Security and Privacy of ML

Empirical Defenses to Adversarial Examples

2/29/2024

Shang-Tse Chen

Department of Computer Science
& Information Engineering
National Taiwan University

Homework 1: Gray-box Attack

- Generate adversarial images on CIFAR-100
- Download the validation set of 500 images and submit the perturbed versions of them
- Allowed perturbation strength: $\|\delta\|_\infty = 8$, i.e., you can change each pixel up to 8 on the scale of [0, 255]

Homework 1: Gray-box Attack

5 standardly trained models from this Github repo will be used, possibly with some preprocessing steps

Table of implemented classification models

Some remarks:

- `Repo` is an author repository, if it exists.
- `a`, `b`, `c`, `d`, and `e` means the implementation of a model for ImageNet-1K, CIFAR-10, CIFAR-100, SVHN, and CUB-200-2011, respectively.
- `A`, `B`, `C`, `D`, and `E` means having a pre-trained model for corresponding datasets.

Model	Gluon	PyTorch	Chainer	Keras	TF	TF2	Paper	Repo
AlexNet	A	A	A	A	A	A	link	link
ZFNet	A	A	A	A	A	A	link	-
VGG	A	A	A	A	A	A	link	-
BN-VGG	A	A	A	A	A	A	link	-
BN-Inception	A	A	A	-	-	A	link	-
ResNet	ABCDE	ABCDE	ABCDE	A	A	ABCDE	link	link
PreResNet	ABCD	ABCD	ABCD	A	A	ABCD	link	link
ResNeXt	ABCD	ABCD	ABCD	A	A	ABCD	link	link
SENet	A	A	A	A	A	A	link	link
SE-ResNet	ABCDE	ABCDE	ABCDE	A	A	ABCDE	link	link

https://github.com/osmr/imgclsmob

Homework 1: Gray-box Attack

- You can use any packages
- Check out official tutorials on adversarial attacks with Tensorflow or PyTorch if you've never done it before
- Your submission will be evaluated on 5 models

Homework 1: Gray-box Attack

- General rule of thumb:
  - Your attack tends to be stronger / more transferable if you can simultaneously attack multiple models (i.e., ensemble)
  - Single step methods (e.g., FGM) usually more transferable than iterative methods (e.g., PGD)

Homework 1: Gray-box Attack

- You can get some ideas / insights from:
  - Past competitions (NIPS 2017, 2018)
  - CIFAR10 Adversarial Examples Challenge

Homework 1: Report

- Write a report with at most 4 pages in NeurIPS format
  - Methods you tried
  - Why you choose certain methods in your submission
  - Experiments that you did
  - Findings or insights you gained

Homework 1: Grading

- Grading Policy:
  - Accuracy : 5%
  - Report: 5%
    - Clarity
    - Experiments / Comparison of different approaches
    - Novelty

Student Group Presentation

- Enter your team members and topic preferences in this Google form by Friday
- Topics and dates are announced on the course website
- Submit your slides by **6 pm** before the presentation day

Review From Last Week

- Given a classifier C and an example x, find an adversarial example $\boldsymbol{x}'$, s.t. $d(\boldsymbol{x}', \boldsymbol{x}) \leq \epsilon$, and $C(\boldsymbol{x}') \neq C(\boldsymbol{x})$
- The distance function $d(\cdot,\cdot)$ is application dependent
  - For mathematical convenience, ℓ_p distance is often used

$$\|\boldsymbol{x} - \boldsymbol{x}'\|_p = \left(\sum_{i=1}^{N} |\boldsymbol{x}_i - \boldsymbol{x}_i{}'|^p \right)^{\frac{1}{p}}$$

$$\|\delta\|_1 = \sum_{i=1}^{N} |\delta_i| \qquad \|\delta\|_2 = \sqrt{\sum_{i=1}^{N} \delta_i^2} \qquad \|\delta\|_\infty = \max\{|\delta_i|: i = 1, \ldots, N\}$$

Training and Attack Are Dual Problems

- Training:

$$\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in S} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})$$

Gradient descent to update model weights $\boldsymbol{\theta}$

- Attack:

(untargeted) $$\max_{\boldsymbol{\delta}\in\boldsymbol{\Delta}} \ell(\boldsymbol{x}+\boldsymbol{\delta}, y; \boldsymbol{\theta})$$

Gradient descent to update input $\boldsymbol{x}$

(targeted) $$\max_{\boldsymbol{\delta}\in\boldsymbol{\Delta}} -\ell(\boldsymbol{x}+\boldsymbol{\delta}, y'; \boldsymbol{\theta})$$

Fast Gradient Method (L_2) [Goodfellow et al., 2014]

$$\ell(\boldsymbol{x} + \boldsymbol{\delta}, y; \boldsymbol{\theta}) \approx \ell(\boldsymbol{x}, y; \boldsymbol{\theta}) + \boldsymbol{\delta} \cdot \nabla_{\mathbf{x}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})$$

Maximize

$$\ell(\boldsymbol{x}, y; \boldsymbol{\theta}) + \boldsymbol{\delta} \cdot \nabla_{\mathbf{x}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})$$

subject to

$$\|\delta\|_2 \leq \epsilon$$

➔

$$\delta = \epsilon \cdot \frac{\nabla_x \ell(\boldsymbol{x}, y; \boldsymbol{\theta})}{\|\nabla_x \ell(\boldsymbol{x}, y; \boldsymbol{\theta})\|_2}$$

Fast Gradient Method (L_∞)

$$\ell(\boldsymbol{x} + \boldsymbol{\delta}, y; \boldsymbol{\theta}) \approx \ell(\boldsymbol{x}, y; \boldsymbol{\theta}) + \boldsymbol{\delta} \cdot \nabla_{\mathbf{x}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})$$

Maximize

$$\ell(\boldsymbol{x}, y; \boldsymbol{\theta}) + \boldsymbol{\delta} \cdot \nabla_{\mathbf{x}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})$$

subject to

$$\|\delta\|_\infty \leq \epsilon$$

➔

$$\delta = \epsilon \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})\right)$$

Also known as Fast Gradient Sign Method (FGSM)

Fast Gradient Method (L_1)

$$\ell(\boldsymbol{x} + \boldsymbol{\delta}, y; \boldsymbol{\theta}) \approx \ell(\boldsymbol{x}, y; \boldsymbol{\theta}) + \boldsymbol{\delta} \cdot \nabla_{\mathbf{x}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})$$

Maximize

$$\ell(\boldsymbol{x}, y; \boldsymbol{\theta}) + \boldsymbol{\delta} \cdot \nabla_{\mathbf{x}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})$$

subject to

$$\|\delta\|_1 \leq \epsilon$$

➔

$$\mathrm{i}^* = \mathrm{argmax}_{\mathrm{i}} \, |\nabla_{\mathbf{x}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})_i|$$

$$\delta_i = \begin{cases} \epsilon \cdot \mathrm{sign}(\nabla_{\mathbf{x}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})_{\boldsymbol{i}}), & \text{if } i = i^* \\ 0, & \text{otherwise} \end{cases}$$

Let’s move on to defenses

Threat Models

- White-box attacks
  - Attacker knows
    - Model architecture
    - Model weights
    - Pre-processing / Post-processing
- Black-box attacks
  - Attacker may or may not know
    - Algorithm (DNN, SVM, …)
    - Features
    - Model architecture
    - Model weights
    - …

Defenses in Black-box Settings

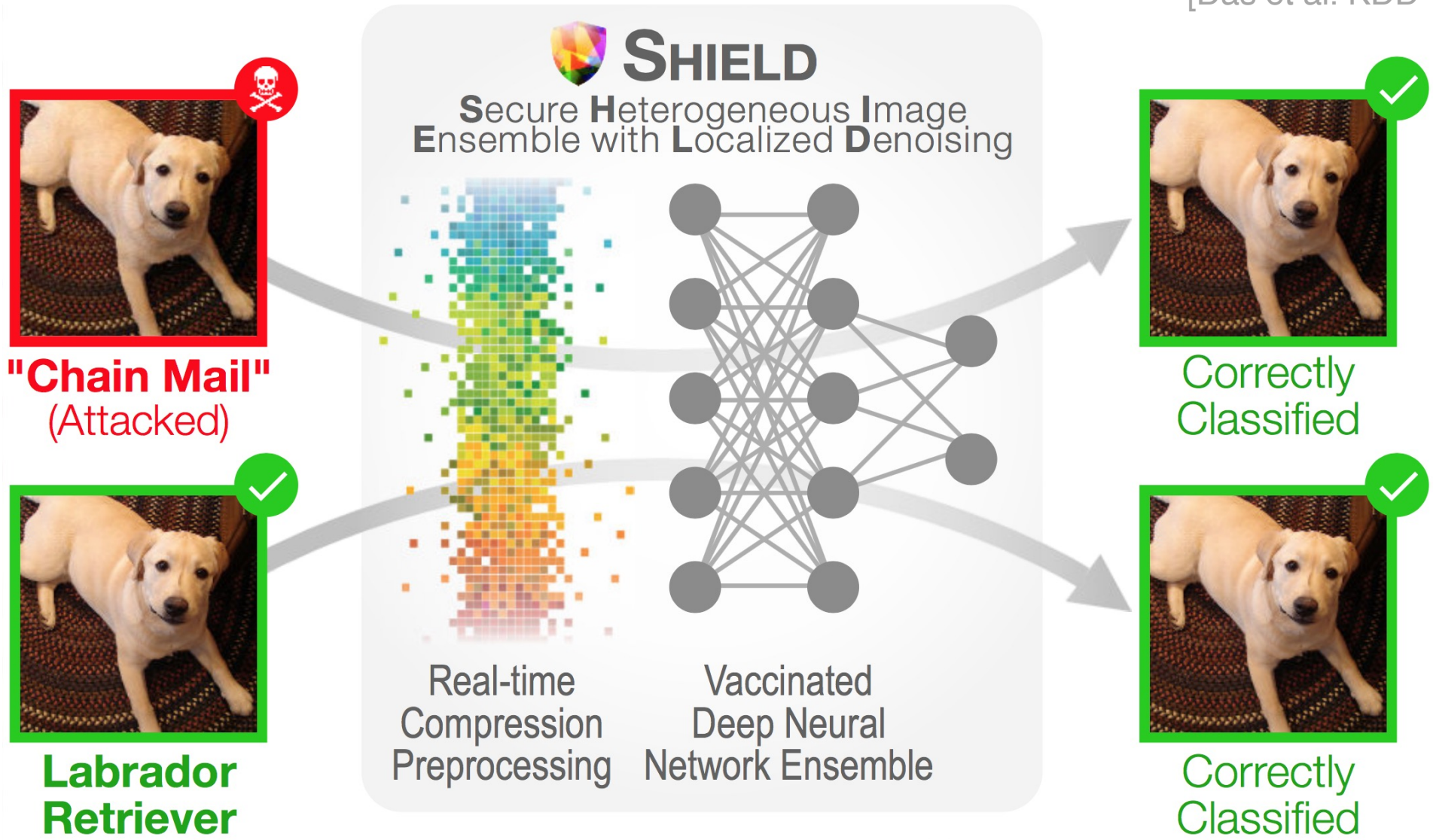
- Many papers published on daily basis
- The attacker has zero/little knowledge of the defense
- Usually (easily) breakable in the white-box setting
- Still useful when you want defenses that are scalable and ready-to-use

Approaches in Defenses

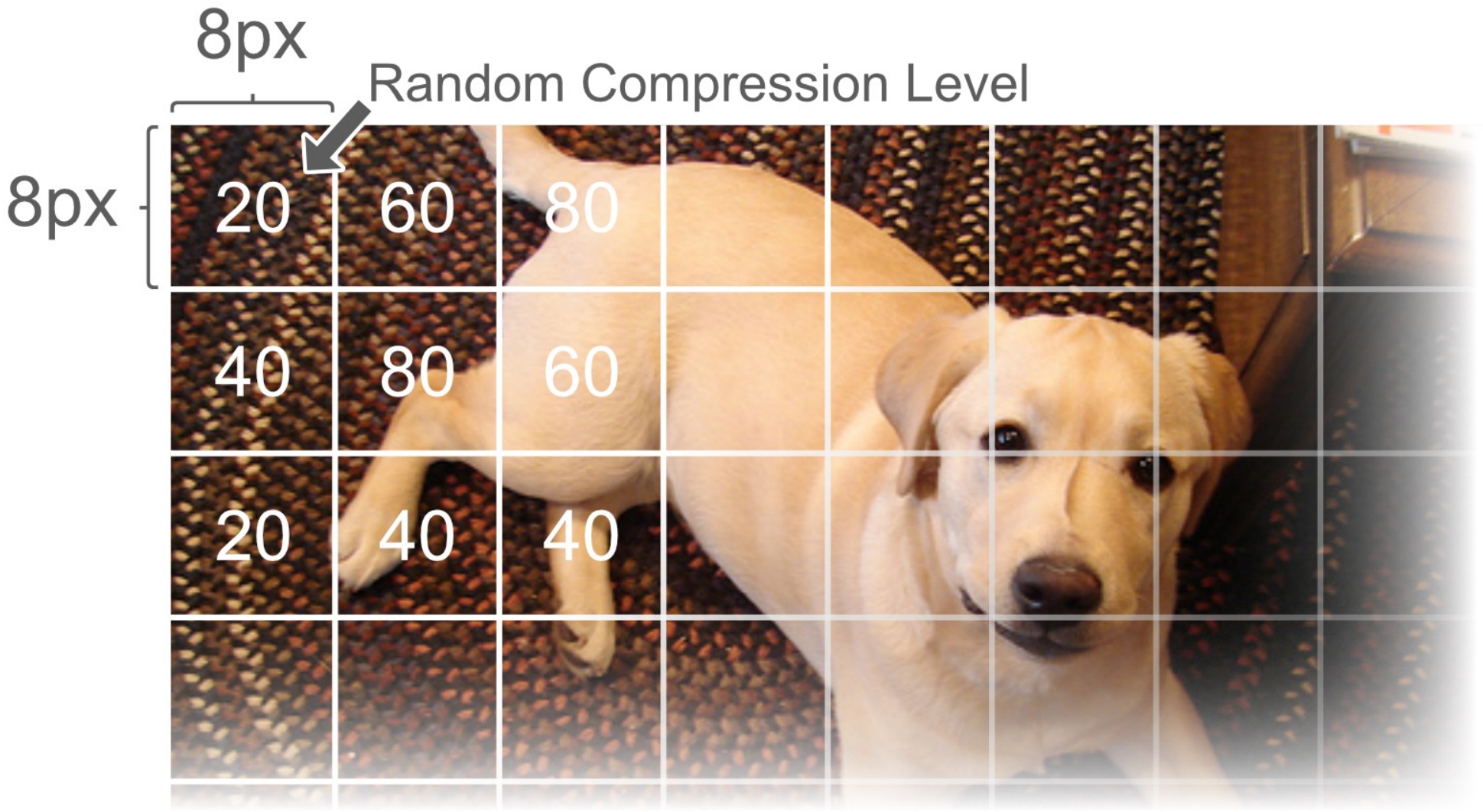
- **Data pre-processing**
  - Remove adversarial noise before feeding to the model
- **Model hardening**
  - Modify architecture and/or training process
- **Detection** (will be discussed in student presentation)
  - Detecting adversarial examples before classification

SHIELD: A Fast, Practical Defense using JPEG

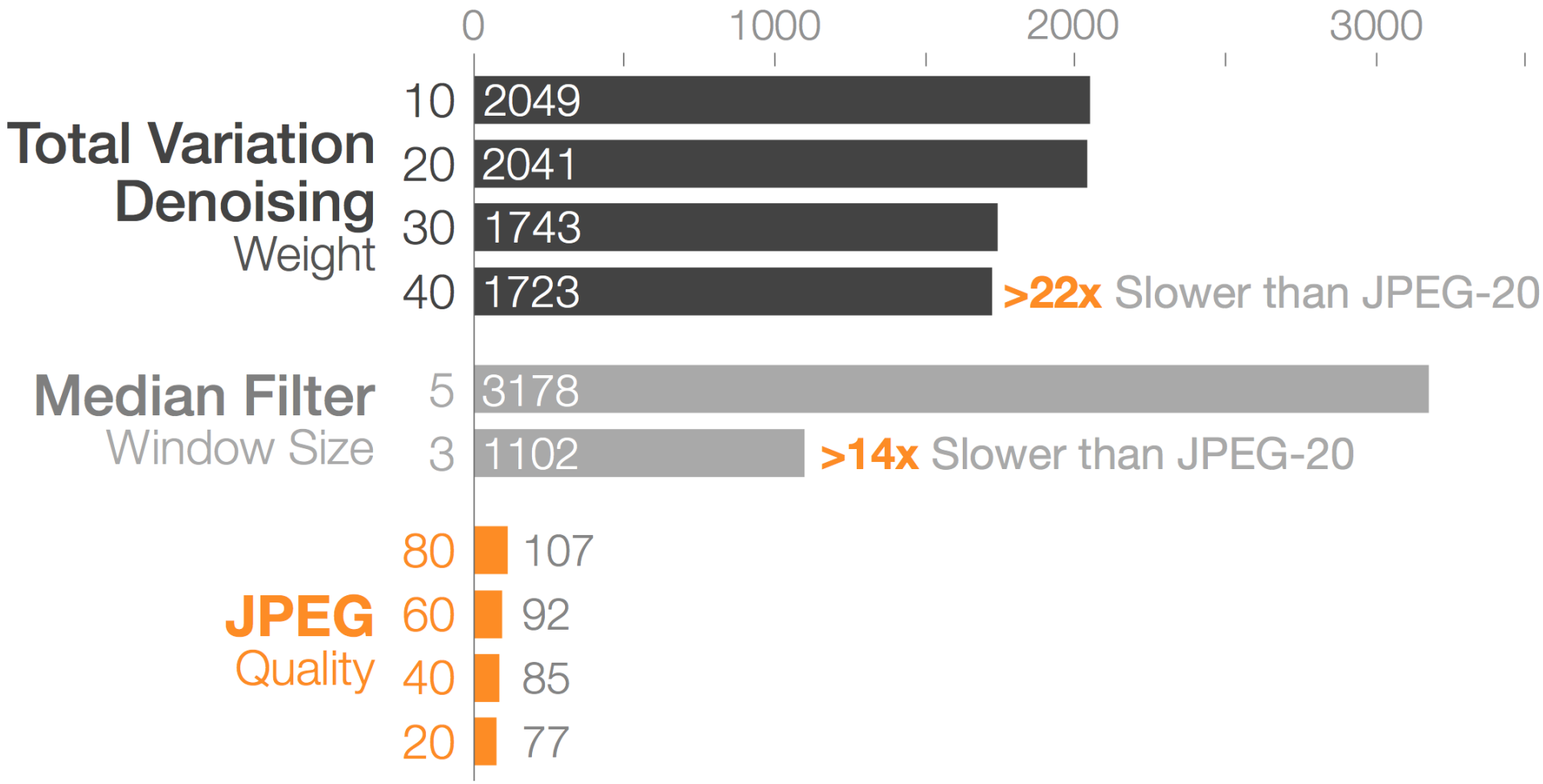
[Das et al. KDD'18]



Stochastic Local Quantization (SLQ)



SHIELD is Fast

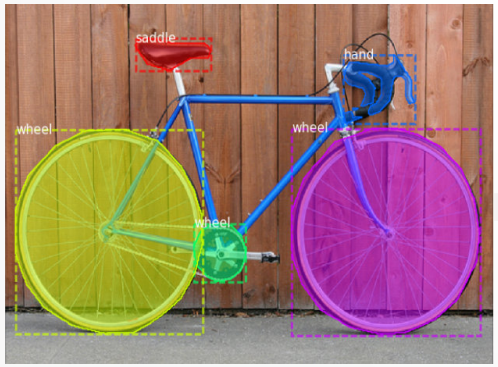


Runtimes in processing 50k ImageNet images (in seconds; shorter is better)

UnMask: Knowledge-Based Defense

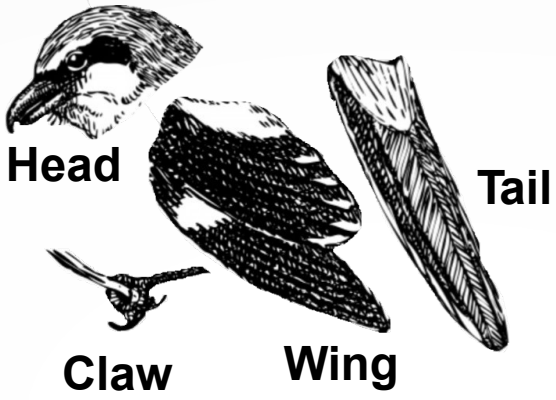
[Freitas et al. '20]

"Bird"
(attacked)

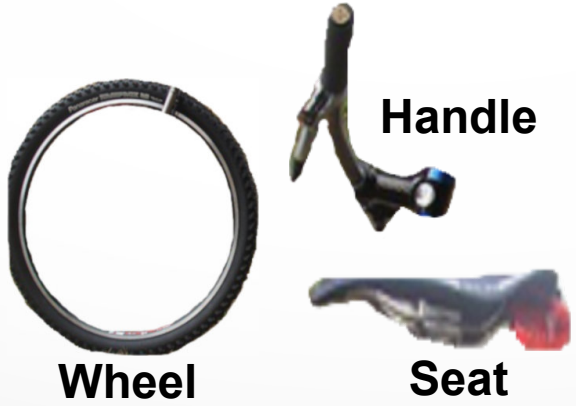


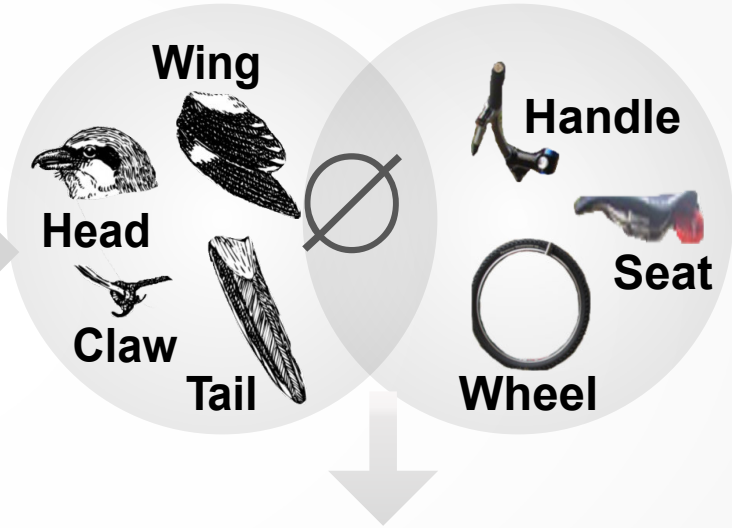
Segmentation Mask

Expected Features of Bird



Extracted Features



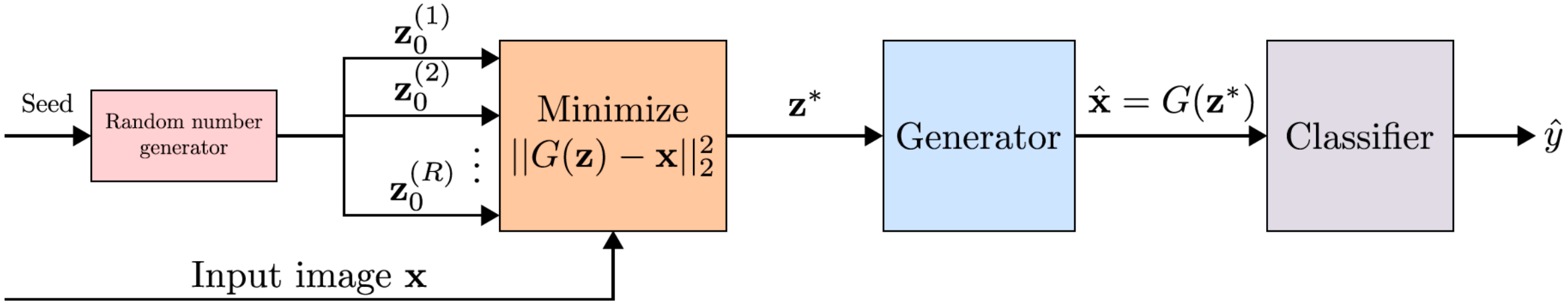


Similarity: 0
Under attack: Yes
Prediction: Bike

Defense-GAN

[Samangouei et al. ICLR'18]

- Train a generator on benign data
- Use the closest generated example for classification in testing
- Claimed to be robust in white-box and black-box settings



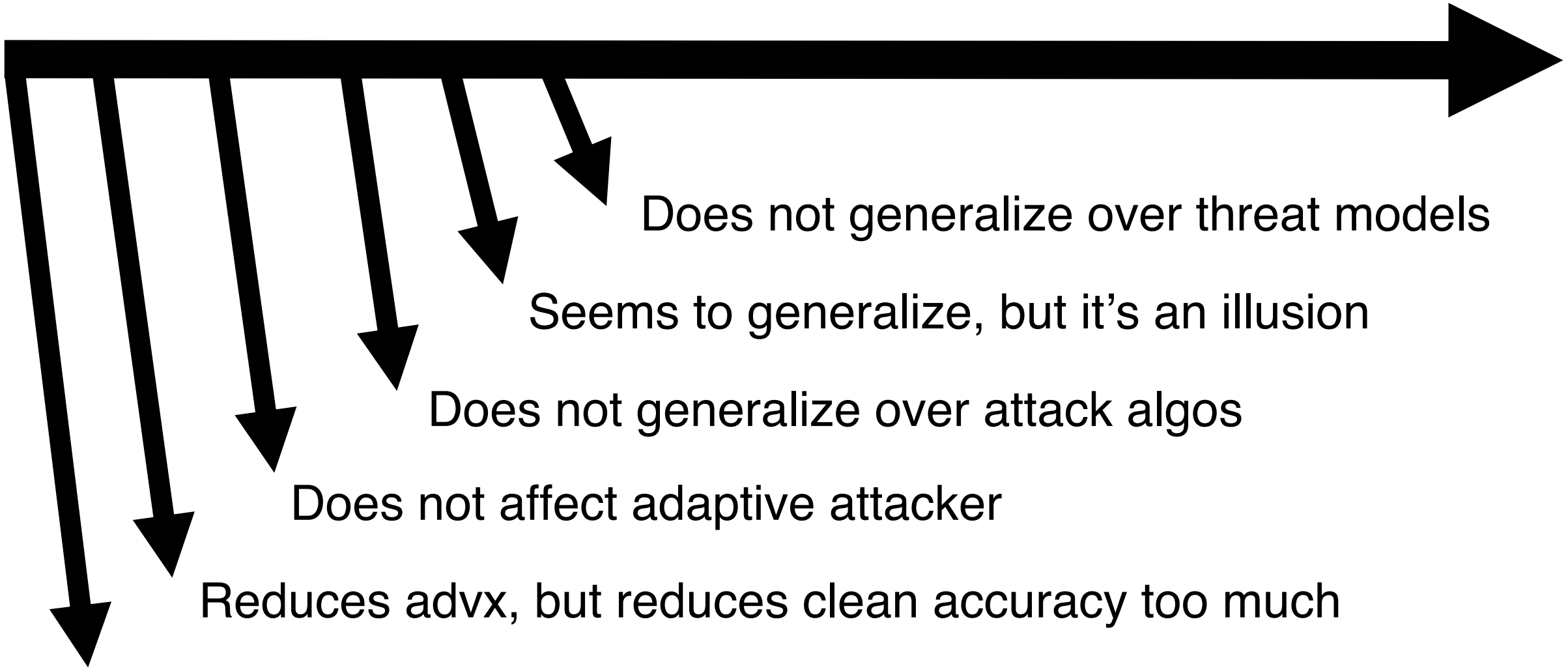
Defenses in Black-box Settings

- A form of “**security through obscurity**”
- Very hard to do evaluation properly
- Not that interesting in academic research now

Defenses in White-box Settings

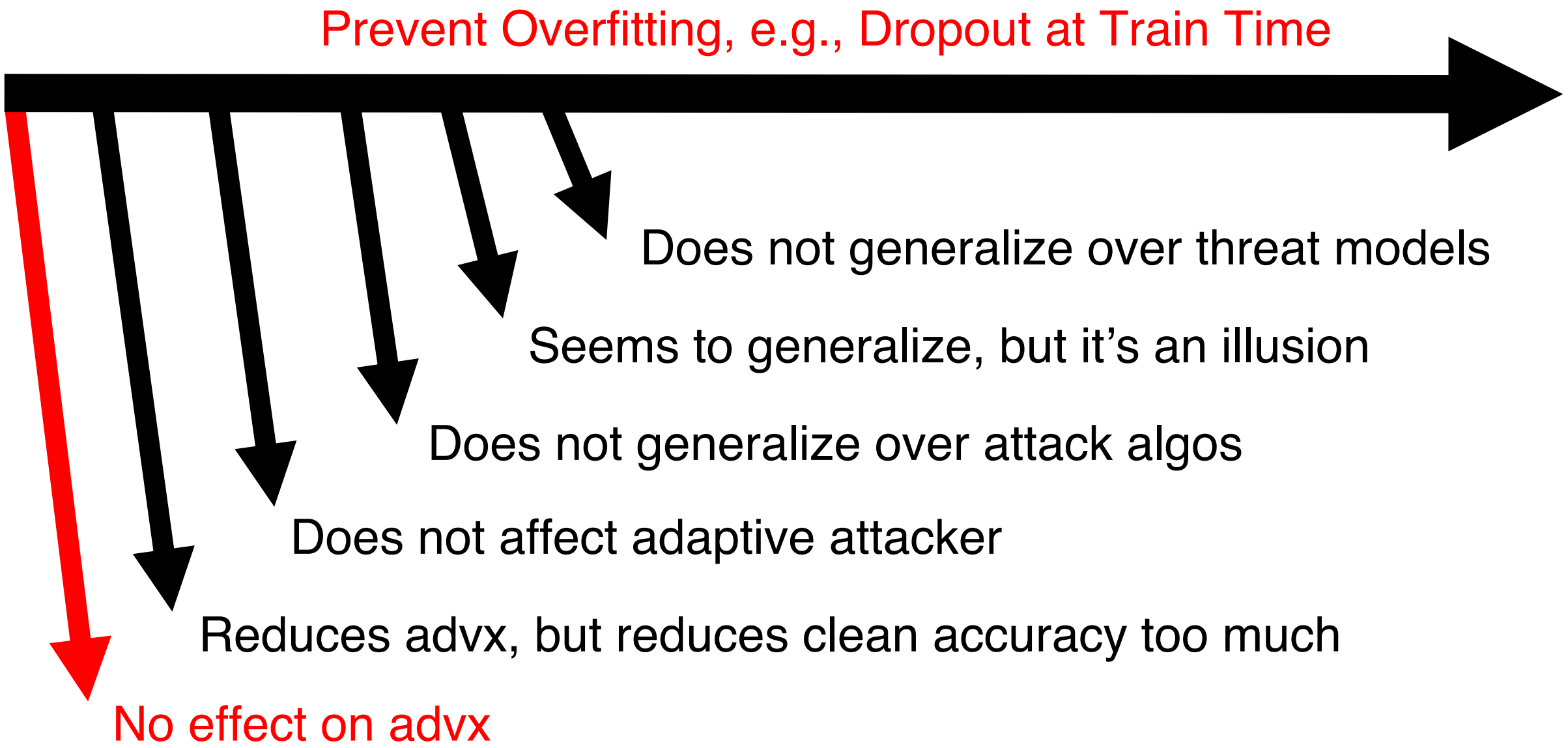
- Also many papers published everyday
- Most of them are found ineffective quickly because
  - Stronger attacks come up
  - Evaluation not thorough enough

Pipeline of Defense Failures



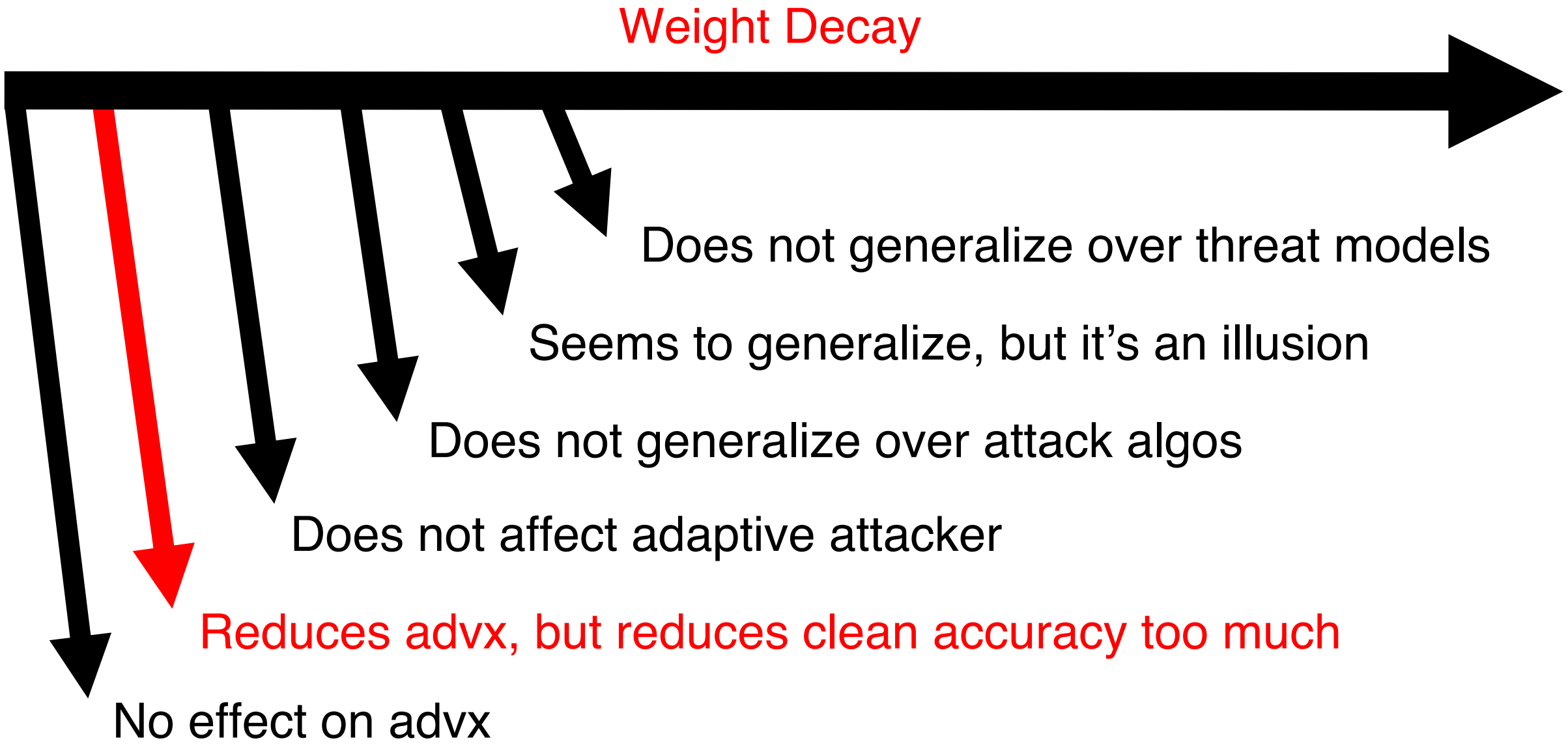
[Slides from Goodfellow 2018]

Pipeline of Defense Failures



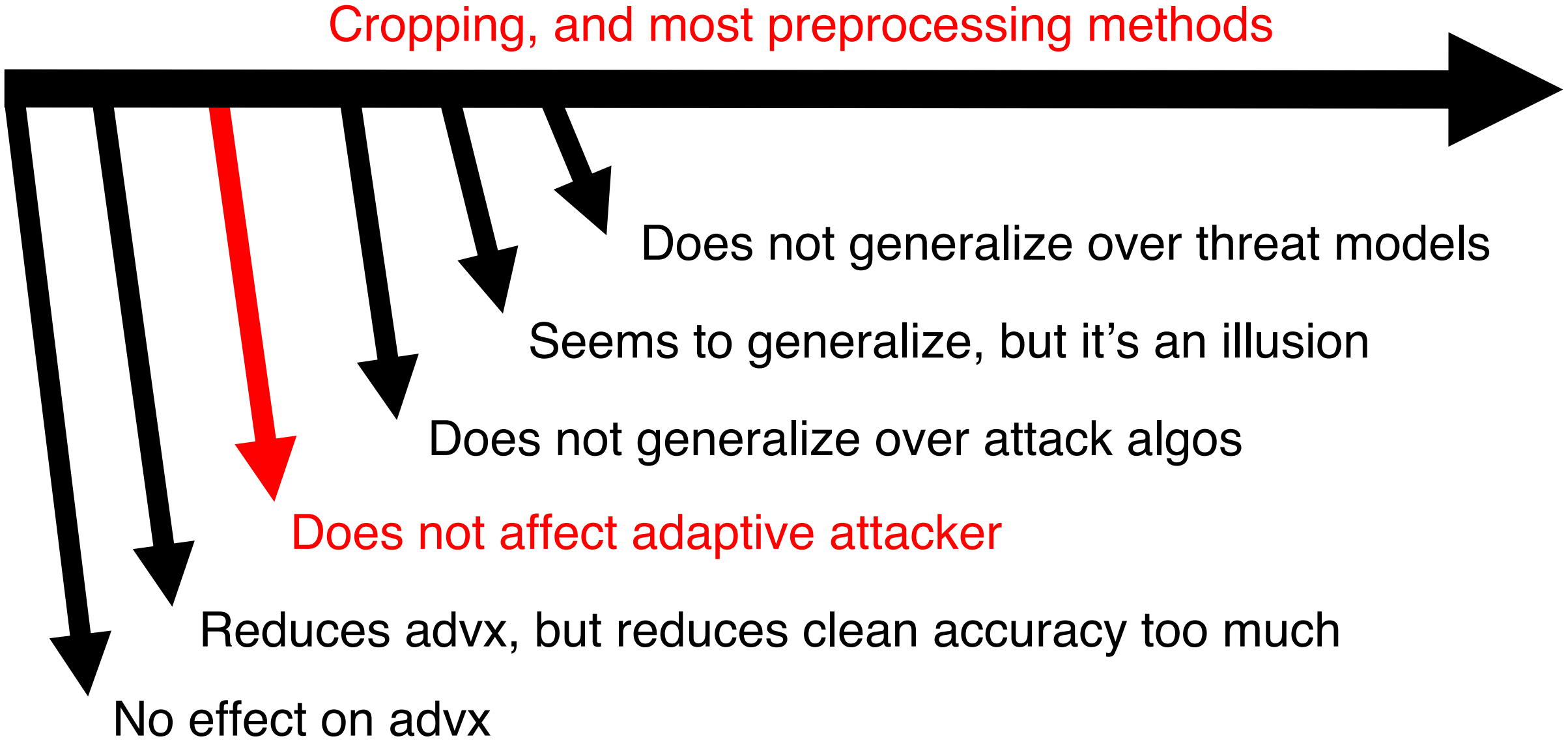
[Slides from Goodfellow 2018]

Pipeline of Defense Failures



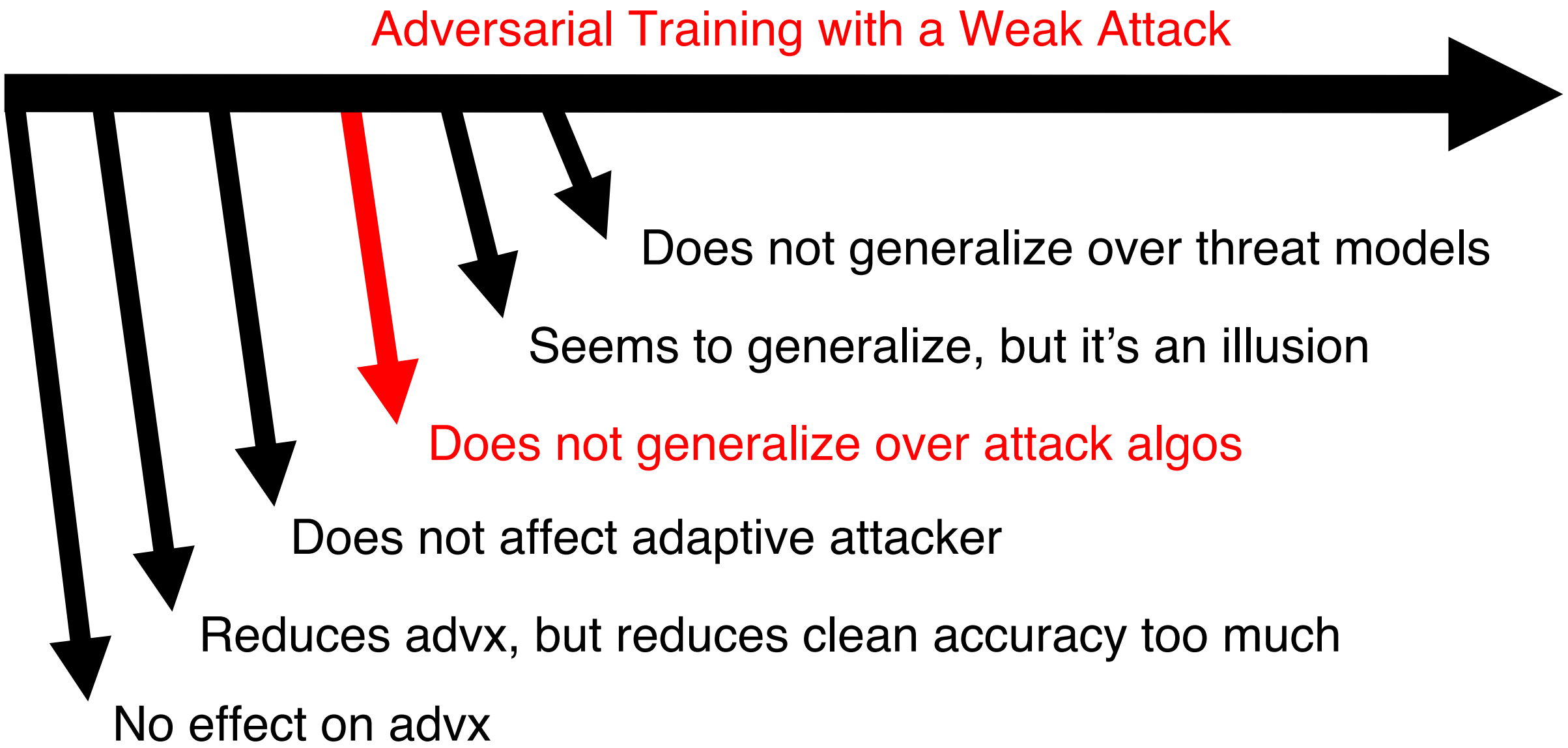
[Slides from Goodfellow 2018]

Pipeline of Defense Failures



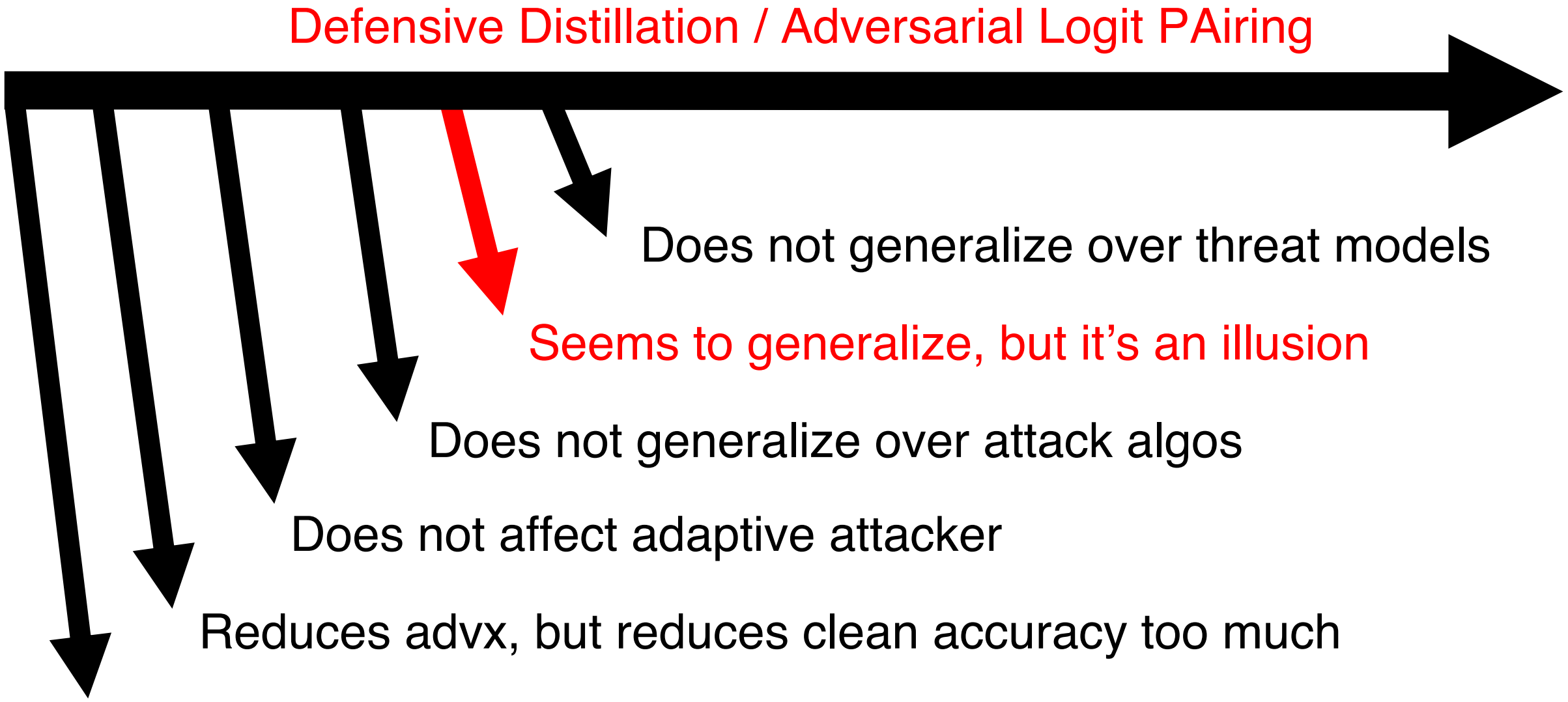
[Slides from Goodfellow 2018]

Pipeline of Defense Failures



[Slides from Goodfellow 2018]

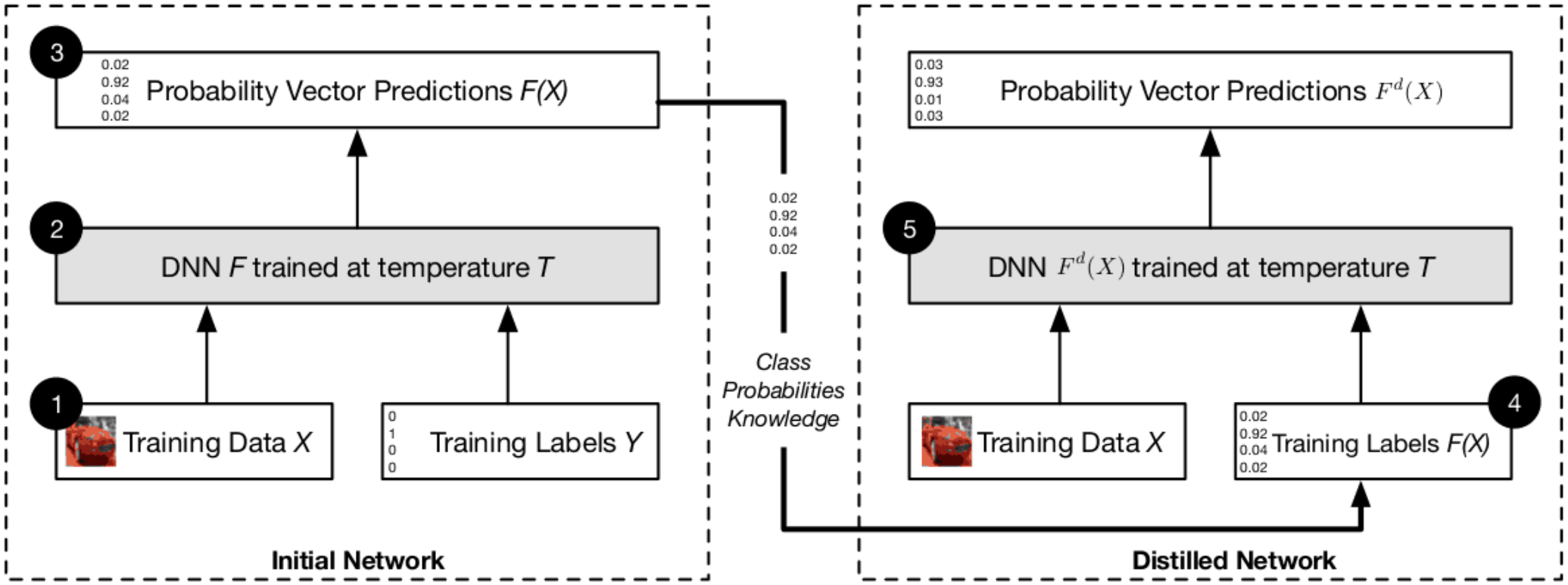
Pipeline of Defense Failures



[Slides from Goodfellow 2018]

Defensive Distillation

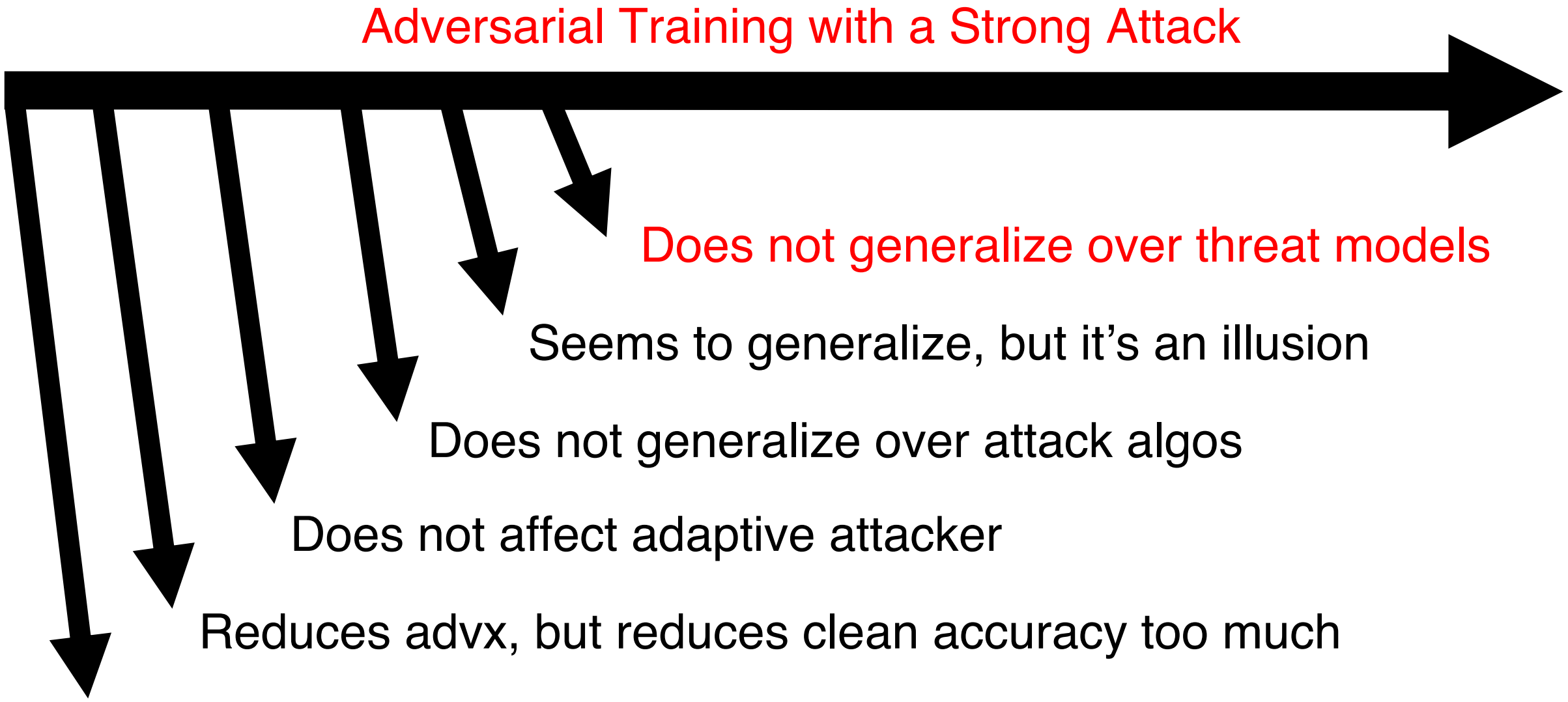
- Was once considered strongest defense
- Broken by transfer attack



Adversarial Logit Pairing [Kannan et al., '18]

- Idea: The logits for the benign and adversarial examples should be close
- Add a regularization term $\lambda \cdot L\big(f(x), f(x')\big)$ in training
- Was the state-of-the-art defense for some time
- Broken by simply increasing #iterations of PGD (up to 1000 iterations) [Engstrom et al., '18] .

Pipeline of Defense Failures



[Slides from Goodfellow 2018]

Obfuscated Gradients [Athalye et al. ICML'18]

- Many defenses are messing with gradient to make gradient-based attacks harder to find successful adversarial examples
- This does not actually make the model more robust

Obfuscated Gradients [Athalye et al. ICML'18]

Thought experiment:

- Instead of probability distribution, let the model only outputs the most likely class
- Small perturbation does not change output
- Gradient is almost always zero
- Model still has same blind-spots, just harder to find now

Obfuscated Gradients [Athalye et al. ICML'18]

Three major types:

- Shattered Gradients
- Stochastic Gradients
- Exploding & Vanishing Gradients

Obfuscated Gradients

[Athalye et al. ICML'18]

7 out of 9 white-box defenses in ICLR 2018 rely on this

- 6 of them are completely broken (~0% accuracy)
- Defense-GAN achieves ~55% accuracy on MNIST

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%

Identify Obfuscated Gradients

[Athalye et al. ICML'18]

1. One-step attacks perform better than iterative attacks
2. Black-box attacks are better than white-box attacks
3. Unbounded attacks do not reach 100% success
4. Random sampling finds adversarial examples
5. Increasing distortion bound does not increase success

Attack technique 1: BPDA

[Athalye et al. ICML'18]

Backward Pass Differentiable Approximation

1. Forward pass through the original network
2. Replace non-differentiable components by differentiable approximation for backward pass
   - E.g., replace JPEG compression with the identify function

Attack technique 2: EoT

[Athalye et al. ICML'18]

Expectation over Transformation

- For attacking defenses with randomness
- Original optimization: $\min_x f(x)$
- New optimization: $\min_x E_{t\sim T}\, f(t(x))$
  - $t(\cdot)$: some transformation function sampled from a distribution of transformations T
- $\nabla_x E_{t\sim T}\, f\big(t(x)\big) = E_{t\sim T}\, \nabla_x f\big(t(x)\big)$

Attack technique 2: EoT

[Athalye et al. ICML'18]

Very useful in physical attack

Optimize over different backgrounds, scales, rotations, lightings

Obfuscated Gradients

[Athalye et al. ICML'18]

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%

Adversarial Training with PGD

Cascade Adversarial Training

Adversarial Training

Standard training:

$$\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in S} \ell(\boldsymbol{x}, y; \boldsymbol{\theta})$$

Adversarial training (a.k.a. robust optimization) :

$$\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in S} \max_{\boldsymbol{\delta}\in\boldsymbol{\Delta}} \ell(\boldsymbol{x}+\boldsymbol{\delta}, y; \boldsymbol{\theta})$$

Hard min-max optimization problem

Adversarial Training: Approximation

$$\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in S} \max_{\boldsymbol{\delta}\in\boldsymbol{\Delta}} \ell(\boldsymbol{x}+\boldsymbol{\delta}, y; \boldsymbol{\theta})$$

Step 2 (outer minimization):

$$\min_{\boldsymbol{\theta}} \ell(\boldsymbol{x}+\boldsymbol{\delta}^*, y; \boldsymbol{\theta})$$

Step 1 (inner maximization):

$$\boldsymbol{\delta}^* = \operatorname*{argmax}_{\boldsymbol{\delta}\in\boldsymbol{\Delta}} \ell(\boldsymbol{x}+\boldsymbol{\delta}, y; \boldsymbol{\theta})$$

Danskin's Theorem:

If ℓ is continuously differentiable w.r.t. $\boldsymbol{x}$ and $\boldsymbol{\theta}$, and $\boldsymbol{\delta}^*$ is the maximizer of the inner problem, then this approximation is perfect

Adversarial Training

- Conditions of Danskin's Theorem do not hold
  - Loss function is not continuously differentiable, due to ReLU and max-pooling
  - We only approximately find the best perturbation
- Empirically, this approach works quite well, **if the perturbation is strong enough**

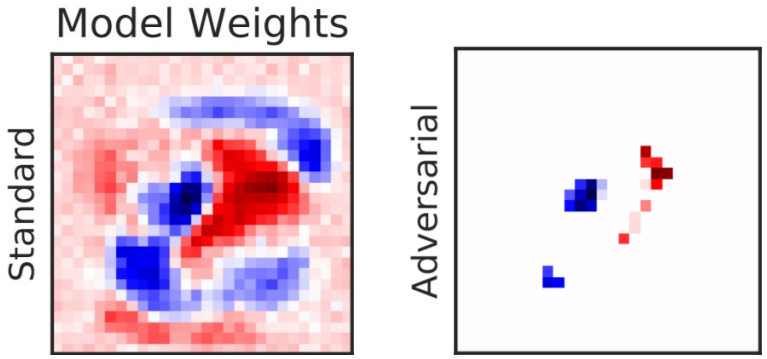
Adversarial Training with FGSM

[Goodfellow ICLR '14]

- Only effective against FGSM attack, but not against stronger attacks like PGD
- FGSM is too weak for solving the inner maximization
- It quickly converges to a model where FGSM can't find adversarial examples successfully
- There is also the label leaking problem

Adversarial Training as Regularization

- Adversarial training is a type of regularization



Linear model trained on binary MNIST (5 vs 7) [Tsipras et al. arXiv'18]

- Standard model utilizes all weakly-correlated pixels
- Adversarially trained model tend to use only highly-correlated features [Tsipras et al. arXiv'18]

Adversarial training as L1-regularization

- Let model $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$ and loss $\ell(\boldsymbol{w}, \boldsymbol{x}, y) = -y\boldsymbol{w} \cdot \boldsymbol{x}$
- Standard training: $\min_{w} -y\boldsymbol{w} \cdot \boldsymbol{x}$ (ignore summation)
- Adversarial training (with FGSM perturbation):

$$\begin{aligned}
&\min_{w} -y\boldsymbol{w} \cdot (\boldsymbol{x} + \epsilon \operatorname{sign}(\nabla_{\boldsymbol{x}} \ell(\boldsymbol{w}, \boldsymbol{x}, y))) \\
&= \min_{w} -y\boldsymbol{w} \cdot (\boldsymbol{x} + \epsilon \operatorname{sign}(-y\boldsymbol{w})) \\
&= \min_{w} -y\boldsymbol{w} \cdot \boldsymbol{x} + \epsilon\, (-\mathrm{y}\mathbf{w}) \cdot \operatorname{sign}(-y\boldsymbol{w}) \\
&= \min_{w} -y\boldsymbol{w} \cdot \boldsymbol{x} + \epsilon \|\boldsymbol{w}\|_1
\end{aligned}$$

Adversarial training as L1-regularization

- The connection does not hold in non-linear cases
- Empirically, “input gradient regularization” still useful
  - $\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in S} \ell(\boldsymbol{x}, y; \boldsymbol{\theta}) + \lambda \cdot \|\nabla_x \ell(\boldsymbol{x}, y; \boldsymbol{\theta})\|_1$
  - Also known as double backpropogation
  - Known to increase generalization [Drucker & LeCun ‘91]

Adversarial Training with PGD

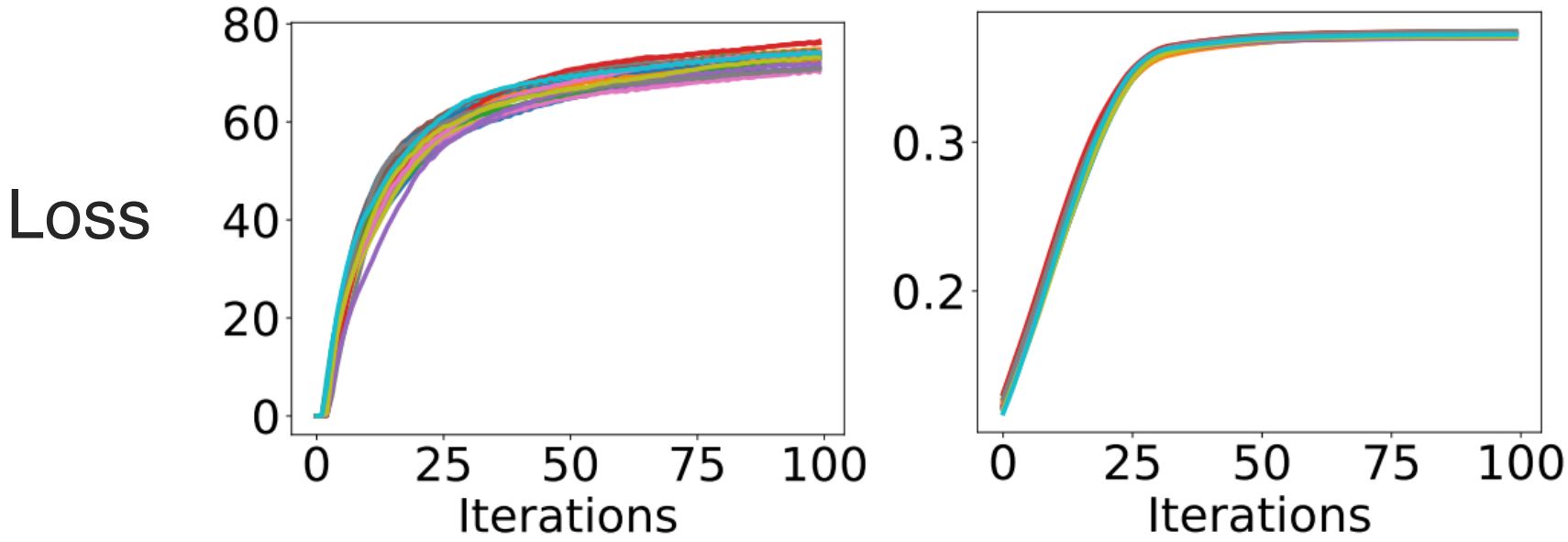
[Madry ICLR ‘18]

- PGD is strong in the sense that it consistently finds the perturbation that maximizes the loss of the model
- The authors conjecture that, if the adversary only uses gradients of the loss, it will not find significantly better local maxima than PGD
- In other words, PGD is the strongest first order attack

Adversarial Training with PGD

[Madry ICLR '18]

- Start PGD from 10^5 uniformly random points around ℓ_∞-ball of the example, and run until the loss plateaus



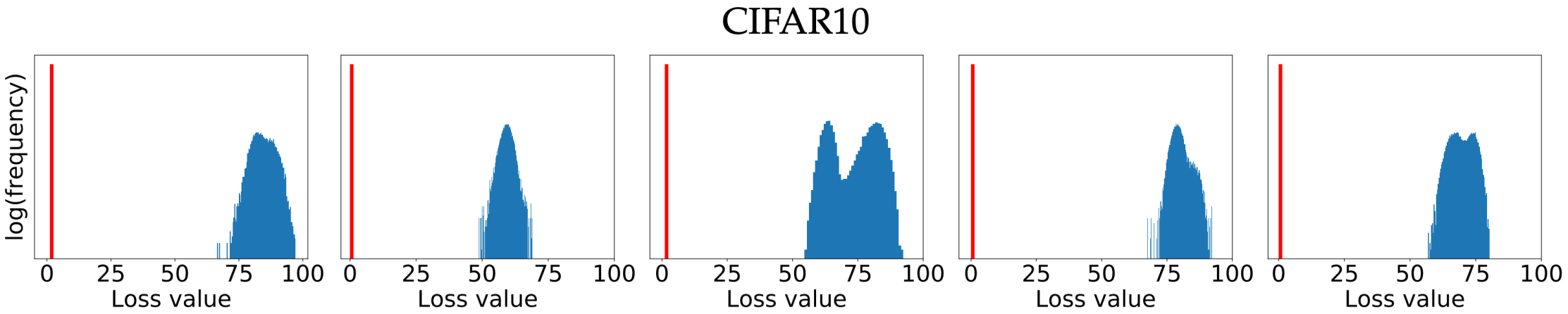
(c) CIFAR10 Natural training

(d) CIFAR10 Adversarial training

Adversarial Training with PGD

[Madry ICLR '18]

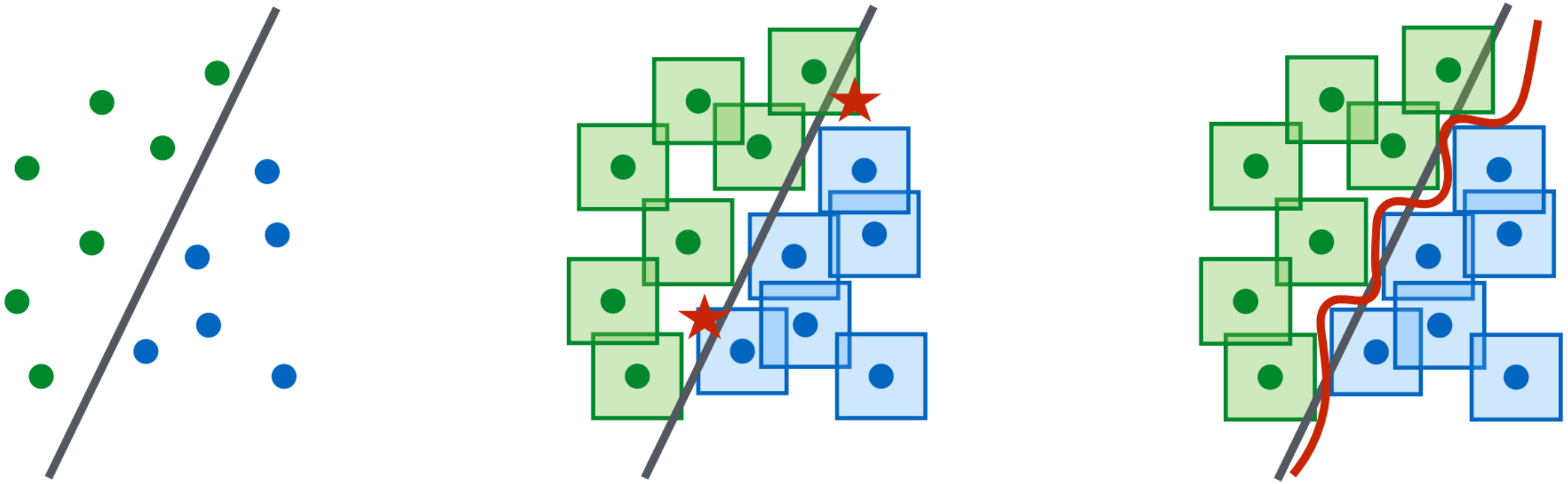
- Start PGD from 10^5 uniformly random points around ℓ_∞-ball of the example, and run until the loss plateaus
- Blue histogram: loss on standard network
- Red histogram: loss on adversarially-trained network



Adversarial Training with PGD

[Madry ICLR '18]

Besides a strong attack, model capacity also important



Adversarial Training with PGD

[Madry ICLR '18]

- Drawback: slow and not scalable
- Each iteration of PGD is computationally as expensive as the network update
- PGD typically uses ≥ 20 iterations
- At least 20 times slower than standard training!

Adversarial Training with PGD

[Madry ICLR '18]

- The current state-of-the-art white-box defense
- Many subsequent works try to speed up the process, but most of them are not thoroughly tested yet, and some are already broken

Faster Adversarial Training?

- AT with FGSM and random initialization [Wong et al.. ICLR '20]
- Cyclic learning rate & Mixed-precision arithmetic

Method	Epochs	Seconds/epoch	Total time (minutes)
DAWNBench + PGD-7	10	104.94	17.49
DAWNBench + Free ($m = 8$)	80	13.08	17.44
DAWNBench + FGSM	15	25.36	6.34
PGD-7 (Madry et al., 2017)[5]	205	1456.22	4965.71
Free ($m = 8$) (Shafahi et al., 2019)[6]	205	197.77	674.39

Stronger Adversarial Training?

Defense Method	Loss Function
Standard	$\mathrm{CE}(\mathbf{p}(\hat{\mathbf{x}}', \boldsymbol{\theta}), y)$
ALP	$\mathrm{CE}(\mathbf{p}(\hat{\mathbf{x}}', \boldsymbol{\theta}), y) + \lambda \cdot \|\mathbf{p}(\hat{\mathbf{x}}', \boldsymbol{\theta}) - \mathbf{p}(\mathbf{x}, \boldsymbol{\theta})\|_2^2$
CLP	$\mathrm{CE}(\mathbf{p}(\mathbf{x}, \boldsymbol{\theta}), y) + \lambda \cdot \|\mathbf{p}(\hat{\mathbf{x}}', \boldsymbol{\theta}) - \mathbf{p}(\mathbf{x}, \boldsymbol{\theta})\|_2^2$
TRADES	$\mathrm{CE}(\mathbf{p}(\mathbf{x}, \boldsymbol{\theta}), y) + \lambda \cdot \mathrm{KL}\big(\mathbf{p}(\mathbf{x}, \boldsymbol{\theta}) \| \mathbf{p}(\hat{\mathbf{x}}', \boldsymbol{\theta})\big)$
MMA	$\mathrm{CE}(\mathbf{p}(\hat{\mathbf{x}}', \boldsymbol{\theta}), y) \cdot \mathbb{1}(h_{\boldsymbol{\theta}}(\mathbf{x}) = y) + \mathrm{CE}(\mathbf{p}(\mathbf{x}, \boldsymbol{\theta}), y) \cdot \mathbb{1}(h_{\boldsymbol{\theta}}(\mathbf{x}) \neq y)$
MART	$\mathrm{BCE}(\mathbf{p}(\hat{\mathbf{x}}', \boldsymbol{\theta}), y) + \lambda \cdot \mathrm{KL}(\mathbf{p}(\mathbf{x}, \boldsymbol{\theta}) \| \mathbf{p}(\hat{\mathbf{x}}', \boldsymbol{\theta})) \cdot (1 - \mathbf{p}_y(\mathbf{x}, \boldsymbol{\theta}))$

[Wang et al.. ICLR ‘20]

How Do I Know If My Defense Is Good?

1. Define concrete “threat model”
   - What does the attacker know?
   - What power / constraints do the attacker have?
   - What dataset is considered?

- **Example**: The attacker has white-box knowledge, and he can perturb each pixel up to 8 in [0, 255] scale. Claim: 85% accuracy on ImageNet

How Do I Know If My Defense Is Good?

2. Write clear and precise description of your defense
   - Make your code and model public
   - Many researcher will help test your method
   - The evaluation is usually more accurate when conducted by different people

How Do I Know If My Defense Is Good?

3. Evaluation
   - Design adaptive attacks targeting your model
   - Check if there are obfuscated gradients
     - Black-box attack, plot distortion-vs-accuracy curve, …
   - Try different hyperparameters, e.g., iterations, step size, …

Next Time

- Why do adversarial examples exist?
- Many theories, but no absolute answer yet