CS480/680: Introduction to Machine Learning

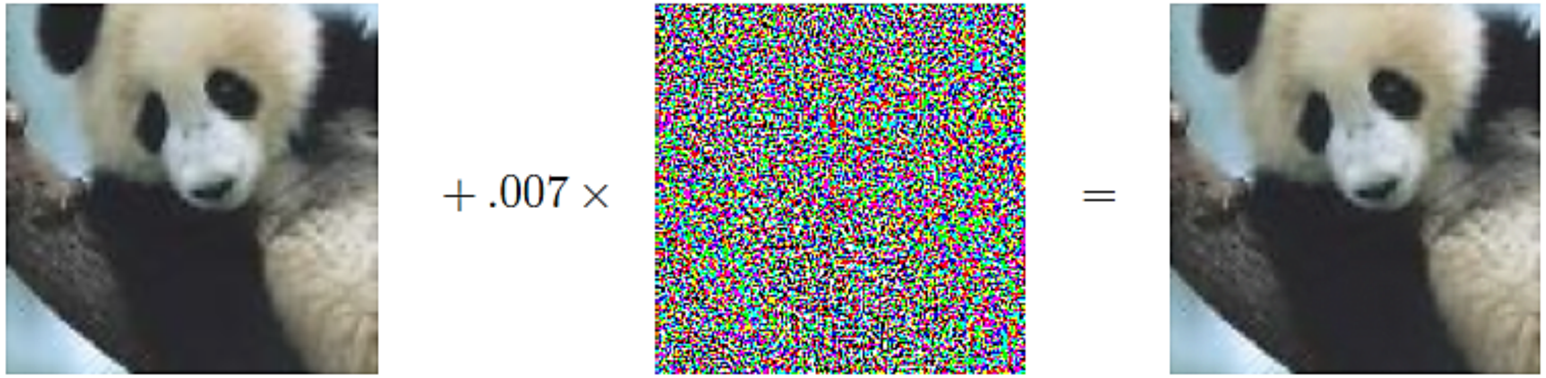
Lecture 15: Adversarial Attacks

Hongyang Zhang

UNIVERSITY OF WATERLOO

July 23, 2025

Adversarial attacks



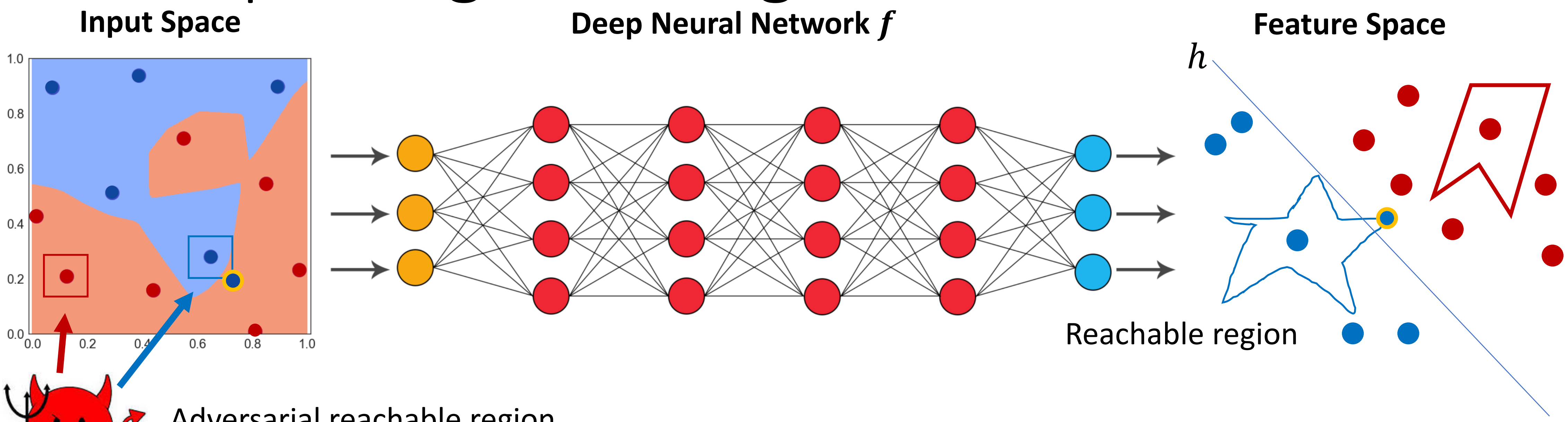
$\boldsymbol{x}$

"panda"
57.7% confidence

$x + \epsilon \cdot \text{sign}\left(\nabla_x \mathcal{L}(C(x, w), y)\right)$

"gibbon"
99.3 % confidence

Principle of generating adversarial attacks

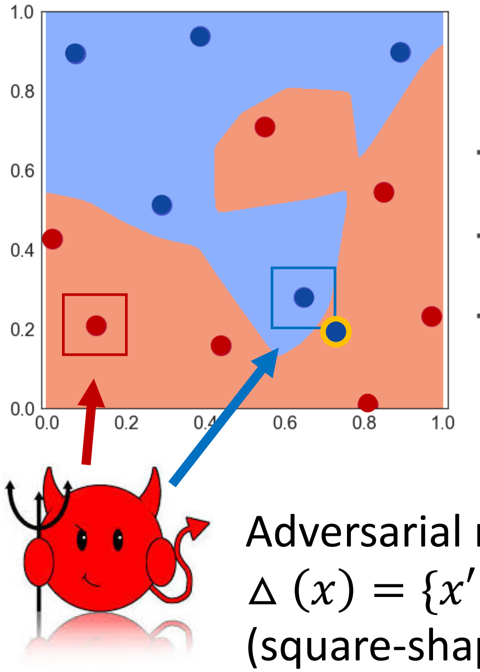


Adversarial reachable region
$\triangle(x) = \{x' : \|x' - x\|_\infty \leq \varepsilon\}$
(square-shaped, $\|\cdot\|_\infty$ is the maximum absolute value of all entries)

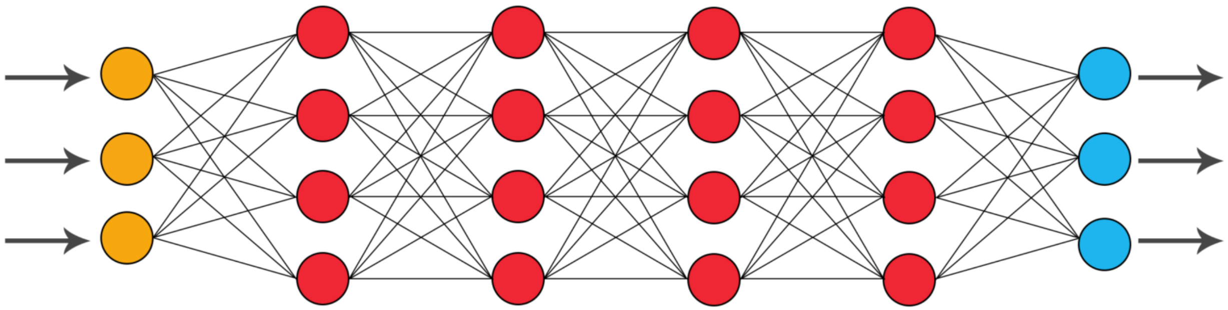
$$\max_{x_{adv} \in \triangle(x)} \mathcal{L}(h(f(x_{adv})), y)$$

Principle of generating adversarial attacks

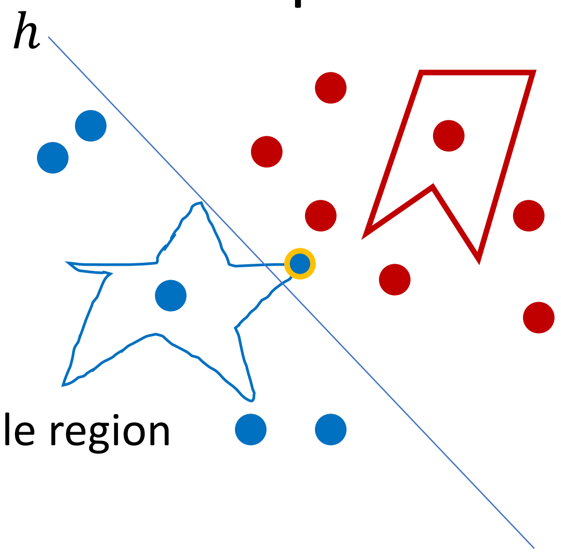
Input Space



Deep Neural Network f



Feature Space



Adversarial reachable region
$\triangle(x) = \{x' : \|x' - x\|_\infty \le \varepsilon\}$
(square-shaped, $\|\cdot\|_\infty$ is the maximum absolute value of all entries)

$$\max_{x_{adv} \in \triangle(x)} \mathcal{L}(C(x_{adv})), y)$$

Composition of h and f

Principle of generating adversarial attacks

- Then generating adversarial examples reduces to the problem of solving

$$\max_{\|x_{adv}-x\|_\infty \le \varepsilon} \mathcal{L}(C(x_{adv})), \mathrm{y})$$

- Different tools in optimizations
  - Zero-order solvers (only access to the output of NN)
    - Black-box attack
  - First-order solvers (access to gradient info, e.g., FGSM, BIM, PGD, CW attack, …)
    - White-box attack
    - Why white-box? Because calculating gradient requires full info about NN
  - Second-order solvers (access to Hessian matrix, e.g., L-BFGS attack)
    - White-box attack

Principle of generating adversarial attacks

- Then generating adversarial examples reduces to the problem of solving

$$\max_{\|x_{adv}-x\|_{\infty}\leq\varepsilon} \mathcal{L}(C(x_{adv})), \mathrm{y})$$

- Different tools in optimizations
  - Zero-order solvers (only access to the output of NN)
    - Black-box attack
  - First-order solvers (access to gradient info, e.g., FGSM, BIM, PGD, CW attack, …)
    - White-box attack
    - Why white-box? Because calculating gradient requires full info about NN
  - Second-order solvers (access to Hessian matrix, e.g., L-BFGS attack)
    - White-box attack

FGSM Attack

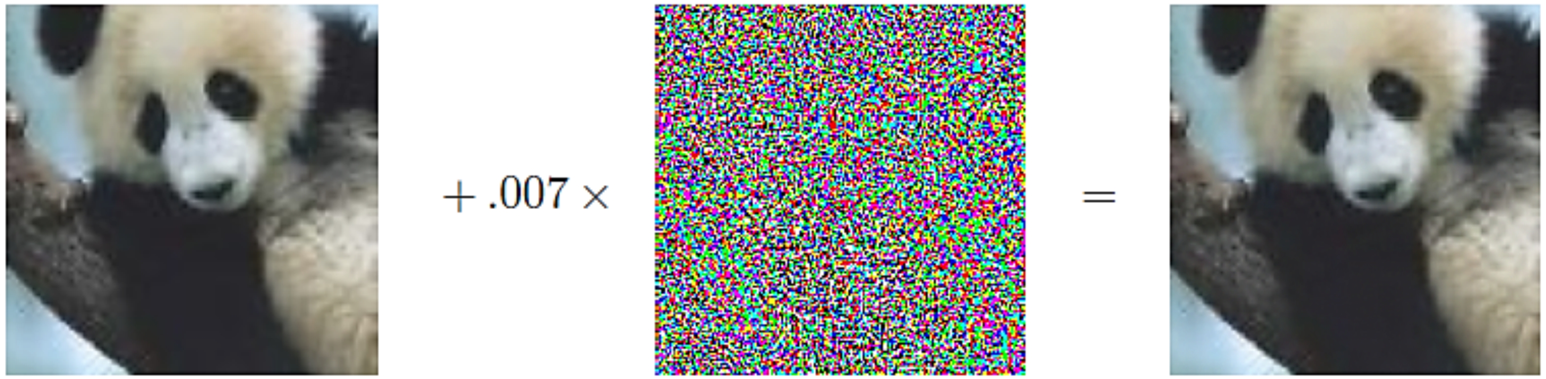
- ***Fast gradient sign method (FGSM) attack***
  - Goodfellow (2015) Explaining and Harnessing Adversarial Examples
- Recall our goal: $\max_{\|x_{adv}-x\|_\infty \le \varepsilon} \mathcal{L}(C(x_{adv})), \mathrm{y})$ (non-convex and hard to solve)
- Let us do linear expansion at x: $\mathcal{L}(C(x_{adv})), \mathrm{y}) \approx \mathcal{L}(C(x)), \mathrm{y}) + \langle x_{adv} - x, \nabla_x \mathcal{L}(C(x), y) \rangle$ (constant)
- So the problem then reduces to $\max_{\|x_{adv}-x\|_\infty \le \varepsilon} \langle x_{adv} - x, \nabla_x \mathcal{L}(C(x), y) \rangle$
- Closed-form solution: $x_{adv}^* = x + \epsilon \cdot \mathrm{sign}\big(\nabla_x \mathcal{L}(C(x), y)\big)$
  - Why?
  - Holder inequality: for any vector a, b, we have $\langle a, b \rangle \le \|a\|_p \|b\|_q$, where $\frac{1}{p} + \frac{1}{q} = 1$ and $p, q \ge 1$
    - $\|\cdot\|_p$ and $\|\cdot\|_q$ are also known as dual norms
    - Examples: $\|\cdot\|_2$ is self-dual, $\|\cdot\|_1$ and $\|\cdot\|_\infty$ are dual

FGSM Attack

- ***Fast gradient sign method (FGSM) attack***
  - [Goodfellow (2015) Explaining and Harnessing Adversarial Examples]()
- Recall our goal: $\max_{\|x_{adv}-x\|_\infty \le \varepsilon} \mathcal{L}(C(x_{adv})), \mathrm{y})$ (non-convex and hard to solve)
- Let us do linear expansion at x: $\mathcal{L}(C(x_{adv})), \mathrm{y}) \approx \mathcal{L}(C(x)), \mathrm{y}) + \langle x_{adv} - x, \nabla_x \mathcal{L}(C(x), y) \rangle$ (constant)
- So the problem then reduces to $\max_{\|x_{adv}-x\|_\infty \le \varepsilon} \langle x_{adv} - x, \nabla_x \mathcal{L}(C(x), y) \rangle$
- Closed-form solution: $x_{adv}^* = x + \epsilon \cdot \mathrm{sign}\big(\nabla_x \mathcal{L}(C(x), y)\big)$ Named FGSM attack
  - Why?
  - $\mathrm{Obj}(x_{adv}) = \langle x_{adv} - x, \nabla_x \mathcal{L}(C(x), y) \rangle \le \|x_{adv} - x\|_\infty \|\nabla_x \mathcal{L}(C(x), y)\|_1 \le \epsilon \|\nabla_x \mathcal{L}(C(x), y)\|_1$ (by Holder inequality)
  - On the other hand, the above solution achieves the upper bound and satisfies the constraint
  - This finishes the proof

Facts about FGSM Attack

- FGSM is a white-box, non-targeted adversarial attack
  - White-box: we need to calculate $\nabla_x \mathcal{L}(C(x), y)$ to create the adversarial image
    - FGSM calculates the gradient only once
  - Non-targeted: the attacker aims to maximize the loss w.r.t. the true label



x | $\text{sign}(\nabla_x \mathcal{L}(C(x, w), y))$ | $x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(C(x), y))$

"panda"
57.7% confidence

"gibbon"
99.3 % confidence

Intuition behind using sign operator?

- Recall that FGSM creates an adversarial image x_{adv} by

$$x_{adv} = x + \epsilon \cdot \text{sign}\left(\nabla_x \mathcal{L}(C(x), y)\right)$$

  - We have proven that it is the closed-form solution of an optimization problem

- Intuition behind using sign operator:
  - Remove the imbalance in the update when the gradient on one pixel is much larger
  - The method automatically reaches the boundary of adversarial reachable region for all pixels $\triangle(x) = \{x' : \|x' - x\|_\infty \leq \varepsilon\}$ (thus, it uses the full power of adversarial budget)
  - Better empirical attack success rate in experiments

Issues with FGSM Attack

- Sometimes, FGSM requires large ϵ in order to succeed (human-perceptible)

Original image

Prediction: car mirror

Adversarial image

Many artifacts

Prediction: sunglasses

Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

BIM Attack

- ***Basic iterative method (BIM) attack***
  - [Kurakin (2017) Adversarial Examples in the Physical World]
- BIM is a variant of FGSM: it repeatedly adds noise to the image *x* in multiple iterations, in order to cause misclassification
  - Let t be the index of iterations, and γ be the step size. BIM is given by

$$x^t = x^{t-1} + \gamma \cdot \text{sign}\left(\nabla_x \mathcal{L}(C(x^{t-1}), y)\right)$$

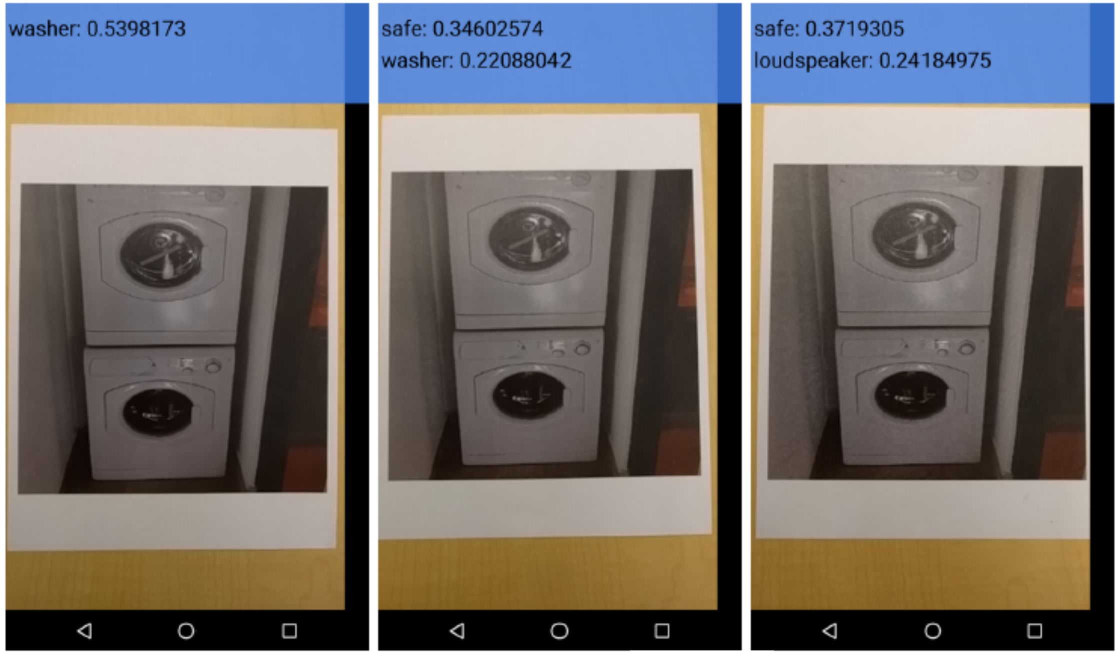
  - Compare with FGSM

$$x_{adv} = x + \epsilon \cdot \text{sign}\left(\nabla_x \mathcal{L}(C(x), y)\right)$$

    - Step size is different
    - BIM uses an iterative procedure while FGSM uses a one-shot procedure

BIM Attack

- Example of BIM attack on the printed image of a washer
- By repeating $x^t = x^{t-1} + \gamma \cdot \text{sign}\left(\nabla_x \mathcal{L}(C(x^{t-1}), y)\right)$, the perturbation size ϵ will become larger and larger



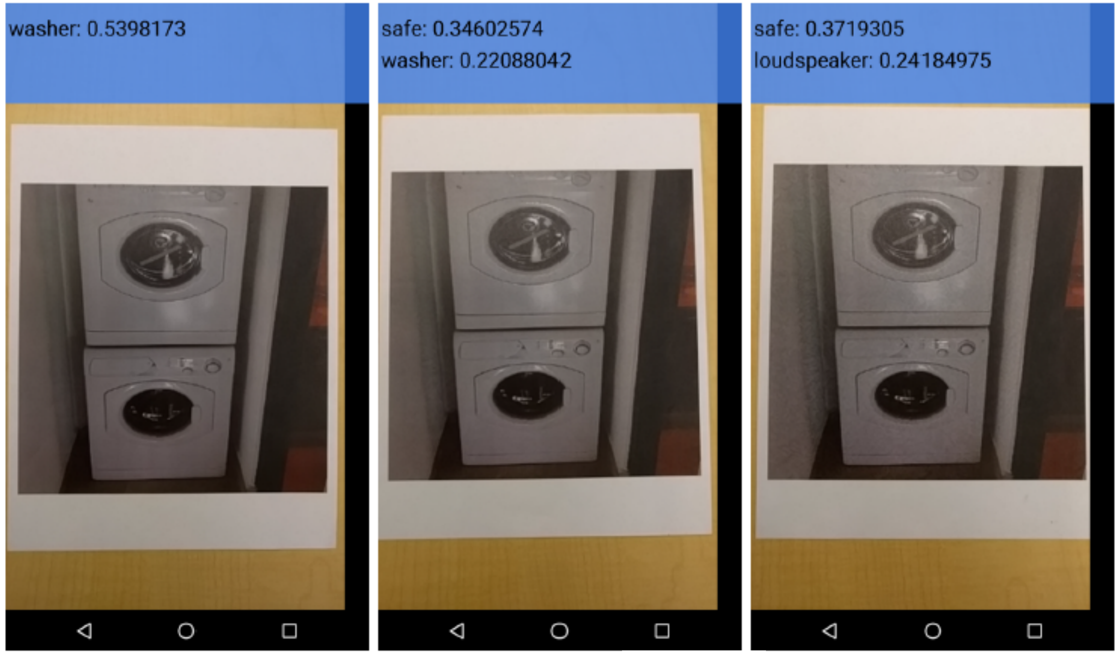
(b) Clean image

(c) Adv. image, Distance 4

(d) Adv. image, Distance 8

Issues with BIM Attack

- Example of BIM attack on the printed image of a washer
- By repeating $x^t = x^{t-1} + \gamma \cdot \mathrm{sign}\big(\nabla_x \mathcal{L}(C(x^{t-1}), y)\big)$, the perturbation size ϵ will become larger and larger
- For a pre-defined ε, x^t may violate the constraint $\|x' - x\|_\infty \leq \varepsilon$ when t is large



(b) Clean image (c) Adv. image, Distance 4 (d) Adv. image, Distance 8

PGD Attack

- ***Projected gradient descent (PGD) attack***
  - Madry (2017) Towards Deep Learning Models Resistant to Adversarial Attacks
- To resolve the issue of BIM, PGD involves a truncation operation:

$$x^t = \text{clip}_{(-\epsilon,\epsilon)}\left(x^{t-1} + \gamma \cdot \text{sign}\left(\nabla_x \mathcal{L}(C(x^{t-1}), y)\right)\right)$$

  - That is, for those pixels with perturbation size larger than ϵ, "clip" truncates it to ϵ
- Another difference from BIM: PGD uses random initialization for x^0, by adding random noise to the original image from a uniform distribution in the range $(-\epsilon, \epsilon)$

PGD Attack

- PGD attack example

Original image

Prediction: baboon

Adversarial image

Prediction: Egyptian cat

Egyptian cat

Fewer artifacts than FGSM

Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

Facts about PGD Attack

- PGD is a white-box, non-targeted adversarial attack
  - White-box, since we need to know the gradients $\nabla_x \mathcal{L}(C(x), y)$ of the model to create the adversarial image
    - PGD calculates the gradient multiple times
  - Non-targeted, since PGD aims to maximize the loss w.r.t. the true label

Targeted PGD Attack

- Gradient approaches (FGSM, BIM, PGD) can also be designed as targeted white-box attacks
  - In this case, the added perturbation noise aims to minimize the loss function of the image for a specific target class
  - But how?

Original image

Prediction: hippopotamus

Adversarial image

Prediction: maraca

Maraca

Picture from: https://blog.floydhub.com/introduction-to-adversarial-machine-learning/

Comparison between Untargeted and Targeted Attacks

Gradient Ascent

- Untargeted objective: $\max_{x_{adv} \in \Delta(x)} \mathcal{L}(C(x_{adv}), y_{\text{true}})$
- Targeted objective: $\min_{x_{adv} \in \Delta(x)} \mathcal{L}(C(x_{adv}), y_{\text{target}})$

Gradient Descent

- Untargeted iteration: $x_{adv}^{t} = \text{clip}_{(-\epsilon,\epsilon)}\left(x^{t-1} + \gamma \cdot \text{sign}(\nabla_x \mathcal{L}(C(x^{t-1}), y_{\text{true}}))\right)$
  - It is based on maximizing the loss function for the true class
- Targeted iteration: $x_{adv}^{t} = \text{clip}_{(-\epsilon,\epsilon)}\left(x^{t-1} - \gamma \cdot \text{sign}\left(\nabla_x \mathcal{L}\left(F(x^{t-1}), y_{\text{target}}\right)\right)\right)$
  - It is based on minimizing the loss function for the target class

Unrestricted Adversarial Examples

- Most works investigated the generation of adversarial examples that are constrained to lie in the neighborhood of clean samples
  - E.g., L_p norm bounded perturbation
  - Such constraints ensure that the adversarial examples are human-imperceptible
  - Such examples are sometimes referred to as restricted adversarial examples
- ***Unrestricted adversarial examples*** are generated without considering any bounds or constraints on the modifications of clean inputs
  - As long as the adversarial examples are human-imperceptible
  - Challenging, because it is mathematically hard to define "human-imperceptible"

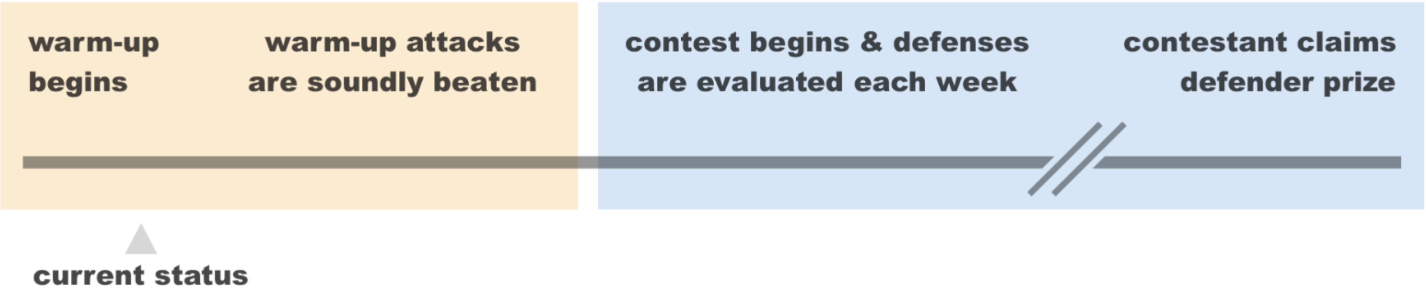
Unrestricted Adversarial Examples Challenge

Unrestricted Adversarial Examples Challenge build passing

In the Unrestricted Adversarial Examples Challenge, attackers submit arbitrary adversarial inputs, and defenders are expected to assign low confidence to difficult inputs while retaining high confidence and accuracy on a clean, unambiguous test set. You can learn more about the motivation and structure of the contest in our recent paper

This repository contains code for the warm-up to the challenge, as well as the public proposal for the contest. We are currently accepting defenses for the warm-up.

Warm-up & Contest Timeline



Unrestricted Adversarial Examples Challenge

The class
of bicycle

The class
of bird

Unrestricted Adversarial Examples Challenge

Clean image:

Defense	Submitted by	Clean data	Common corruptions	Spatial grid attack	SPSA attack	Boundary attack	Submission Date
Pytorch ResNet50 (trained on bird-or-bicycle extras)	TRADES	100.0%	100.0%	99.5%	100.0%	95.0%	Jan 17th, 2019 (EST)
Keras ResNet (trained on ImageNet)	Google Brain	100.0%	99.2%	92.2%	1.6%	4.0%	Sept 29th, 2018
Pytorch ResNet (trained on bird-or-bicycle extras)	Google Brain	98.8%	74.6%	49.5%	2.5%	8.0%	Oct 1st, 2018

Unrestricted Adversarial Examples Challenge

Clean image:

Corrupted image:

Defense	Submitted by	Clean data	Common corruptions	Spatial grid attack	SPSA attack	Boundary attack	Submission Date
Pytorch ResNet50 (trained on bird-or-bicycle extras)	TRADES	100.0%	100.0%	99.5%	100.0%	95.0%	Jan 17th, 2019 (EST)
Keras ResNet (trained on ImageNet)	Google Brain	100.0%	99.2%	92.2%	1.6%	4.0%	Sept 29th, 2018
Pytorch ResNet (trained on bird-or-bicycle extras)	Google Brain	98.8%	74.6%	49.5%	2.5%	8.0%	Oct 1st, 2018

Unrestricted Adversarial Examples Challenge

Clean image:

Corrupted image:

Defense	Submitted by	Clean data	Common corruptions	Spatial grid attack	SPSA attack	Boundary attack	Submission Date
Pytorch ResNet50 (trained on bird-or-bicycle extras)	TRADES	100.0%	100.0%	99.5%	100.0%	95.0%	Jan 17th, 2019 (EST)
Keras ResNet (trained on ImageNet)	Google Brain	100.0%	99.2%	92.2%	1.6%	4.0%	Sept 29th, 2018
Pytorch ResNet (trained on bird-or-bicycle extras)	Google Brain	98.8%	74.6%	49.5%	2.5%	8.0%	Oct 1st, 2018

Unrestricted Adversarial Examples Challenge

Clean image:

Corrupted image:

Defense	Submitted by	Clean data	Common corruptions	Spatial grid attack	SPSA attack	Boundary attack	Submission Date
Pytorch ResNet50 (trained on bird-or-bicycle extras)	TRADES	100.0%	100.0%	99.5%	100.0%	95.0%	Jan 17th, 2019 (EST)
Keras ResNet (trained on ImageNet)	Google Brain	100.0%	99.2%	92.2%	1.6%	4.0%	Sept 29th, 2018
Pytorch ResNet (trained on bird-or-bicycle extras)	Google Brain	98.8%	74.6%	49.5%	2.5%	8.0%	Oct 1st, 2018

Unrestricted Adversarial Examples Challenge

Clean image:

Corrupted image:

Defense	Submitted by	Clean data	Common corruptions	Spatial grid attack	SPSA attack	Boundary attack	Submission Date
Pytorch ResNet50 (trained on bird-or-bicycle extras)	TRADES	100.0%	100.0%	99.5%	100.0%	95.0%	Jan 17th, 2019 (EST)
Keras ResNet (trained on ImageNet)	Google Brain	100.0%	99.2%	92.2%	1.6%	4.0%	Sept 29th, 2018
Pytorch ResNet (trained on bird-or-bicycle extras)	Google Brain	98.8%	74.6%	49.5%	2.5%	8.0%	Oct 1st, 2018

List of Adversarial Evasion Attacks

Attack	Publication	Similarity	Attacking Capability	Algorithm	Apply Domain
L-BFGS	(Szegedy et al., 2013)	l_2	White-Box	Iterative	Image Classification
FGSM	(Goodfellow et al., 2014b)	l_∞,l_2	White-Box	Single-Step	Image Classification
Deepfool	(Moosavi-Dezfooli et al., 2016)	l_2	White-Box	Iterative	Image Classification
JSMA	(Papernot et al., 2016a)	l_2	White-Box	Iterative	Image Classification
BIM	(Kurakin et al., 2016a)	l_∞	White-Box	Iterative	Image Classification
C & W	(Carlini & Wagner, 2017b)	l_2	White-Box	Iterative	Image Classification
Ground Truth	(Carlini et al., 2017)	l_0	White-Box	SMT solver	Image Classification
Spatial	(Xiao et al., 2018b)	Total Variation	White-Box	Iterative	Image Classification
Universal	(Metzen et al., 2017b)	l_∞, l_2	White-Box	Iterative	Image Classification
One-Pixel	(Su et al., 2019)	l_0	White-Box	Iterative	Image Classification
EAD	(Chen et al., 2018)	$l_1 + l_2, l_2$	White-Box	Iterative	Image Classification
Substitute	(Papernot et al., 2017)	l_p	Black-Box	Iterative	Image Classification
ZOO	(Chen et al., 2017)	l_p	Black-Box	Iterative	Image Classification
Biggio	(Biggio et al., 2012)	l_2	Poisoning	Iterative	Image Classification
Explanation	(Koh & Liang, 2017)	l_p	Poisoning	Iterative	Image Classification
Zugner's	(Zügner et al., 2018)	Degree Distribution, Coocurrence	Poisoning	Greedy	Node Classification
Dai's	(Dai et al., 2018)	Edges	Black-Box	RL	Node & Graph Classification
Meta	(Zügner & Günnemann, 2019)	Edges	Black-Box	RL	Node Classification
C & W	(Carlini & Wagner, 2018)	max dB	White-Box	Iterative	Speech Recognition
Word Embedding	(Miyato et al., 2016)	l_p	White-Box	One-Step	Text Classification
HotFlip	(Ebrahimi et al., 2017)	letters	White-Box	Greedy	Text Classification
Jia & Liang	(Jia & Liang, 2017)	letters	Black-Box	Greedy	Reading Comprehension
Face Recognition	(Sharif et al., 2016)	physical	White-Box	Iterative	Face Recognition
RL attack	(Huang et al., 2017)	l_p	White-Box	RL	