

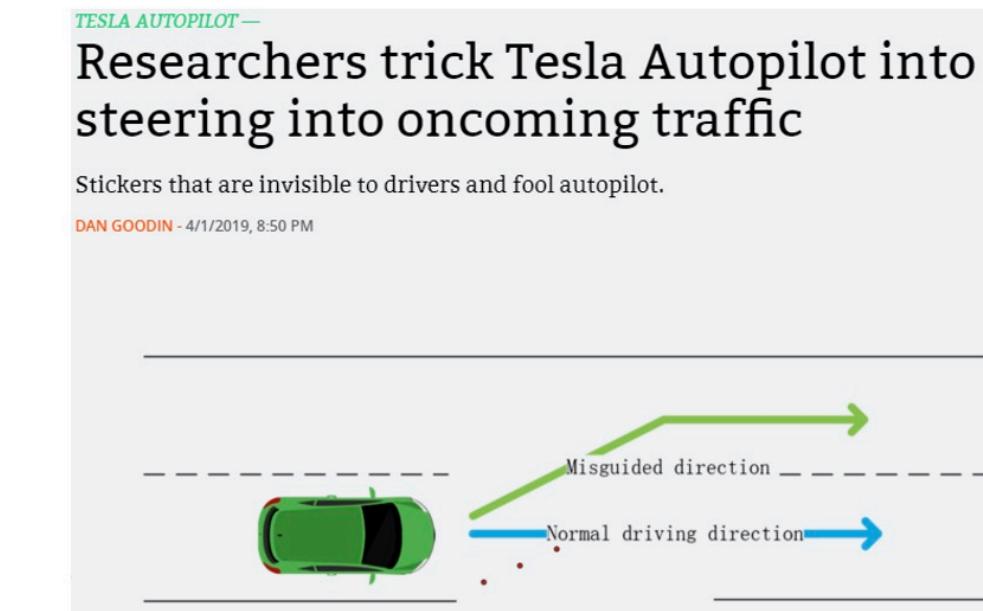
# **COMP5212: Machine Learning**

**Lecture 21**

**Minhao CHENG**

# 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

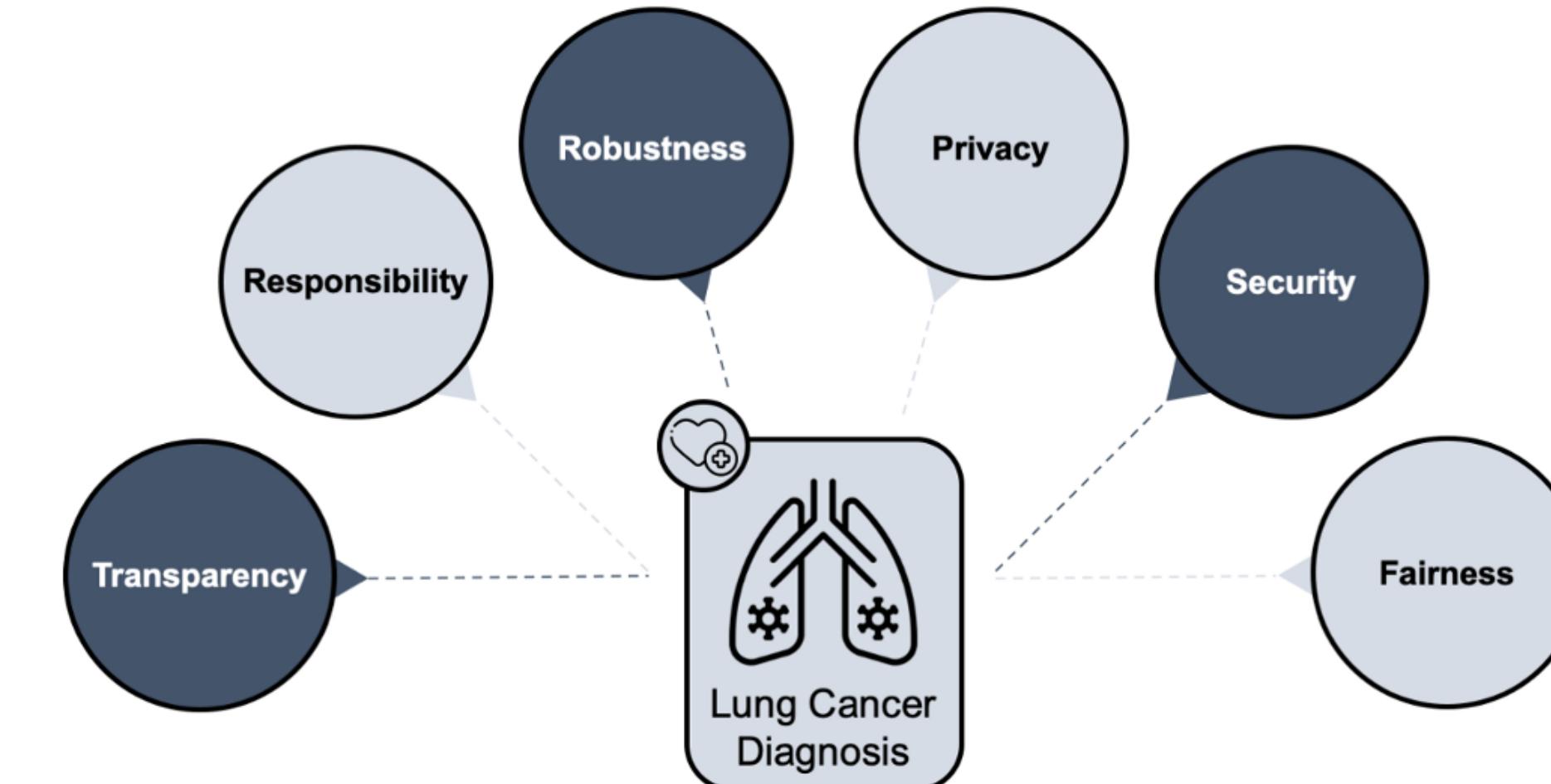


Microsoft's newly launched A.I.-powered bot called Tay, which was responding to tweets and chats on GroupMe and Kik, has already been shut down due to concerns with its inability to recognize when it was making offensive or racist statements. Of course, the bot wasn't coded to be racist, but it "learns" 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

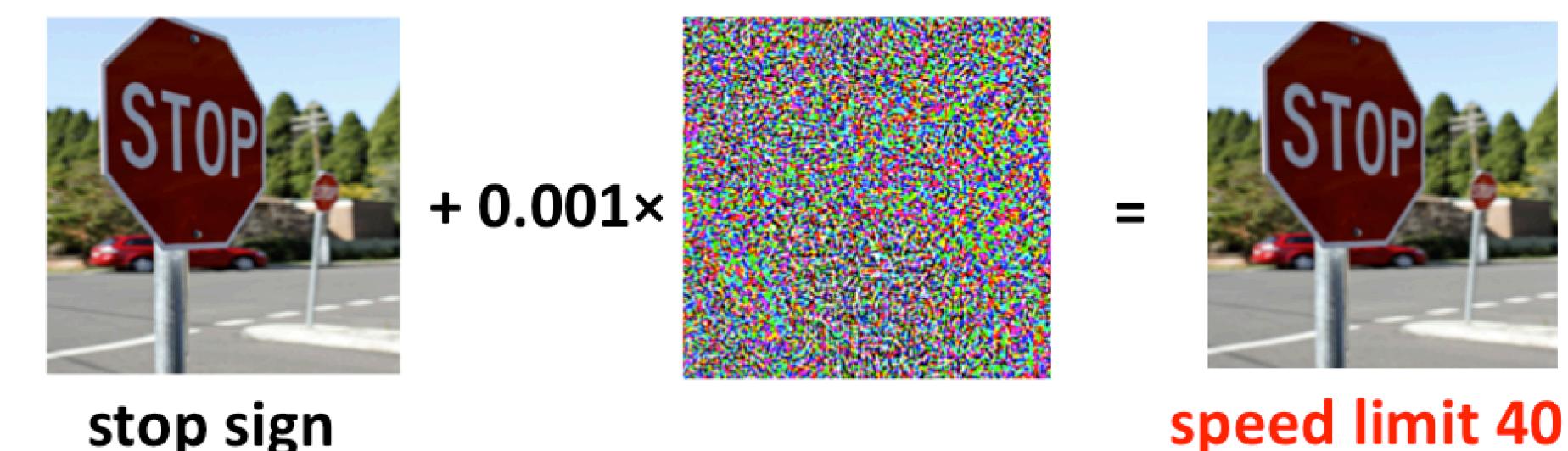
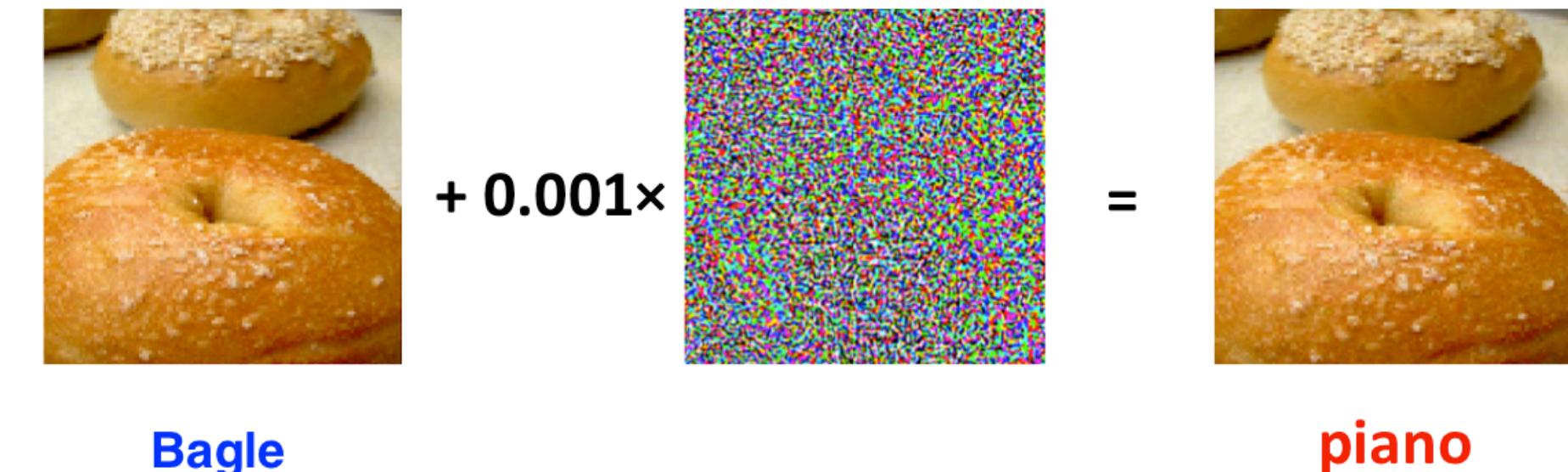
- Training-time integrity and Testing-time integrity

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



# Adversarial examples

## Definition

- Given a  $K$ -way multi-class classification model  $f: \mathbb{R}^d \rightarrow \{1, \dots, K\}$  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

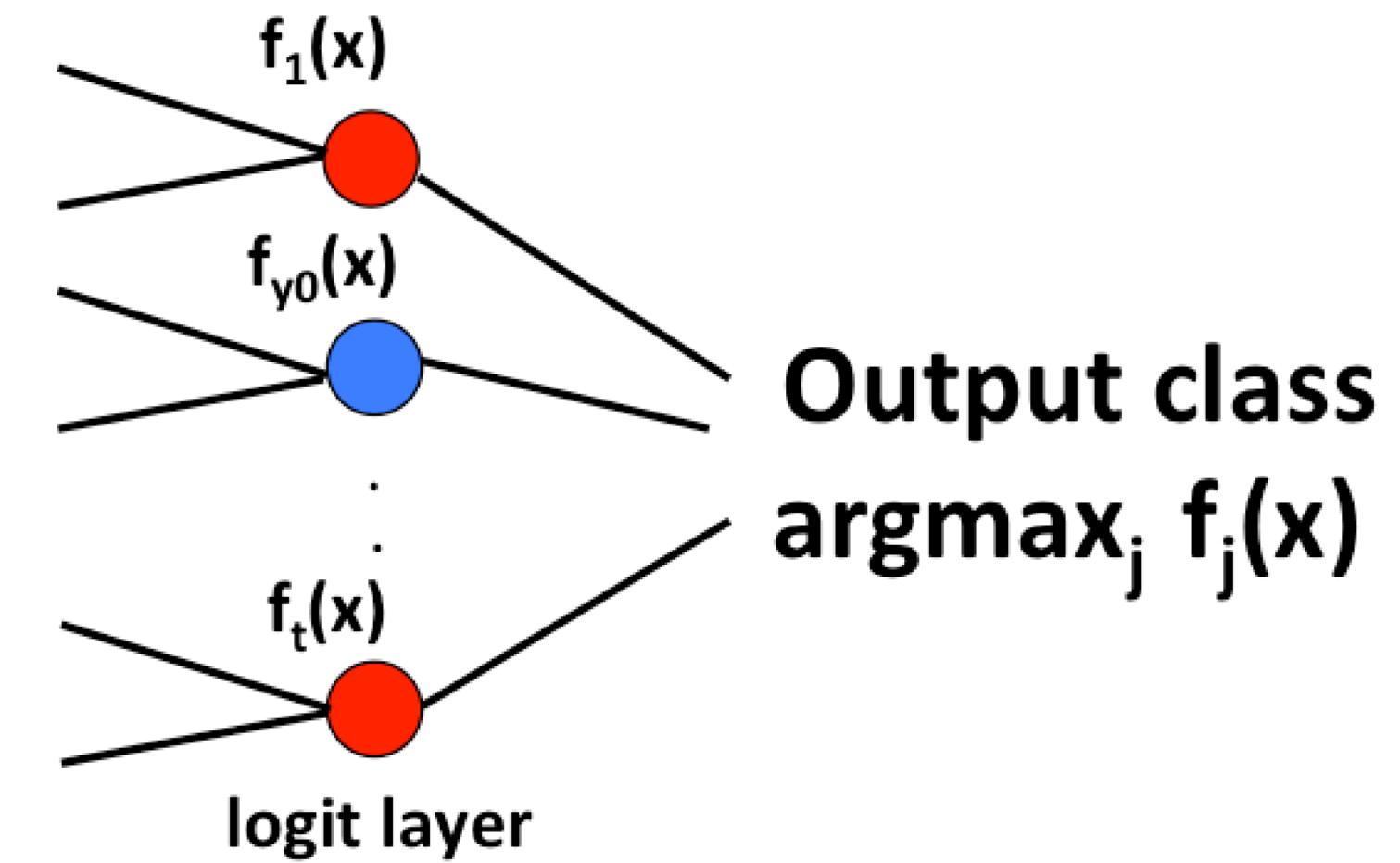
## Attack as an optimization problem

- 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

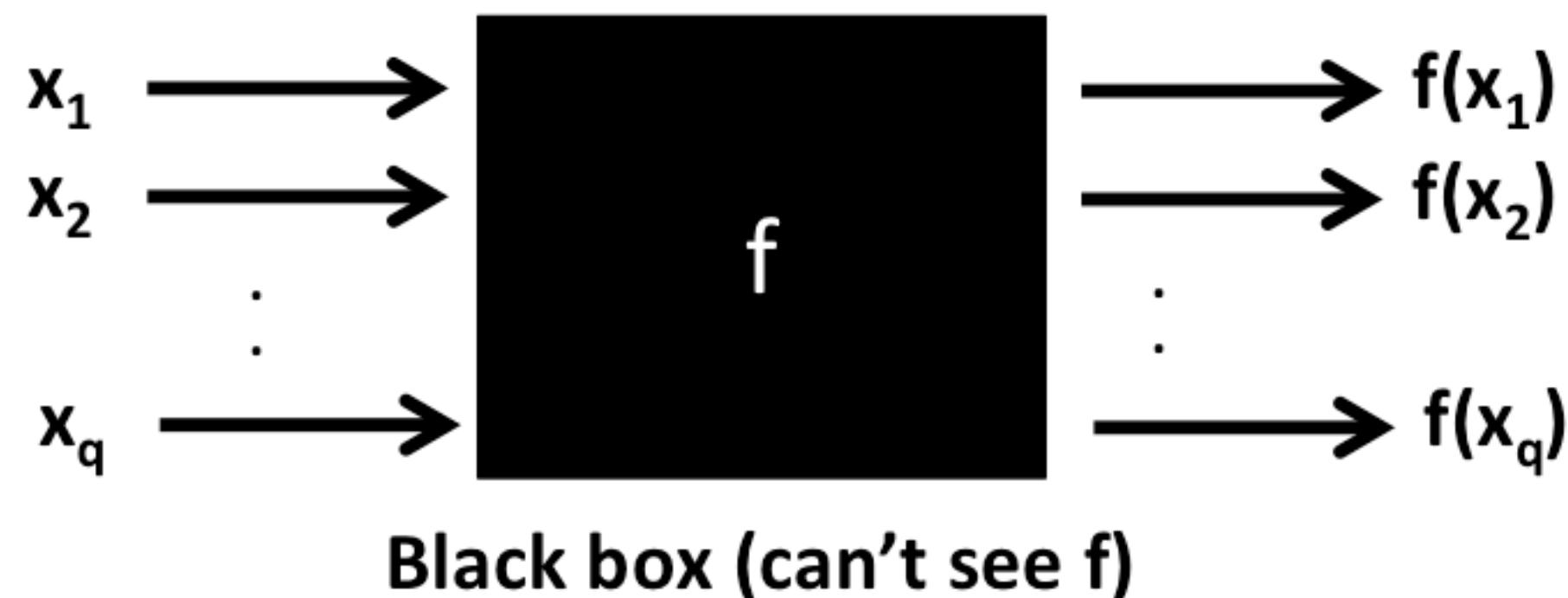
## White-box adversarial attack

- 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  $\nabla_x$

# 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**

$$x \rightarrow \boxed{f} \rightarrow f(x) = \begin{cases} 0 \\ 1 \\ . \\ . \\ K \end{cases}$$

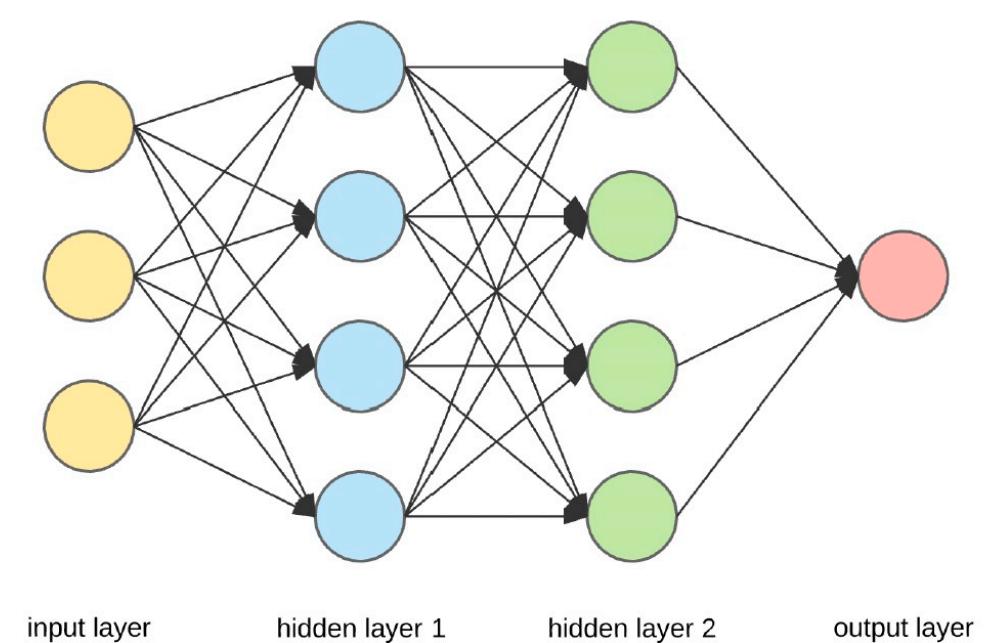
- ( $K$ : number of classes)

## Why Hard-label

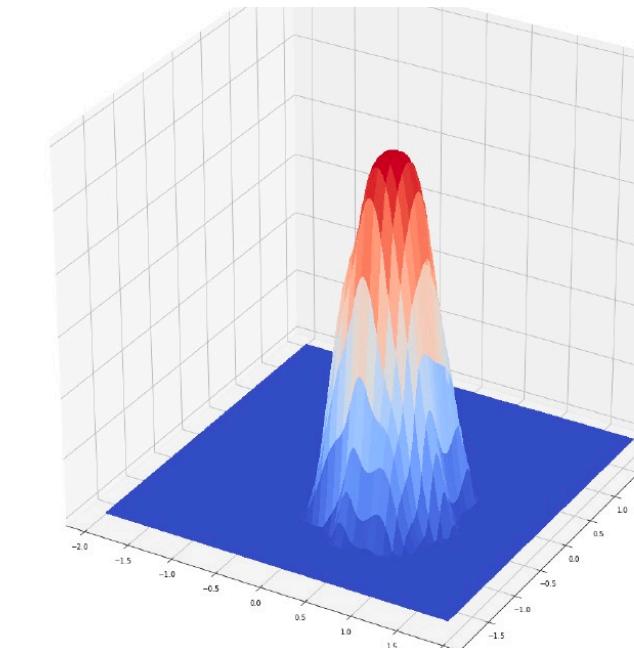
- More practical setting for attacker
- Discrete and complex models (e.g quantization, projection, detection)
- Framework friendly

# The difficulty

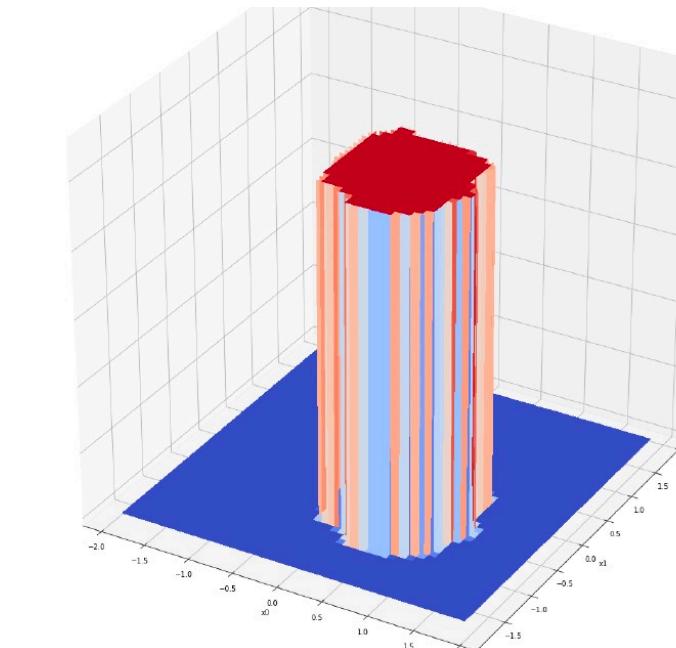
- Hard-label attack on a simple 3-layer neural network yields a discontinuous optimization problem



(a) neural network  $f(x)$

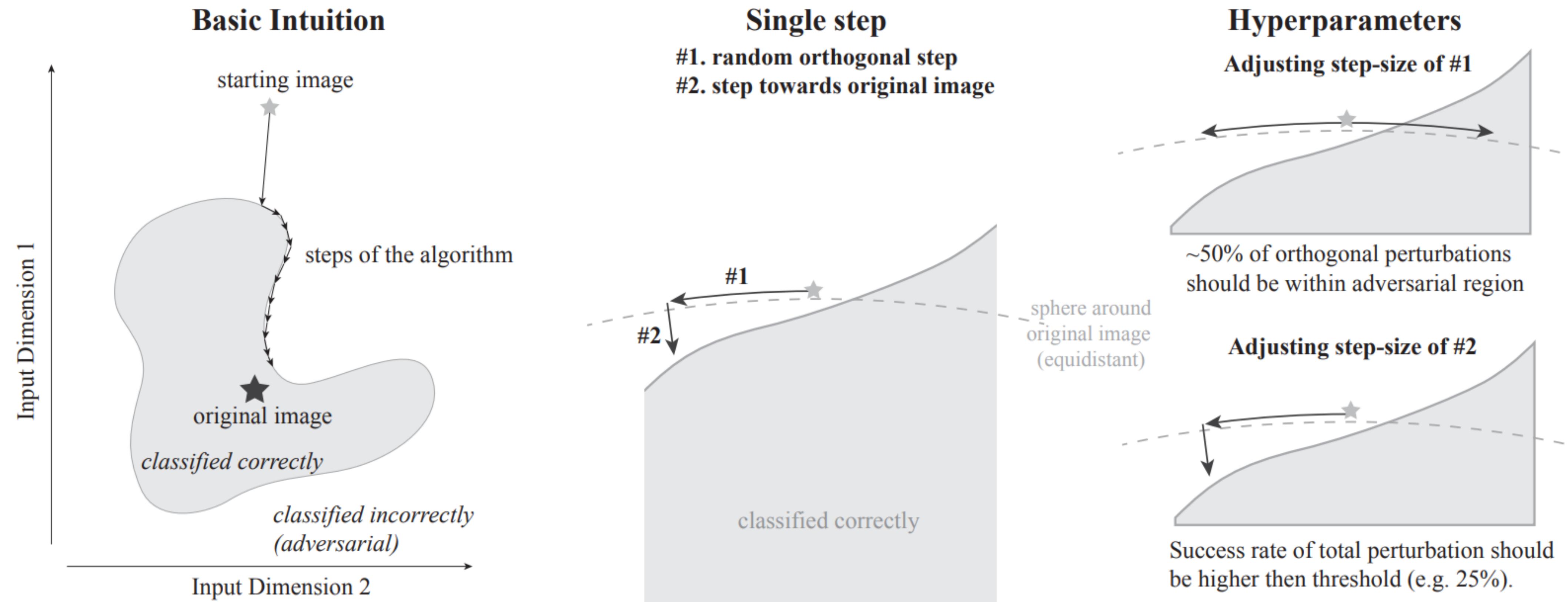


(b)  $h(Z(x))$



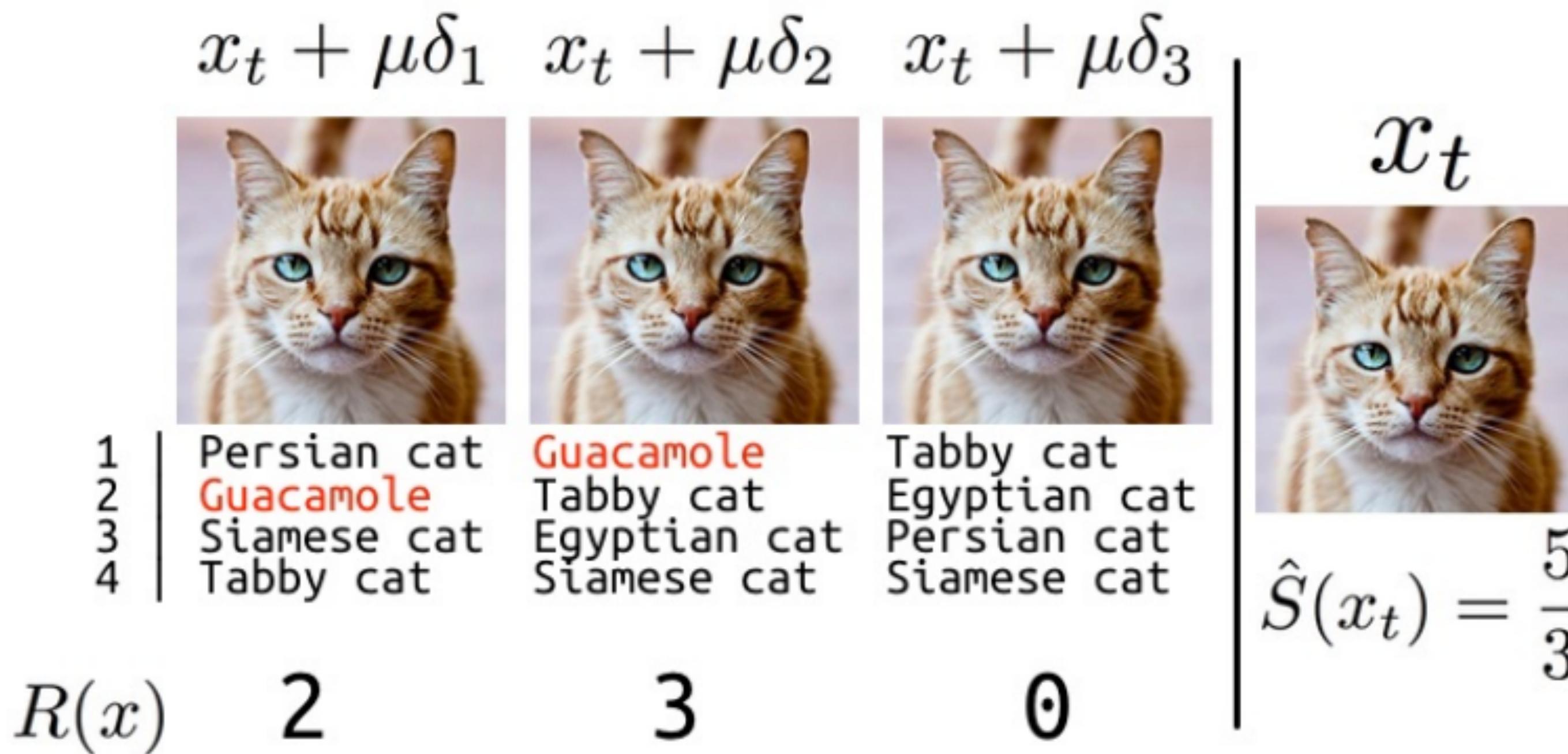
(c)  $h(f(x))$

# Boundary attack: based on random walk



# Limited attack

- Limited Attack: Monte Carlo method to get the probability output

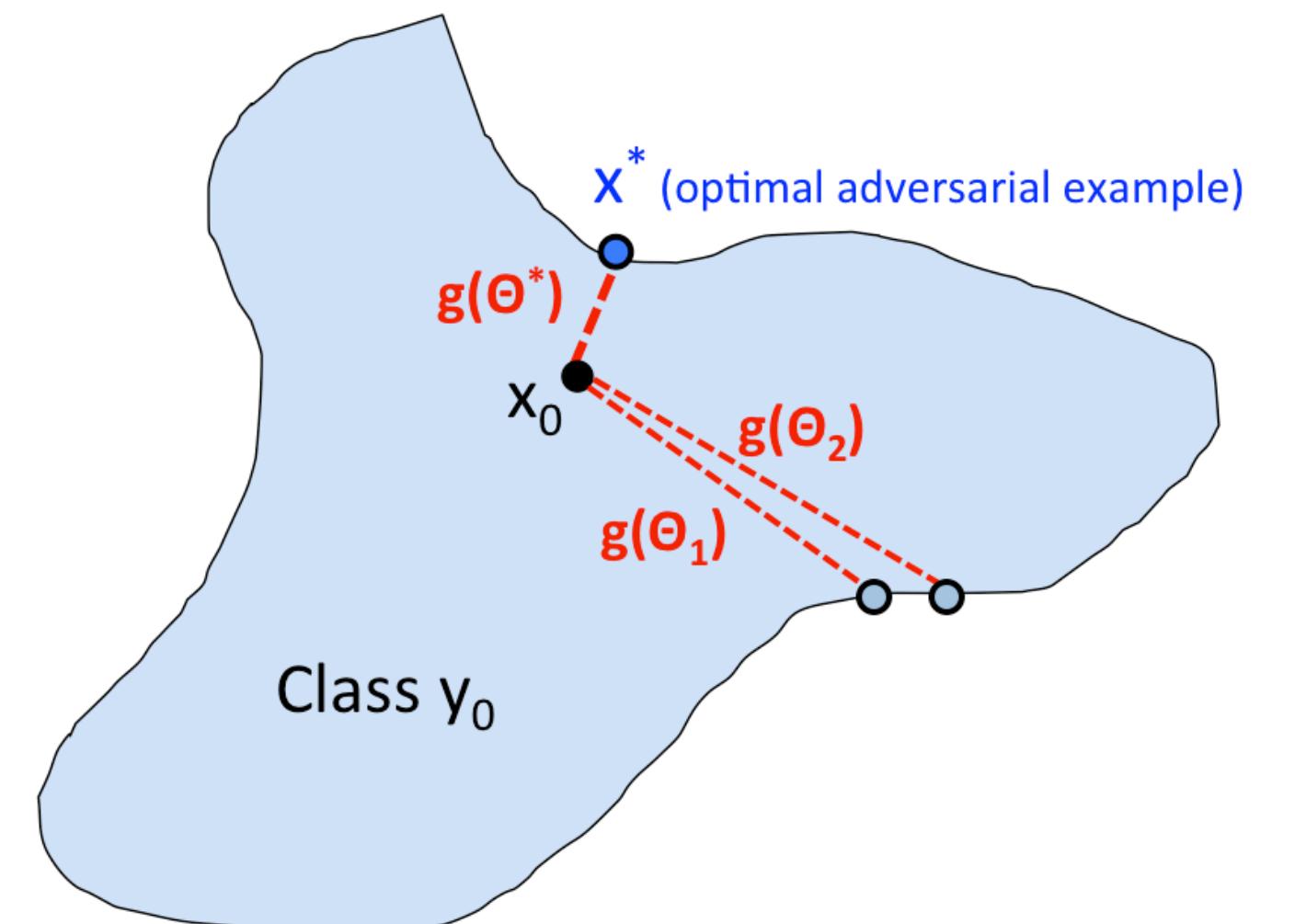


# Reformulation

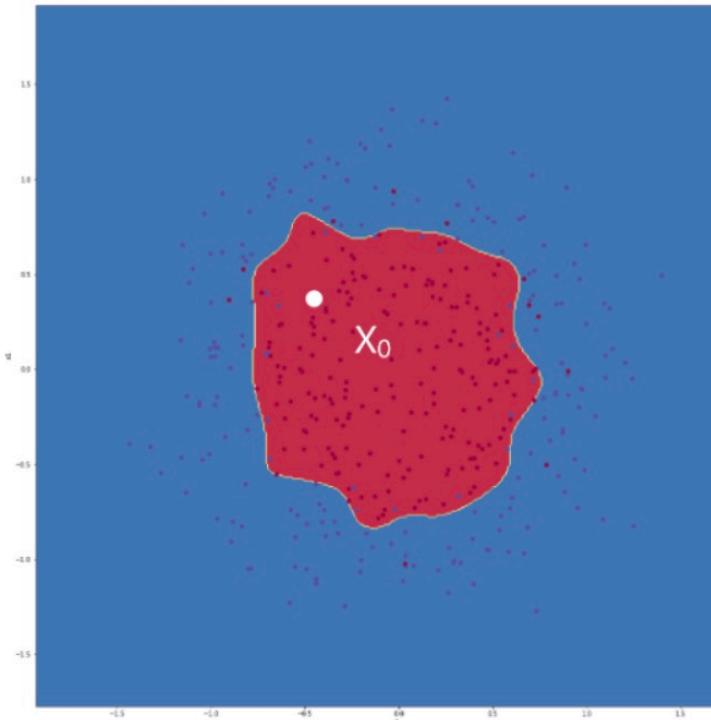
- We reformulate the attack optimization problem (untargeted attack):

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

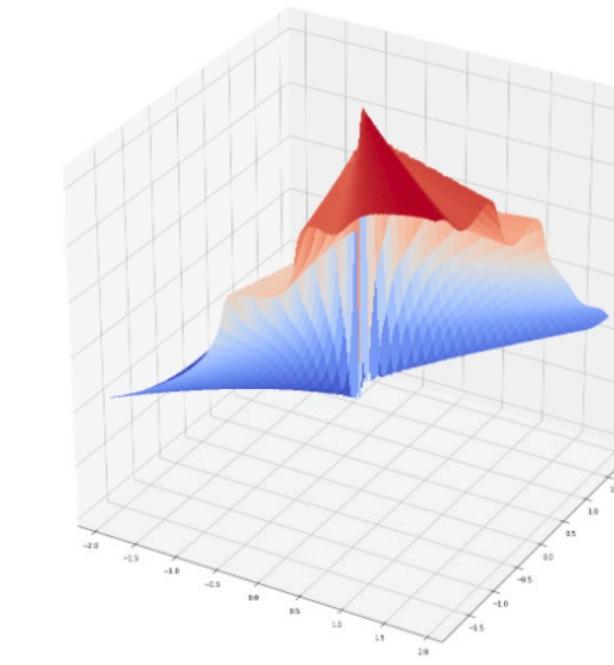
- where  $g(\theta) = \operatorname{argmin}_{\lambda > 0} \left( f(x_0 + \lambda \frac{\theta}{\|\theta\|}) \neq y_0 \right)$
- $\theta$ : the direction of adversarial example



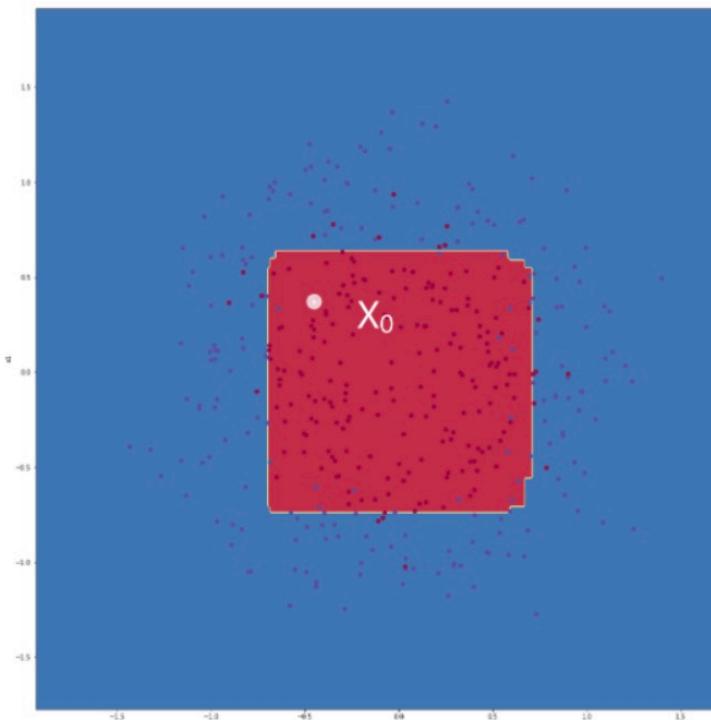
# Examples



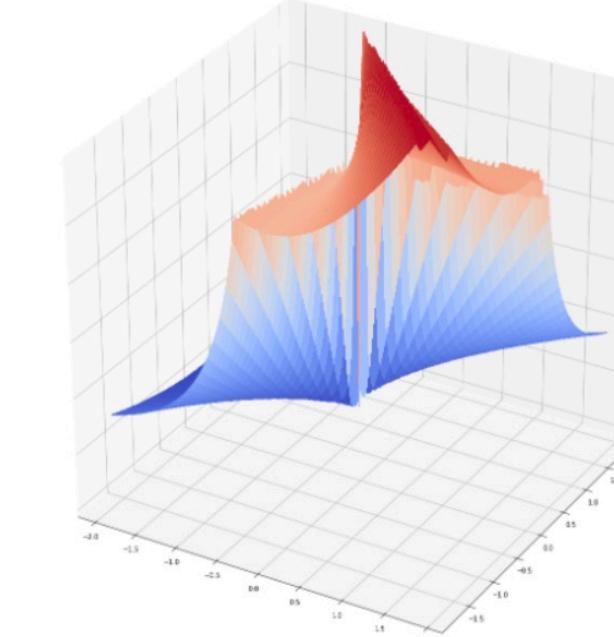
Neural network decision function



$$g(\theta)$$



Boosting Tree decision function



$$g(\theta)$$

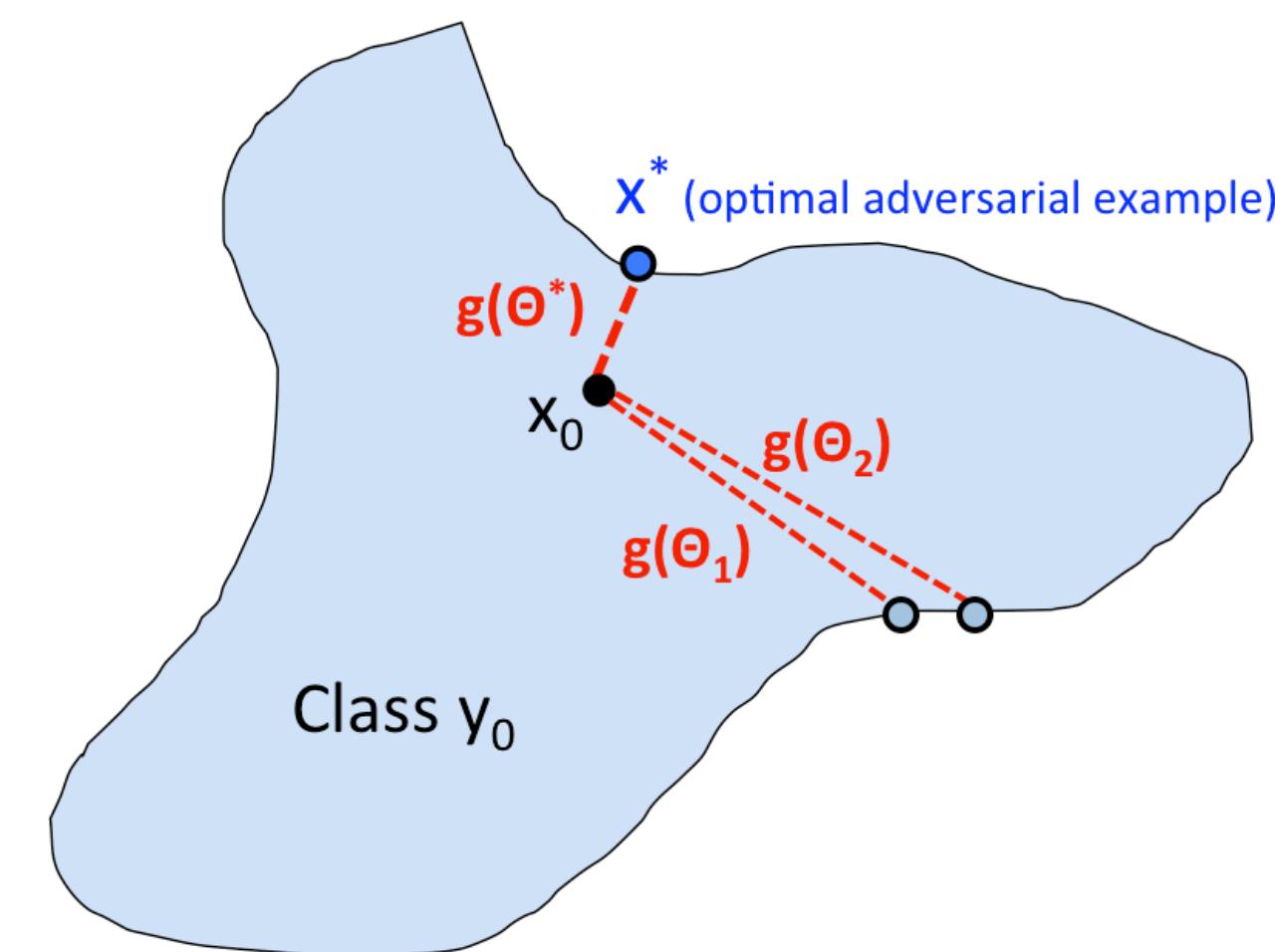
## Two things unaddressed

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

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

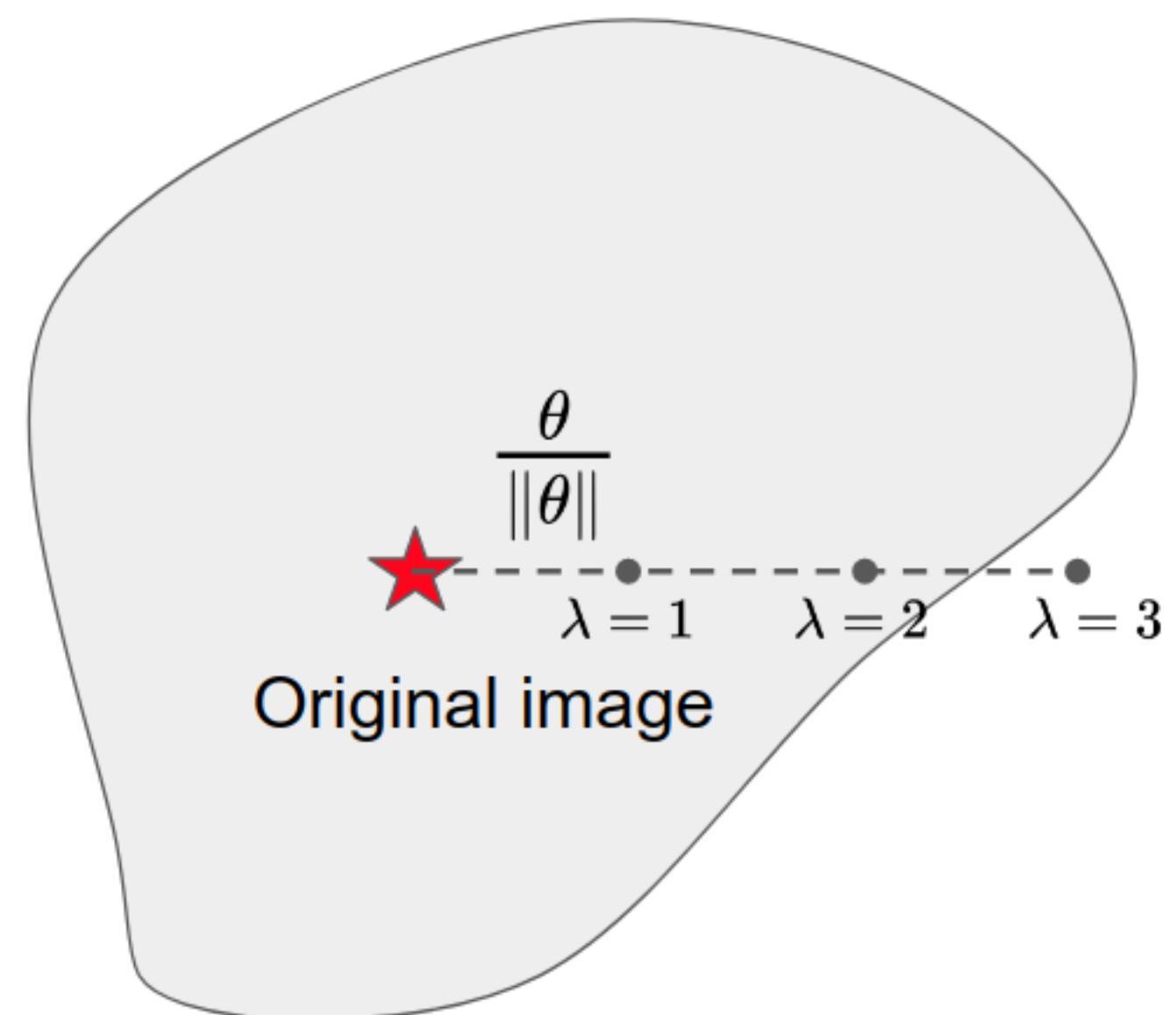
# Computing Function Value

- 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  $u$  ( $\sim 20$ )
- Zeroth order optimization for minimizing  $g(\theta)$

# Algorithm

---

## Algorithm 1 OPT attack (ICLR '19)

---

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

# Algorithm

---

## Algorithm 2 OPT attack (ICLR '19)

---

- 1: **Input:** Hard-label model  $f$ , original image  $x_0$ , initial  $\theta_0$ .
- 2: **for**  $t = 0, 1, 2, \dots, T$  **do**
- 3:     Randomly choose  $u_t$  from a zero-mean Gaussian distribution
- 4:     Evaluate  $g(\theta_t)$  and  $g(\theta_t + \beta u)$
- 5:     Compute  $\hat{g} = \frac{g(\theta_t + \beta u) - g(\theta_t)}{\beta} \cdot u$
- 6:     Update  $\theta_{t+1} = \theta_t - \eta_t \hat{g}$
- 7: **return**  $x_0 + g(\theta_T)\theta_T$

---

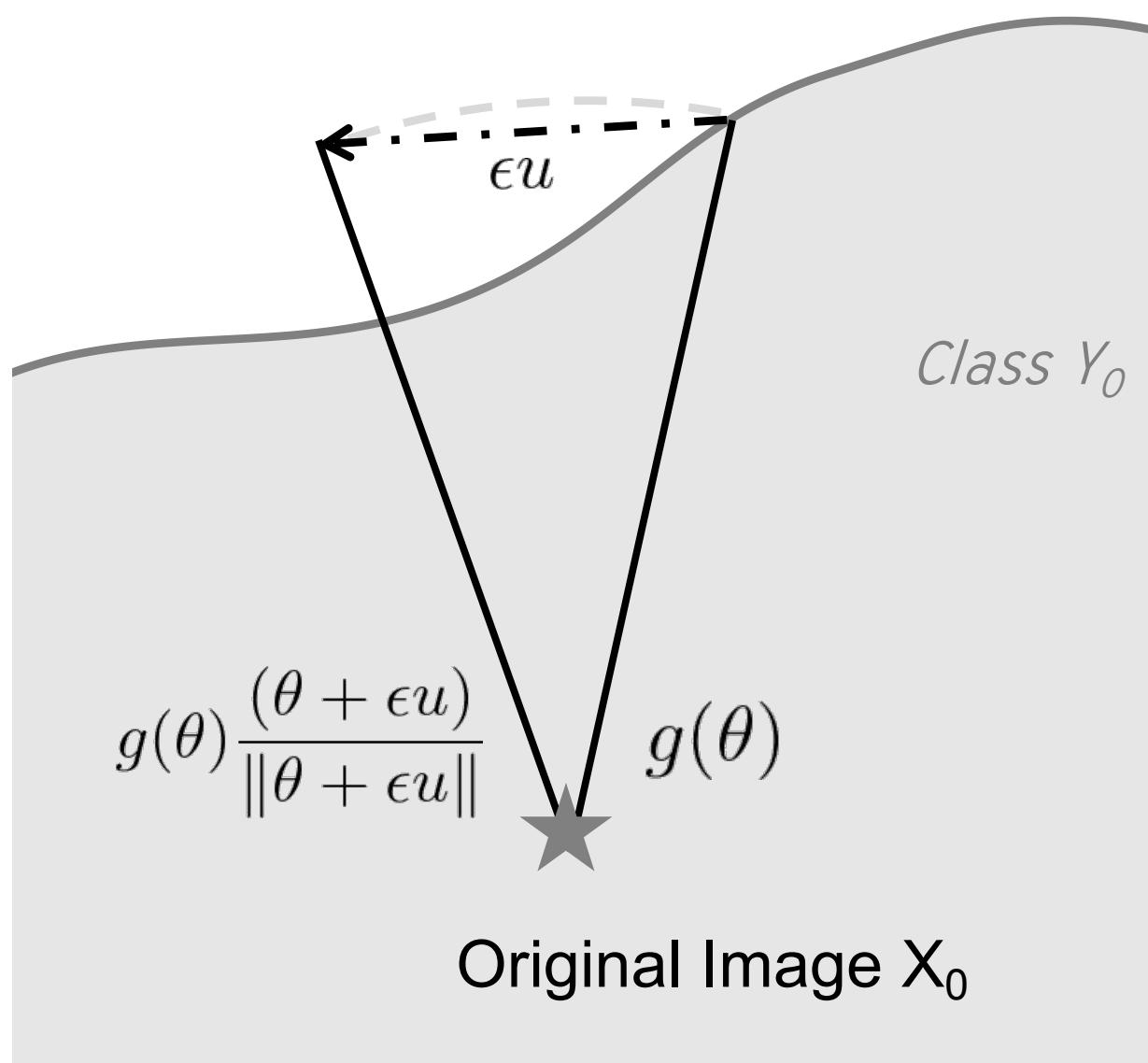
- $g(\theta_t)$  and  $g(\theta_t + \beta u)$  in the gradient estimation takes most of queries, how to further reduce it?

# 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  $\theta_0$

for  $t = 1, 2, \dots, T$  do

    Randomly sample  $u_1, \dots, u_Q$  from a Gaussian or Uniform distribution

    Evaluate  $g(\theta_t)$

$$\hat{g} = \frac{\cancel{g(\theta_t + \beta u) - g(\theta_t)}}{\cancel{\beta}} \cdot u \Rightarrow \text{sign}\left(\frac{g(\theta_t + \beta u) - g(\theta_t)}{\beta}\right) \cdot u$$

    Update  $\theta_{t+1} \leftarrow \theta_t - \eta \hat{g}$

    Evaluate  $g(\theta_t)$  using the same search algorithm

---

# Adversarial defense

## Adversarial training

- Adversarial training [MMS18]:

$$\min_{\theta} \mathbb{E}_x \left[ \max_{\|x'-x\|_\infty \leq \epsilon} loss(\theta, x') \right]$$

- TRADES

$$\min_{\theta} \mathbb{E}_x \left[ \underbrace{loss(\theta, x)}_{\text{clean acc}} + \lambda \underbrace{\max_{\|x'-x\|_\infty \leq \epsilon} loss(\theta, x')}_{\text{robust reg}} \right]$$

# Adversarial defense

## Customized adversarial training

- Adversarial training [MMS18]:

- $$\min_{\theta} \mathbb{E}_x \left[ \max_{\|x'-x\|_{\infty} \leq \epsilon} loss(\theta, x') \right]$$

- Problems:

- Same large  $\epsilon$  uniformly for all samples.
- Force the prediction to match the one-hot label

- Solutions:

- Adaptively assigns a suitable  $\epsilon$  for each example

- $$\epsilon_i = \arg \min_{\epsilon} \{ \max_{x'_i \in \mathcal{B}_p(x_i, \epsilon)} f_{\theta}(x'_i) \neq y_i \}$$

- Adaptive label smoothing

- $$\tilde{y}_i = (1 - c\epsilon_i)y_i + c\epsilon_i \text{Dirichlet}(\beta).$$

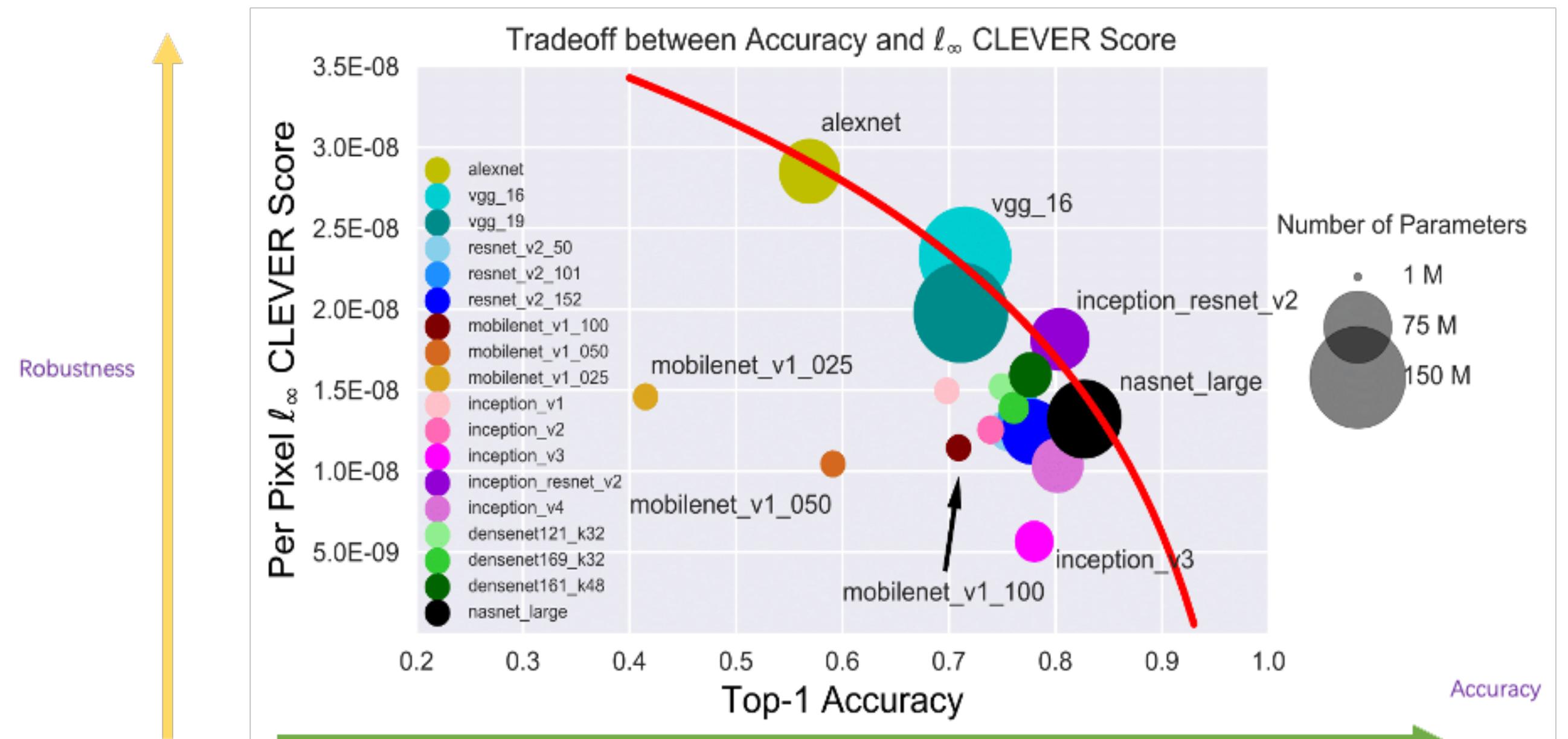
# Adversarial defense

## Limitations

- Adversarial training [MMS18]:

$$\min_{\theta} \mathbb{E}_x \left[ \max_{\|x' - x\|_\infty \leq \epsilon} loss(\theta, x') \right]$$

- Attack dependency: The **max** doesn't have a closed form solution and is normally done by using adversarial attack (i.e. need several back-propagations).
- Adversarially trained network are **sacrificing** accuracy



# Robustness verification

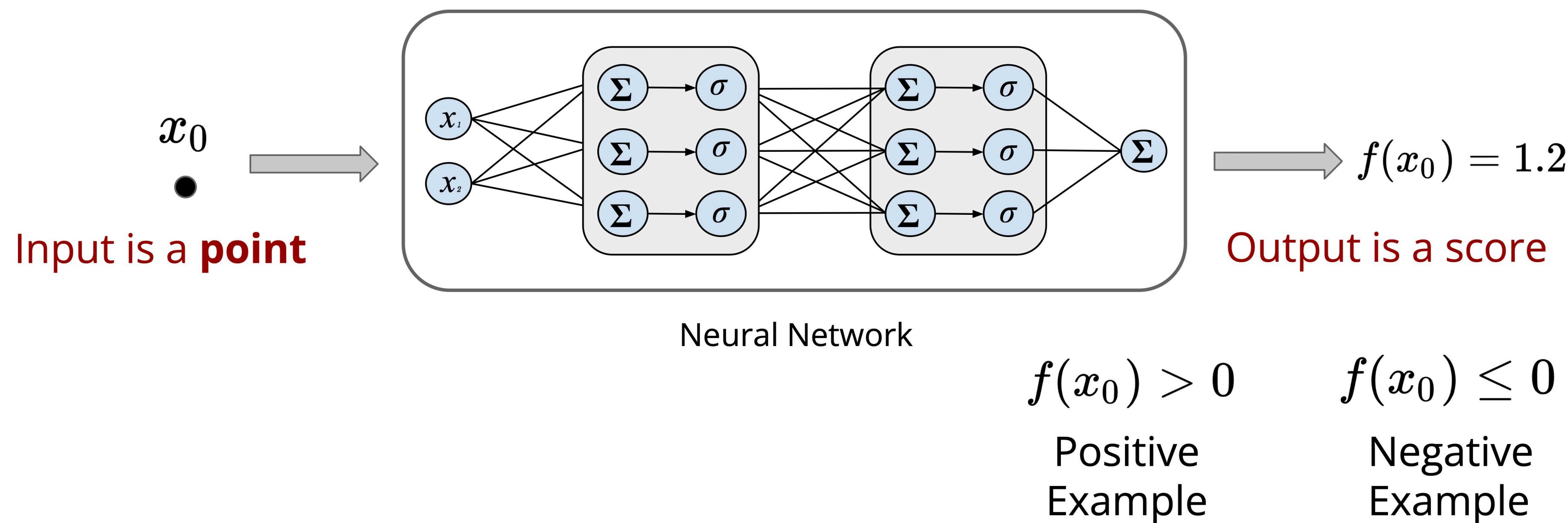
## Why

- Many heuristic defense was broken under stronger attacks
- A verified model cannot be attacked by any attacks (including unforeseen ones)

# Robustness verification

## Basic formulation

- Consider a binary classification case:

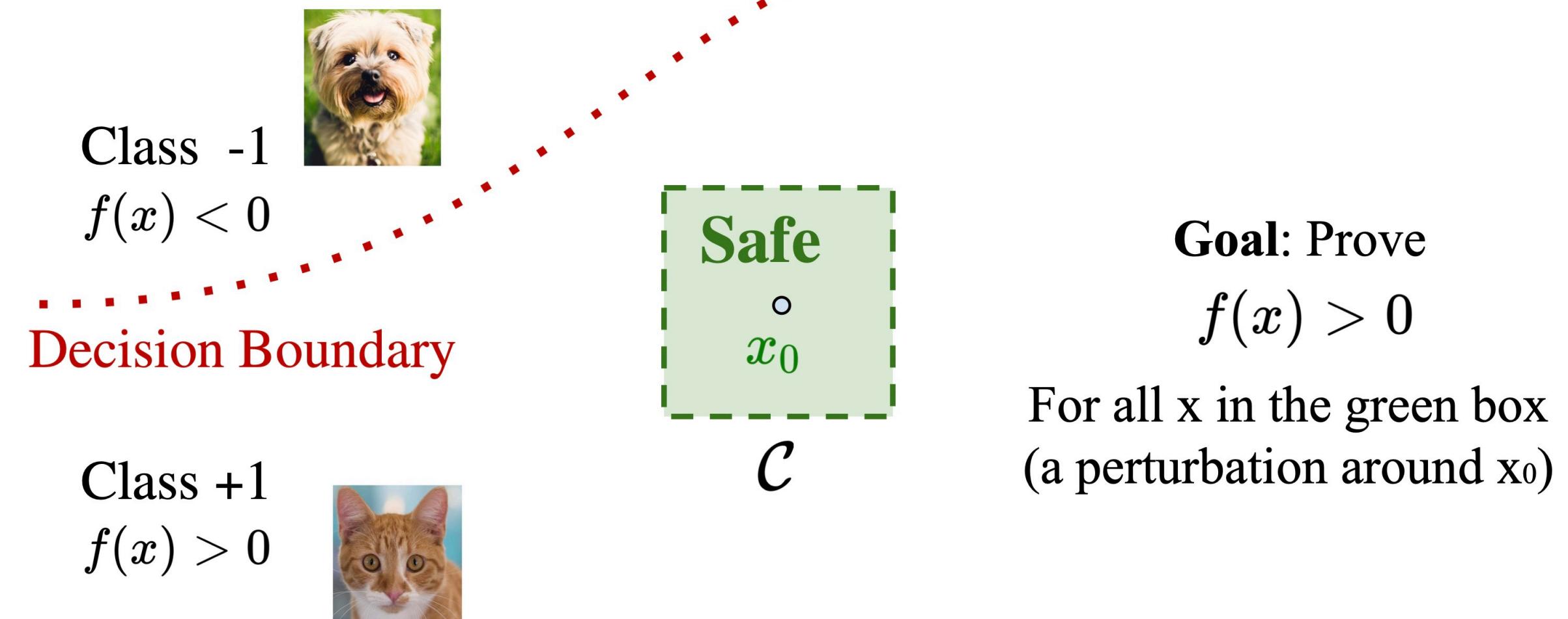


# Robustness verification

## Basic formulation

- Suppose  $f(x_0) > 0$ , can we verify this property:

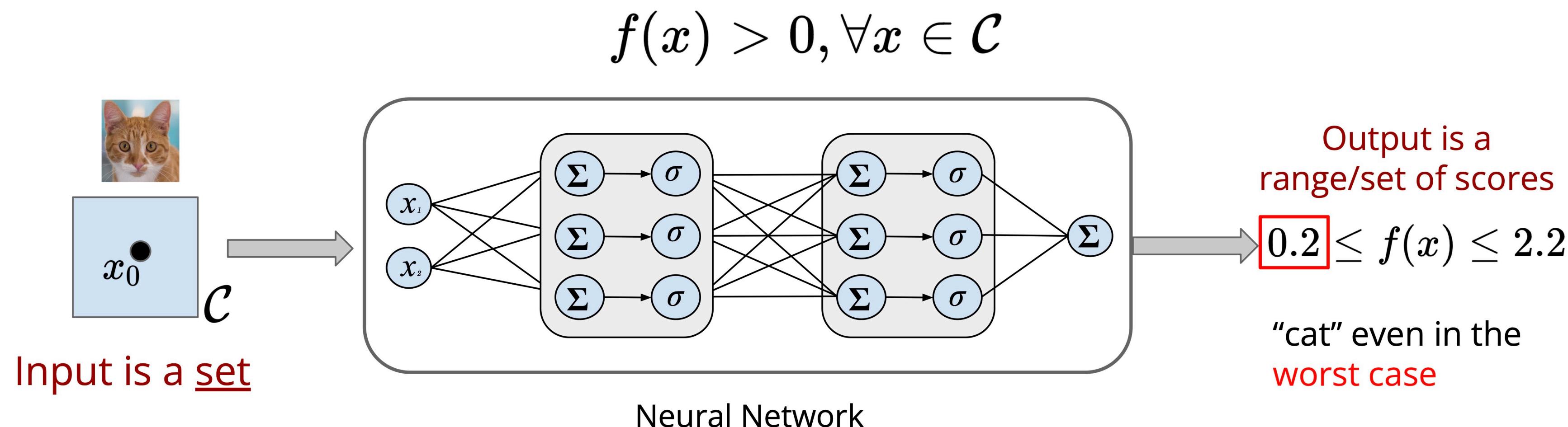
$$f(x) > 0, \forall x \in \mathcal{C}$$



# Robustness verification

## Basic formulation

- Suppose  $f(x_0) > 0$ , can we verify this property:



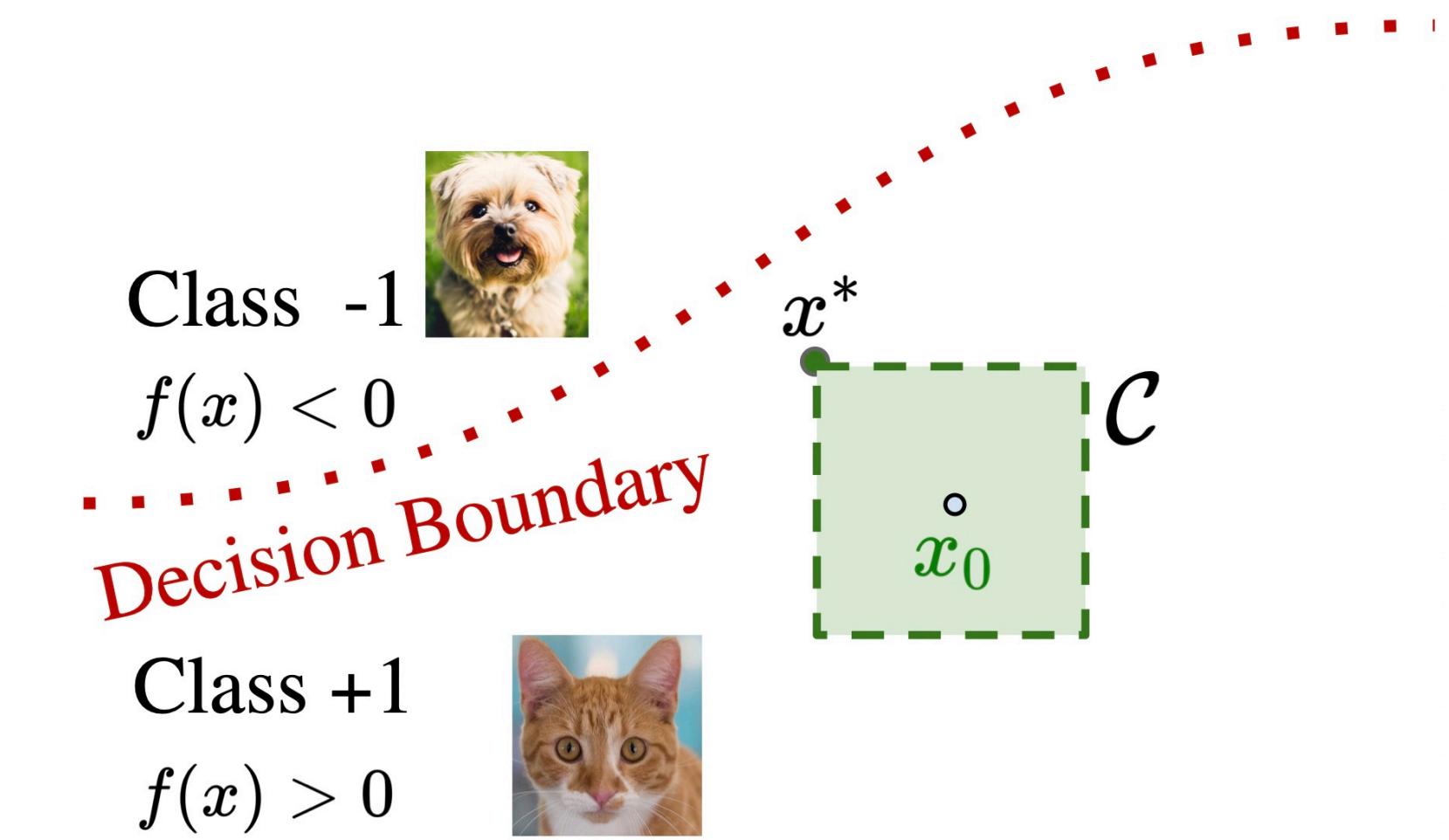
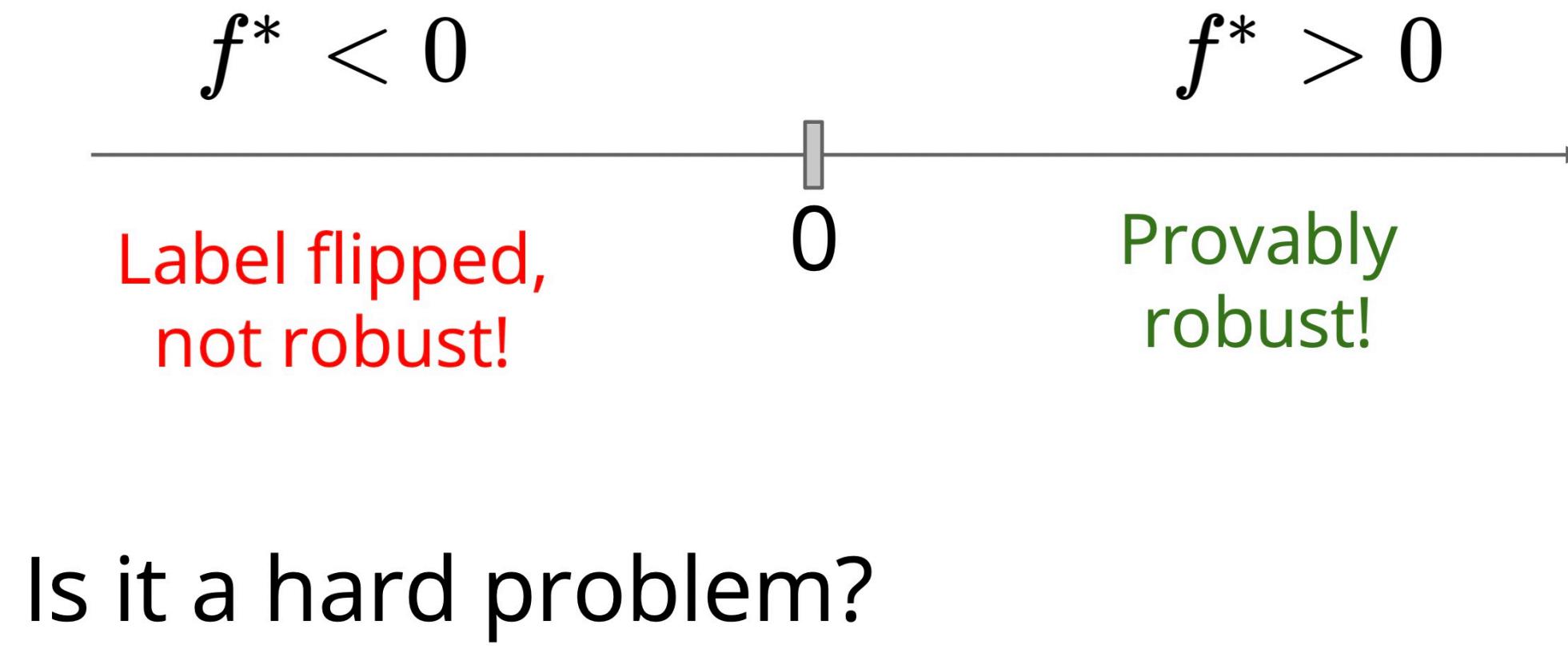
# Robustness verification

## Basic formulation

Assuming  $f(x_0) > 0$ , we solve the optimization problem to find the worst case:

$$f^* = \min_{x \in \mathcal{C}} f(x)$$

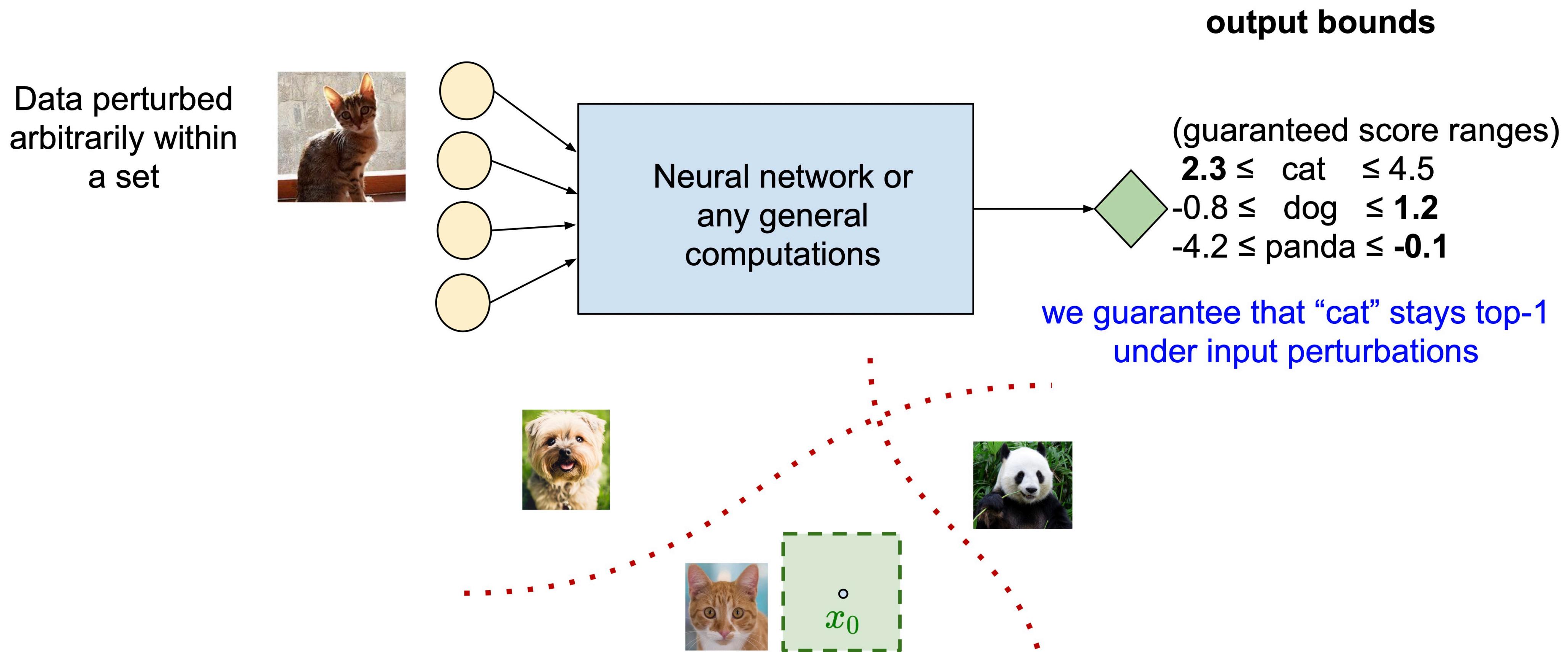
$\mathcal{C}$  is usually a perturbation set “around”  $x_0$ , e.g.,  $\mathcal{C} := \{x \mid \|x - x_0\|_p \leq \epsilon\}$



# Robustness verification

## Basic formulation

Multi-class case:



# Robustness verification

## How to solve?

This is the fundamental problem we want to solve (Wong & Kolter 2018, Salman et al. 2019):

$$\begin{aligned} f^* = \min z^{(L)} && \text{Last layer output } f(x), \text{ at layer L} \\ \text{pre-activation} \quad \text{s.t.} \quad z^{(i)} = W^{(i)} \hat{z}^{(i-1)} + b^{(i)} && i \in \{1, \dots, L\} \quad \text{Linear constraints} \\ \hat{z}^{(i)} = \sigma(z^{(i)}) && i \in \{1, \dots, L-1\} \quad \text{Non-linear, non-convex constraints} \\ \text{post-activation} \quad \hat{z}^{(0)} = x, \quad x \in \mathcal{C} && \text{Input perturbations} \end{aligned}$$

