

# Adversarial Attacks and Defenses

CPSC680: Trustworthy Deep Learning

Rex Ying

# Readings

- Readings are updated on the website (syllabus page)
- **Readings:**
  - [Adversarial Examples: Attacks and Defenses for Deep Learning](#)
  - [Review of Adversarial Attacks and Defense](#)

# Content

- Introduction to Adversarial Attack
- Adversarial Attack Types
- Evasion Attack and Defense
- Poisoning Attack and Defense
- Exploratory Attack and Defense

# Content

- Introduction to Adversarial Attack
- Adversarial Attack Types
- Evasion Attack and Defense
- Poisoning Attack and Defense
- Exploratory Attack and Defense

# Deep Learning Performance

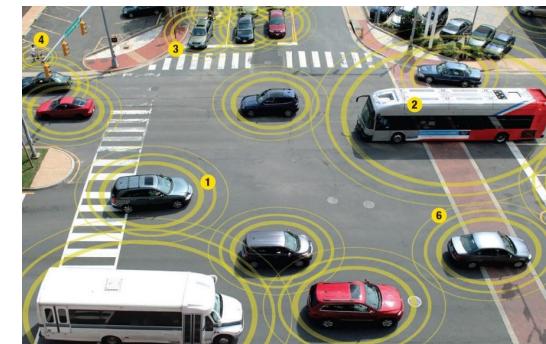
- Recent years have seen **impressive performance of deep learning models in a variety of applications.**
  - Examples: ResNet, AlphaGo, BERT, GPT, AlphaFold ...
- **Are these models ready to be deployed in real world?**



Autonomous Driving



Malware Detection



Smart Transportation

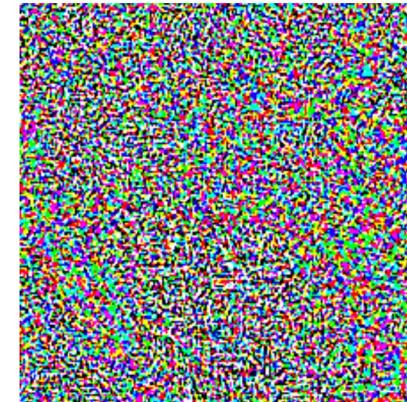
# Adversarial Examples

- Deep convolutional neural networks are vulnerable to **adversarial attacks**:
  - **Imperceptible noise** changes the original prediction.



“Panda”  
57.7% confidence

$+ .007 \times$



Carefully  
calculated noise

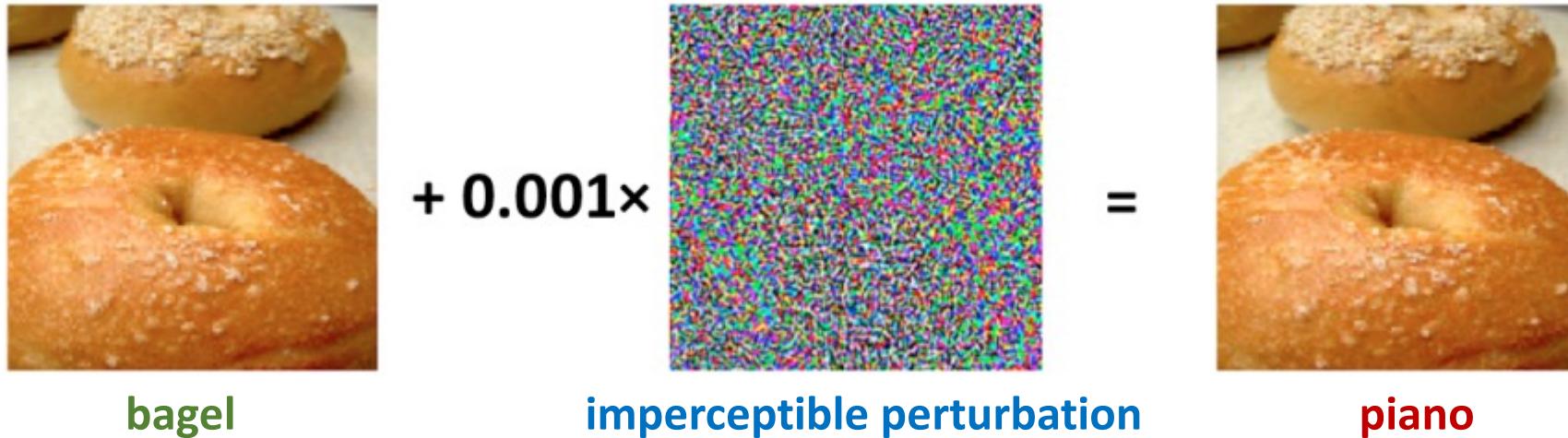


“Gibbon”  
93.3% confidence

Goodfellow, I., Shlens, J., & Szegedy, C. "Explaining and Harnessing Adversarial Examples."

# Adversarial Examples

- Two **key characteristics of adversarial attacks**
  - The adversarial examples make changes **imperceptible to human**
  - The prediction results are dramatically altered

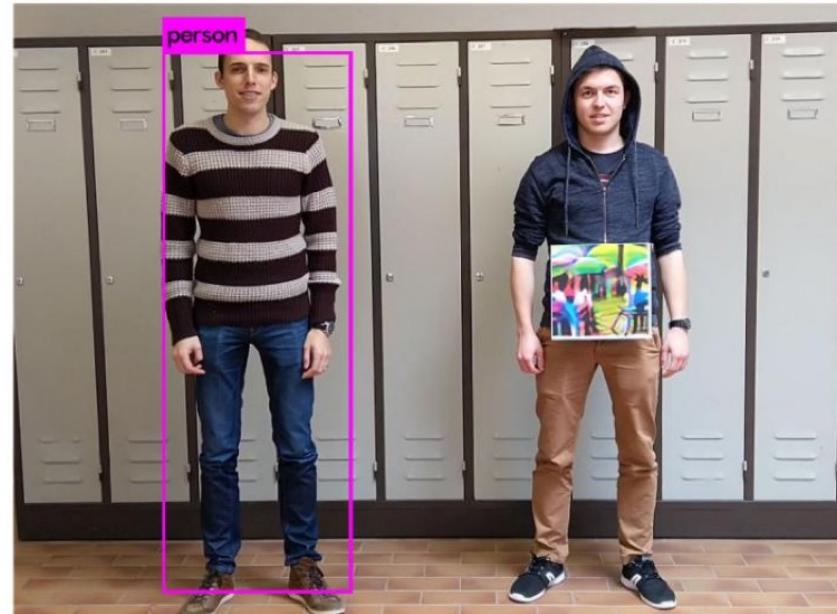


Li, Yao, et al. "A Review of Adversarial Attack and Defense for Classification Methods."

- Adversarial examples are also reported in natural language, graph-structured data, medical image, etc.

# Adversarial Examples in Object Detection

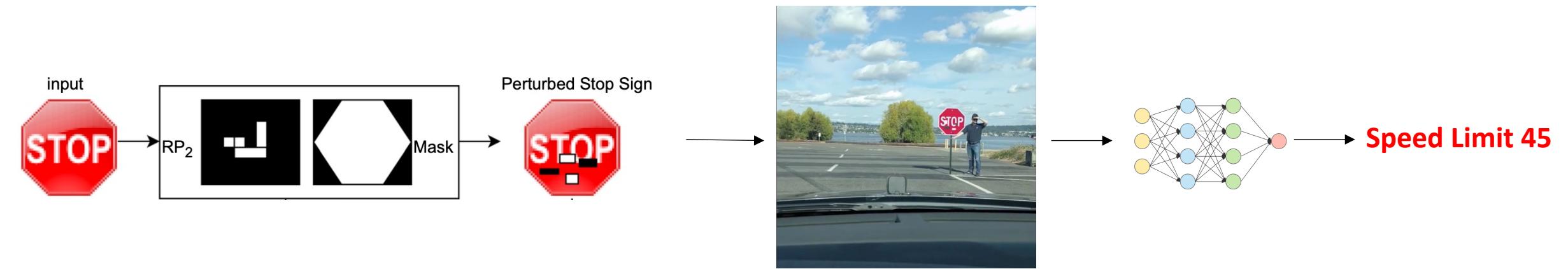
- A person wearing an adversarial patch is not detected by a person detector model using YOLOv2.
  - The adversaries can use it to bypass the surveillance cameras.



Thys et al. “Fooling automated surveillance cameras: adversarial patches to attack person detection”

# Adversarial Examples in Object Detection

- An adversary adds adversarial patches into the **STOP** sign that causes the DL model of a self-driving car to misclassify it as a **Speed Limit 45** sign



Eykholt et al. "Robust Physical-World Attacks on Deep Learning Visual Classification."

# Adversarial Examples in Natural Language

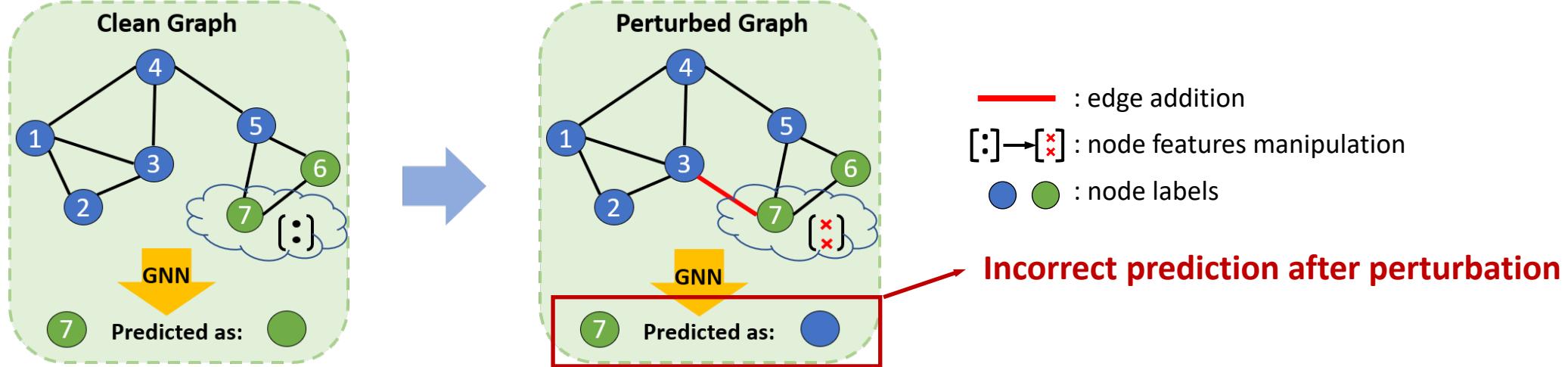
## Examples of sentiment analysis:

<b>Original Input</b>	Connoisseurs of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: <b>Positive (77%)</b>
<b>Adversarial example [Visually similar]</b>	<u>Onnoisseurs</u> of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: <b>Negative (52%)</b>
<b>Adversarial example [Semantically similar]</b>	Connoisseurs of Chinese <u>footage</u> will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: <b>Negative (54%)</b>

Morris, John X., et al. "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp."

- Adversarial examples are very “**similar**” to the original input:
  - **Visually similar**: a few characters change away; even just “typos”
  - **Semantically similar**: semantically indistinguishable from the original input; paraphrase of original input
- **Similar** adversarial examples cause **different** prediction from the model

# Adversarial Examples in Graphs

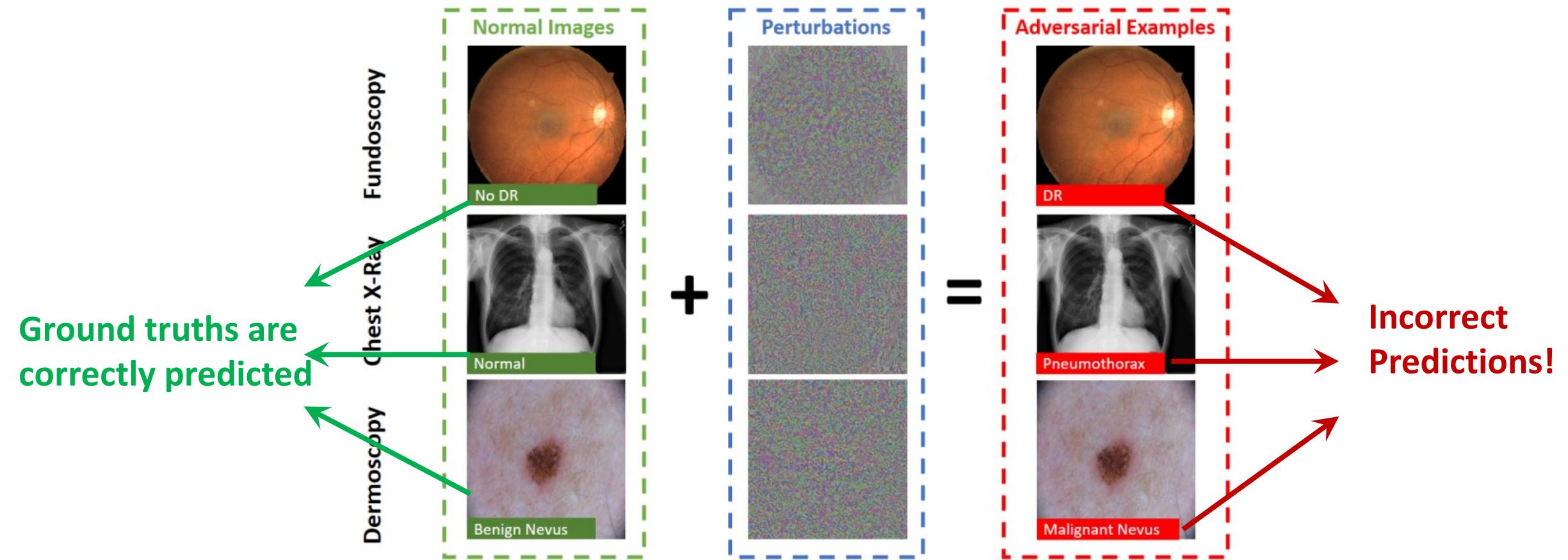


Jin, Wei, et al. "Adversarial Attacks and Defenses on Graphs: A Review, A Tool and Empirical Studies."

- Edge-level perturbation:
  - adding / removing / rewiring edges / manipulating the edge attributes
- Node-level perturbation:
  - adding / removing nodes / manipulating the features of target nodes
- Cause **dramatically different** results in node-level/edge-level/graph-level tasks

# Adversarial Examples – Medical Applications

- medical dataset [Fundoscopy](#), [Chest X-Ray](#), [Dermoscopy](#)



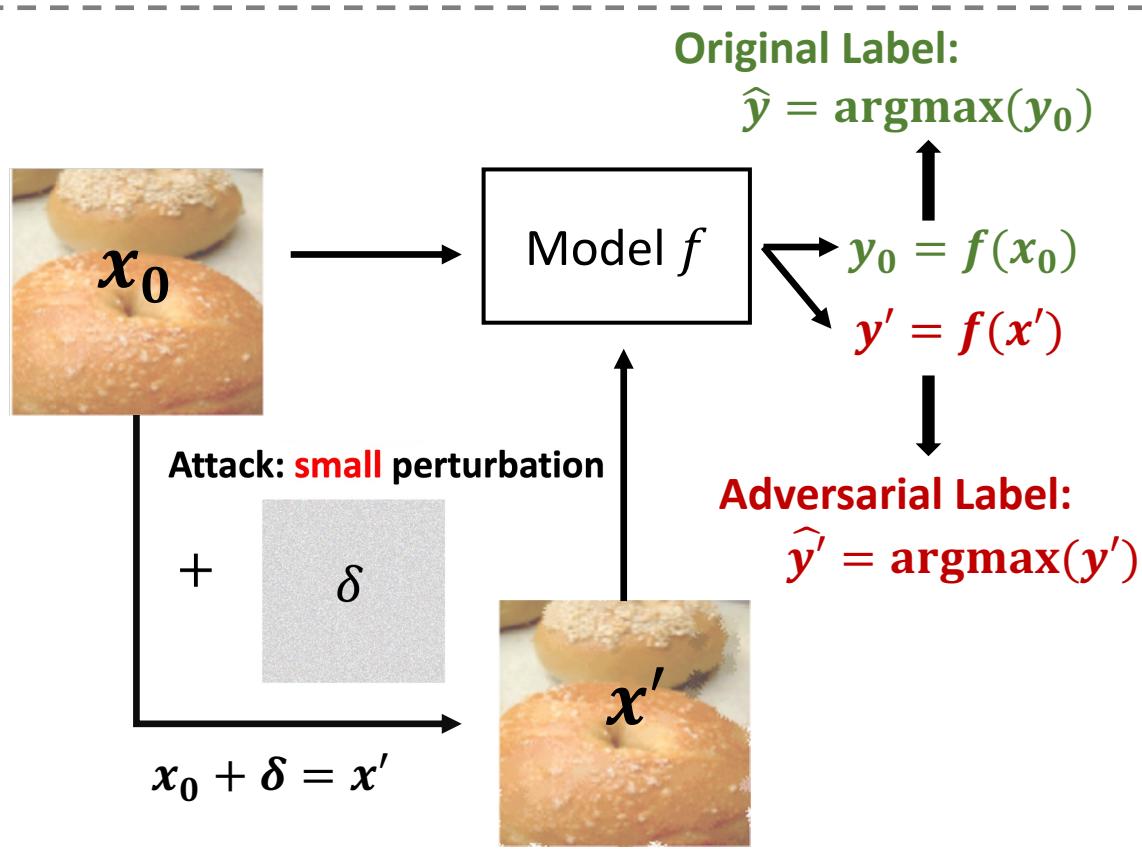
# Implication of Adversarial Examples

- **The existence of adversarial examples prevents the reliable deployment of deep learning models to the real world.**
  - Adversaries may try to actively hack a vulnerable deep learning model.
  - The model performance can become much worse than we expect.
- **Deep learning models are often not robust.**
  - It is an active area of research to make these models robust against adversarial examples.

# Content

- Introduction to Adversarial Attack
- Adversarial Attack Types
- Evasion Attack and Defense
- Poisoning Attack and Defense
- Exploratory Attack and Defense

# Goals of Attacks (1)



- **Untargeted attacks**

- The predicted label of adversarial example can be arbitrary except the original one.
- Ensure  $\hat{y}' \neq \hat{y}$

- **Targeted attacks:**

- Misguide deep neural networks to a specific class  $y^{\text{target}}$ .
- Ensure  $\hat{y}' = y^{\text{target}} \neq \hat{y}$

# Types of Attacks (1)

- **Different Adversary Knowledge:**

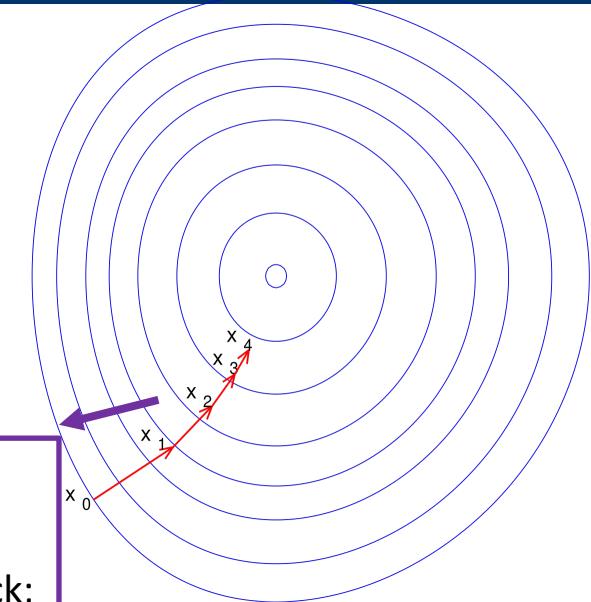
- **White-box Attack:**

- The attacker **has access to model architecture and parameters**
  - **Gradient-based methods** are straightforward and effective

Reverse direction to **maximize loss**:  
$$L(x) = \text{CrossEntropy}(y', \hat{y})$$
  
 $y'$  is the model's output after attack;  
 $\hat{y}$  is the original label

- **Black-box Attack:**

- The attacker **has no access to the model's parameters**
  - The type of architecture might be known
  - A different model or no model is used to generate adversarial examples in the hopes that these will transfer to the target model



# Types of Attacks (2)

- **Different Adversary Strategy:**

- **White-box Attack:**

- Based on the **gradient** of the network loss function **w.r.t. to the input**
  - Instead of optimizing parameters, white-box attack **optimizes the input** via gradient descend
  - Easy to attack

- **Black-box Attack:**

- **Gain restricted knowledge** by providing a series of **carefully crafted inputs** and **observing network's outputs**.
  - Usually **build models locally** that are **functionally close to target model**
  - Relatively hard to attack

# Adversarial Attack Methods

- **Evasion Attack:**

- Add malicious perturbation to samples **during testing phase**
- Do not affect the training data

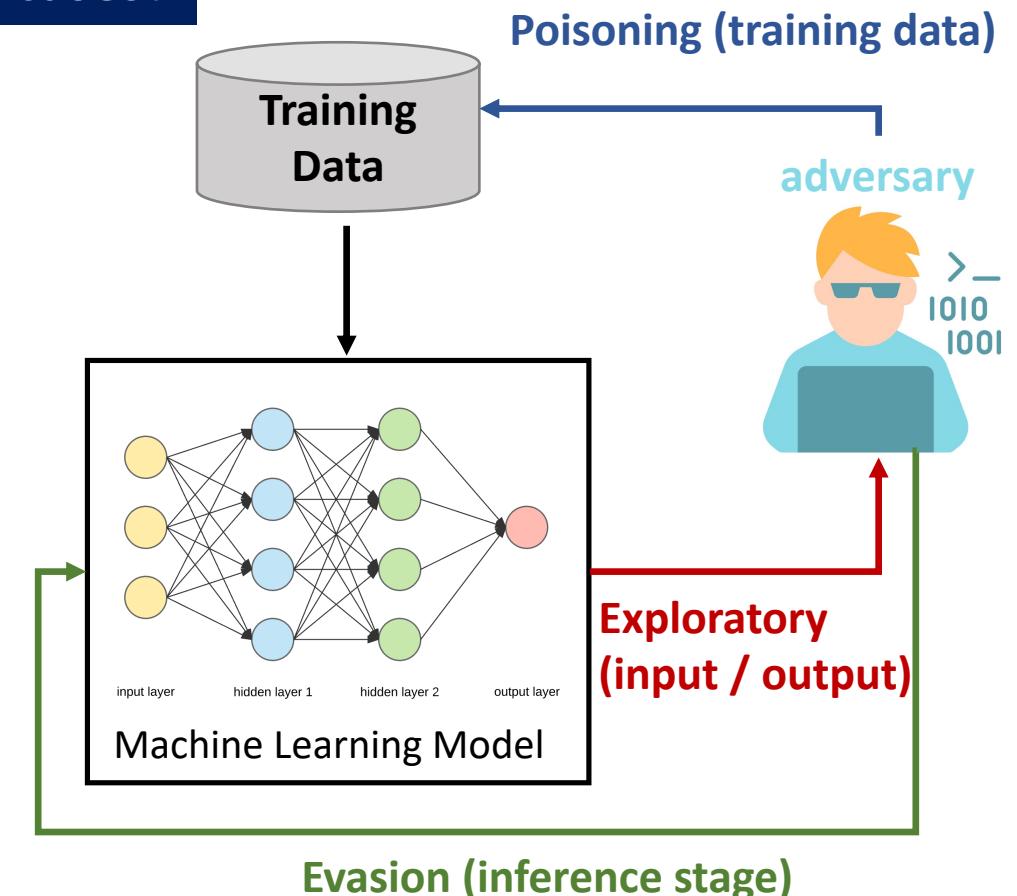
- **Poisoning Attack:**

- Contaminate the **training data** by carefully modifying samples

- **Exploratory Attack:**

- Given **black-box access** of the model, try to gain knowledge by **analyzing the input and output information** of the model

What are some use cases?

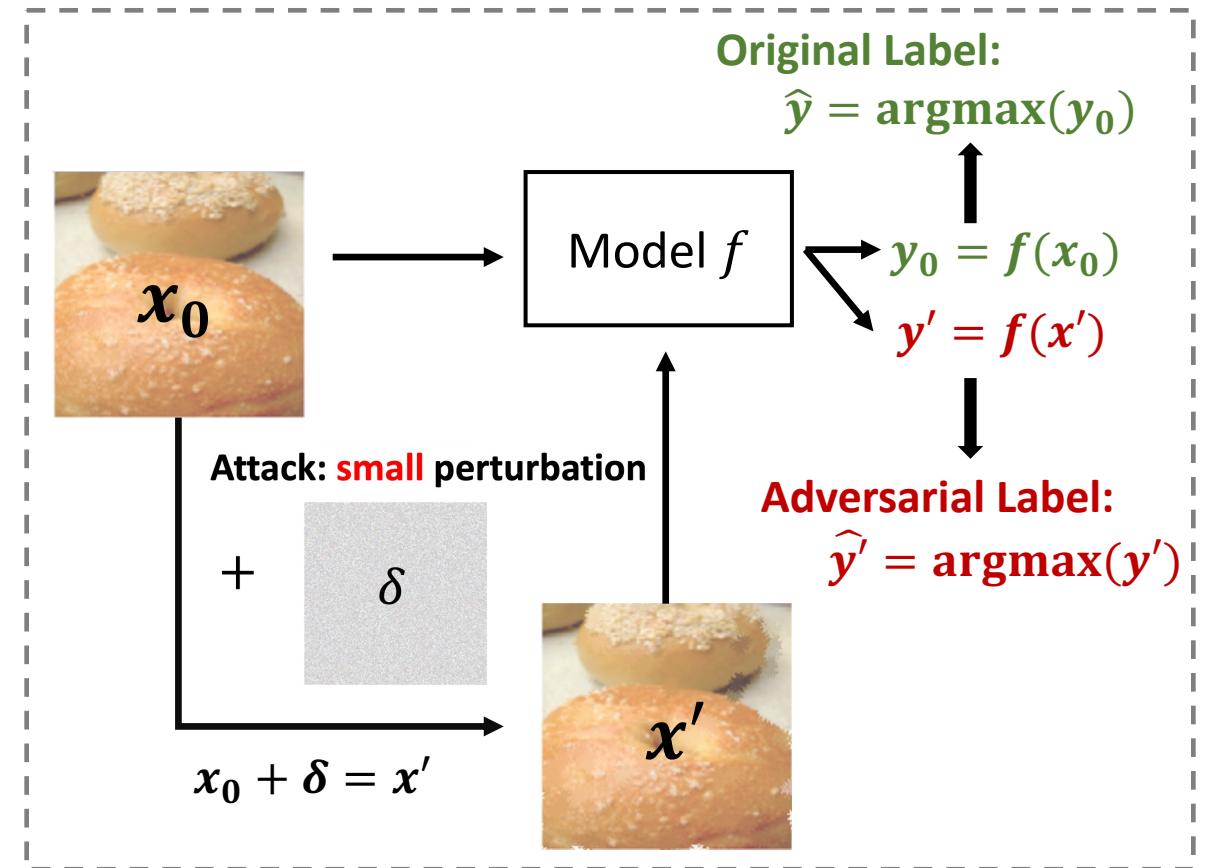


# Content

- Introduction to Adversarial Attack
- Adversarial Attack Types
- Evasion Attacks and Defenses
- Poisoning Attacks and Defenses
- Exploratory Attacks and Defenses

# Evasion Attack (1)

- In **evasion attack**, malicious samples are modified **during testing phase**
- No influence over the training dataset
- **Goal:** create a manipulated image (adversarial example) that is ***similar*** to the original image but causes a ***different prediction*** by the ML model.



# Evasion Attack (2)

- **Goal:** create a manipulated image (adversarial example) that is *similar* to the original image but causes a *different prediction* by the ML model.

- **Attack objective:**  $x^* = \arg \max_{x'} \boxed{\mathcal{L}(x')}$

Objective function ensuring different prediction

$s.t. \quad d(x', x_0) \leq \varepsilon$

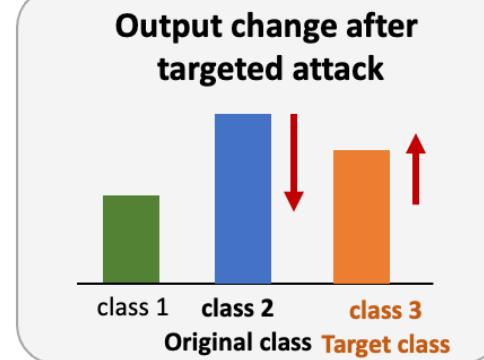
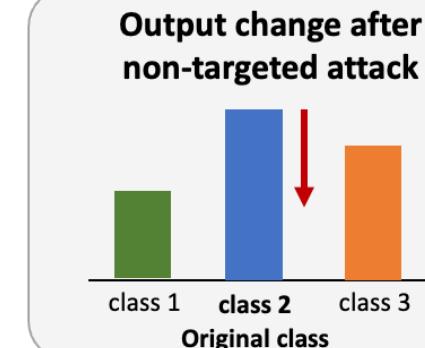
$d(x', x_0)$ : distance between  $x'$  and  $x_0$  ensuring similarity

# Evasion Attack (2)

- **Goal:** create a manipulated image (adversarial example) that is *similar* to the original image but causes a *different prediction* by the ML model.

- **Attack objective:**  $x^* = \arg \max_{x'} \mathcal{L}(x')$   
s.t.  $d(x', x_0) \leq \varepsilon$

Objective function ensuring different prediction



- **Non-targeted:**  $\mathcal{L}(x') = CE(f(x'), \hat{y})$

- **Targeted:**  $\mathcal{L}(x') = CE(f(x'), \hat{y}) - CE(f(x'), y^{target})$

push  $\hat{y}'$  to be far away from  $\hat{y}$   
but close to  $y^{target}$

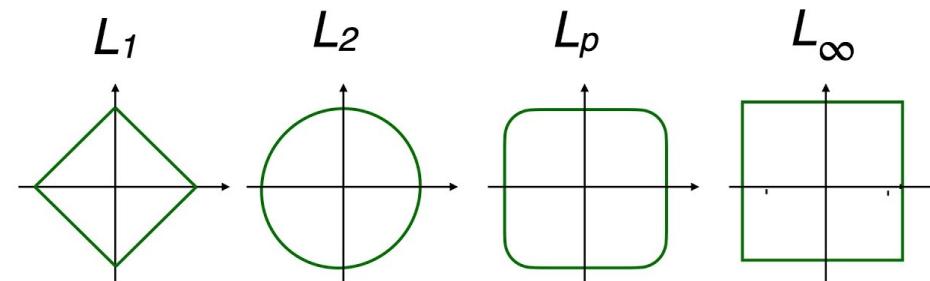
CE: CrossEntropy Loss

# Evasion Attack (2) - Imperceptible Perturbation

- **Perturbation:**

$$\begin{matrix} \mathbf{x}_0 \\ \left[ \begin{matrix} x_{11} & \cdots & x_{n1} \\ \vdots & \ddots & \vdots \\ x_{1n} & \cdots & x_{nn} \end{matrix} \right] \end{matrix} + \begin{matrix} \boldsymbol{\delta} \\ \left[ \begin{matrix} \Delta x_{11} & \cdots & \Delta x_{n1} \\ \vdots & \ddots & \vdots \\ \Delta x_{1n} & \cdots & \Delta x_{nn} \end{matrix} \right] \end{matrix} = \begin{matrix} \mathbf{x}' \\ \left[ \begin{matrix} x'_{11} & \cdots & x'_{n1} \\ \vdots & \ddots & \vdots \\ x'_{1n} & \cdots & x'_{nn} \end{matrix} \right] \end{matrix}$$

- $L_2$  norm:  $d(\mathbf{x}_0, \mathbf{x}') = \|\boldsymbol{\delta}\|_2 = ((\Delta x_{11})^2 + \cdots + (\Delta x_{nn})^2)^{\frac{1}{2}}$
- $L_\infty$  norm:  $d(\mathbf{x}_0, \mathbf{x}') = \|\boldsymbol{\delta}\|_\infty = \max\{|\Delta x_{11}|, \dots, |\Delta x_{nn}|\}$
- For **human perception**,  $L_\infty$  norm usually better captures the perturbation.



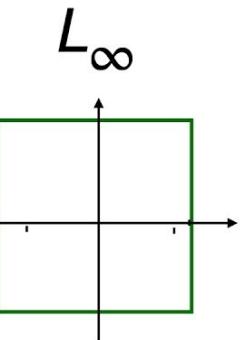
2-dimensional vectors with norm 1

# Evasion Attack (2)

- **Goal:** create a manipulated image (adversarial example) that is *similar* to the original image but causes a *different prediction* by the ML model.
- **Consider the setting:** non-targeted attack,  $L_\infty$ -norm perturbation, the optimization problem becomes:

$$x^* = \arg \max_{\delta} CE(f(x_0 + \delta), \hat{y})$$

$$\text{s.t. } \|\delta\|_\infty \leq \varepsilon$$



How to solve this optimization  
when  $f$  is known / unknown?



White-box: having access to model architecture and gradients

Black-box: queries only

# White-Box Evasion Attack: FGSM

- **Fast Gradient Sign Method (FGSM):**

Gradient ascent away from  $y^{true}$  class

$$x^* = x_0 + \varepsilon \cdot sign(\nabla_x \mathcal{L}(f(x), y^{true}))$$

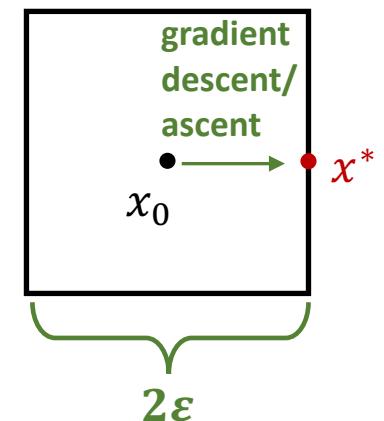
- **Target Class Method:**

Targeted attack

Gradient descent toward from  $y^{target}$  class

$$x^* = x_0 - \varepsilon \cdot sign(\nabla_x \mathcal{L}(f(x), y^{target}))$$

$x^*$  is on the vertex or midpoint of one side



- $x_0$  denotes the original sample,  $f(\cdot)$  denotes **logits output**
- $\text{Sign}(t) = 1$  if  $t > 0$ , otherwise  $\text{Sign}(t) = -1$
- $\varepsilon$  ensures that  $\|x^* - x_0\|_\infty \leq \varepsilon$ , perturbation within  $\varepsilon$ -box around the original sample

How many possible perturbations can there be?

# White-Box Evasion Attack: FGSM

- **Fast Gradient Sign Method (FGSM):**

Gradient ascent away from  $y^{true}$  class

$$x^* = x_0 \boxed{+} \varepsilon \cdot sign(\nabla_x \mathcal{L}(f(x), y^{true}))$$

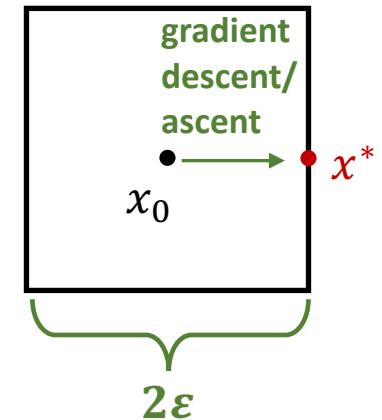
- **Target Class Method:**

Targeted attack

Gradient descent toward from  $y^{target}$  class

$$x^* = x_0 \boxed{-} \varepsilon \cdot sign(\nabla_x \mathcal{L}(f(x), y^{target}))$$

$x^*$  is on the vertex or midpoint of an edge



**Theorem:** If  $f$  is a *binary linear classifier*, FGSM is the *exact optimal evasion attack* under  $L_\infty$  constraint

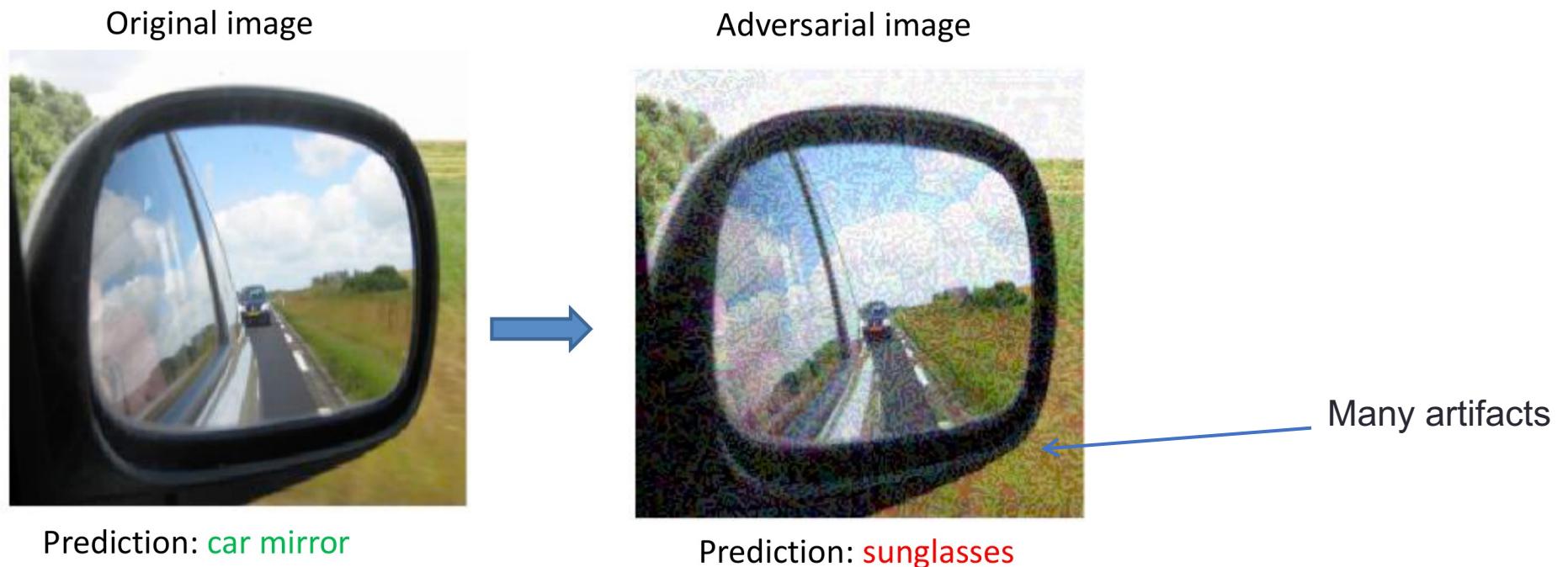
**Proof:** Exercise

Why  $sign(\cdot)$  function?

The gradient is zero or undefined everywhere → harder to defend  
Small gradient → precision problem  
Large gradient → problematic for max norm constraint

# White-Box Evasion Attack: FGSM

- Example



- FGSM is **one-step attack**
- **Pros:** Fast, easier to compute

**Cons:** Not very effective for highly non-linear models,  
Require large  $\epsilon$  to be effective (human-perceptible)

# White-Box Evasion Attack: PGD $\sim$ Iterative FGSM

- **Projected Gradient Descent (PGD)**

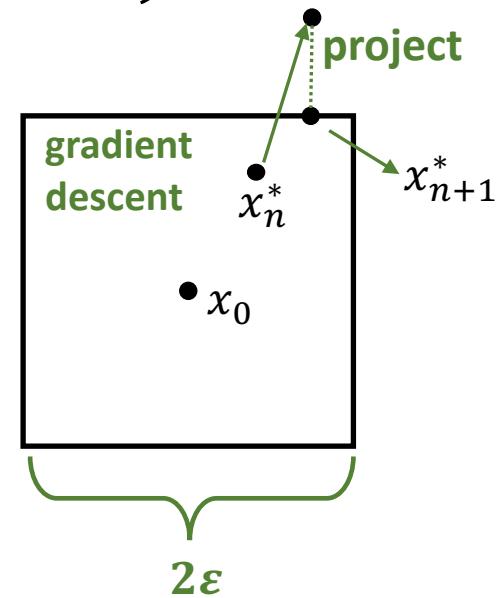
## Iterative approach

$$x_0^* = x_0 ; x_{n+1}^* = \text{proj}_{x,\epsilon}(x_n^* + \alpha \text{ sign}(\nabla_x \mathcal{L}(f(x), y^{true})))$$

- $x$  denotes the original sample,  $\alpha$  (*typically*  $< \epsilon$ ) is the step size.

$$\text{proj}_{x_i,\epsilon}(t) = \begin{cases} x_i - \epsilon & \text{if } t \leq x_i - \epsilon \\ t & \text{if } x_i - \epsilon < t < x_i + \epsilon, \text{ for each feature } x_i \text{ in } x \\ x_i + \epsilon & \text{if } t \geq x_i + \epsilon \end{cases}$$

- PGD **iteratively maximize the adversarial objective** within **restricted perturbation**



example of 2-dimensional case

# White-Box Evasion Attack: PGD ~ Iterative FGSM

## Examples

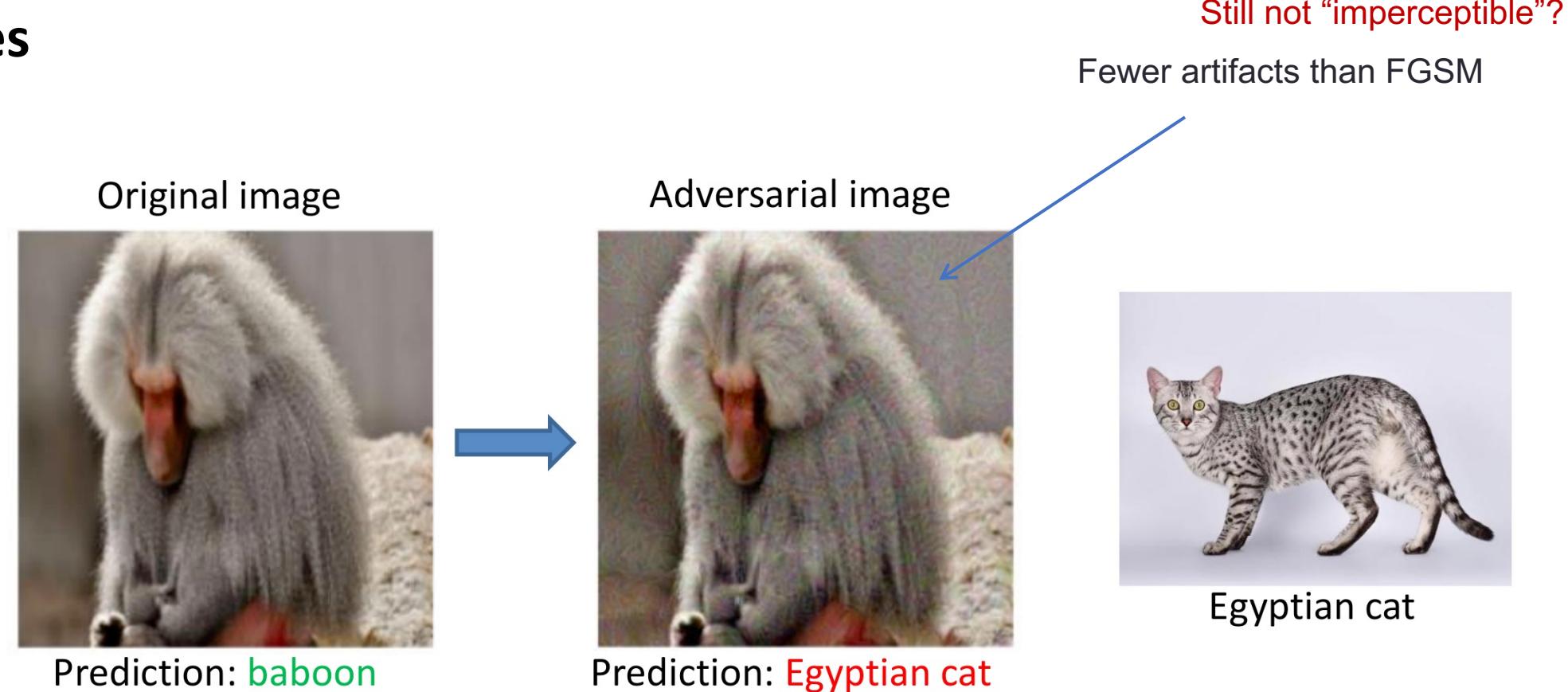


Image credit: <https://blog.floydhub.com/introduction-to-adversarial-machine-learning/>

# White-Box Evasion Attack: Deepfool

- **FGSM & PGD:** Find an adversarial example that *maximizes the likelihood* of being misclassified.

$$x^* = \arg \max_{\delta} CE(f(x_0 + \delta), \hat{y})$$

$$\text{s.t. } \|\delta\|_{\infty} \leq \varepsilon$$

norm-bounded attack

- **DeepFool:** Find an adversarial example with *minimal perturbation* to fool the classifier.

$$x^* = \boxed{\arg \max_{\delta} \|\delta\|_{\infty}}$$

$$\text{s.t. } f(x_0 + \delta) \neq \hat{y}$$

norm-unbounded attack

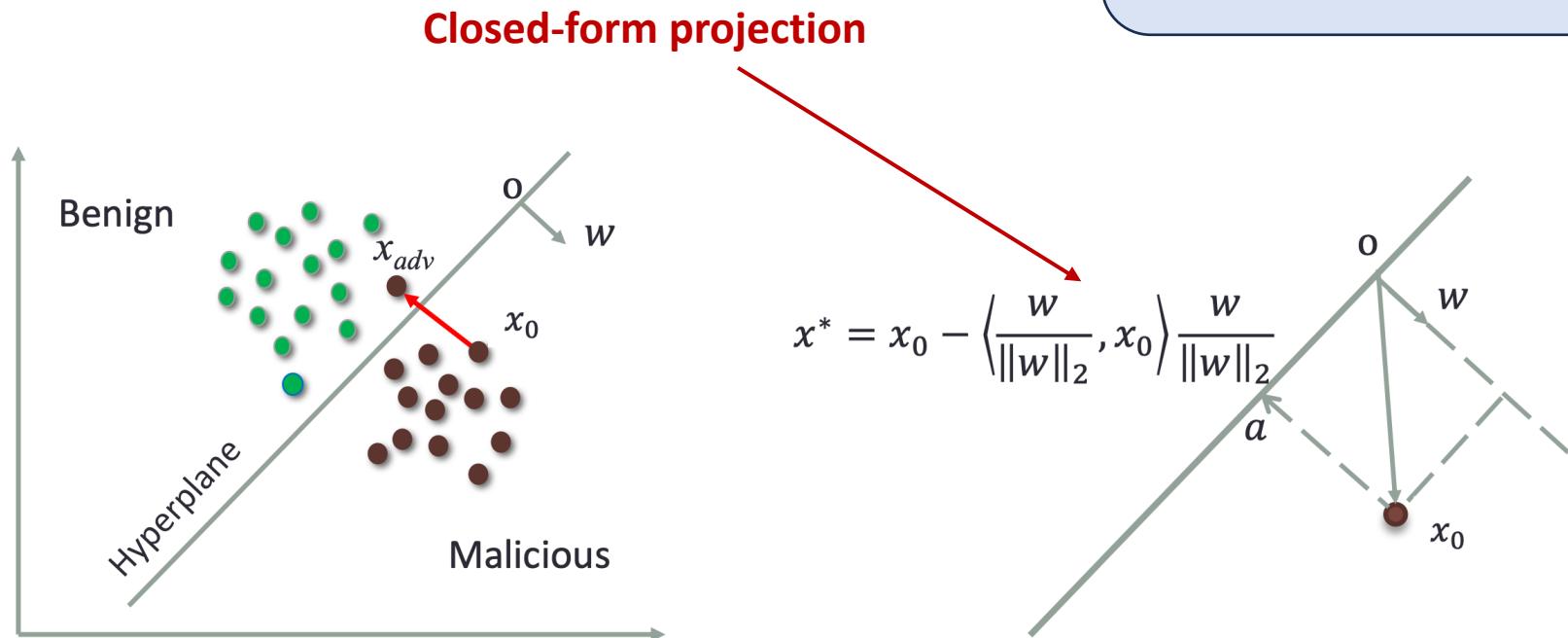
How to solve?

# White-Box Evasion Attack: Deepfool

- If  $f$  is a linear classifier?
- A simple (closed-form) projection would do the job.

$$x^* = \arg \max_{\delta} \|\delta\|_{\infty}$$

s.t.  $f(x_0 + \delta) \neq \hat{y}$



# White-Box Evasion Attack: Deepfool

- If  $f$  is **not** a linear classifier?
- An **iterative** projection **with linear approximation** would do the job.

$$x^* = \arg \max_{\delta} \|\delta\|_{\infty} \\ \text{s.t. } f(x_0 + \delta) \neq \hat{y}$$

## Algorithm

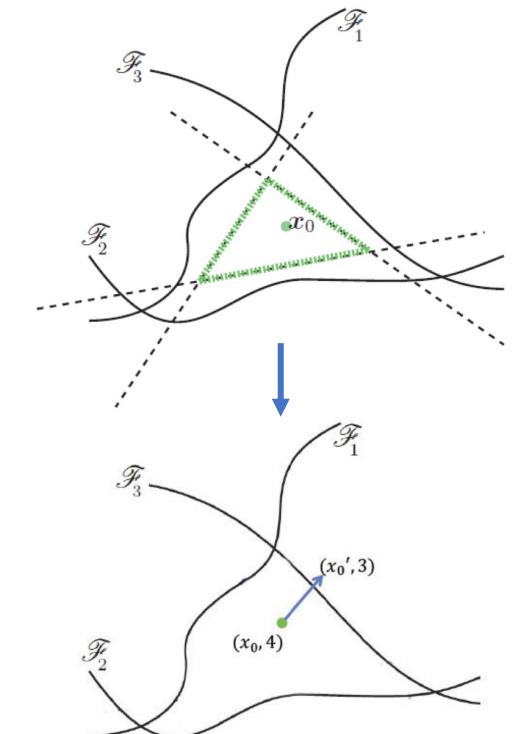
1. Linearize the classifier function around the current input  $x_i$ .

$$f(x_{i+1}) = f(x_i) + \nabla f(x_i)^T (x_{i+1} - x_i)$$

2. Project  $x_i$  onto the approximated hyperplane

$$x_{i+1} = x_i - \frac{f(x_i)}{\|\nabla f(x_i)\|_2^2} \nabla f(x_i)$$

Repeat (1) and (2) until  $f(x_i) \neq \hat{y}$



# White-Box Evasion Attack: Deepfool

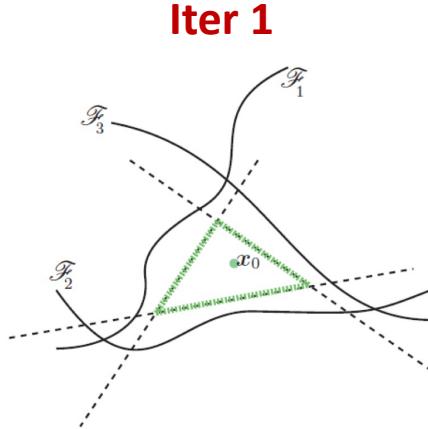
- If  $f$  is **not** a linear classifier?
- An **iterative** projection **with linear approximation** would do the job.

Algorithm

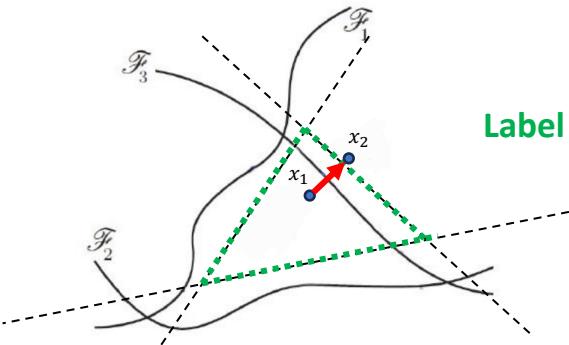
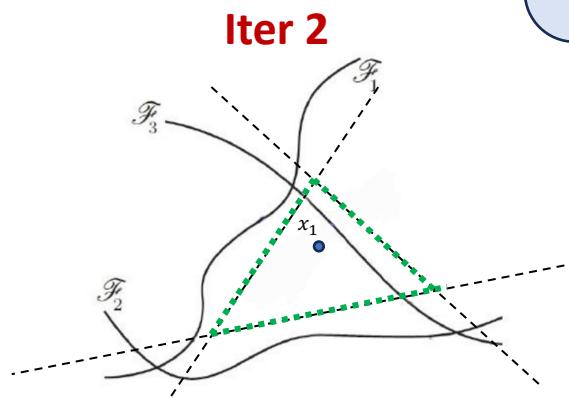
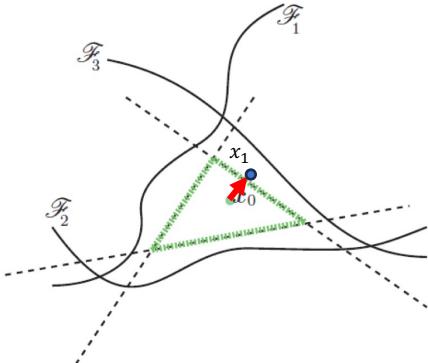
$$x^* = \arg \max_{\delta} \|\delta\|_{\infty}$$

s.t.  $f(x_0 + \delta) \neq \hat{y}$

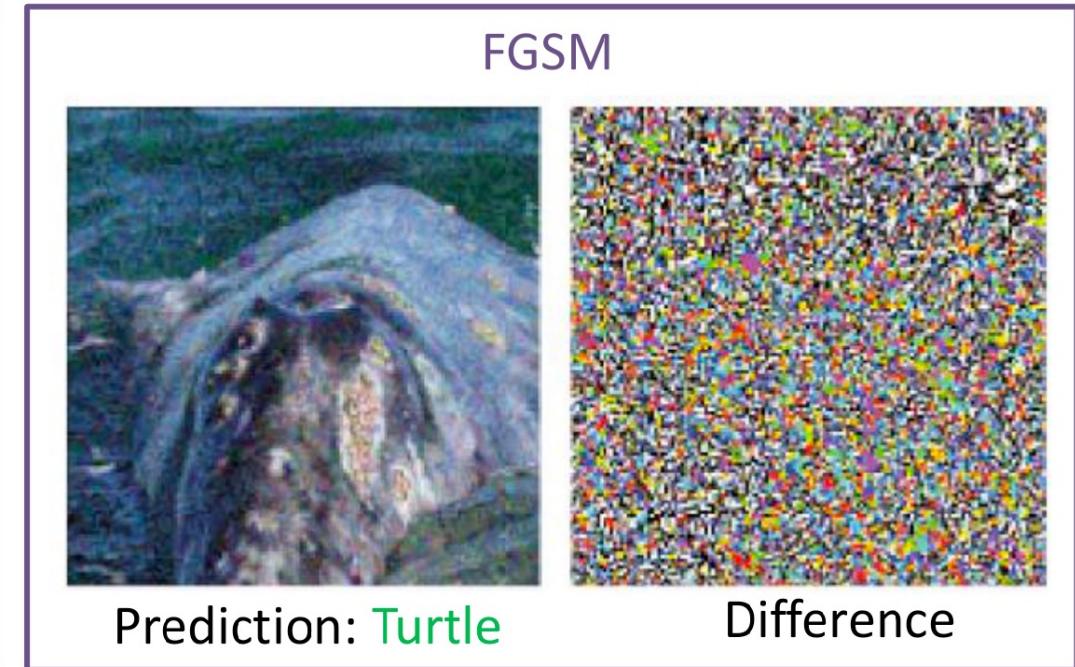
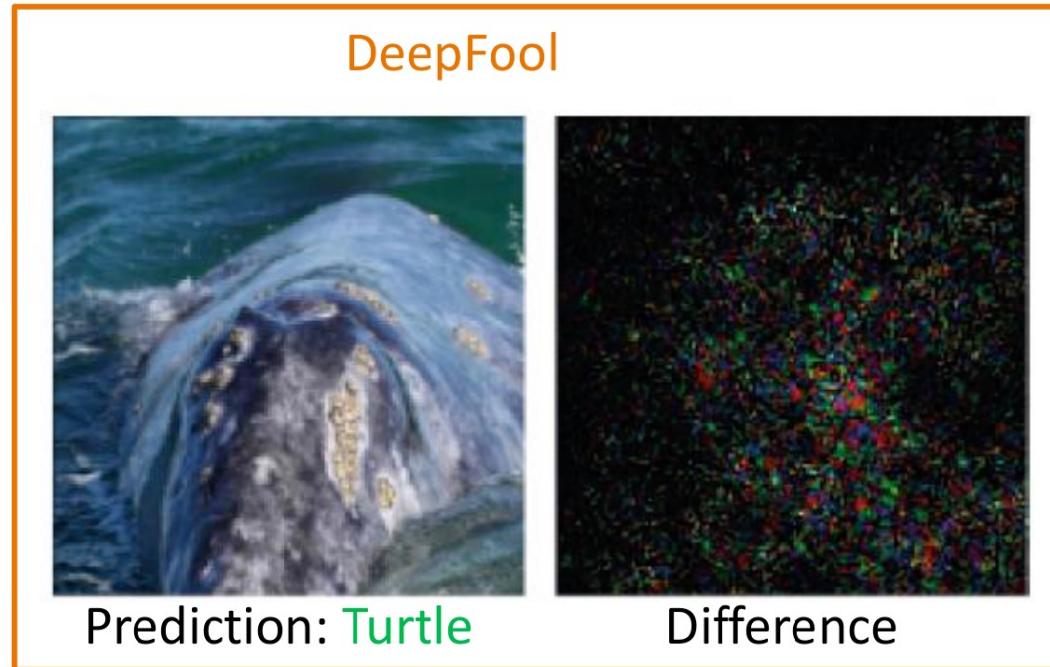
Step 1



Step 2



# White-Box Evasion Attack: Deepfool



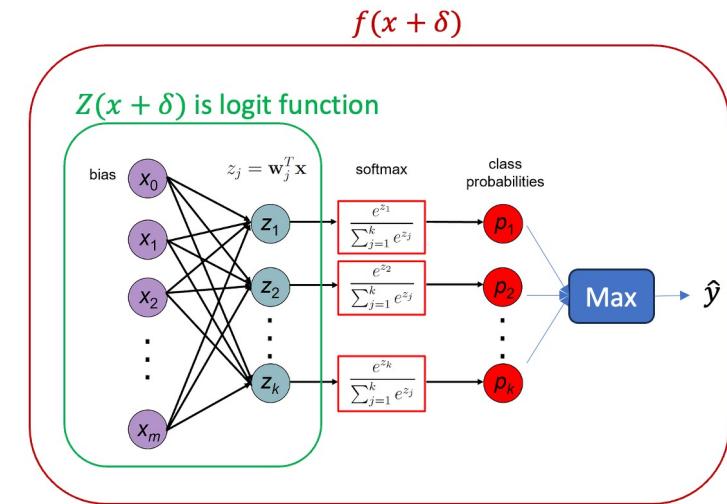
Comparing DeepFool and FGSM: Both flip the model prediction, but DeepFool requires smaller perturbation

# White-Box Evasion Attack: Carlini & Wagner

Recall the optimization problem for targeted attack

$$x^* = \arg \max_{x'} d(x_0, x') \\ \text{s.t. } f(x') = y^{target}$$

This constraint is highly non-linear → hard to solve



# White-Box Evasion Attack: Carlini & Wagner

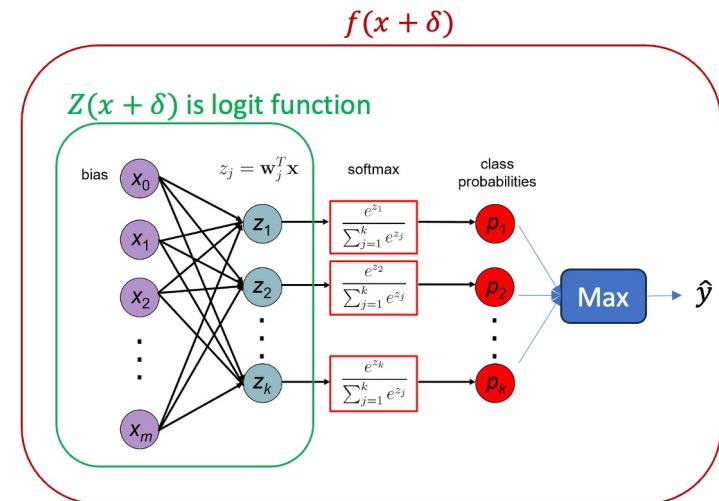
Recall the optimization problem for targeted attack

$$x^* = \arg \max_{x'} d(x_0, x') \\ s.t. F(x') \leq 0$$

where  $F(x') = \max_{i \neq y^{\text{target}}} (Z(x')_i) - Z(x')_{y^{\text{target}}}$

the difference in the logits between *the target class* and *the closest-to-the-target class* (a.k.a. *logit margin loss*)

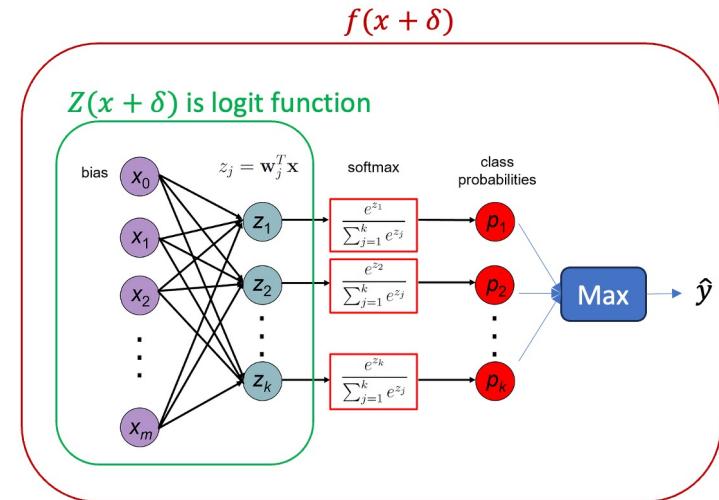
still constrained optimization → Lagrange multiplier



# White-Box Evasion Attack: Carlini & Wagner

Recall the optimization problem for targeted attack

$$x^* = \arg \max_{x'} d(x_0, x') \\ s.t. F(x') \leq 0$$



Reformulation using Lagrange multipliers

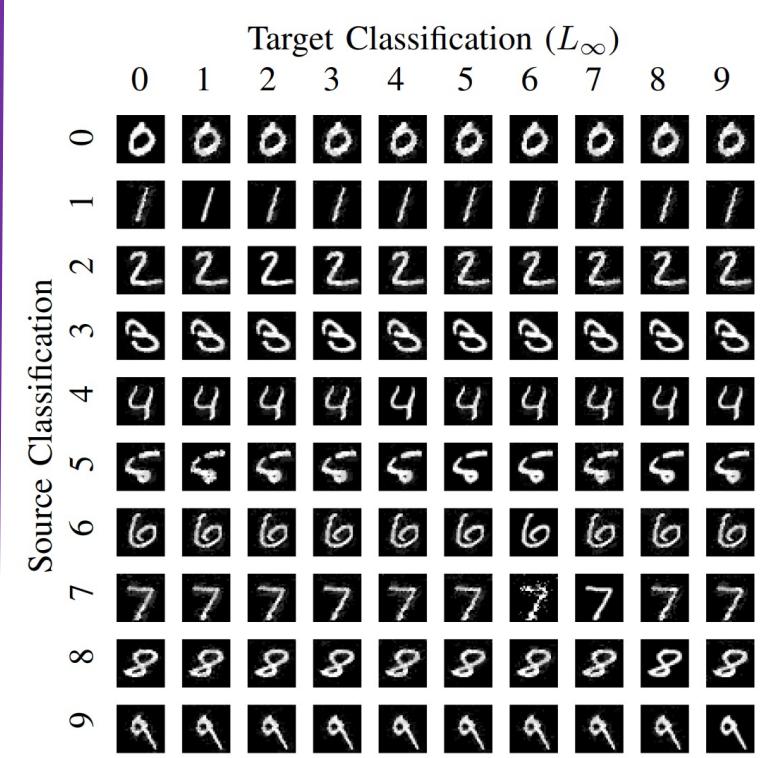
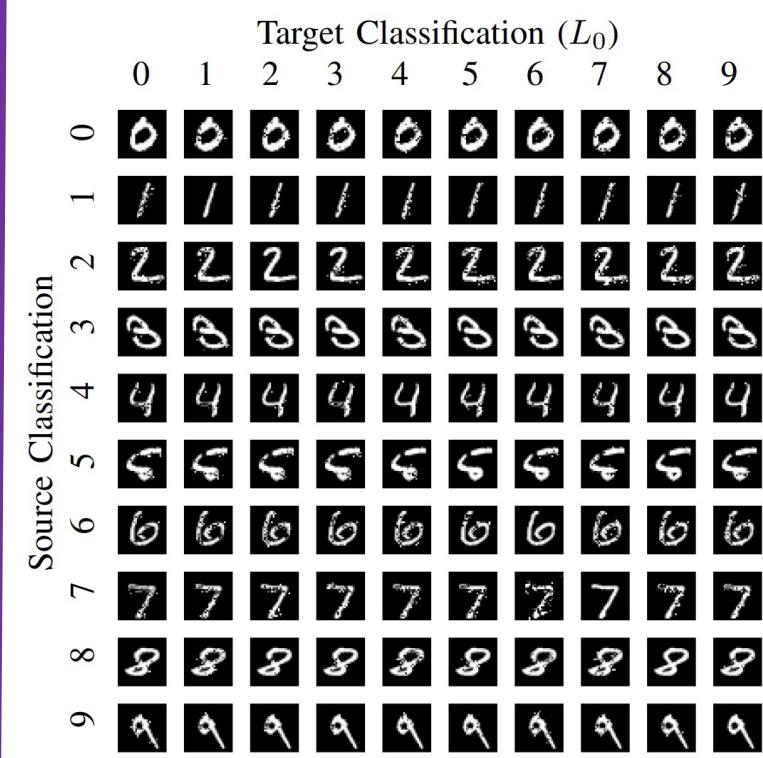
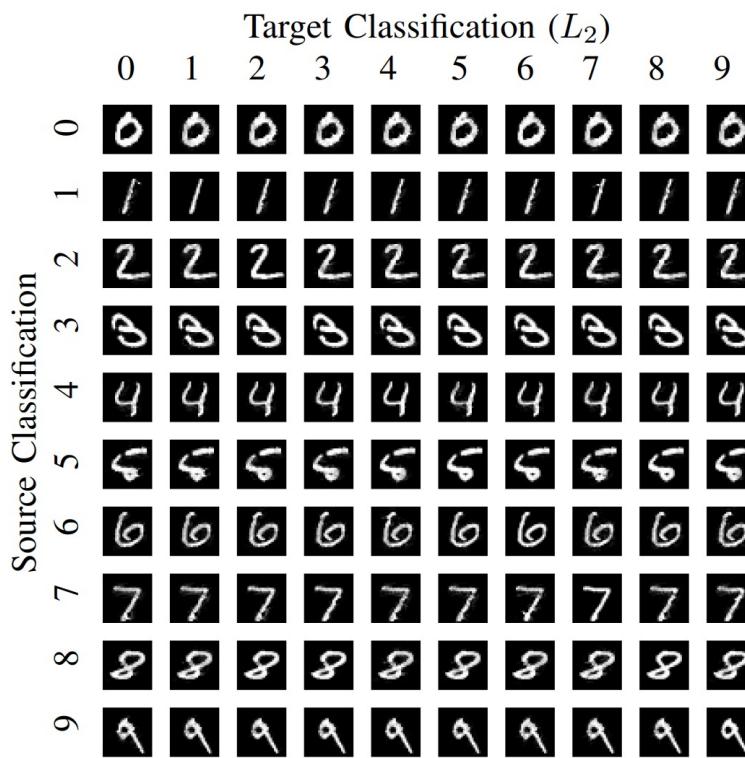
$$x^* = \arg \max_{x'} d(x_0, x') + \lambda F(x')$$

$$\text{where } F(x') = \max_{i \neq y^{\text{target}}} (Z(x')_i - Z(x')_{y^{\text{target}}})$$

**C-W Attack**

Easier to solve by standard solver (ADAM)

# Perturbation on MNIST



# White-Box Evasion Attack: Carlini & Wagner

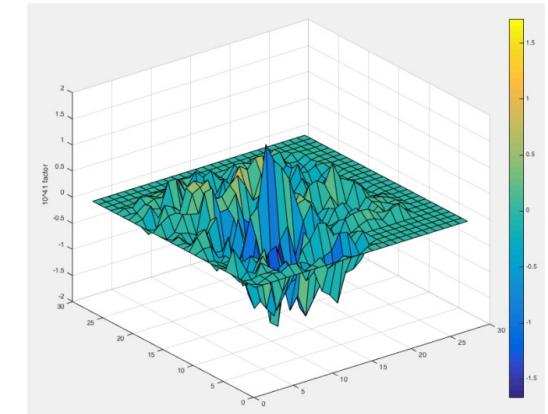
- Mean is the perturbation size (the smaller , the better)
  - Attacking Inception V3 on Imagenet
  - Untargeted: Untargeted attack
  - Average Case: Select the target uniformly at random
  - Least Likely: select the most difficult class as the target to attack

	Untargeted		Average Case		Least Likely	
	Mean	Prob	Mean	Prob	Mean	Prob
CW	0.004	100%	0.006	100%	0.01	100%
FGSM	0.004	100%	0.064	2%	-	0%
PGD	0.004	100%	0.01	99%	0.03	98%

# White-Box Evasion Attack: JSMA

- **Jacobian-based Saliency Map Attack (JSMA)**
- Assuming that the output of  $f$  is a vector of class probabilities

$$\text{Saliency Map : } \nabla f(x) = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \dots & \frac{\partial f_1(x)}{\partial x_M} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N(x)}{\partial x_1} & \dots & \frac{\partial f_N(x)}{\partial x_M} \end{bmatrix}$$

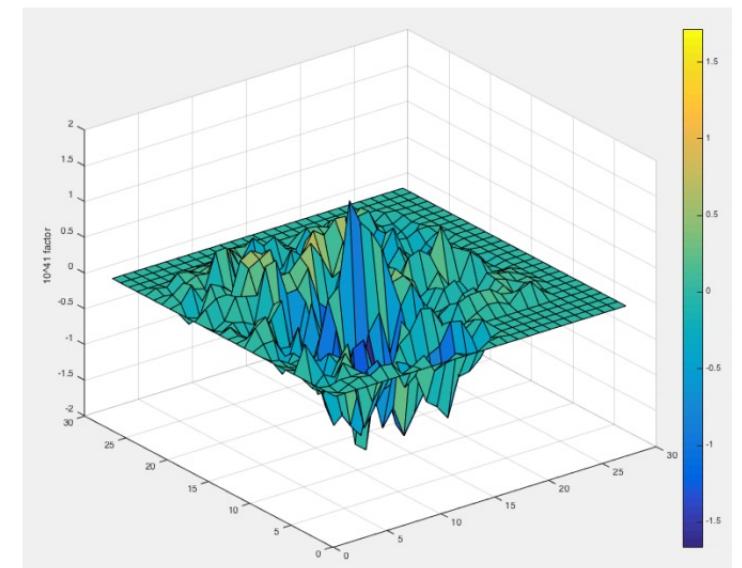


$\nabla f(x)$  Jacobian matrix

- $\nabla f(x) \in \mathbb{R}^{N \times M}$
- $N$  is the number of classes,  $M$  is the number of features of  $x$
- Features with **large saliency values** have **large impact on the output** when perturbed
- Incurs high computational cost to compute the Saliency Map

# White-Box Evasion Attack: JSMA

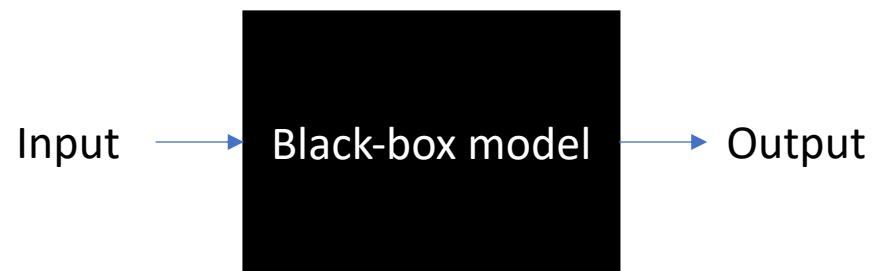
- **Jacobian-based Saliency Map Attack (JSMA)**
- $\nabla f(x)$  **Adversarial saliency maps** can be used to indicate which features an adversary should perturb in order to impact the predicted class by the model
  - Identify the most impactful features in the saliency map and then perturb them by  $\theta$  value in order to realize the adversary's goal (e.g.,  $x_i \rightarrow x_i - \theta$ )
  - The algorithm perturbs limited number of impactful features at each step



An example where explainability can help adversarial robustness

# Black-Box Evasion Attack

- Assuming the adversaries **do not** have access to the model's internal information (architecture, gradient, etc.)
- The adversaries can only query inputs and observe corresponding outputs



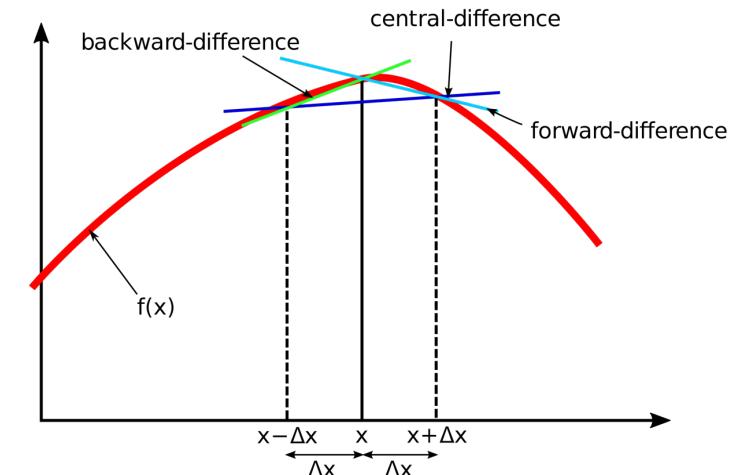
# Black-Box Evasion Attack: GEA

- **Gradient Estimation Attack (GEA):**
  - Estimate the gradient w.r.t. the input from (input, output) queries via **finite difference**
  - Use the estimated gradient to perform **first-order** attacks (e.g., FGSM, PGD, C&W)

- **Finite Difference (FD)** estimates the gradient of a function  $f$  w.r.t. input  $x \in \mathbb{R}^d$  by

$$\text{FD}_x(f(x), \delta) = \begin{bmatrix} \frac{f(x+\delta e_1) - f(x-\delta e_1)}{2\delta} \\ \dots \\ \frac{f(x+\delta e_d) - f(x-\delta e_d)}{2\delta} \end{bmatrix}$$

where  $\delta$  is a small scalar (e.g., 0.01) and  $e_i$  is the standard basis vector w.r.t. pixel  $i$ .



- **Relation to gradient:**  $\lim_{\delta \rightarrow 0} \text{FD}_x(f(x), \delta) \approx \nabla_x f(x)$

Left-hand, right-hand derivatives

# Black-Box Evasion Attack: GEA

- **Gradient Estimation Attack (GEA):**
  - Estimate the gradient w.r.t. the input from (input, output) queries via **finite difference**
  - Use the estimated gradient to perform **first-order** attacks (e.g., FGSM, PGD, C&W)
- Assuming the loss function is the cross-entropy loss we have

$$\begin{aligned}\mathcal{L}(f(x), y) &= \text{CE}(f(x), y) = -\log f(x)_y \\ \Rightarrow \nabla_x \mathcal{L}(f(x), y) &= -\frac{\nabla_x f(x)_y}{f(x)_y} = -\frac{\text{FD}_x(f(x)_y, \delta)}{f(x)_y}\end{aligned}$$

- **Approximated FGSM:**

$$x^* = x + \epsilon \cdot \text{sign} \left( -\frac{\text{FD}_x(f(x)_y, \delta)}{f(x)_y} \right)$$

# Black-Box Evasion Attack: GEA

- **Gradient Estimation Attack (GEA):**
  - Estimate the gradient w.r.t. the input from (input, output) queries via **finite difference**
  - Use the estimated gradient to perform **first-order** attacks (e.g., FGSM, PGD, C&W)
- Assuming the loss function is the cross-entropy loss we have

$$\begin{aligned}\mathcal{L}(f(x), y) &= \text{CE}(f(x), y) = -\log f(x)_y \\ \Rightarrow \nabla_x \mathcal{L}(f(x), y) &= -\frac{\nabla_x f(x)_y}{f(x)_y} = -\frac{\text{FD}_x(f(x)_y, \delta)}{f(x)_y}\end{aligned}$$

- **Approximated targeted FGSM:**

$$x^* = x - \epsilon \cdot \text{sign} \left( -\frac{\text{FD}_x(f(x)_{y^{\text{target}}}, \delta)}{f(x)_{y^{\text{target}}}} \right)$$

# Black-Box Evasion Attack: GEA

- **Gradient Estimation Attack (GEA):**
  - Estimate the gradient w.r.t. the input from (input, output) queries via **finite difference**
  - Use the estimated gradient to perform **first-order** attacks (e.g., FGSM, PGD, C&W)
- Recall the **logit margin loss** from the C&W attack

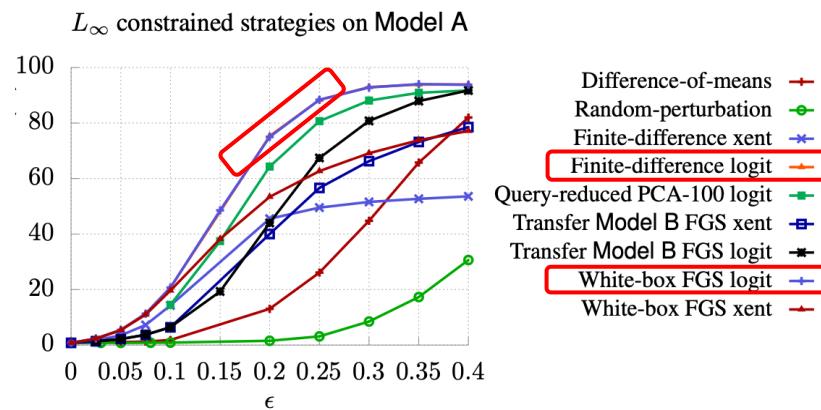
$$\begin{aligned}\mathcal{L}(f(x), y) &= \max_{i \neq y^{\text{target}}} \{Z(x)_i\} - Z(x)_{y^{\text{target}}} \\ \Rightarrow \nabla_x \mathcal{L}(f(x), y) &= \text{FD}_x \left( \max_{i \neq y^{\text{target}}} \{Z(x')_i\} - Z(x')_{y^{\text{target}}}, \delta \right)\end{aligned}$$

- **Approximated targeted C&W attack:**

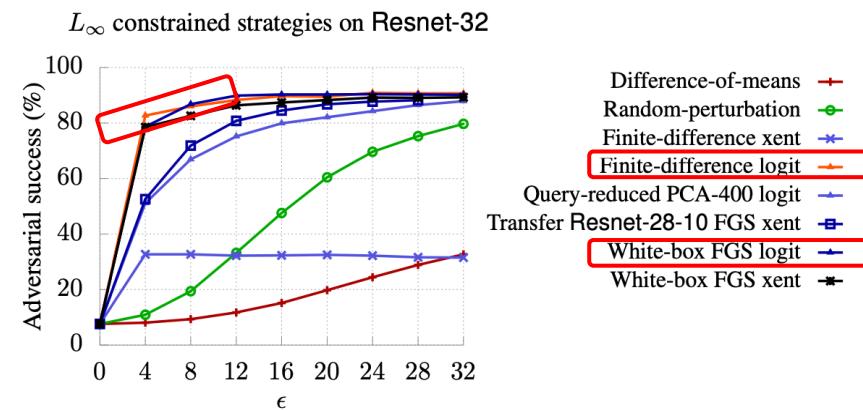
$$x^* = x - \epsilon \cdot \text{sign} \left( \text{FD}_x \left( \max_{i \neq y^{\text{target}}} \{Z(x')_i\} - Z(x')_{y^{\text{target}}}, \delta \right) \right)$$

# Black-Box Evasion Attack: GEA

- Success rate-perturbation size curve
  - Black-box attack (Finite-difference logit) has almost the same curve as white-box C&W attack (White-box FGS logit)



(a) Model A (MNIST)



(b) Resnet-32 (CIFAR-10)

What is the drawback?

# Drawback of GEA

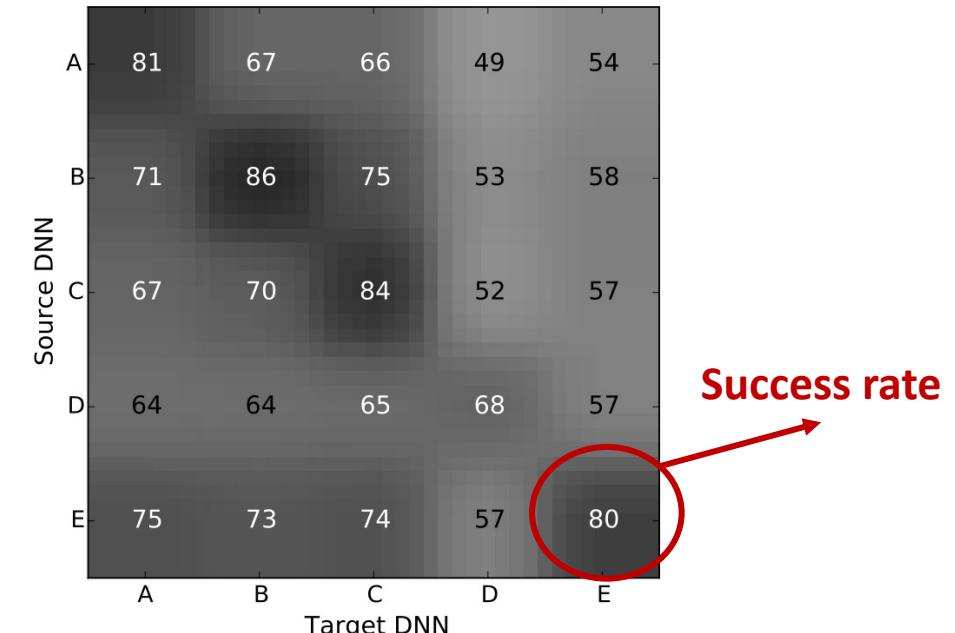
**for 28x28 image, require 28x28x2 (=1568) queries**

$$x^* = x - \epsilon \cdot \text{sign} \left( -\frac{\text{FD}_x(f(x)_{y^{\text{target}}}, \delta)}{f(x)_{y^{\text{target}}}} \right)$$

- Query reduction:
  - Random grouping: The gradient is estimated only for a random group of selected features.
  - PCA (Principal Component Analysis): Compute the gradient only along a number of principal component vectors.

# Black-Box Evasion Attack: Substitute model attack

- **Substitute model attack ~ Zero-query attack**
  - **Idea:** Train a substitute model
    - Generate adversarial examples for the substitute model
    - Transfer the generated adversarial samples to the target model.
  - **Rationale:** Adversarial examples often work across different ML models and datasets.
- **Intra-technique transferability:**  
The substitute and target model have the *same* architecture

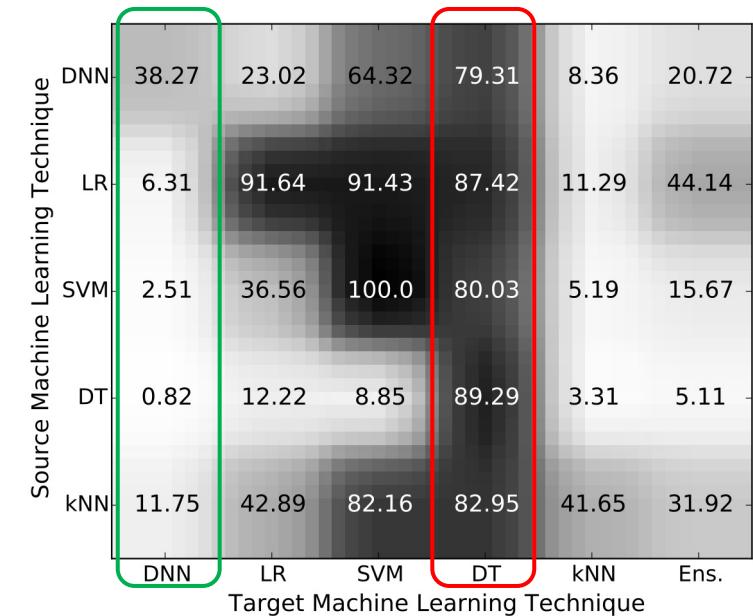


# Black-Box Evasion Attack: Substitute model attack

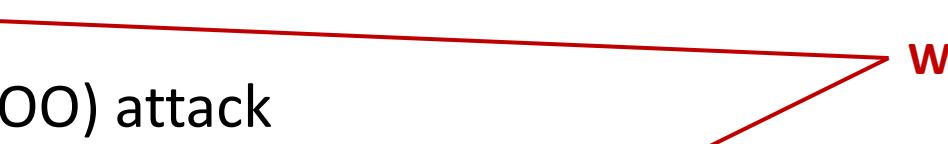
- **Substitute model attack ~ Zero-query attack**
  - **Idea:** Train a substitute model
    - Generate adversarial examples for the substitute model
    - Transfer the generated adversarial samples to the target model.
  - **Rationale:** Adversarial examples often work across different ML models and datasets.
- **Cross-technique transferability:**

The substitute and target model have the *different* architecture

  - **Decision Tree (DT):** is the most vulnerable model.
  - **DNNs:** is the most robust model.



# Black-Box Evasion Attack: Other attacks

- **Query-based attacks**
    - Gradient estimation attack
    - Zeroth-order optimization (ZOO) attack
    - Boundary attack
    - HopSkipJump attack
    - Simple black-box attack
  - **Transfer-based attacks**
    - Substitute model attack
    - Ensemble of models attack
- 
- We have covered

Reading: Akhtar et al. “[Advances in adversarial attacks and defenses in computer vision: A survey](#)”