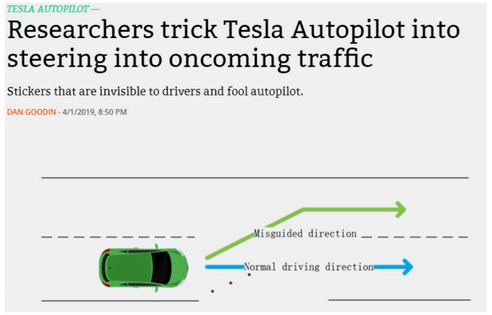
COMP5212: Machine Learning

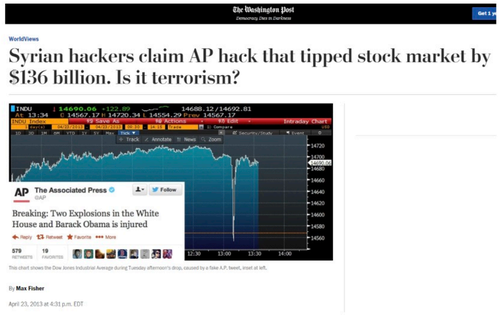
Lecture 18: Adversarial Machine Learning

Sen LI **Zeyu QIN**

Machine learning

Beyond Accuracy





Microsoft silences its new A.I. bot Tay, after Twitter users teach it racism [Updated]

Sarah Perez @sarahintampa / 10:16 am EDT • March 24, 2016

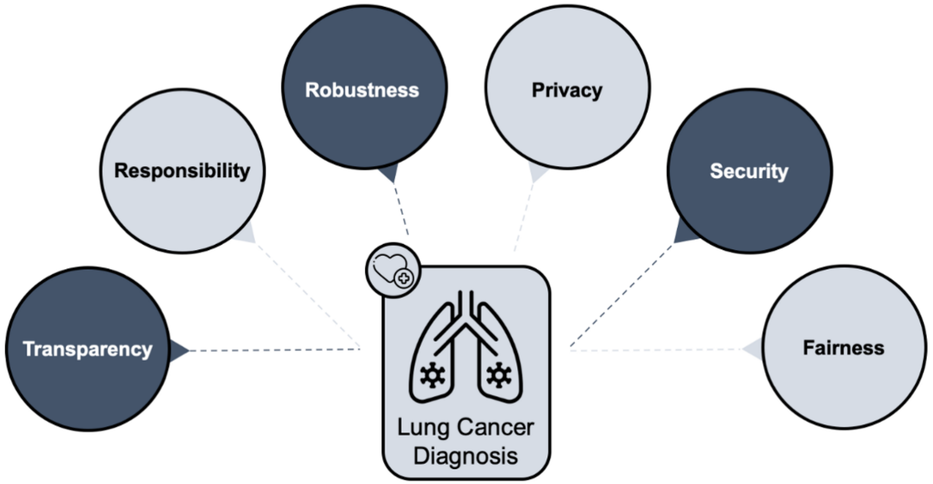
Comment

Microsoft's newly launched A.I.-powered bot called Tay, which was responding to tweets and chats on GroupMe and Kik, has already been shut down due to concerns with its inability to recognize when it was making offensive or racist statements. Of course, the bot wasn't *coded* to be racist, but it "learns" from those it interacts with. And naturally, given that this is the Internet, one of the first things online users taught Tay was how to be racist, and how to spout back ill-informed or inflammatory political opinions. [**Update**: Microsoft now says it's "making adjustments" to Tay in light of this problem.]

Trustworthy ML

What and why

- Not alchemy
  - Explainability
  - Security
  - Privacy
  - Fairness
  - Integrity
  - …
- Establish model understanding



THE NATIONAL SECURITY COMMISSION ON ARTIFICIAL INTELLIGENCE

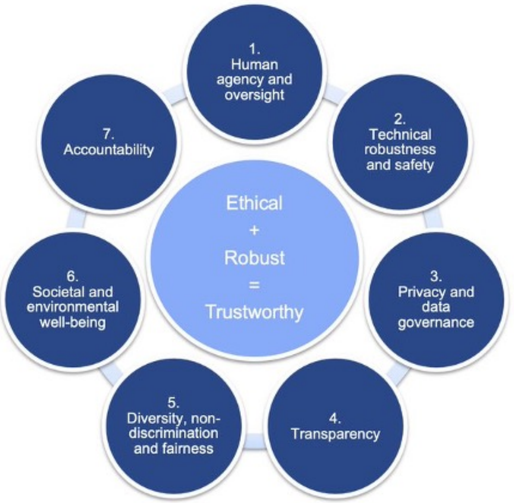
人工智能安全测评白皮书

（2021）

国家语音及图像识别产品质量监督检验中心

国家工业信息安全发展研究中心人工智能所

2021 年 10 月



Trustworthy ML

Integrity

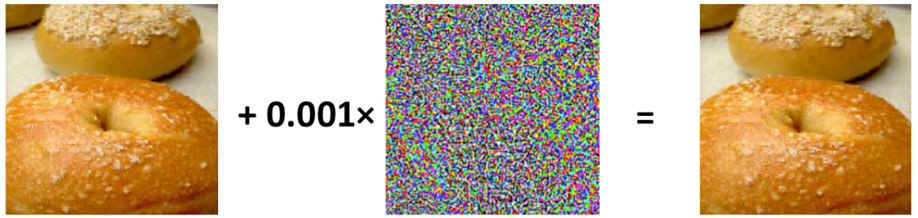
- Training-time integrity and Testing-time integrity

Attack Category	Attack Target	Attack Mechanism	Training Process	Inference Process
Backdoor Attack	Misclassify attacked samples; Behave normal on benign samples.	Excessive learning ability of models.	Under control.	Out of control.
Adversarial Attack	Misclassify attacked samples; Behave normal on benign samples.	Behavior differences between models and humans.	Out of control.	Attackers need to generate adversarial perturbation through an iterative optimization process.
Data Poisoning	Reduce model generalization.	Overfitting to bad local optima.	Can only modify the training set.	Out of control.

Testing time integrity

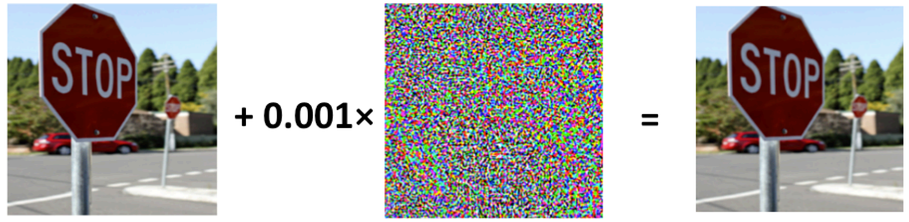
Adversarial examples

- An adversarial example can easily fool a deep network
- Robustness is critical in real systems



Bagle

piano



stop sign

speed limit 40

Adversarial examples

Definition

- Given a K-way multi-class classification model $f : \mathbb{R}^d \rightarrow \{1, \ldots, K\}$ and an original example x_0, the goal is to generate an adversarial example x such that
  - x is close to x_0 and $\arg\max_i f_i(x) \neq \arg\max_i f_i(x_0)$
  - i.e., x has a different prediction with x_0 by model f.

Adversarial example

Attack as an optimization problem

- Craft adversarial example by solving
  - $\arg\min_x \ ||x - x_0||^2 + c \cdot h(x)$
- $||x - x_0||^2$: the distortion

Adversarial example

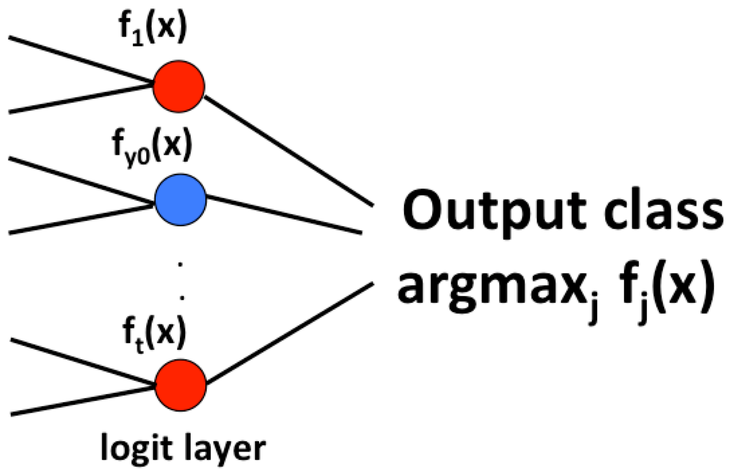
Attack as an optimization problem

- Craft adversarial example by solving
  - $\arg\min_x \ ||x - x_0||^2 + c \cdot h(x)$
- $||x - x_0||^2$: the distortion
- $h(x)$: loss to measure the successfulness of attack

Adversarial example

Attack as an optimization problem

- Craft adversarial example by solving
  - $\arg\min_x \ \|x - x_0\|^2 + c \cdot h(x)$
- $\|x - x_0\|^2$: the distortion
- $h(x)$: loss to measure the successfulness of attack
- Untargeted attack: success if $\arg\max_j f_j(x) \neq y_0$
  - $h(x) = \max\{f_{y_0}(x) - \max_{j \neq y_0} f_j(x), 0\}$



How to find adversarial examples

White-box vs black-box setting

- Attackers knows the model structure and weights (white-box)
- Can query the model to get probability output (soft-label)
- Can query the model to get label output (hard-label)
- No information about the model (universal)

Adversarial example

White-box setting

- $\arg\min_x \ \|x - x_0\|^2 + c \cdot h(x)$
- Model (network structure and weights) is revealed to attacker
  - $\Rightarrow$ gradient of $h(x)$ can be computed
  - $\Rightarrow$ attacker minimizes the objective by gradient descent

Adversarial example

White-box adversarial attack

- C&W attack [CW17]:
  - $h(x) = \max\{[Z_{y_0}(x) - \max_{j \neq y} Z_j(x)], -\kappa\}$
  - Where $Z(x)$ is the pre-softmax layer output

Adversarial example

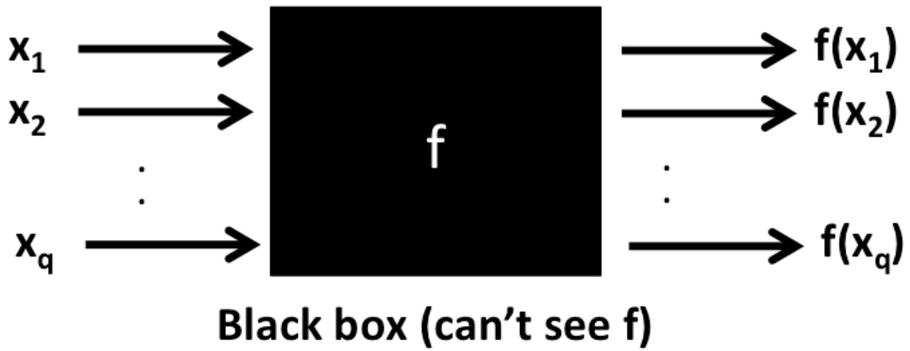
White-box adversarial attack

- If there is $\|x - x_0\|_\infty$ constraint, we could turn to solve by
- FGSM attack [GSS15]:
  - $x \leftarrow proj_{x+\mathcal{S}}(x_0 + \alpha \text{sign}(\nabla_{x_0} \ell(\theta, x, y)))$
- PGD attack [KGB17, MMS18]
  - $x^{t+1} \leftarrow proj_{x+\mathcal{S}}(x^t + \alpha \text{sign}(\nabla_{x^t} \ell(\theta, x, y)))$

Adversarial example

Black-box Soft-label Setting

- Black-box Soft Label setting (practical setting):
  - Structure and weights of deep network are not revealed to attackers
  - Attacker can query the ML model and get the probability output



  - Cannot compute gradient ∇_x

Soft-label Black-box Adversarial attack

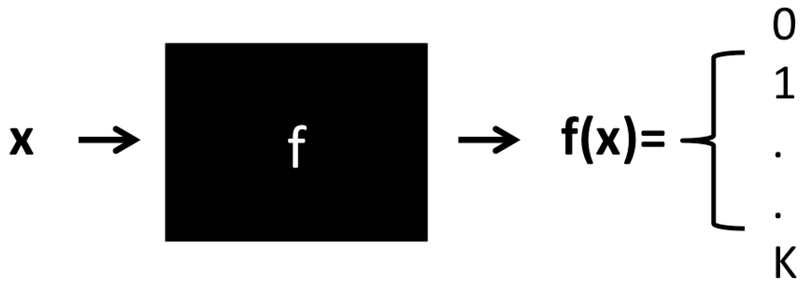
- Soft-label Black-box: query to get the **probability output**
- Key problem: how to estimate gradient?
- Gradient-based [CZS17,IEAL18]:
  - $\nabla_x = \frac{h(x+\beta u) - h(x)}{\beta} \cdot u$
- Genetic algorithm [ASC19]

Soft-label Black-box Adversarial attack

- Query based
- Transfer based:
  - Train a substitute model and conduct the white-box attack

Hard-label Black-box Attack

- Model is not known to the attacker
- Attacker can make query and observe hard-label multi-class output



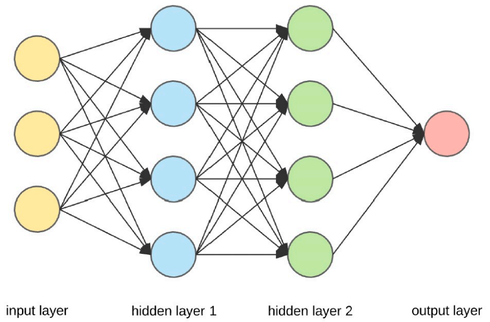
- (K: number of classes)

Why Hard-label

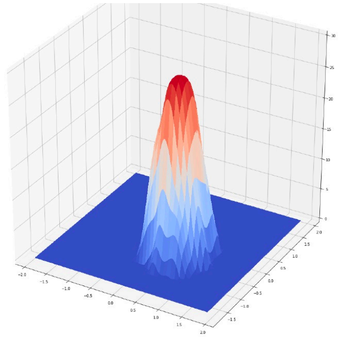
- More practical setting for attacker
- Discrete and complex models (e.g quantization, projection, detection)
- Framework friendly

The difficulty

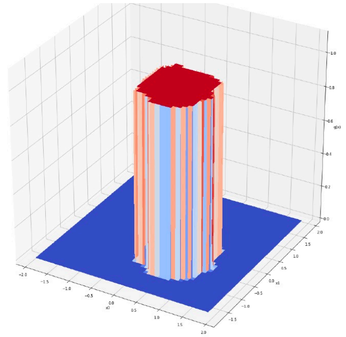
- Hard-label attack on a simple 3-layer neural network yields a discontinuous optimization problem



(a) neural network $f(x)$



(b) $h(Z(x))$



(c) $h(f(x))$

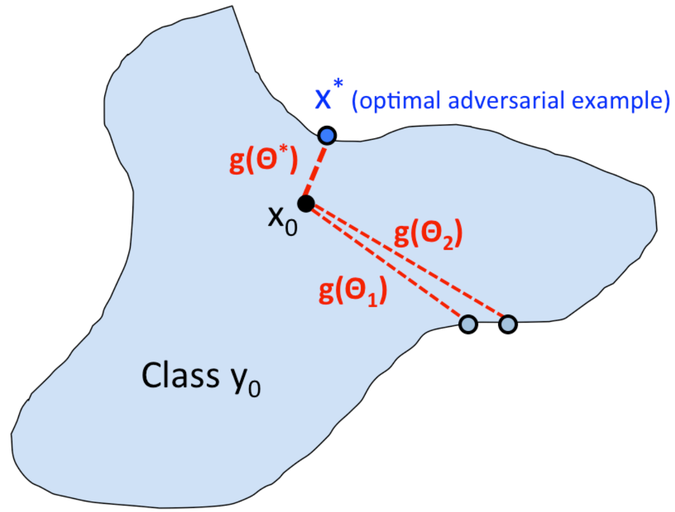
Reformulation

- We reformulate the attack optimization problem (untargeted attack):

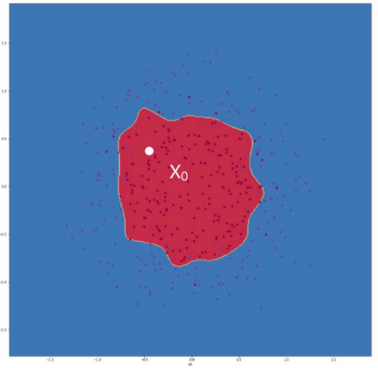
$$\theta^* = \arg\min_{\theta} \; g(\theta)$$

- where $g(\theta) = \text{argmin}_{\lambda>0} \left(f(x_0 + \lambda \frac{\theta}{\|\theta\|}) \neq y_0 \right)$

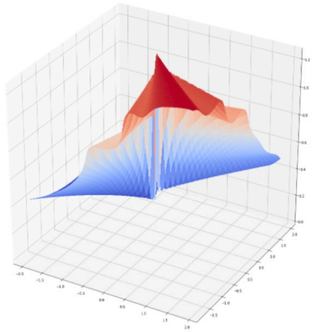
- θ: the direction of adversarial example



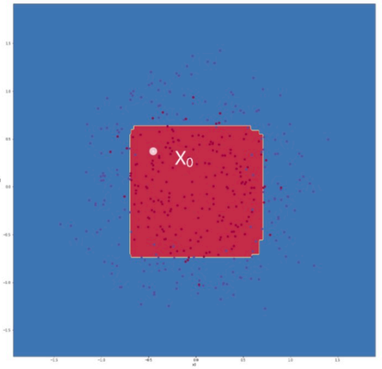
Examples



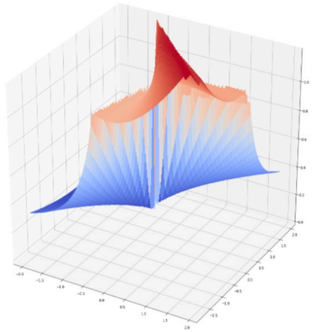
Neural network decision function



$g(\boldsymbol{\theta})$



Boosting Tree decision function



$g(\boldsymbol{\theta})$

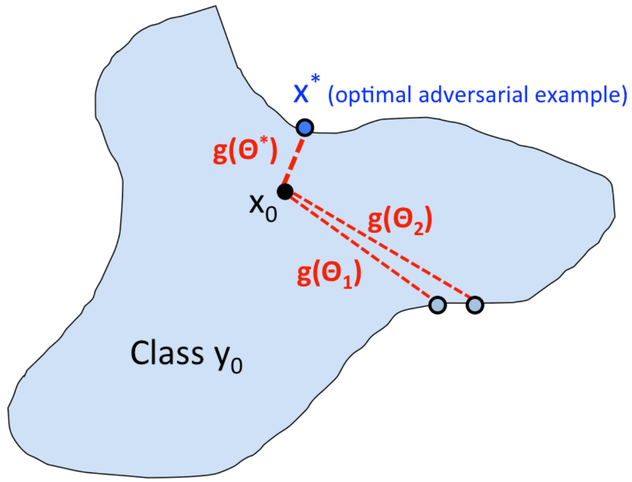
Two things unaddressed

$$\theta^* = \arg\min_{\theta} \; g(\theta)$$

- where $g(\theta) = \text{argmin}_{\lambda>0} \left(f(x_0 + \lambda \frac{\theta}{\|\theta\|}) \neq y_0 \right)$
- How to estimate $g(\theta)$
- How to find θ^*

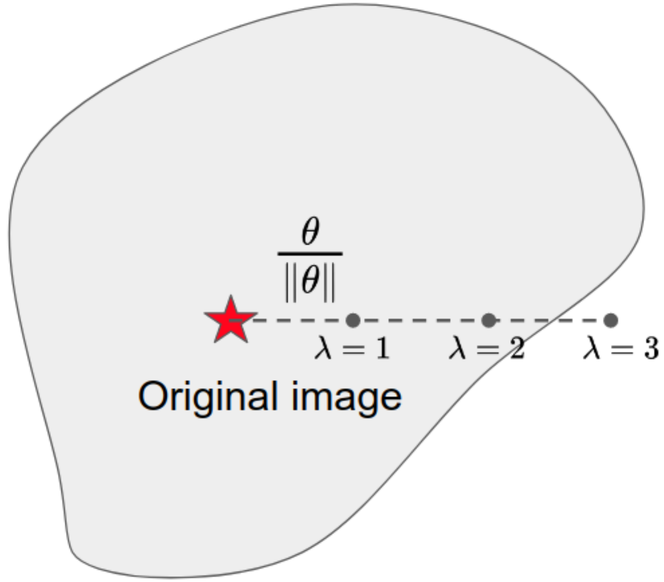
Computing Function Value

- Can't compute the gradient of g
- However, we can compute the function value of g using queries of $f(\cdot)$
- Implemented using fine-grained search + binary search



Estimation of $g(\theta)$

- Fine-grained search
- Binary search
  - Prediction unchanged enlarge g
  - Prediction changed shrink g



How to optimize $g(\theta)$

- The gradient of g is available by
  - $$\nabla g(\theta) \approx \frac{g(\theta + \beta u) - g(\theta)}{\beta} \cdot u$$
- One u is too noisy, better to use multiple u (~ 20)
- Zeroth order optimization for minimizing $g(\theta)$

Algorithm

Algorithm 1 OPT attack (ICLR '19)

Input: Hard-label model f, original image x_0, initial $\boldsymbol{\theta}_0$.
for $t = 0, 1, 2, \ldots, T$ **do**
    Randomly choose $\boldsymbol{u}$ from a zero-mean Gaussian distribution
    Evaluate $g(\boldsymbol{\theta}_t)$ and $g(\boldsymbol{\theta}_t + \beta \boldsymbol{u})$
    Compute $\hat{\boldsymbol{g}} = \dfrac{g(\boldsymbol{\theta}_t + \beta \boldsymbol{u}) - g(\boldsymbol{\theta}_t)}{\beta} \cdot \boldsymbol{u}$
    Update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \hat{\boldsymbol{g}}$
return $\boldsymbol{x}_0 + g(\boldsymbol{\theta}_T)\boldsymbol{\theta}_T$

Algorithm

Algorithm 2 OPT attack (ICLR '19)

Input: Hard-label model f, original image x_0, initial $\boldsymbol{\theta}_0$.
for $t = 0, 1, 2, \ldots, T$ **do**
    Randomly choose $\boldsymbol{u}_t$ from a zero-mean Gaussian distribution
    Evaluate $g(\boldsymbol{\theta}_t)$ and $g(\boldsymbol{\theta}_t + \beta \boldsymbol{u})$
    Compute $\hat{\boldsymbol{g}} = \frac{g(\boldsymbol{\theta}_t + \beta \boldsymbol{u}) - g(\boldsymbol{\theta}_t)}{\beta} \cdot \boldsymbol{u}$
    Update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \hat{\boldsymbol{g}}$
return $x_0 + g(\boldsymbol{\theta}_T)\boldsymbol{\theta}_T$

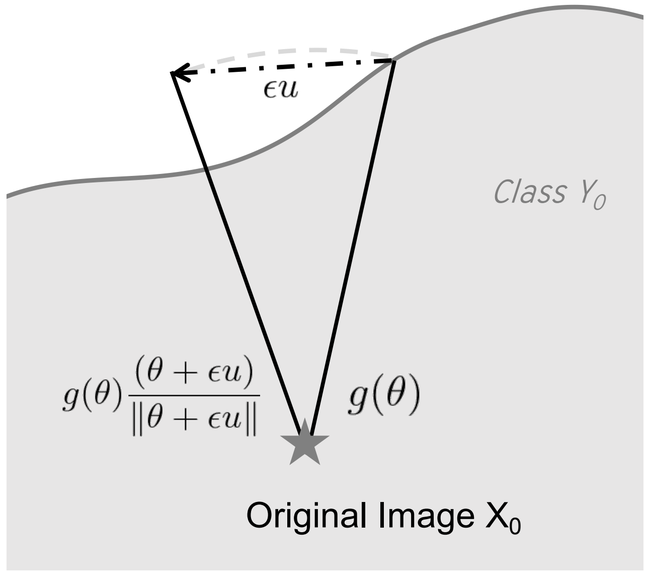
- $g(\theta_t)$ and $g(\theta_t + \beta u)$ in the gradient estimation takes most of queries, how to further reduce it?

Sign is enough!

- Binary search to estimate $g(\theta)$ in the gradient estimation takes most of queries.
- Gradient sign is powerful ! (FGSM)
- How to get the gradient sign efficiently ?

Single query oracle

- $\text{sign}(g(\theta+\epsilon u)-g(\theta)) = \begin{cases} +1, & f(x_0+g(\theta)\frac{(\theta+\epsilon u)}{\|\theta+\epsilon u\|}) = y_0, \\ -1, & \text{Otherwise.} \end{cases}$



Sign-OPT attack

Algorithm 3 Sign-OPT attack (ICLR '20)

Input: Hard-label model f, original image x_0, initial $\boldsymbol{\theta}_0$
for $t = 1, 2, \ldots, T$ **do**
  Randomly sample $u_1, \ldots, u_Q$ from a Gaussian or Uniform distribution
  Evaluate $g(\boldsymbol{\theta}_t)$
  $\hat{\boldsymbol{g}} = \frac{g(\boldsymbol{\theta}_t+\beta u)-g(\boldsymbol{\theta}_t)}{\beta} \cdot u \Rightarrow \text{sign}(\frac{g(\boldsymbol{\theta}_t+\beta u)-g(\boldsymbol{\theta}_t)}{\beta}) \cdot u$
  Update $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\hat{\boldsymbol{g}}$
  Evaluate $g(\boldsymbol{\theta}_t)$ using the same search algorithm

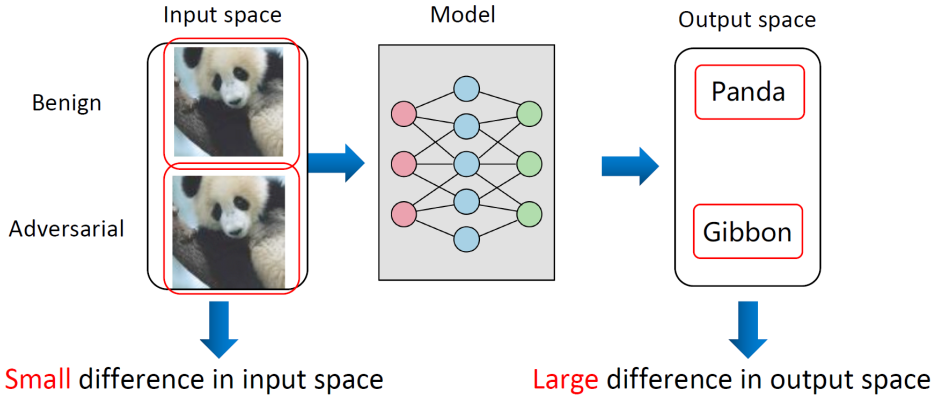
Transferability of Adversarial Examples

a.k.a Adversarial Transferability

Zeyu Qin

HKUST, AI Safety Group

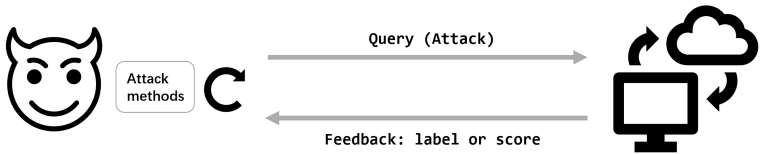
Adversarial Examples



- Imperceptible: $\|\boldsymbol{x}_{adv} - \boldsymbol{x}\|_p \leq \epsilon$
- Misclassified: $\boldsymbol{y} \neq Prediction(\boldsymbol{x}_{adv})$

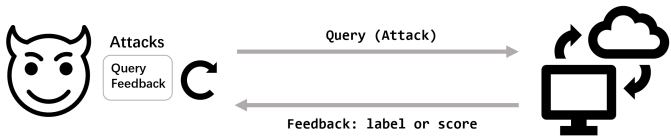
Severe Threats from Black-box Attacks

- In real scenarios, the users always get access to the DNN applications by querying API.
- The adversary does not know any knowledge about the target model $\mathcal{M}_T$, such as parameters, the architecture, and possibly the dataset.



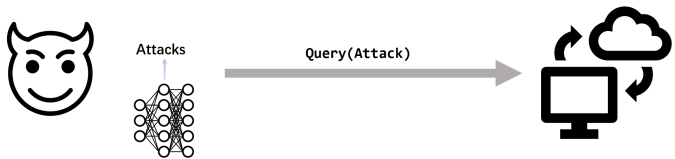
Black-box Attacks

- **Query-based attacks: only utilizing the feedback from $\mathcal{M}_T$**
  - Only know **the model feedback for each query (labels or confidence scores)** from $\mathcal{M}_T$.
  - By iteratively querying $\mathcal{M}_T$, the attackers generate $\boldsymbol{x}_{adv}$ based on exact feedback of each query.
  - It is more practical and only needs dozens of queries to generate the successful attacks.



Black-box Attacks

- **Transfer attacks: using $\boldsymbol{x}_{adv}$ from $\mathcal{M}_S$ to attack $\mathcal{M}_T$**
  - The attackers can **utilize same dataset to train the surrogate model $\mathcal{M}_S$**
  - Generating $\boldsymbol{x}_{adv}$ (white-box attacks) on $\mathcal{M}_S$, then attacking $\mathcal{M}_T$.
  - Don't need to iteratively query but it is not practical and performs poor attack performance.



How to generate adversarial examples (white-box attacks)

- generating attacks

$$\arg\min_{\boldsymbol{x}^{adv}} f(\boldsymbol{x}^{adv}) := \mathcal{L}(\mathcal{M}(\boldsymbol{x}^{adv}), \boldsymbol{y}_t),$$
$$s.t.\ \|\boldsymbol{x}^{adv} - \boldsymbol{x}\|_p \leq \epsilon, \quad \boldsymbol{x}^{adv} \in [0,1]$$

where p norm could be ℓ_∞, ℓ_2 or ℓ_1 norm. $\mathcal{L}$ could be CE loss, and $\mathcal{M}(\cdot)$ is Softmax output or logits (**returned feedback under black-box setting**).

Multi-step Projection Gradient Method (PGD) [1]:

$$\boldsymbol{x}^{adv} \leftarrow \mathrm{Proj}_{\mathcal{B}(\boldsymbol{x}^{adv})}\left(\boldsymbol{x} - \alpha \cdot \mathrm{sign}\left(\nabla\mathcal{L}\left(\mathcal{M}\left(\boldsymbol{x}^{adv}\right), \boldsymbol{y}_t\right)\right)\right)$$
$$\mathcal{B} = \left\{\boldsymbol{x}^{adv} \mid [0,1] \cap \left\|\boldsymbol{x}^{adv} - \boldsymbol{x}\right\|_\infty \leq \epsilon\right\}$$

[1]. Alexey Kurakin, Ian Goodfellow, Samy Bengio, "Adversarial examples in the physical world", Arxiv 2017

Transfer attacks

- Taking the targeted attack as an example, the general formulation of many existing transfer attack methods can be written as follows:

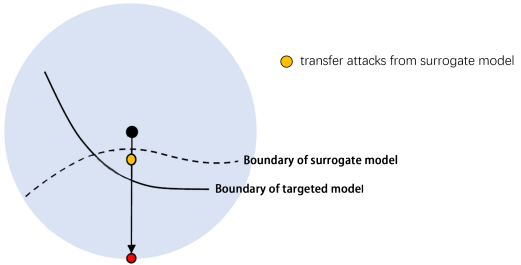
$$\min_{\boldsymbol{x}^{adv} \in \mathcal{B}_\epsilon(\boldsymbol{x})} \mathcal{L}(\mathcal{M}^s(\boldsymbol{x}^{adv}; \boldsymbol{\theta}), y_t). \tag{1}$$

  where $\mathcal{L}$ is the adversarial loss function computed on **surrogate model** $\mathcal{M}^S$, $\boldsymbol{y}_t$ is target label.

- Then, the attackers use $\boldsymbol{x}^{adv}$ from $\mathcal{M}^S$ to attack the target model, $\mathcal{M}^T$.

Transfer attacks overfits to $\mathcal{M}_S$

- The existing transfer attack methods exhibit poor transferability on $\mathcal{M}_T$ (not successfully attacking $\mathcal{M}_T$)
- $\boldsymbol{x}_{adv}$ severely depends on (overfits to) the decision boundaries of $\mathcal{M}_S$ and there are huge differences of decision boundaries between $\mathcal{M}_S$ and $\mathcal{M}_T$. [1,2]



[1] Tramer et al., Ensemble Adversarial Training: Attacks and Defenses, ICLR 2018.
[2] Demontis et al., Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks, ACM CCS 2019.

How can we improve transfer attack performance?

- Remember our goal: to improve $\boldsymbol{x}_{adv}$ on unseen models (but trained on the same training dataset, like ImageNet). ('Sample' Generalization)
- It seems like the other important problem: to improve M on unseen data. (Model Generalization)
- Sample Generalization $\Leftrightarrow$ Model Generalization ?

Inspiration from Improving Model Generalization

- You could borrow some **advanced optimization methods** to generate $\boldsymbol{x}_{adv}$, like momentum [1,3], and variance reduction [4].
- Strong Augmentations: like crop and resizing [2] and Mixup [5].
- Unseen models: we could use much more diverse models during attack generation: model ensembling [6]
  - More diverse architectures
  - Bayesian model sampling (adding noise on parameters of M_s)

[1] Dong et al., Boosting Adversarial Attacks with Momentum, CVPR 2018.
[2] Xie et al., Improving transferability of adversarial examples with input diversity, CVPR 2019.
[3] Lin et al., Nesterov Accelerated Gradient and Scale Invariance for Adversarial Attacks, ICLR 2020.
[4] Wang et al., Enhancing the Transferability of Adversarial Attacks through Variance Tuning, CVPR 2021.
[5] Xiaosen Wang et al., Admix: Enhancing the Transferability of Adversarial Attacks, ICCV 2021.
[6] Tramer et al., Ensemble Adversarial Training: Attacks and Defenses, ICLR 2018.

Poor targeted attack performance

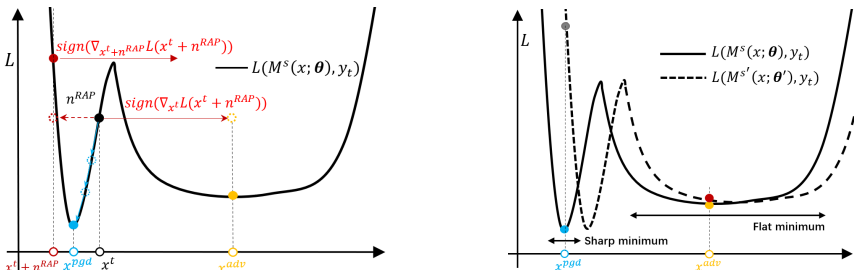
Figure from the recent NeurIPS 2021 paper:

Table 1: Targeted transfer success rates (%) in the single-model transfer setting. We consider three attacks with different loss functions: cross-entropy (CE), Poincaré distance with Triplet loss (Po+Trip) [21], and the logit loss. Results with 20/100/300 iterations are reported.

Attack	Source Model: Res50			Source Model: Dense121		
	→Dense121	→VGG16	→Inc-v3	→Res50	→VGG16	→Inc-v3
CE	26.9/39.4/42.6	17.3/27.3/30.4	2.4/3.8/4.1	13.1/17.3/19.4	7.7/10.8/10.9	1.9/3.3/3.5
Po+Trip	26.7/53.0/54.7	18.8/34.2/34.4	2.9/6.0/5.9	10.1/14.7/14.7	6.7/8.3/7.7	2.1/3.0/2.7
Logit	**29.3/63.3/72.5**	**24.0/55.7/62.7**	**3.0/7.2/9.4**	**17.2/39.7/43.7**	**13.5/35.3/38.7**	**2.7/6.9/7.6**
Attack	Source Model: VGG16			Source Model: Inc-v3		
	→Res50	→Dense121	→Inc-v3	→Res50	→Dense121	→VGG16
CE	0.7/0.4/0.6	0.5/0.3/0.1	0/0.1/0	0.6/**2.1**/2.4	0.8/2.5/2.9	0.7/1.6/2.0
Po+Trip	0.6/0.8/0.5	0.6/0.6/0.7	0.2/0.1/0.1	0.6/2.0/2.5	0.8/**3.1**/3.3	0.5/2.1/2.0
Logit	**3.3/8.7/11.2**	**3.6/11.7/13.2**	**0.2/0.7/0.9**	**0.8**/1.6/**2.9**	**1.2**/2.8/**5.3**	**0.7/2.2/3.7**

Further Improving Transferability

- Qin et al., propose a new perspective to interpret the adversarial transferability, the flatness of (adversarial) loss landscape of $\boldsymbol{x}^{adv}$ on $\mathcal{M}^S$.
- The $\boldsymbol{x}^{adv}$ located at the flat local minimum is less sensitive to the changes of decision boundary (the difference of $\mathcal{M}^S$ and $\mathcal{M}^T$). Therefore, it could have better adversarial transferability.



[1] Qin et al., [1] Boosting the Transferability of Adversarial Attacks with Reverse Adversarial Perturbation, NeurIPS 2022.

Finding x_{adv} located at a local flat region

- Qin et al., encourage that **not only x_{adv} itself has low loss value, but also the points in the vicinity of x_{adv} have similarly low loss values**.
- Qin et al., proposes to **minimize the maximal loss value within a local neighborhood region** around x_{adv}.
- The maximal loss is implemented by perturbing x_{adv} (adding perturbation n) to maximize the adversarial loss, named **Reverse Adversarial Perturbation (RAP)**. So, we aim to solve this problem,

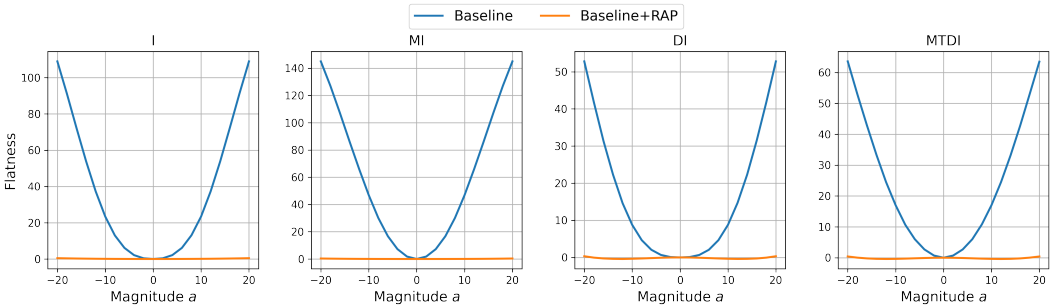
$$\min_{\boldsymbol{x}_{adv} \in \mathcal{B}_\epsilon(\boldsymbol{x})} \mathcal{L}\left(\mathcal{M}_S\left(\boldsymbol{x}_{adv} + \boldsymbol{n}_{adv}; \boldsymbol{\theta}\right), y_t\right)$$

Where,

$$\boldsymbol{n}_{adv} = \arg\max_{\|\boldsymbol{n}\|_\infty \leq \epsilon_n} \mathcal{L}\left(\mathcal{M}_S\left(\boldsymbol{x}_{adv} + \boldsymbol{n}; \boldsymbol{\theta}\right), y_t\right)$$

A Closer Look at RAP

- First we **visualize the loss landscape around $\boldsymbol{x}^{adv}$ on $\mathcal{M}^S$** by plotting the loss variations. We can observe that RAP could help find $\boldsymbol{x}^{adv}$ located at the flat region.



RAP achieves the better attack performance

- Combined with existing attacks, RAP **further boosts their transferability for both untargeted and targeted attacks.**
  The below tables show the transfer targeted attack performance ($\mathcal{M}^S \Longrightarrow \mathcal{M}^T$).

Attack	**ResNet-50** $\Longrightarrow$			**DenseNet-121** $\Longrightarrow$		
	Dense-121	VGG-16	Inc-v3	Res-50	VGG-16	Inc-v3
MTDI / +RAP / +RAP-LS	74.9 / 78.2 / **88.5**	62.8 / 72.9 / **81.5**	10.9 / 28.3 / **33.2**	44.9 / 64.3 / **74.5**	38.5 / 55.0 / **65.5**	7.7 / 23.0 / **26.5**
MTDSI / +RAP / +RAP-LS	86.3 / 88.4 / **93.3**	70.1 / 77.7 / **84.7**	38.1 / 51.8 / **58.0**	55.0 / 71.2 / **75.8**	42.0 / 58.4 / **62.3**	19.8 / 39.0 / **39.2**
MTDAI / +RAP / +RAP-LS	91.4 / 89.4 / **93.6**	79.9 / 79.0 / **86.3**	50.8 / 57.1 / **64.1**	69.1 / 74.2 / **82.1**	54.7 / 63.1 / **69.3**	32.0 / 43.5 / **49.3**

Attack	**VGG-16** $\Longrightarrow$			**Inc-v3** $\Longrightarrow$		
	Res-50	Dense-121	Inc-v3	Res-50	Dense-121	VGG-16
MTDI / +RAP / +RAP-LS	11.8 / 16.7 / **22.9**	13.7 / 19.4 / **27.4**	0.7 / 3.4 / **4.6**	1.8 / **8.3** / 7.5	4.1 / **14.8** / 13.4	2.9 / 8.0 / **9.8**
MTDSI / +RAP / +RAP-LS	31.0 / 35.3 / **38.7**	41.7 / 44.4 / **49.6**	9.6 / **15.2** / 13.7	5.6 / **11.9** / 10.7	10.4 / **21.2** / 20.9	4.2 / **8.9** / 8.6
MTDAI / +RAP / +RAP-LS	36.2 / 39.0 / **43.1**	48.0 / 45.1 / **55.2**	11.6 / 17.1 / **17.6**	9.6 / 13.6 / **16.7**	17.9 / 27.5 / **31.6**	8.4 / 12.0 / **12.1**

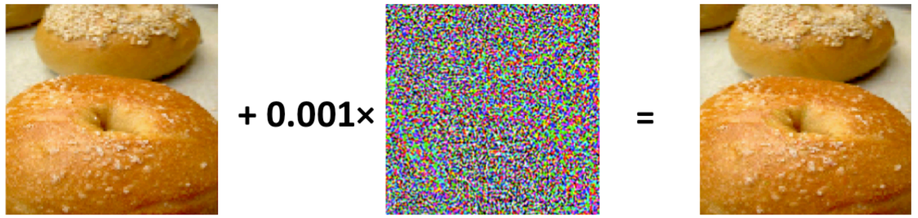
Adversarial Defense Methods

Adversarial Training

Test-time integrity

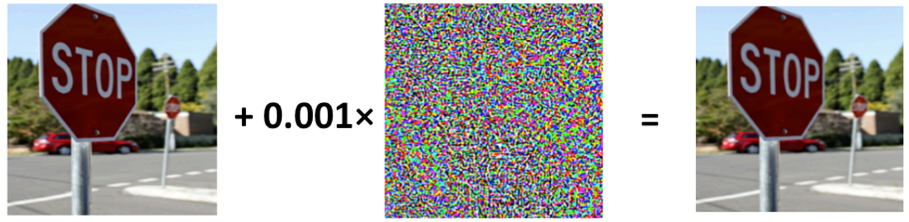
Adversarial examples

- An adversarial example can easily fool a deep network
- Robustness is critical in real systems



Bagle

piano



stop sign

speed limit 40

Adversarial example

White-box adversarial attack

- If there is $\|x - x_0\|_\infty$ constraint, we could turn to solve by
- FGSM attack [GSS15]:
  - $x \leftarrow \text{proj}_{x+\mathcal{S}}(x_0 + \alpha \text{sign}(\nabla_{x_0} \ell(\theta, x, y)))$
- PGD attack [KGB17, MMS18]
  - $x^{t+1} \leftarrow \text{proj}_{x+\mathcal{S}}(x^t + \alpha \text{sign}(\nabla_{x^t} \ell(\theta, x, y)))$

Adversarial defense

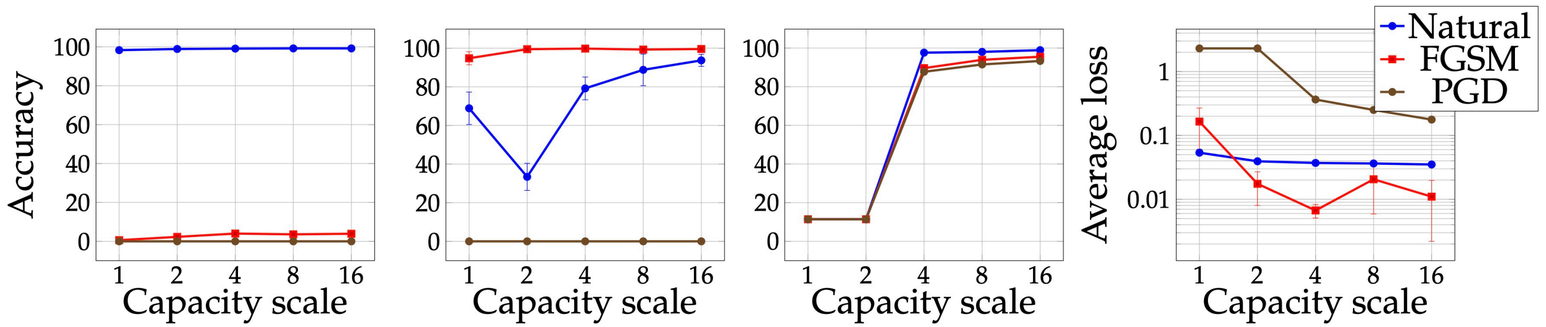
Adversarial training

- Adversarial training [MMS18]:
  - $$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y) \right] .$$
- Solve the inner loop by
  - $$x^{t+1} = \Pi_{x+\mathcal{S}} \left(x^t + \alpha \operatorname{sgn}(\nabla_x L(\theta, x, y)) \right)$$

Adversarial training

Capacity is crucial

MNIST



CIFAR10

	Simple	Wide
Natural	92.7%	95.2%
FGSM	27.5%	32.7%
PGD	0.8%	3.5%

(a) Standard training

	Simple	Wide
Natural	87.4%	90.3%
FGSM	90.9%	95.1%
PGD	0.0%	0.0%

(b) FGSM training

	Simple	Wide
Natural	79.4%	87.3%
FGSM	51.7%	56.1%
PGD	43.7%	45.8%

(c) PGD training

	Simple	Wide
Natural	0.00357	0.00371
FGSM	0.0115	0.00557
PGD	1.11	0.0218

(d) Training Loss

Figure 4: The effect of network capacity on the performance of the network. We trained MNIST and CIFAR10 networks of varying capacity on: (a) natural examples, (b) with FGSM-made adversarial examples, (c) with PGD-made adversarial examples. In the first three plots/tables of each dataset, we show how the standard and adversarial accuracy changes with respect to capacity for each training regime. In the final plot/table, we show the value of the cross-entropy loss on the adversarial examples the networks were trained on. This corresponds to the value of our saddle point formulation (2.1) for different sets of allowed perturbations.

Adversarial training

Problems

- Huge overhead
  - Increase training time by an order magnitude (7x if 7 step PGD)
- Fast method like FGSM doesn’t work
  - Easily be attacked by strong attackers such as C&W attack

Fast Adversarial training

Algorithm 3 FGSM adversarial training for T epochs, given some radius ϵ, N PGD steps, step size α, and a dataset of size M for a network f_θ

for $t = 1 \dots T$ **do**
  for $i = 1 \dots M$ **do**
    // Perform FGSM adversarial attack
    $\delta = \text{Uniform}(-\epsilon, \epsilon)$
    $\delta = \delta + \alpha \cdot \text{sign}(\nabla_\delta \ell(f_\theta(x_i + \delta), y_i))$
    $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
    $\theta = \theta - \nabla_\theta \ell(f_\theta(x_i + \delta), y_i)$ *// Update model weights with some optimizer, e.g. SGD*
  end for
end for

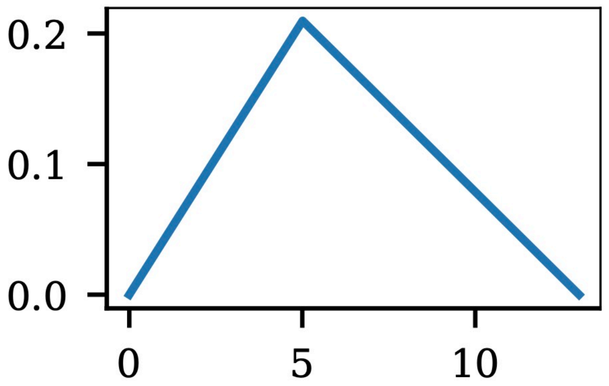
The magic of random initialization

Method	Standard accuracy	PGD ($\epsilon = 8/255$)	Time (min)
FGSM + DAWNBench			
+ zero init	85.18%	0.00%	12.37
+ early stopping	71.14%	38.86%	7.89
+ previous init	86.02%	42.37%	12.21
+ random init	85.32%	44.01%	12.33
+ $\alpha = 10/255$ step size	83.81%	46.06%	12.17
+ $\alpha = 16/255$ step size	86.05%	0.00%	12.06
+ early stopping	70.93%	40.38%	8.81
"Free" ($m = 8$) (Shafahi et al., 2019)[1]	85.96%	46.33%	785
+ DAWNBench	78.38%	46.18%	20.91
PGD-7 (Madry et al., 2017)[2]	87.30%	45.80%	4965.71
+ DAWNBench	82.46%	50.69%	68.8

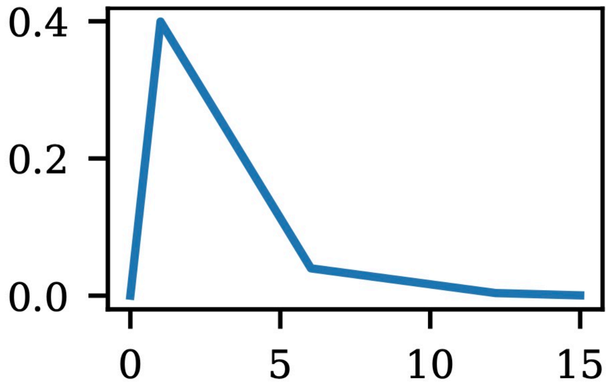
DAWNBench Improvement

Reduce # of training epochs

- Cyclic learning rate
- Mixed-precision arithmetic



(a) CIFAR10



(b) ImageNet

Catastrophic overfitting