much Are They Effective?

- [

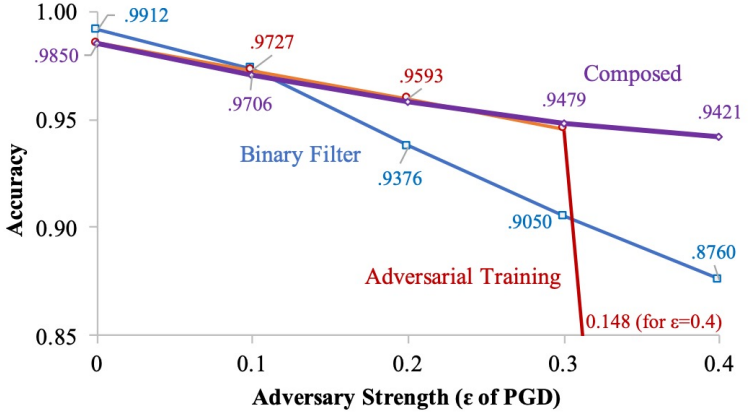
	Configuration			L_∞ Attacks				L_2 Attacks			L_0 Attacks				Overall Detection Rate
	Squeezer	Parameters	Threshold	FGSM	BIM	CW_∞		Deep Fool	CW_2		CW_0		JSMA		
						Next	LL		Next	LL	Next	LL	Next	LL	
CIFAR-10	Bit Depth	1-bit	1.9997	0.063	0.075	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000	0.000	0.013
		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**
ImageNet	Bit Depth	1-bit	1.9942	0.151	0.444	0.042	0.021	0.048	0.064	0.000	0.000	0.000	-	-	0.083
		2-bit	1.9512	0.132	0.511	0.500	0.354	0.286	0.170	0.306	0.218	0.191	-	-	0.293
		3-bit	1.4417	0.132	0.556	**0.979**	**1.000**	0.476	0.787	**1.000**	0.836	**1.000**	-	-	0.751
		4-bit	0.7996	0.038	0.089	0.813	**1.000**	0.381	0.915	**1.000**	0.727	**1.000**	-	-	0.664
		5-bit	0.3528	0.057	0.022	0.688	0.958	0.310	**0.957**	**1.000**	0.473	**1.000**	-	-	0.606
	Median Smoothing	2x2	1.1472	0.358	0.422	0.958	**1.000**	0.714	0.894	**1.000**	**0.982**	**1.000**	-	-	0.816
		3x3	1.6615	0.264	0.444	0.917	0.979	0.500	0.723	0.980	0.909	**1.000**	-	-	0.749
	Non-local Mean	11-3-2	0.7107	0.113	0.156	0.813	0.979	0.357	0.936	0.980	0.418	0.830	-	-	0.618
		11-3-4	1.0387	0.208	0.467	0.958	**1.000**	0.548	0.936	**1.000**	0.673	0.957	-	-	0.747
		13-3-2	0.7535	0.113	0.156	0.813	0.979	0.357	0.936	0.980	0.418	0.851	-	-	0.620
		13-3-4	1.0504	0.226	0.444	0.958	**1.000**	0.548	0.936	**1.000**	0.709	0.957	-	-	0.751
	Best Attack-Specific Single Squeezer		-	0.358	0.556	**0.979**	**1.000**	0.714	**0.957**	**1.000**	**0.982**	**1.000**	-	-	-
	Best Joint Detection (5-bit, 2x2, 11-3-4)		1.2128	**0.434**	**0.644**	**0.979**	**1.000**	**0.786**	0.915	**1.000**	**0.982**	**1.000**	-	-	**0.859**

RQ 3: When Combined with Adversarial Training (AT)

- Effectiveness of the Squeezers + 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.

RQ 4: Is Feature Squeezing Effective against Adaptive Attacks?

- Adaptive attacker
  - Difference: Know the feature squeezing is used
  - 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:

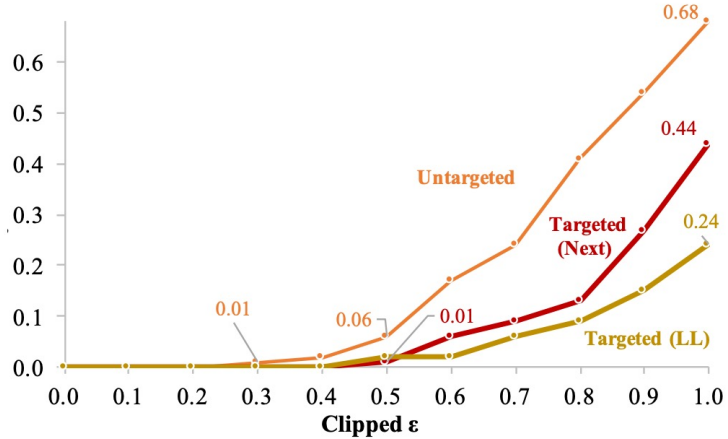


Fig. 7: Adaptive adversary success rates.

Motivation

- Research Questions
  - RQ 1: What are the **squeezers** available for a defender?
    - Bit-width reduction
    - Smoothing (local or non-local)
  - RQ 2: How much are they **effective** against existing adversarial attacks?
    - Reduce the attack success rate by 87—100%
    - Detection rate is up to 100% when squeezers are jointly used
  - RQ 3: How much are they effective when **combined with existing defenses**?
    - On MNIST, it improves the robustness over what AT can provides
  - RQ 4: How much is feature-squeezing effective under **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%)

Recap

- ML Matters
- Evasion (Test-time Adversarial) Attack
  - Threat model
  - Attacks:
    - White-box:
      - FGSM / BIM
      - C&W / PGD attacks
    - Black-box:
      - Practicality
      - Transfer-based attacks
      - Optimization-based attacks
  - Mitigation:
    - Adversarial training (AT)
    - Systematic defense (*e.g.*, FeatureSqueezing)

Thank You!

Mon/Wed 12:00 – 1:50 pm

Sanghyun Hong

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

Oregon State University

SAIL

Secure AI Systems Lab