

CS 6290

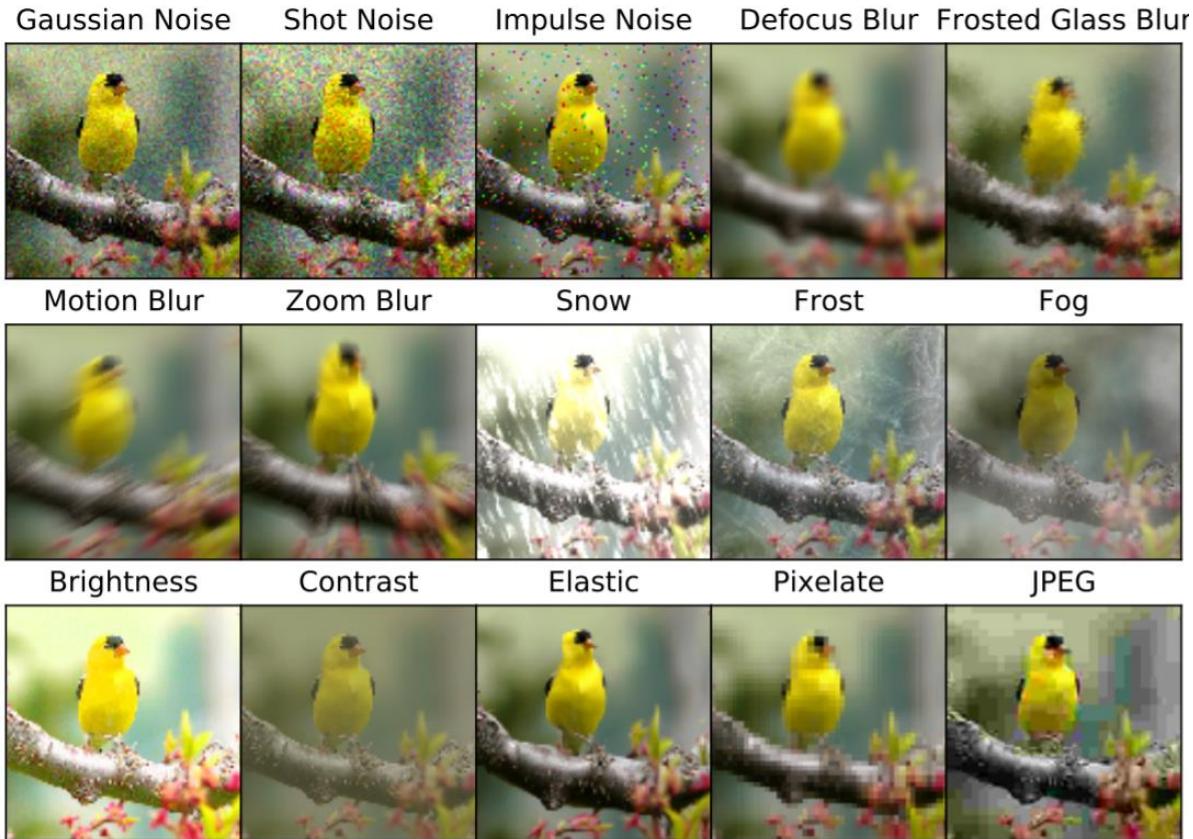
Privacy-enhancing Technologies

Department of Computer Science

Tutorial 11 – Adversarial Examples and Defenses

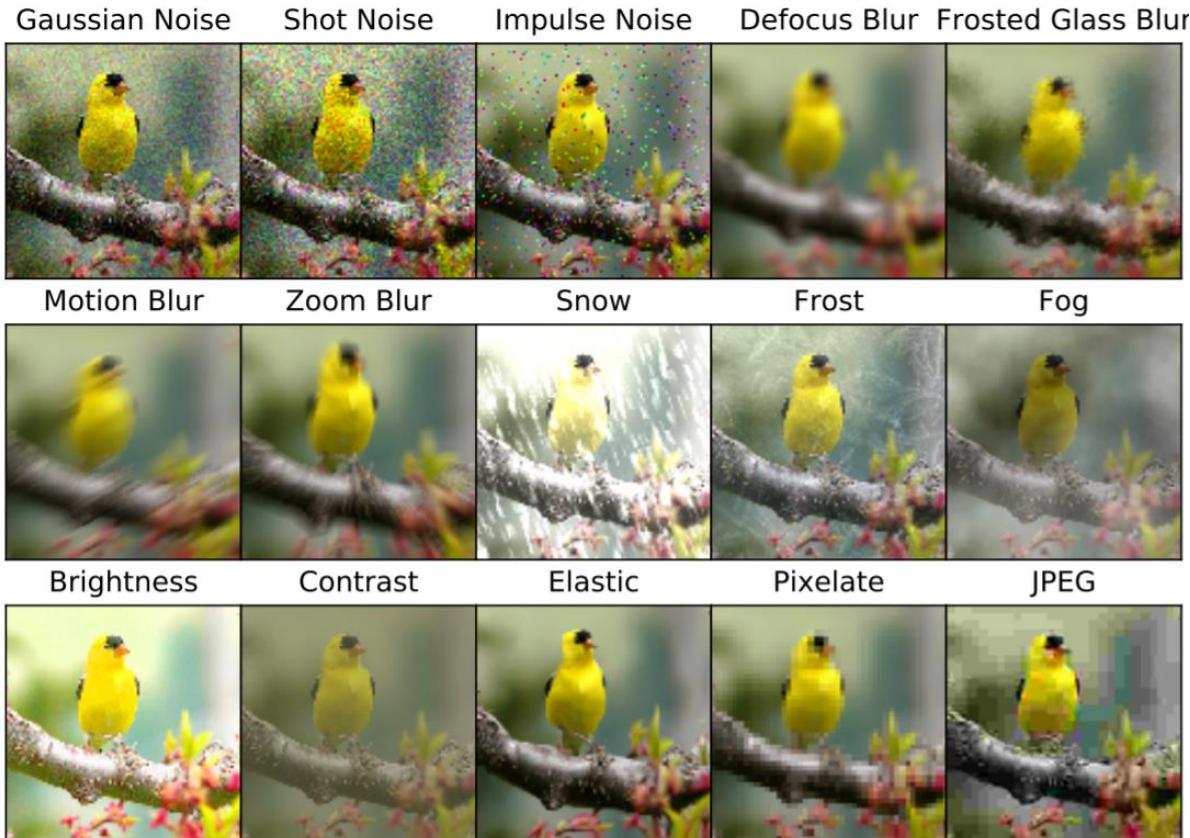
Yufei CHEN
CS Department
City University of Hong Kong

Noisy Examples



Average-case

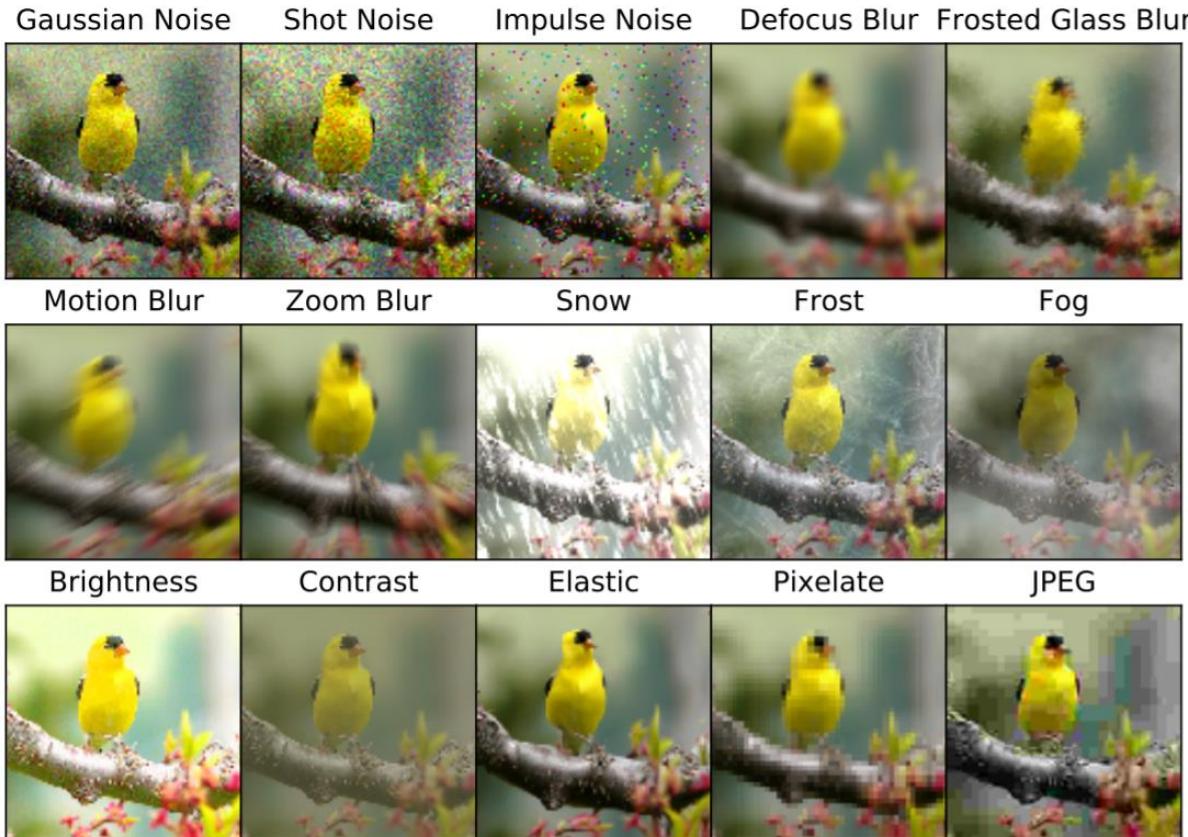
Noisy Examples → Adversarial Examples



Worst-case
(optimized)

Average-case

Noisy Examples → Adversarial Examples



Average-case

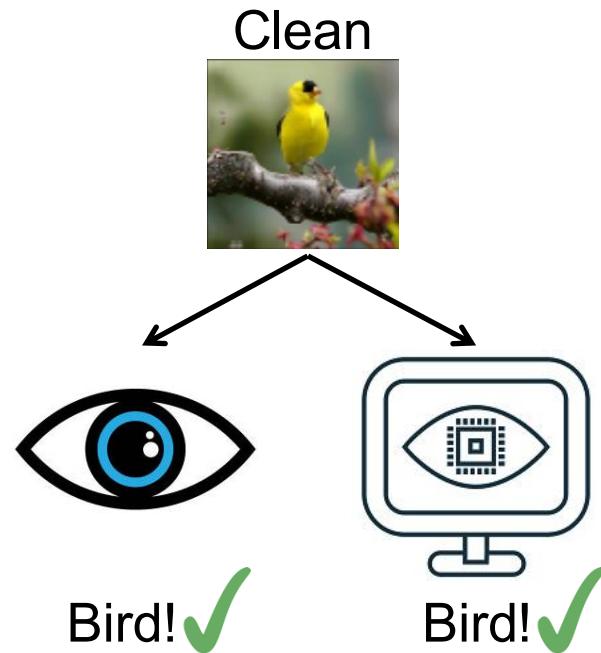
2013
Christian Szegedy,
Ian Goodfellow
Intriguing properties of
neural networks



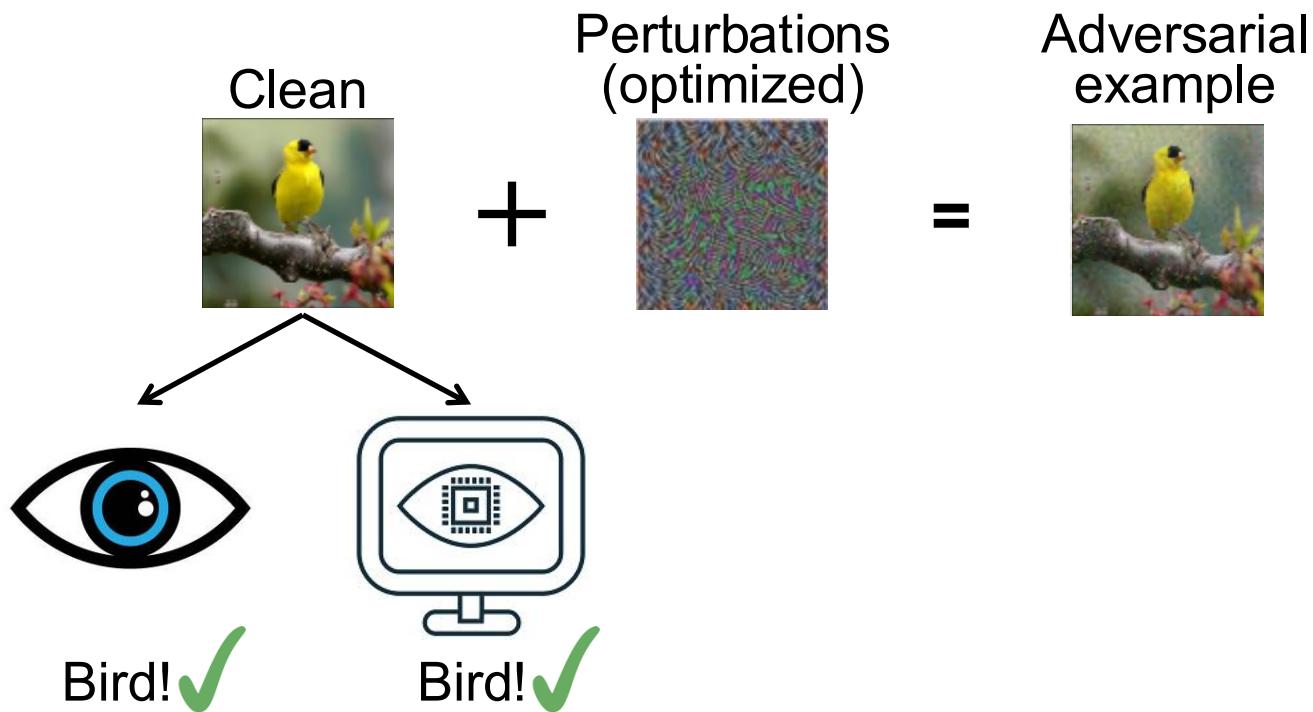
Worst-case
(optimized)

2014
Ian Goodfellow
Explaining and
harnessing adversarial
examples

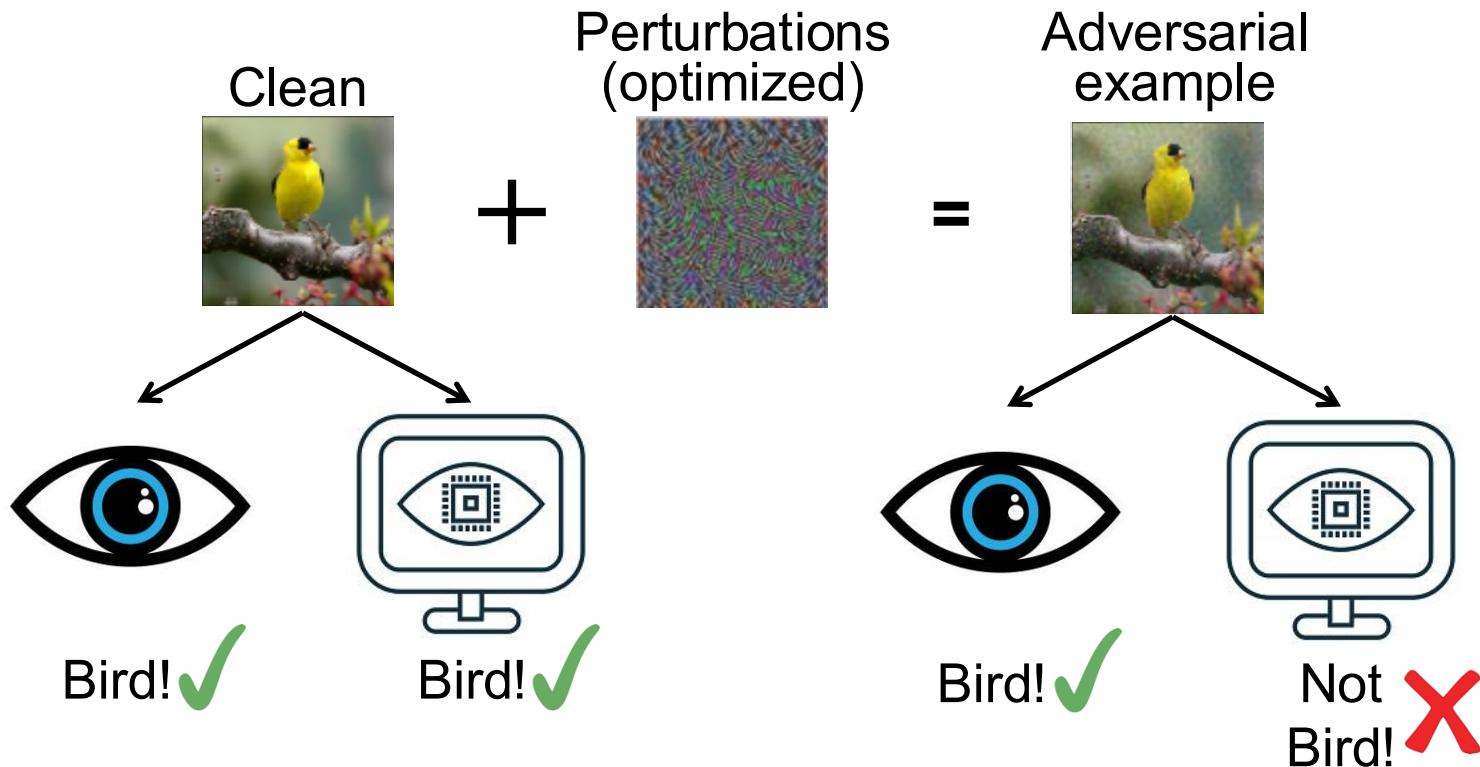
Definition



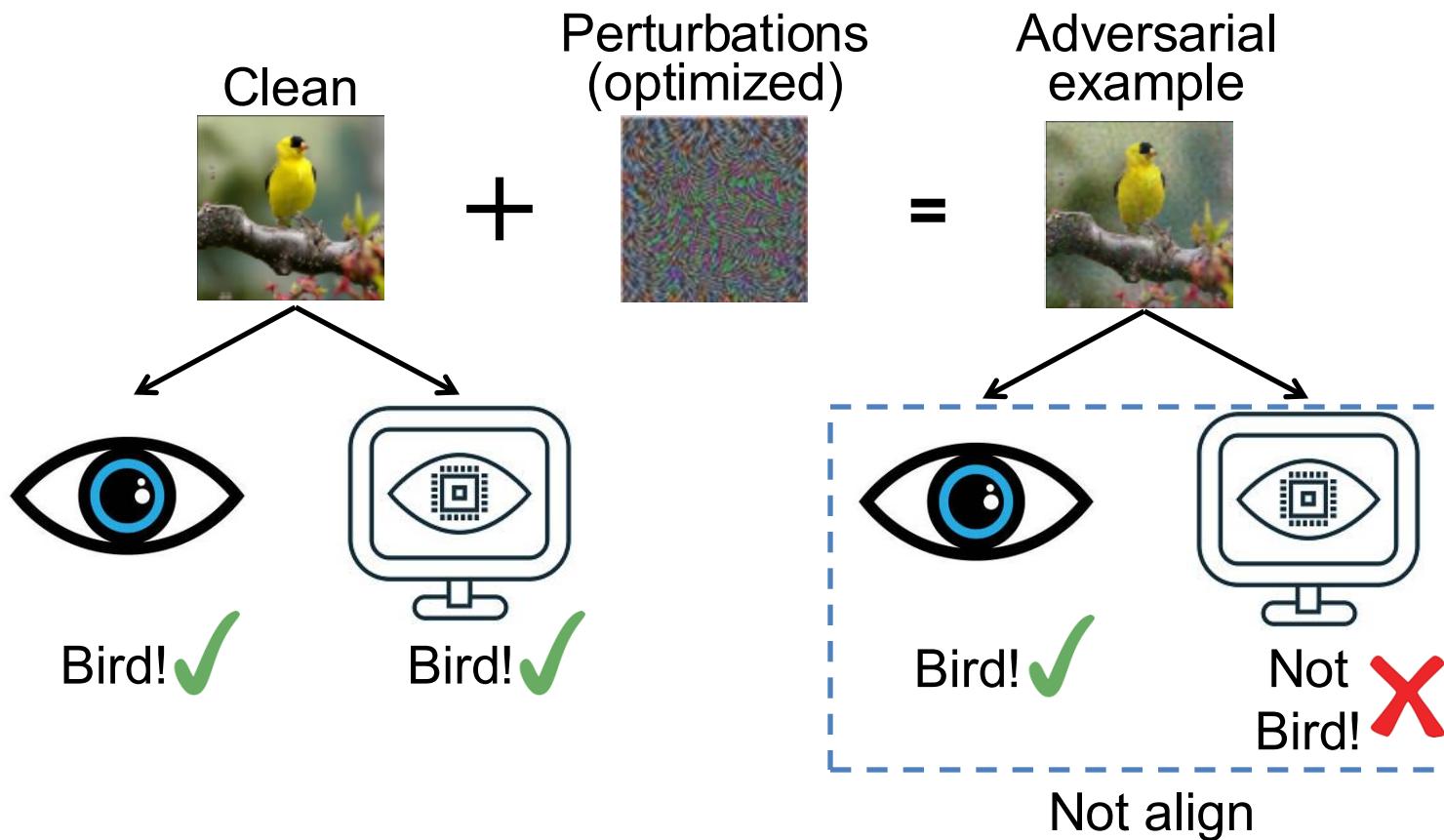
Definition



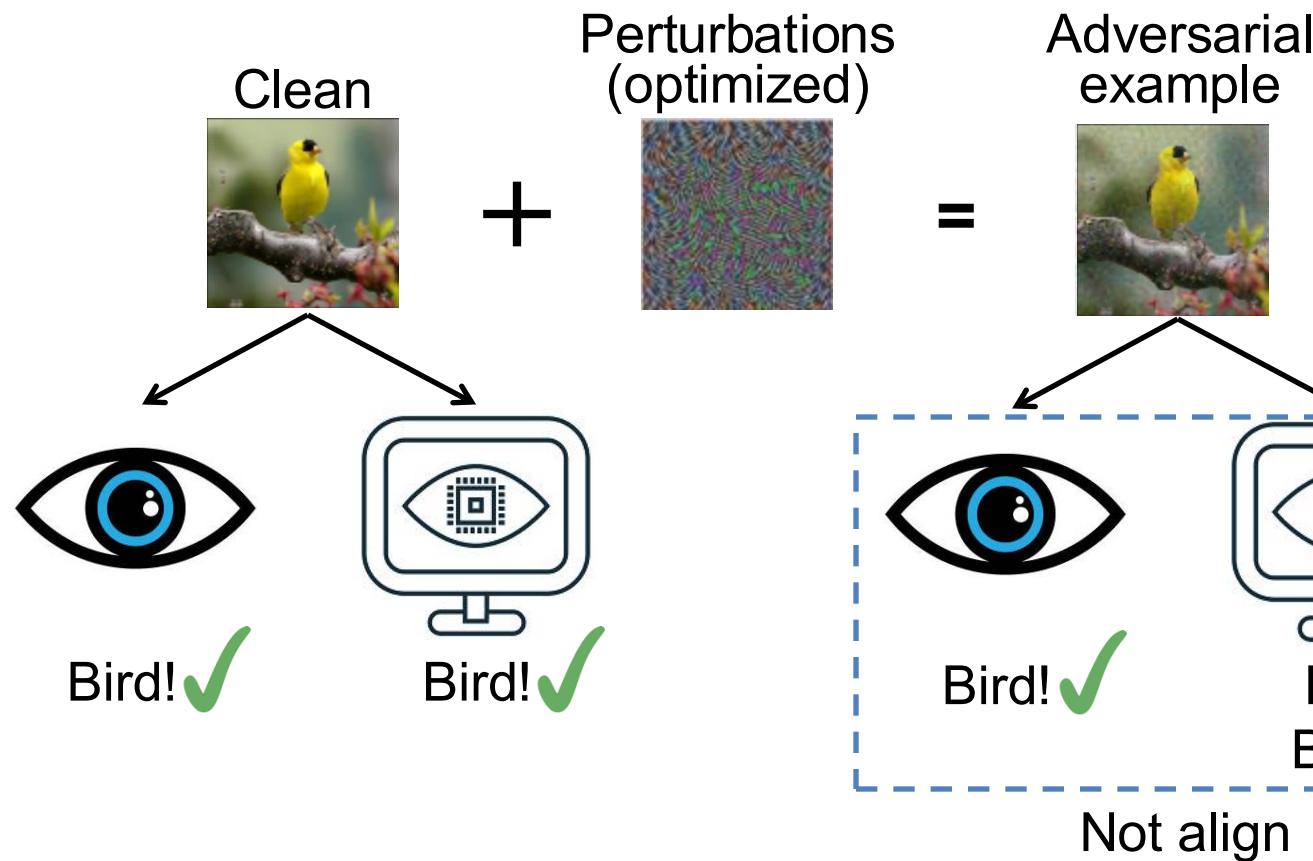
Definition



Definition



Definition



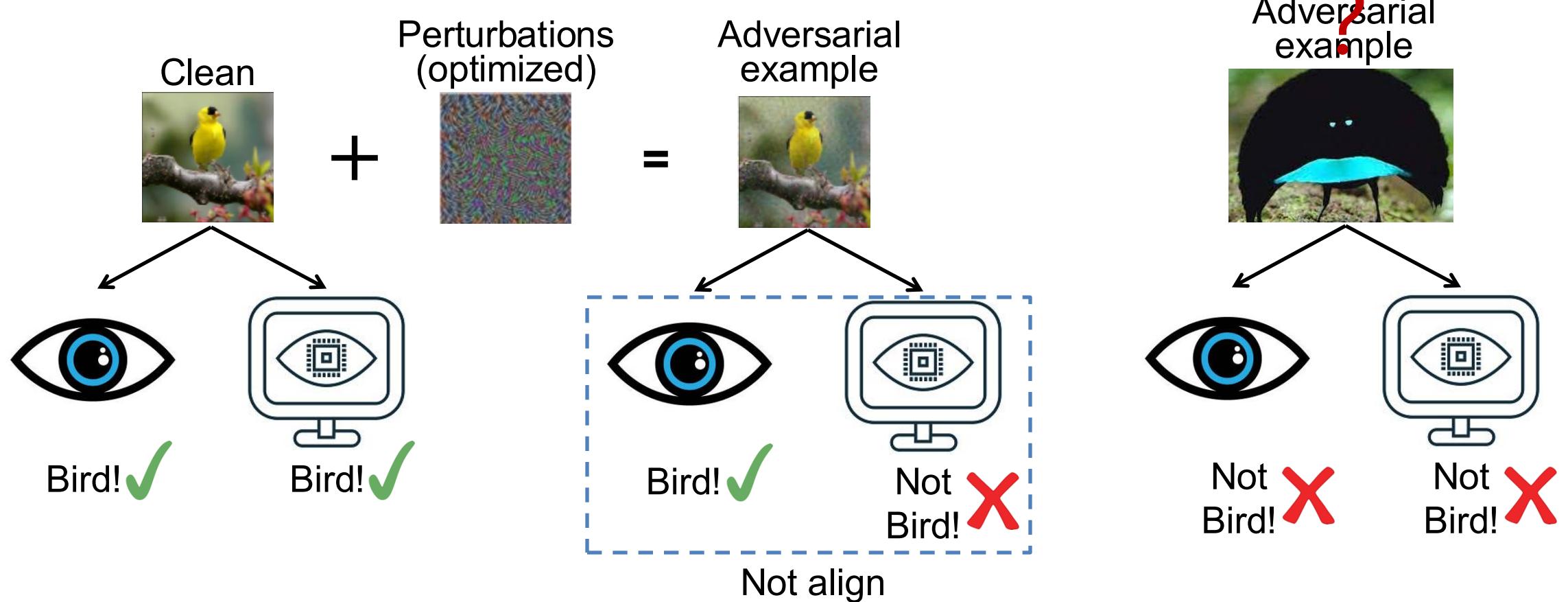
CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart)

Please click each image containing a planet

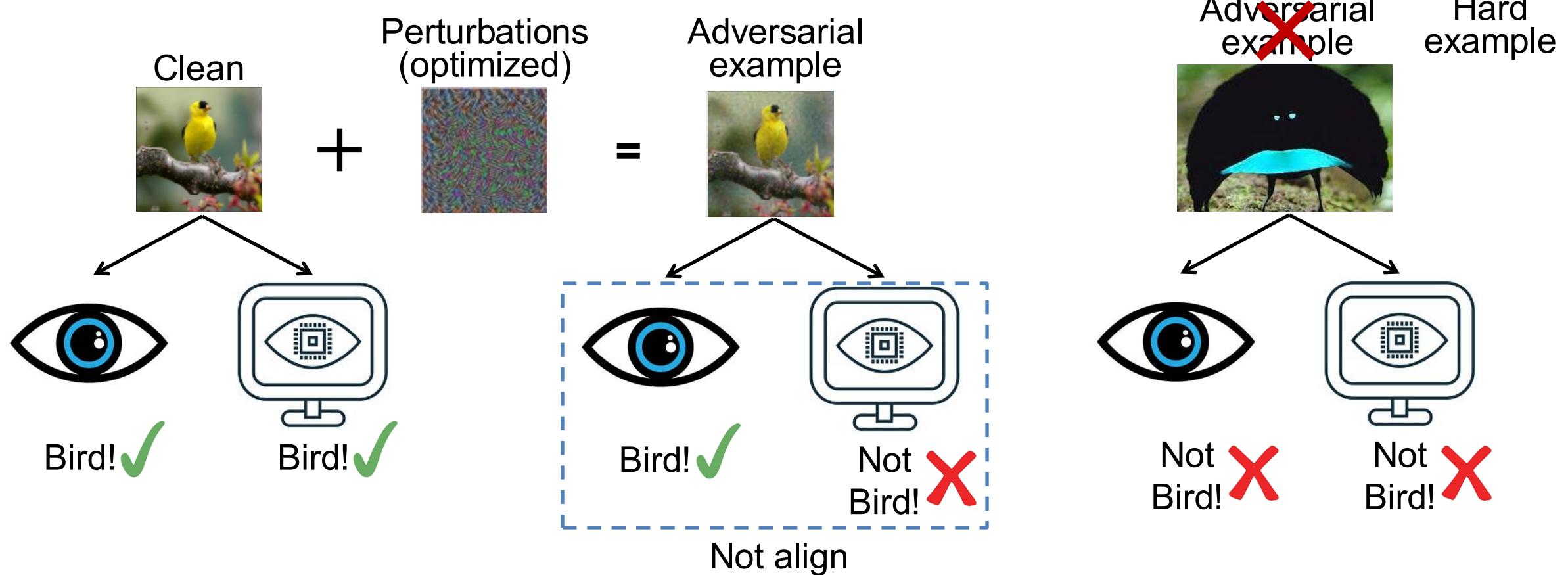
If there are None, click Skip



Definition



Definition



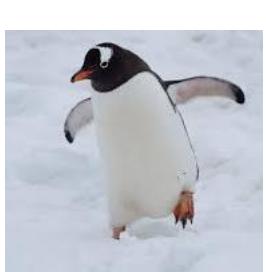
Why do Adversarial Examples Exist?

Train



Why do Adversarial Examples Exist?

Train



Test



Why do Adversarial Examples Exist?

Train



Test



Why do Adversarial Examples Exist?

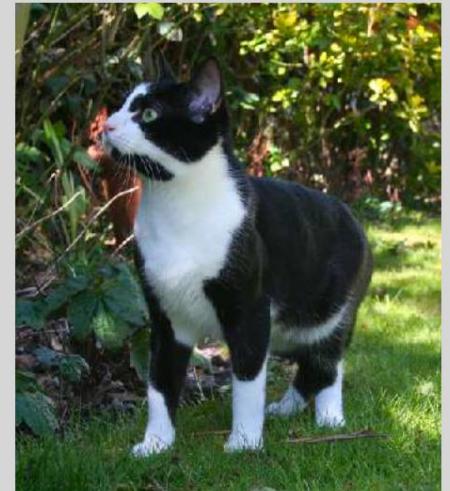
Training data
“dog”



features

- sunny
- dog
- moving

Training data
“cat”

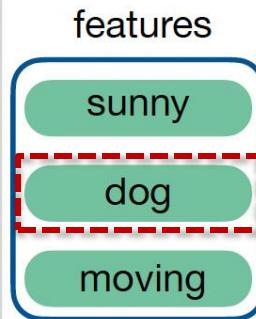
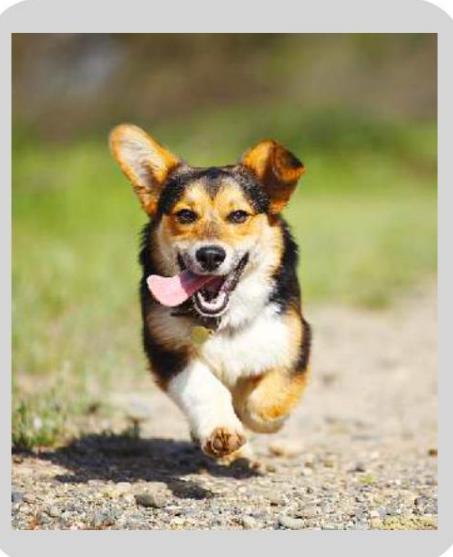


features

- shady
- cat
- still

Why do Adversarial Examples Exist?

Training data
“dog”

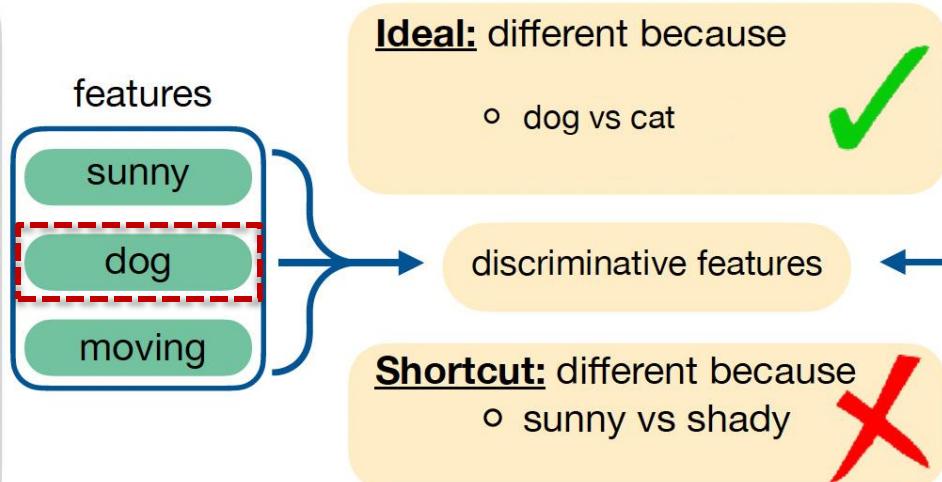


Training data
“cat”



Why do Adversarial Examples Exist?

Training data
“dog”



Training data
“cat”



Why do Adversarial Examples Exist?

Train



Test



Why do Adversarial Examples Exist?

Train



Test



Why do Adversarial Examples Exist?

Train



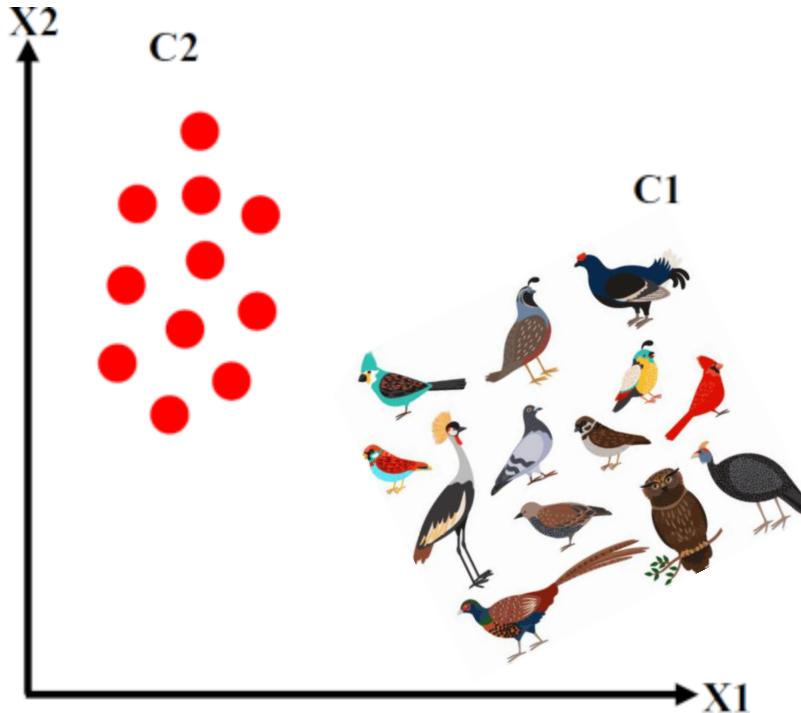
Test



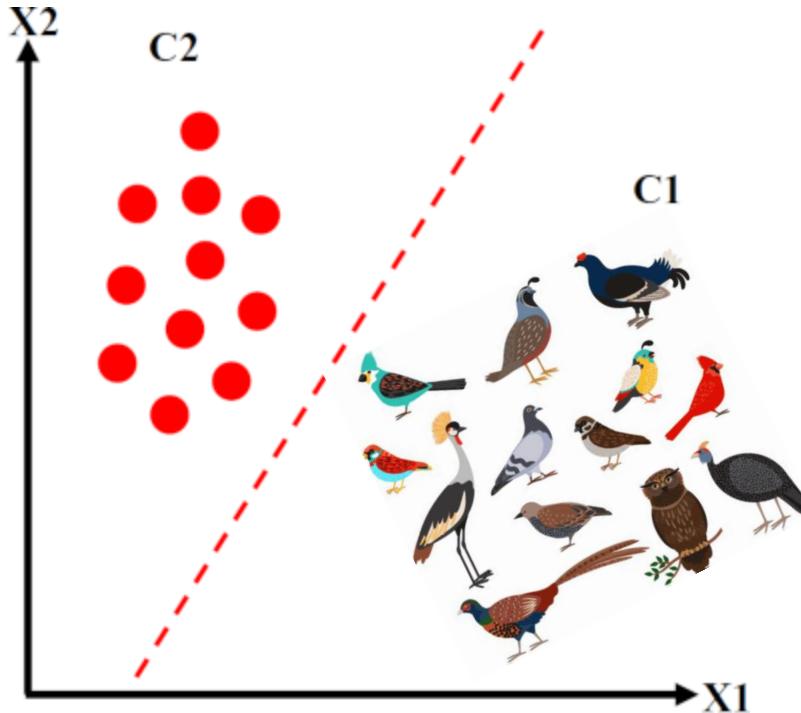
Noise
(optimized)



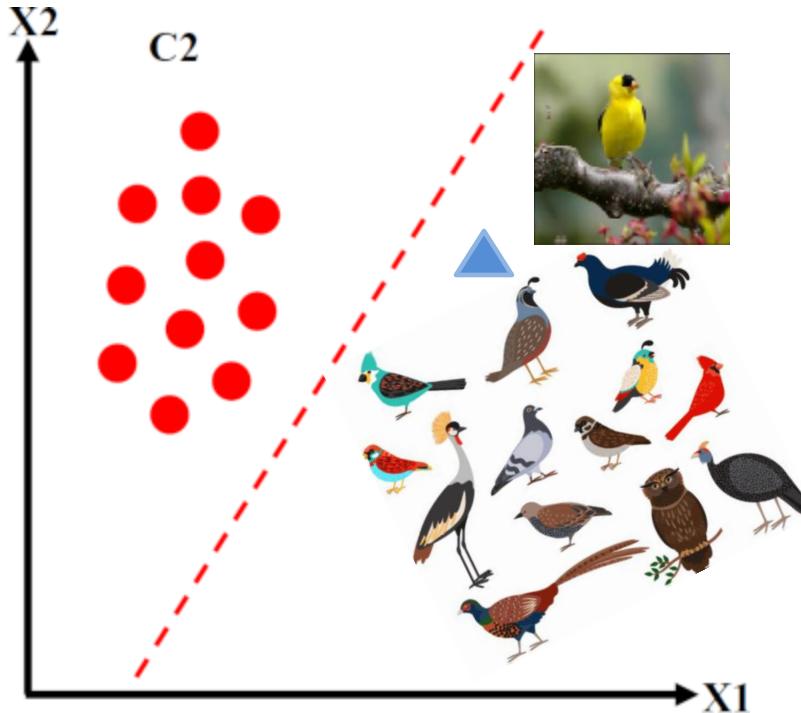
How Can We Generate Adversarial Examples?



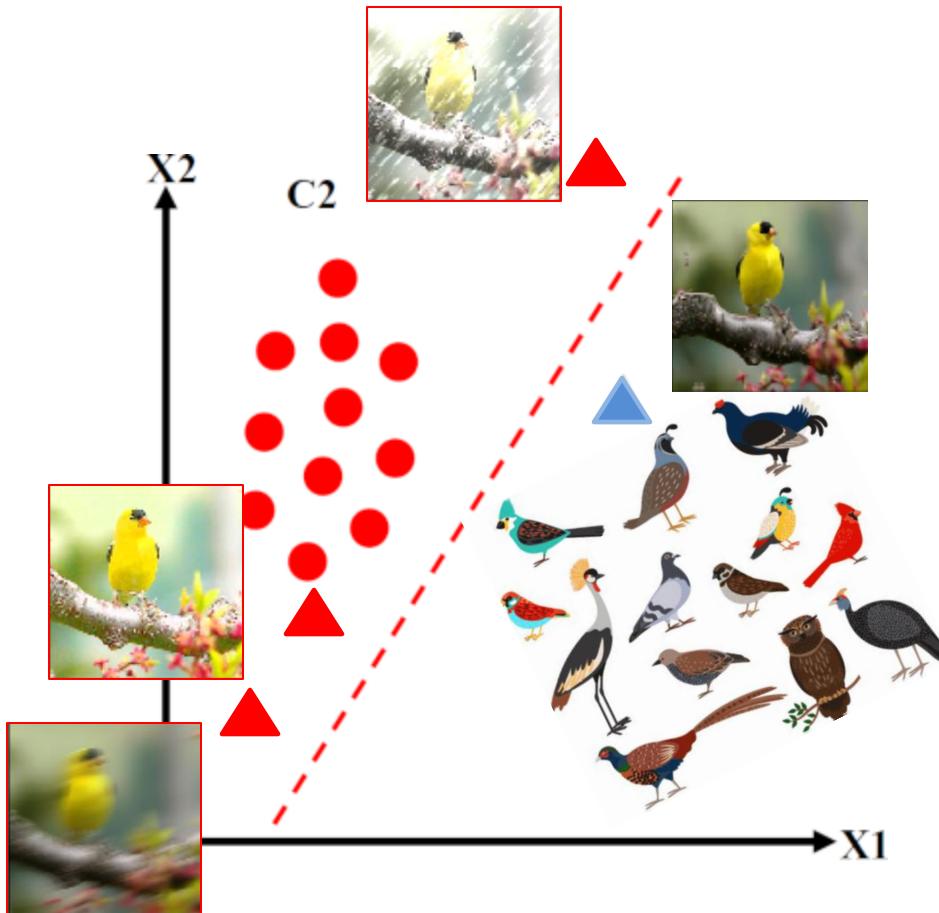
How Can We Generate Adversarial Examples?



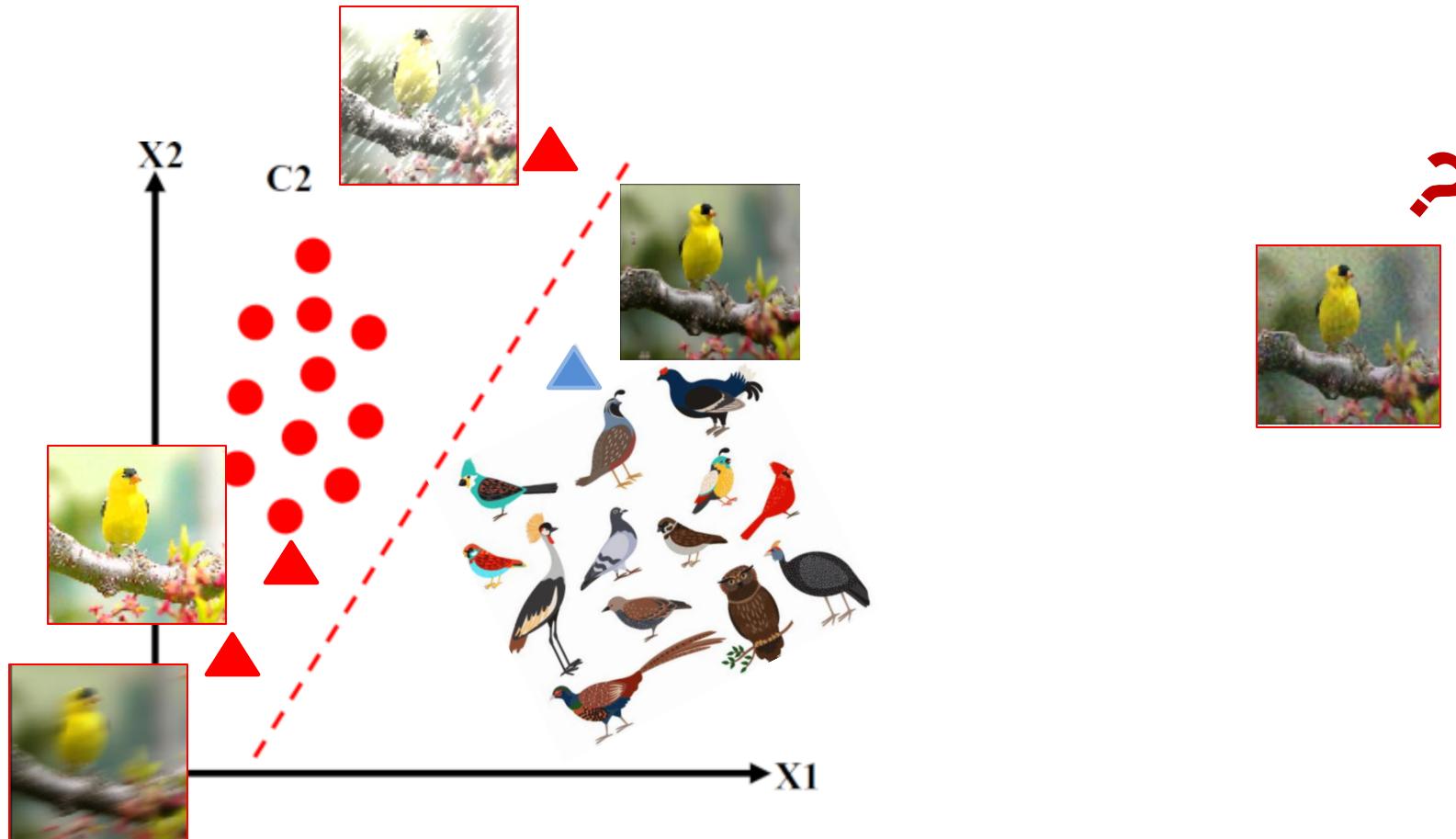
How Can We Generate Adversarial Examples?



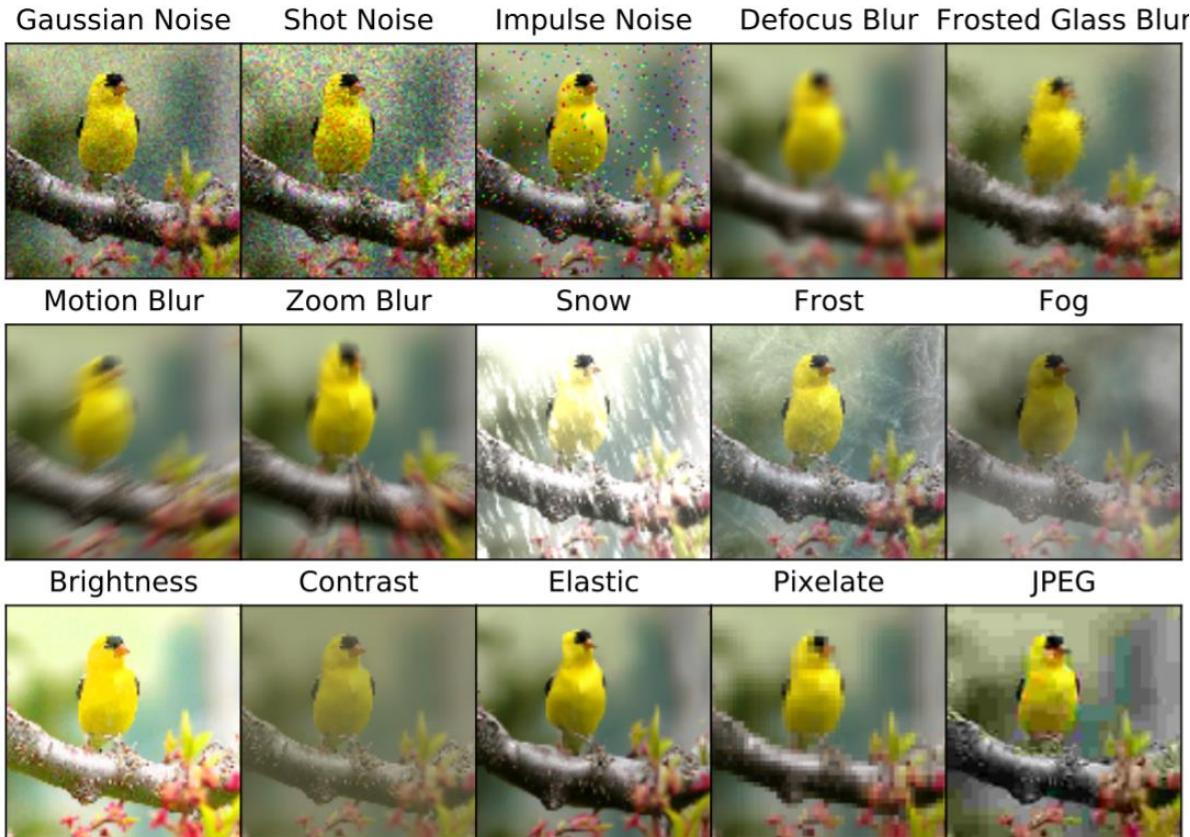
How Can We Generate Adversarial Examples?



How Can We Generate Adversarial Examples?



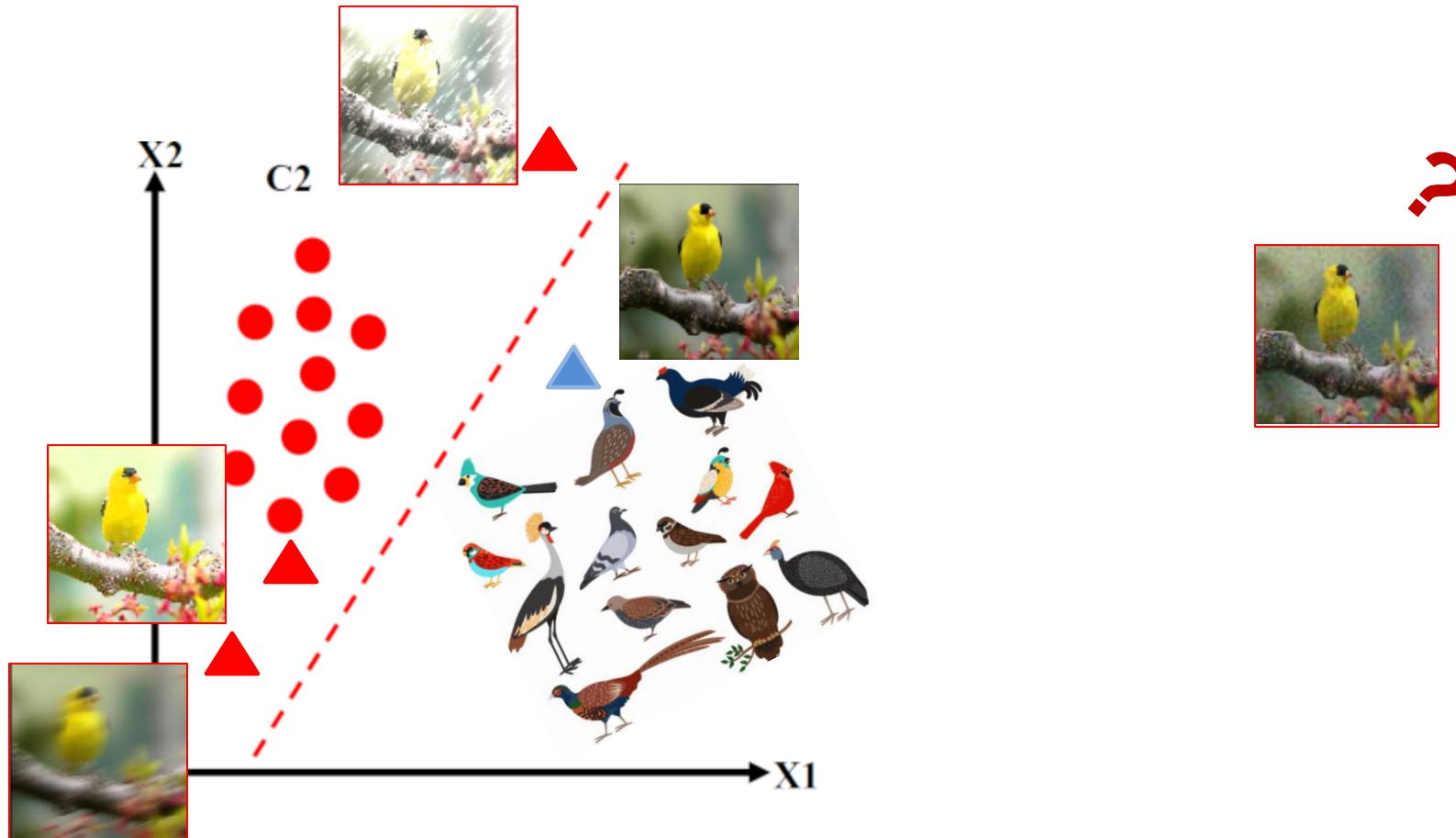
Noisy Examples → Adversarial Examples



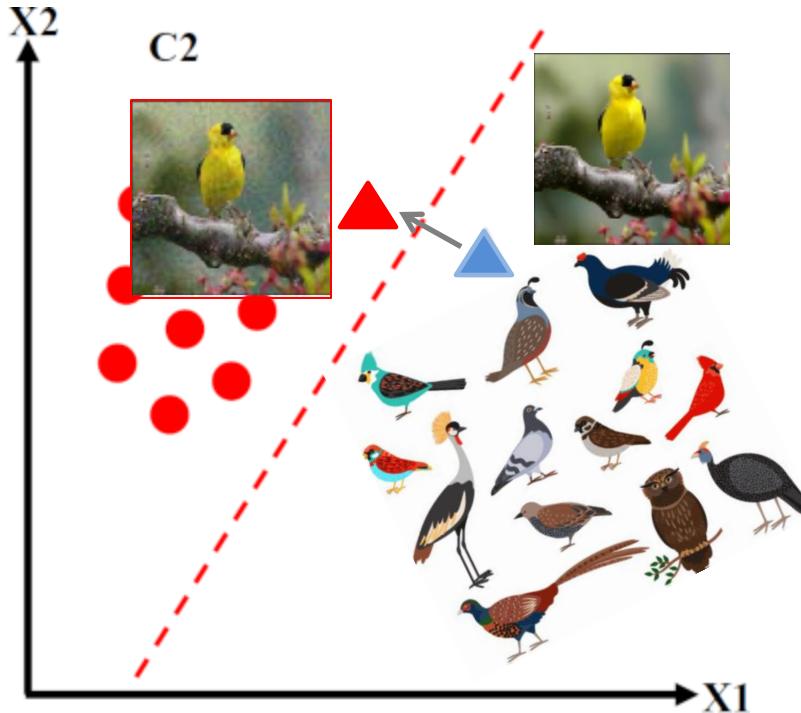
Worst-case
(optimized)

Average-case

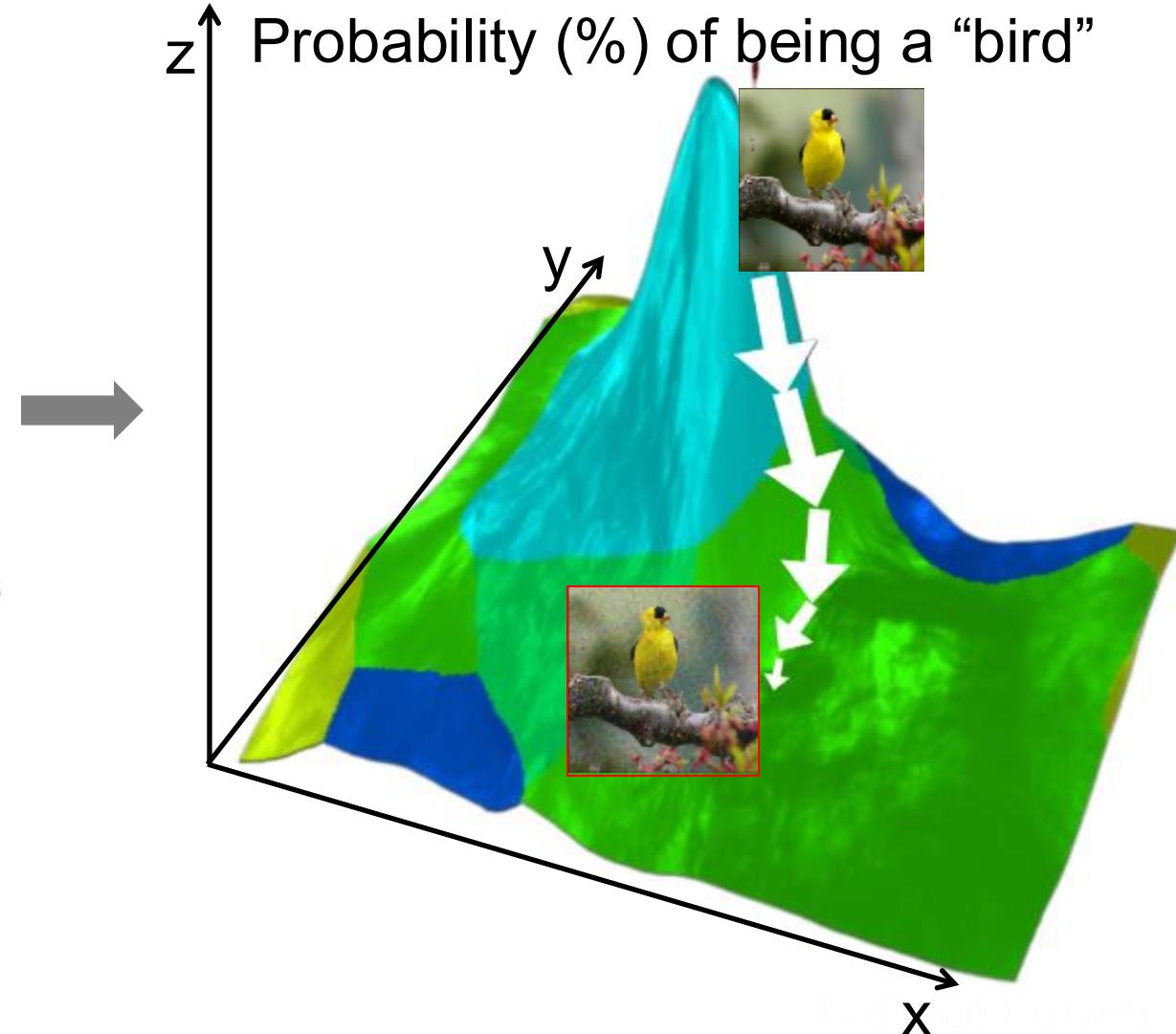
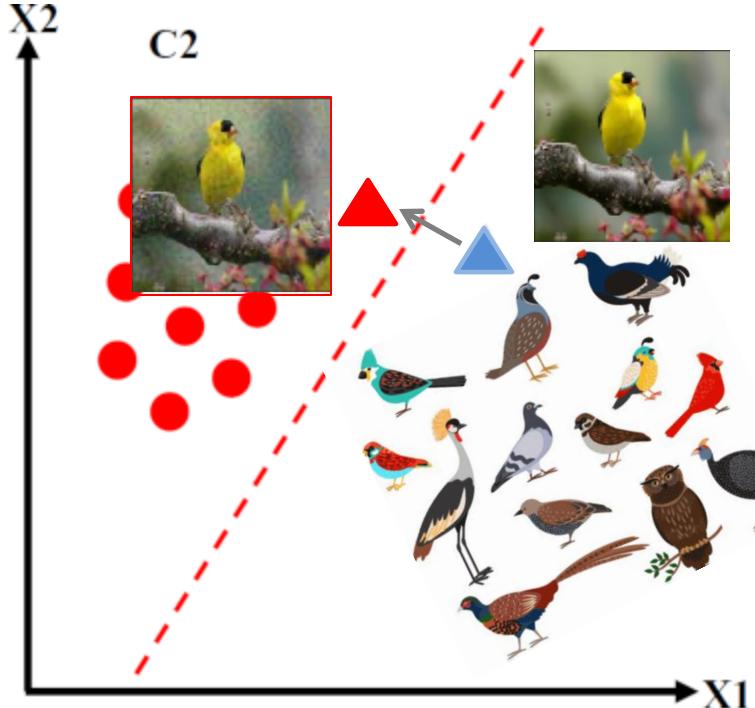
How Can We Generate Adversarial Examples?



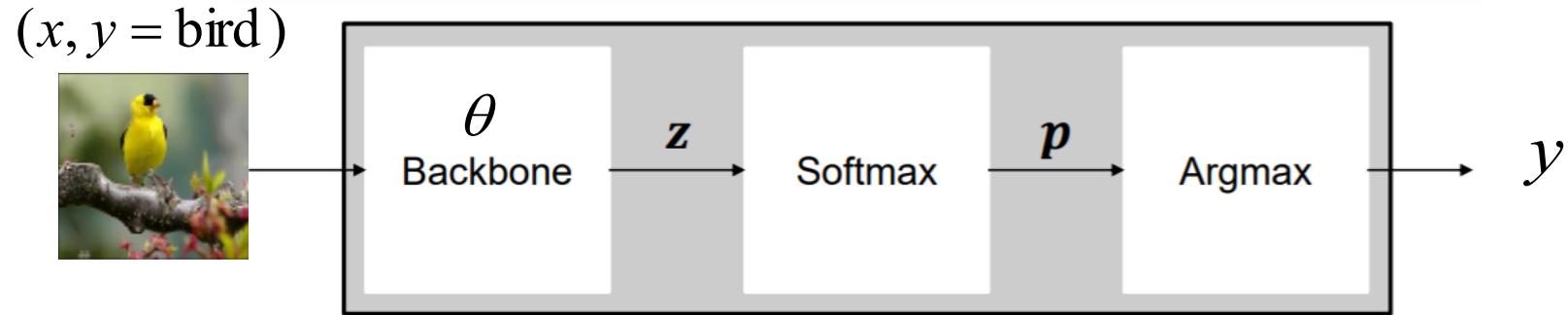
How Can We Generate Adversarial Examples?



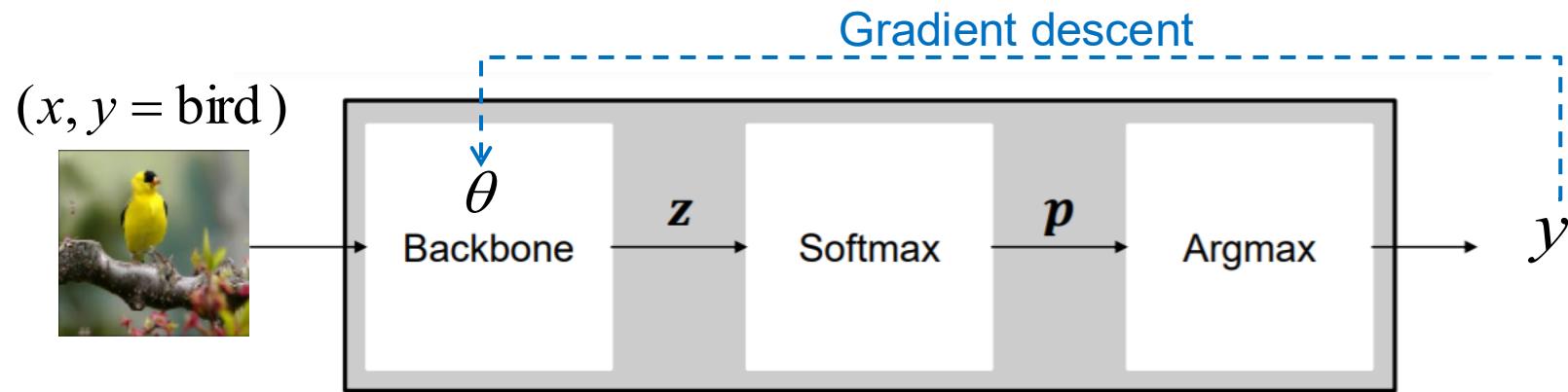
How Can We Generate Adversarial Examples?



Optimization: Formulation



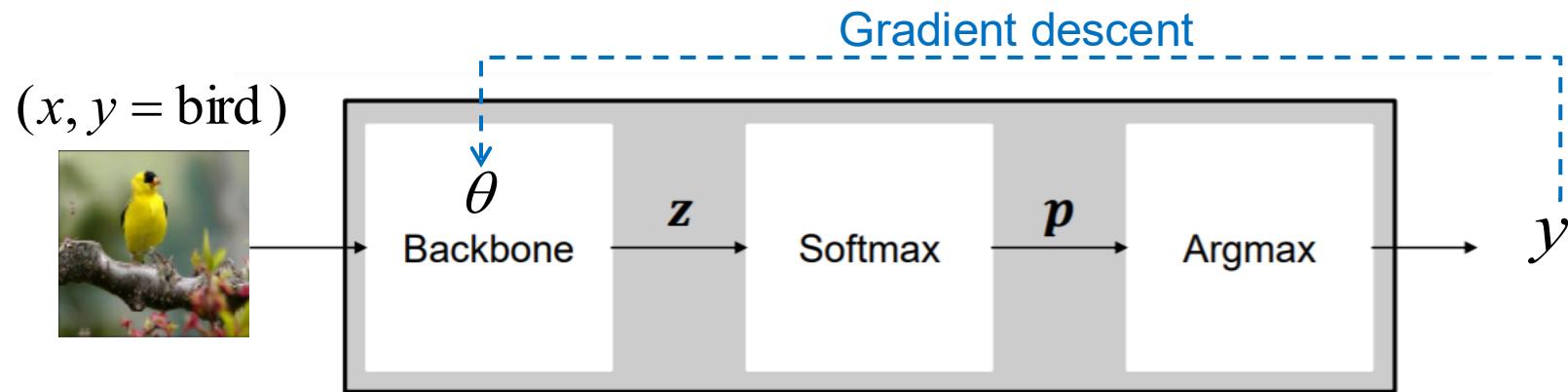
Optimization: Formulation



Model training

$$\theta' = \arg \min_{\theta} d(y, y_{\text{bird}})$$

Optimization: Formulation

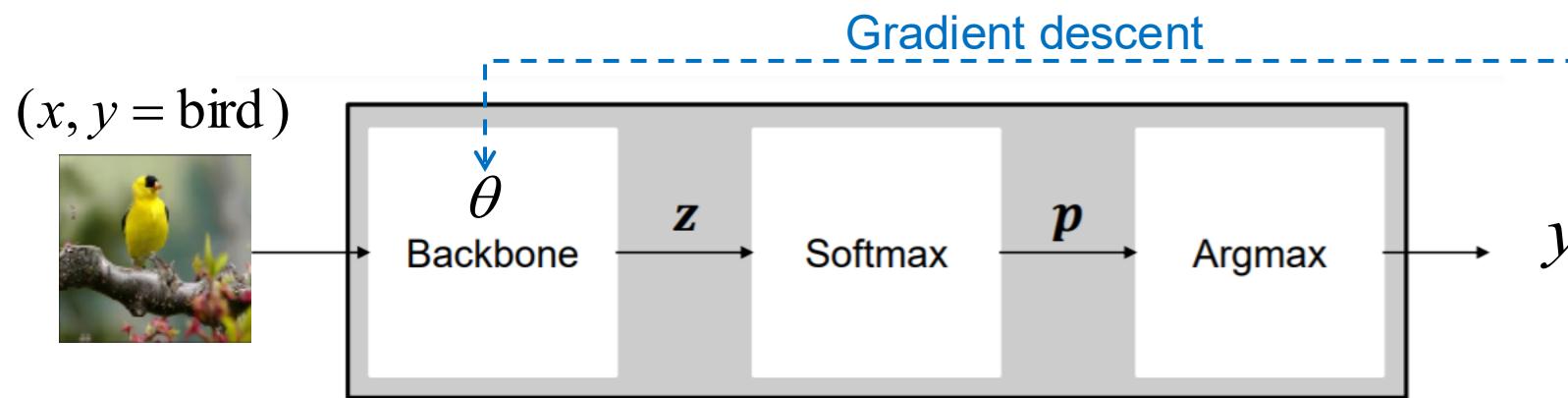


Model training

$$\theta' = \arg \min_{\theta} d(y, y_{\text{bird}})$$

$$d(y, y_{\text{bird}}) = -y_{\text{bird}} \log(p)$$

Optimization: Formulation



Model training

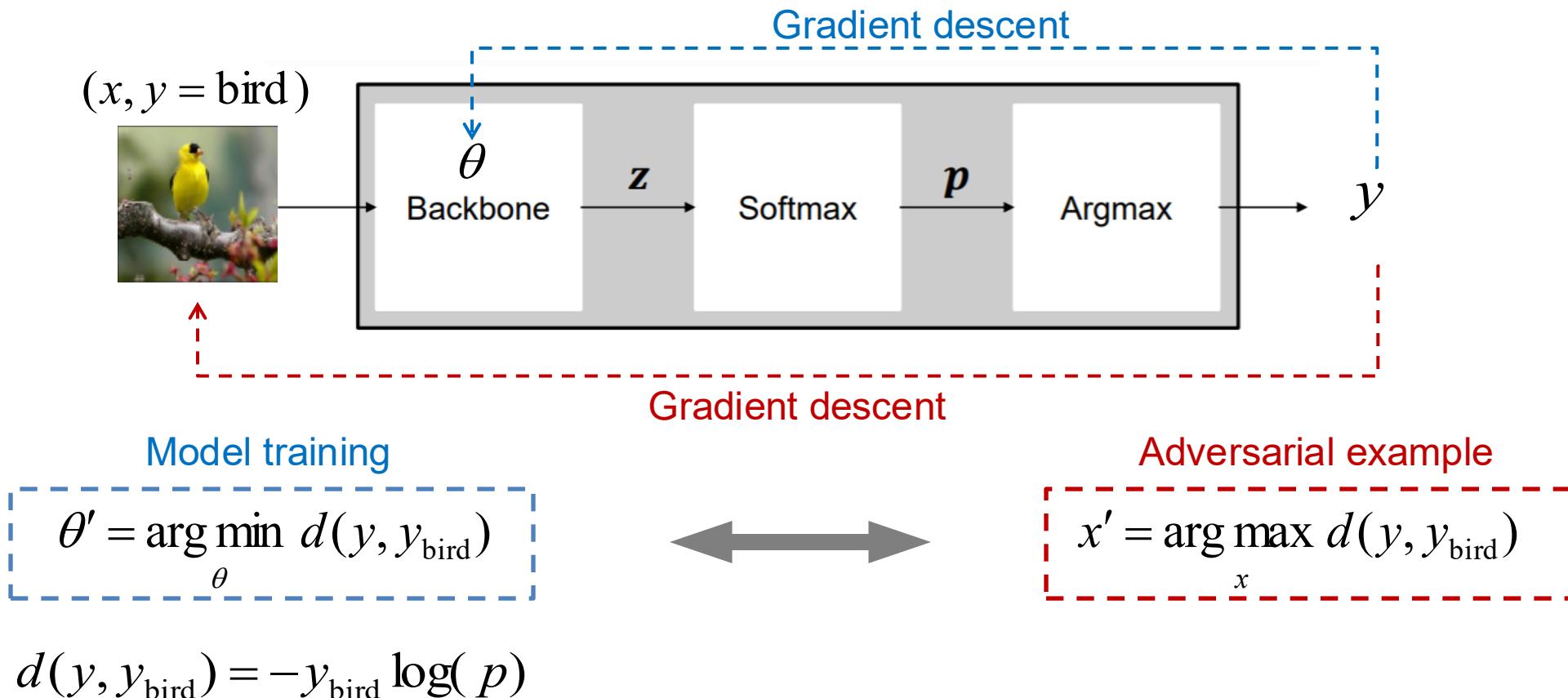
$$\theta' = \arg \min_{\theta} d(y, y_{\text{bird}})$$

$$d(y, y_{\text{bird}}) = -y_{\text{bird}} \log(p)$$

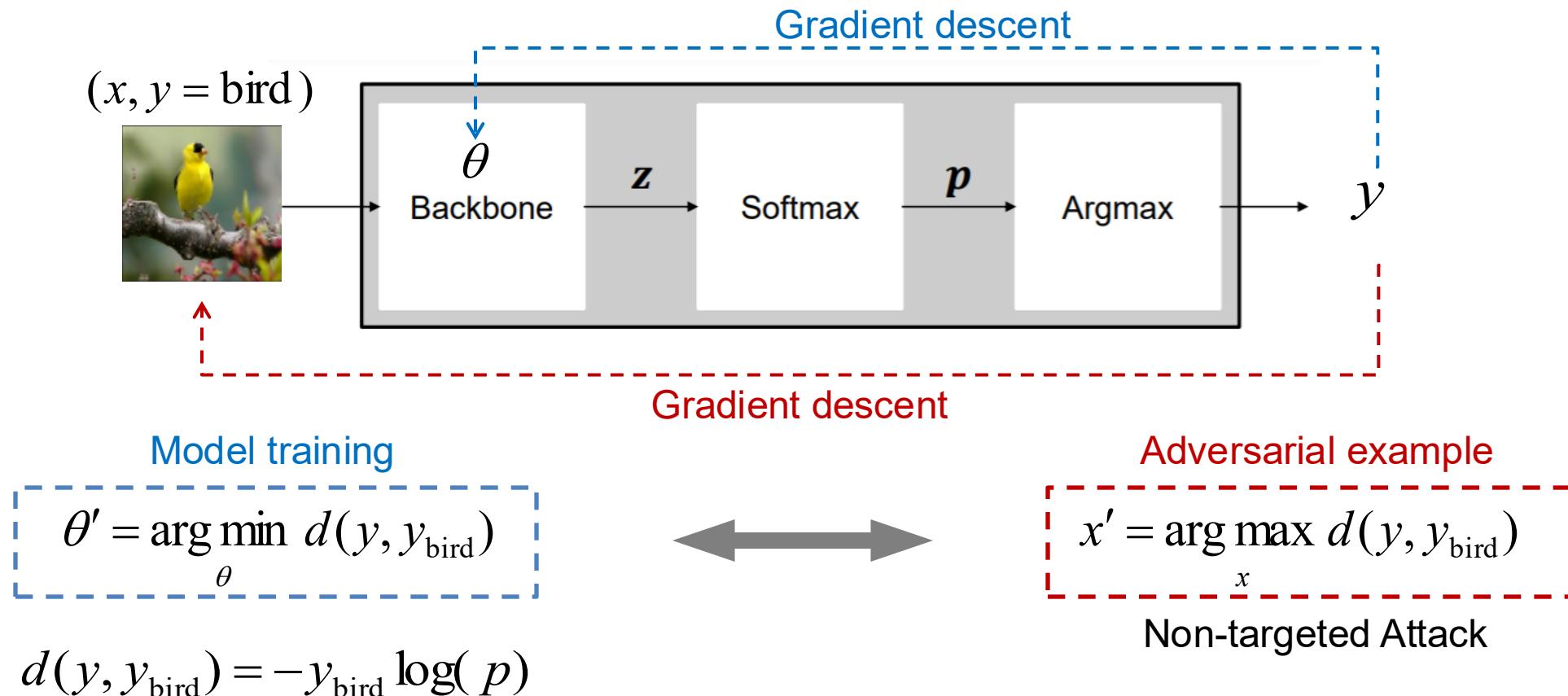
Adversarial example

$$x' = \arg \max_x d(y, y_{\text{bird}})$$

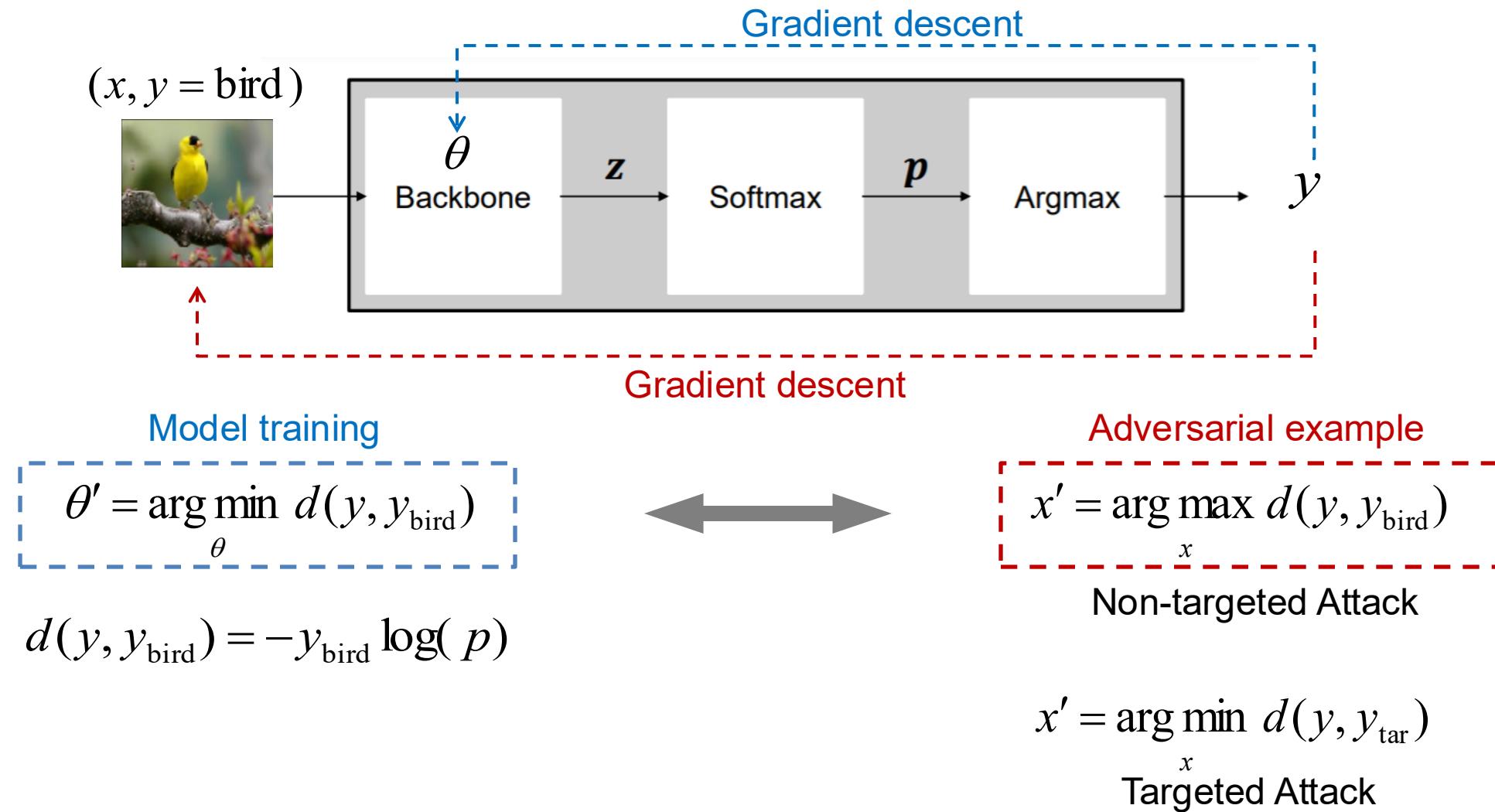
Optimization: Formulation



Optimization: Formulation



Optimization: Formulation



Optimization: Perturbation Constraint ϵ

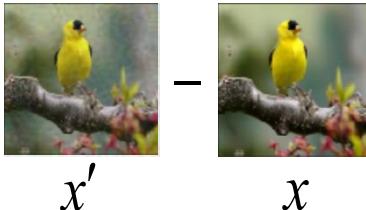
Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$

s.t. $\left\| \begin{matrix} \text{[Image of a bird]} \\ - \\ x' \end{matrix} - \begin{matrix} \text{[Image of a bird]} \\ - \\ x \end{matrix} \right\|_p \leq \epsilon$

Optimization: Perturbation Constraint ϵ

Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$

s.t. $\|x' - x\|_p \leq \epsilon$



L0-norm :

$d = \text{num}(\Delta x_n)$; number



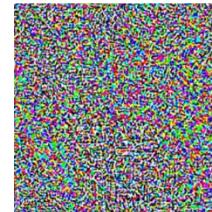
Bird (Airplane)

L1-norm :

$d = |\Delta x_1| + |\Delta x_2| + \dots$; total value

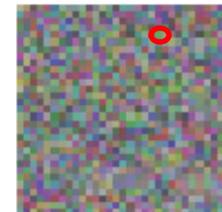
L2-norm :

$d = \Delta x_1^2 + \Delta x_2^2 + \dots$; total value



L^∞ -norm:

$d = \max(\Delta x_1, \Delta x_2, \dots)$; max value



Optimization: Three Methods

Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$

s.t. $\| \begin{matrix} \text{[Image of a bird]} \\ - \\ x' \end{matrix} - \begin{matrix} \text{[Image of a bird]} \\ - \\ x \end{matrix} \|_p \leq \varepsilon$

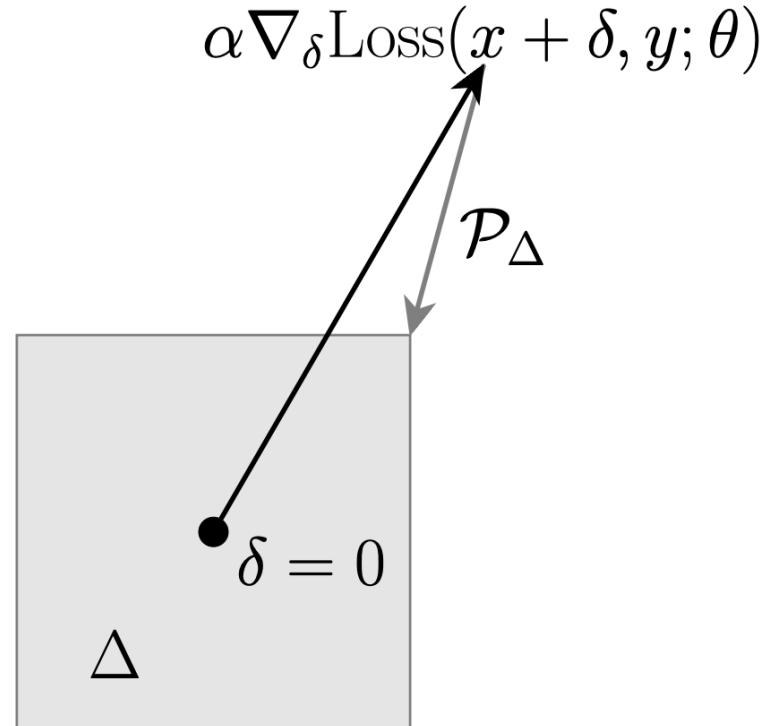
Method 1: Fast Gradient Sign Method (FGSM)

To be more concrete, take Δ to be the ℓ_∞ ball, $\Delta = \{\delta: \|\delta\|_\infty \leq \epsilon\}$, so projection takes the form

$$P_\Delta(\delta) = \text{Clip}(\delta, [-\epsilon, \epsilon])$$

As $\alpha \rightarrow \infty$, we always reach “corner” of the box, called fast gradient sign method (FGSM)
[Goodfellow et al., 2014]

$$\delta = \epsilon \cdot \text{sign}(\nabla_\delta \text{Loss}(x + \delta, y; \theta))$$



Method 2: Projected Gradient Descent (PGD)

Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$

s.t. $\| \begin{matrix} \text{image of bird} \\ - \\ x' \end{matrix} - \begin{matrix} \text{image of bird} \\ - \\ x \end{matrix} \|_p \leq \varepsilon$

Method 2: Projected Gradient Descent (PGD)

Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$

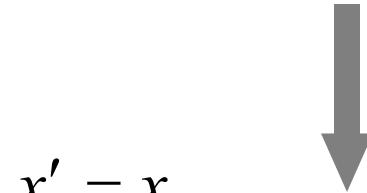
s.t. $\left\| \frac{\text{Image of bird}}{x'} - \frac{\text{Image of bird}}{x} \right\|_p \leq \varepsilon$

$$x'_0 = x,$$

$$x'_{i+1} = x'_i - \alpha(\nabla_x d(y, y_{\text{bird}}))$$

Method 2: Projected Gradient Descent (PGD)

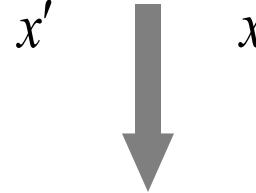
Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$



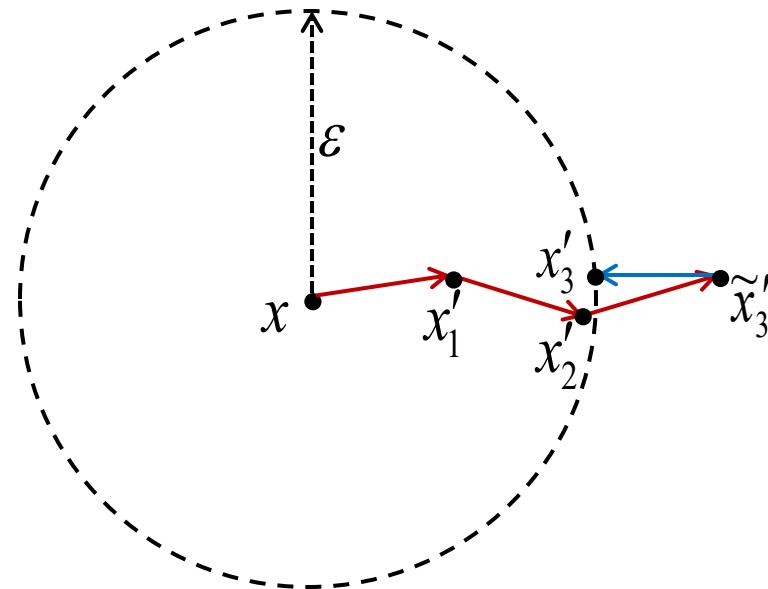
$$x'_0 = x,$$

$$x'_{i+1} = x'_i - \alpha(\nabla_x d(y, y_{\text{bird}}))$$

s.t. $\| \begin{array}{c} \text{[Image of a bird]} \\ - \end{array} \|_p \leq \varepsilon$



$$x' \leftarrow \text{project}(x' - x, -\varepsilon, \varepsilon)$$



Method 2: Projected Gradient Descent (PGD)

```
# PGD Attack
# MNIST init
def pgd_attack(model, images, labels, eps=0.3, alpha=2/255, iters=40) :
    images = images.to(device)
    labels = labels.to(device)
    loss = nn.CrossEntropyLoss()

    ori_images = images.data

    for i in range(iters) :
        images.requires_grad = True
        outputs = model(images)

        model.zero_grad()
        cost = loss(outputs, labels).to(device)
        cost.backward()

        adv_images = images + alpha*images.grad.sign()
        eta = torch.clamp(adv_images - ori_images, min=-eps, max=eps)
        images = torch.clamp(ori_images + eta, min=0, max=1).detach_()

    return images
```

<https://github.com/Harry24k/PGD-pytorch/blob/master/PGD.ipynb>

Method 3: Joint Optimization

Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$ s.t. $\|x' - x\|_p \leq \varepsilon$



$$x' = \arg \max_x d(y, y_{\text{bird}}) - \lambda \cdot \|x' - x\|_p$$

Method 3: Joint Optimization

Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$

s.t. $\| \begin{matrix} \text{[Image of a bird]} \\ - \\ \text{[Image of a bird]} \end{matrix} \|_p \leq \varepsilon$



x' x



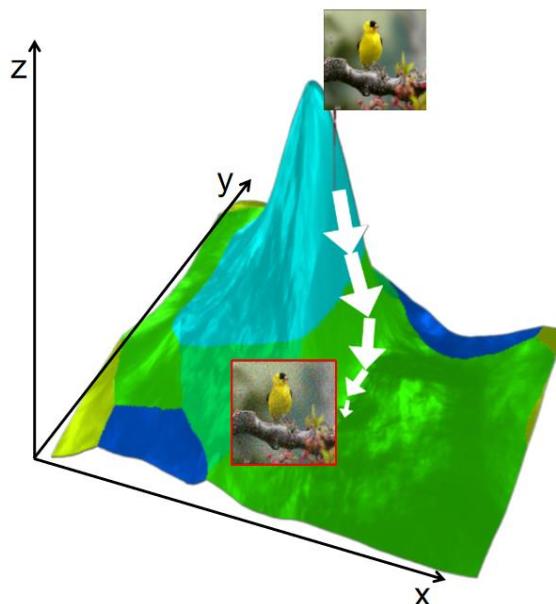
$$x' = \arg \max_x d(y, y_{\text{bird}}) - \lambda \cdot \|x' - x\|_p$$

$$\|x' - x\|_{p=2} = \sqrt{\sum_{k=1}^n (x'_k - x_k)^2}$$

Method 3: Joint Optimization

Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$

s.t. $\|x' - x\|_p \leq \varepsilon$



$$x' = \arg \max_x d(y, y_{\text{bird}}) - \lambda \cdot \|x' - x\|_p$$

$$L_{\text{joint}}$$

$$x'_0 = x,$$

$$x'_{i+1} = x'_i - \alpha \cdot (\nabla_x L_{\text{joint}})$$

$$\|x' - x\|_{p=2} = \sqrt{\sum_{k=1}^n (x'_k - x_k)^2}$$

(Empirical) Defenses

- **Black-box:**
Attack doesn't know defense
- **White-box:**
Attack knows and can be adapted to defense

(Empirical) Defenses

- **Black-box:**
Attack doesn't know defense
- **White-box:**
Attack knows and can be adapted to defense



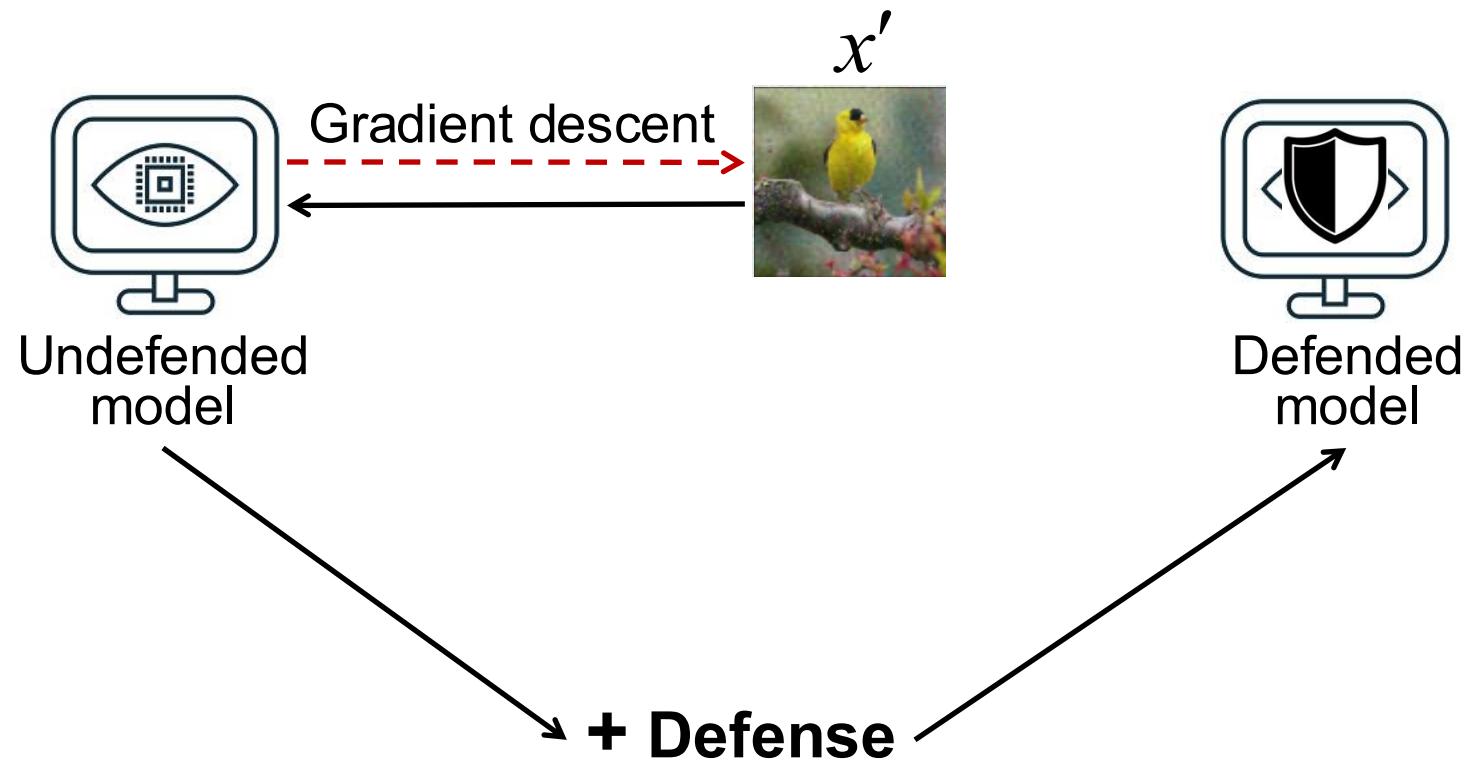
Undefended
model



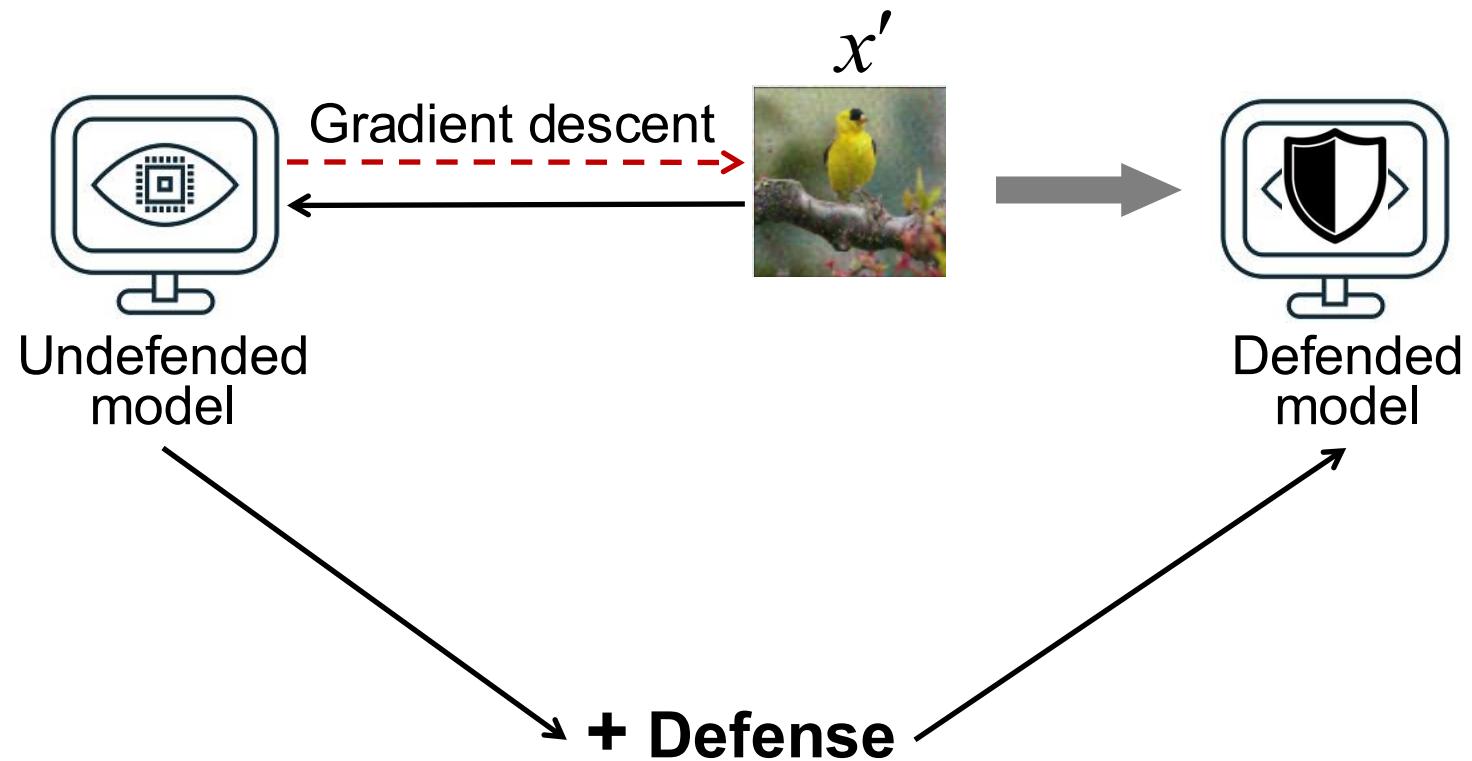
Defended
model

+ Defense

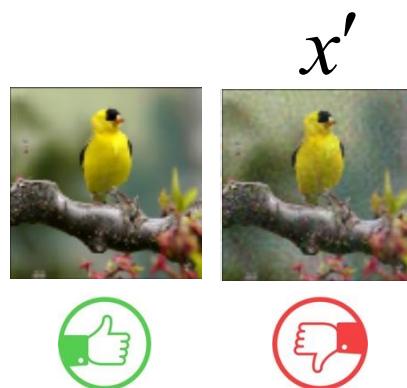
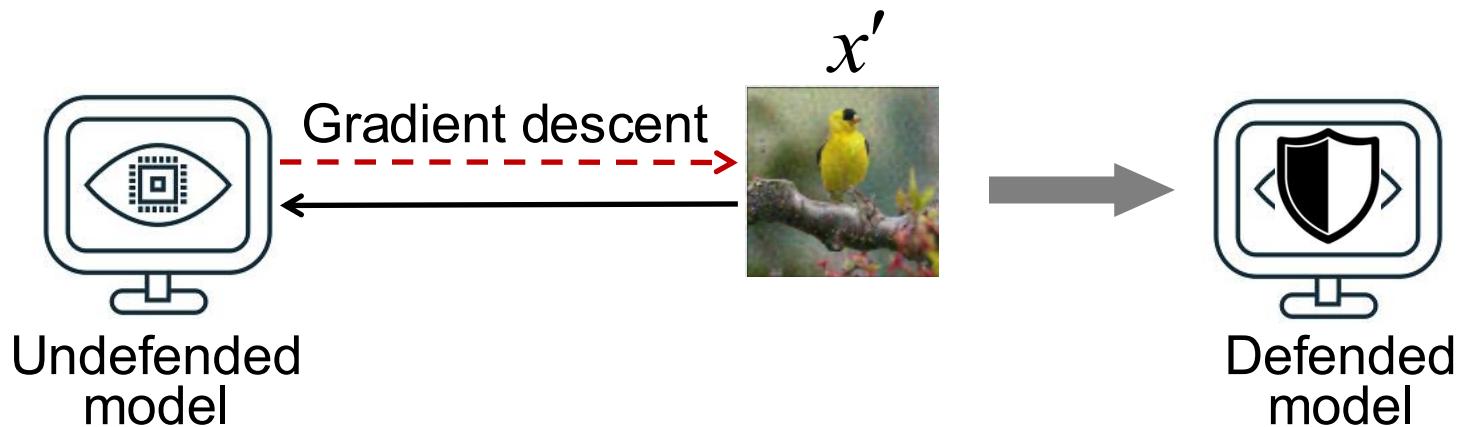
Black-box Defense



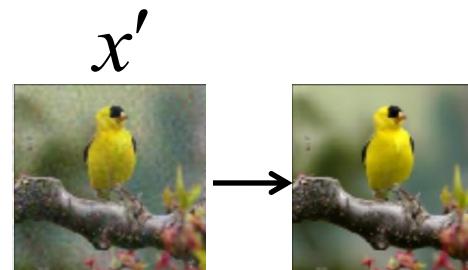
Black-box Defense



Black-box Defense



1. (test) Detection

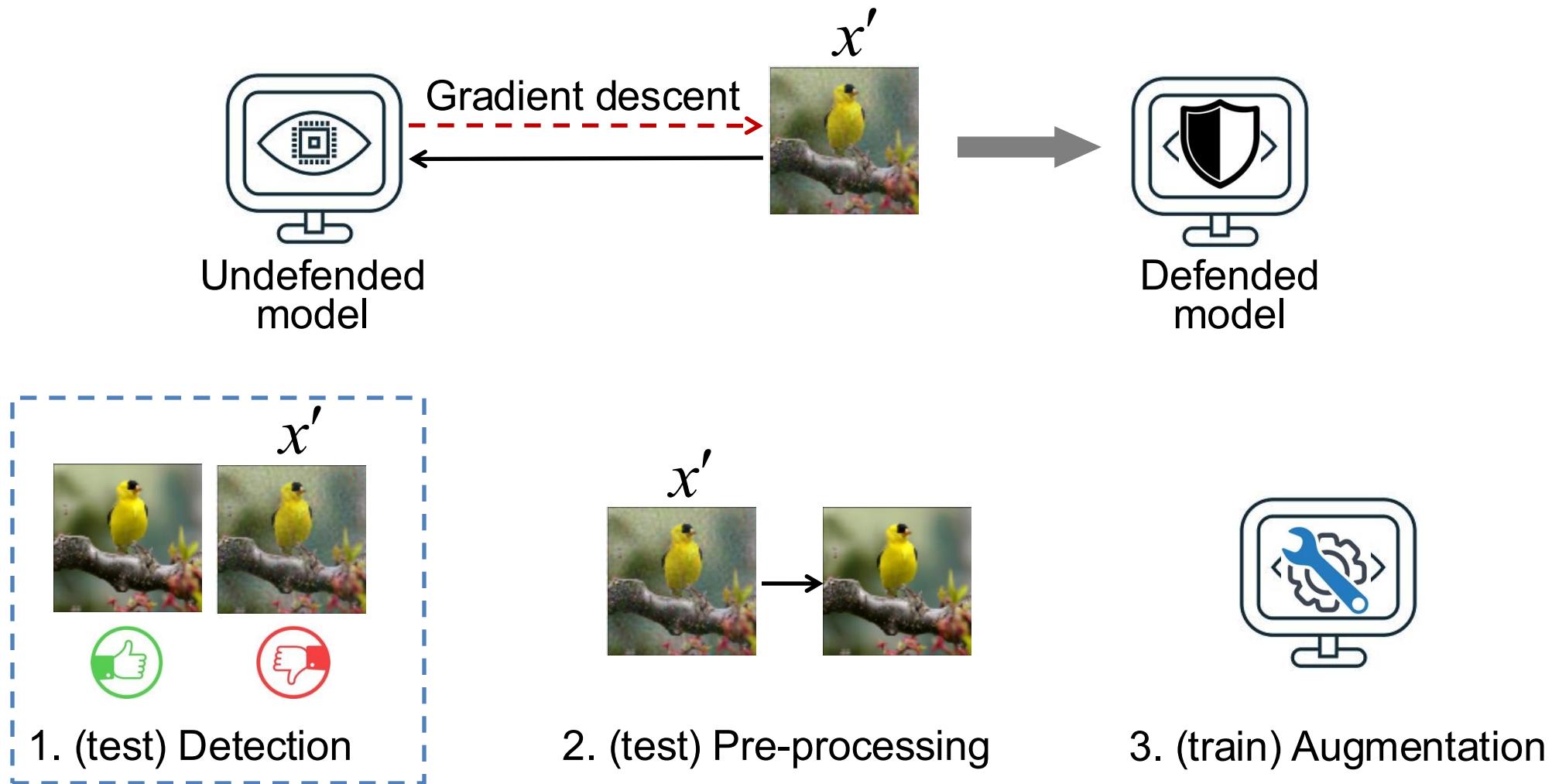


2. (test) Pre-processing

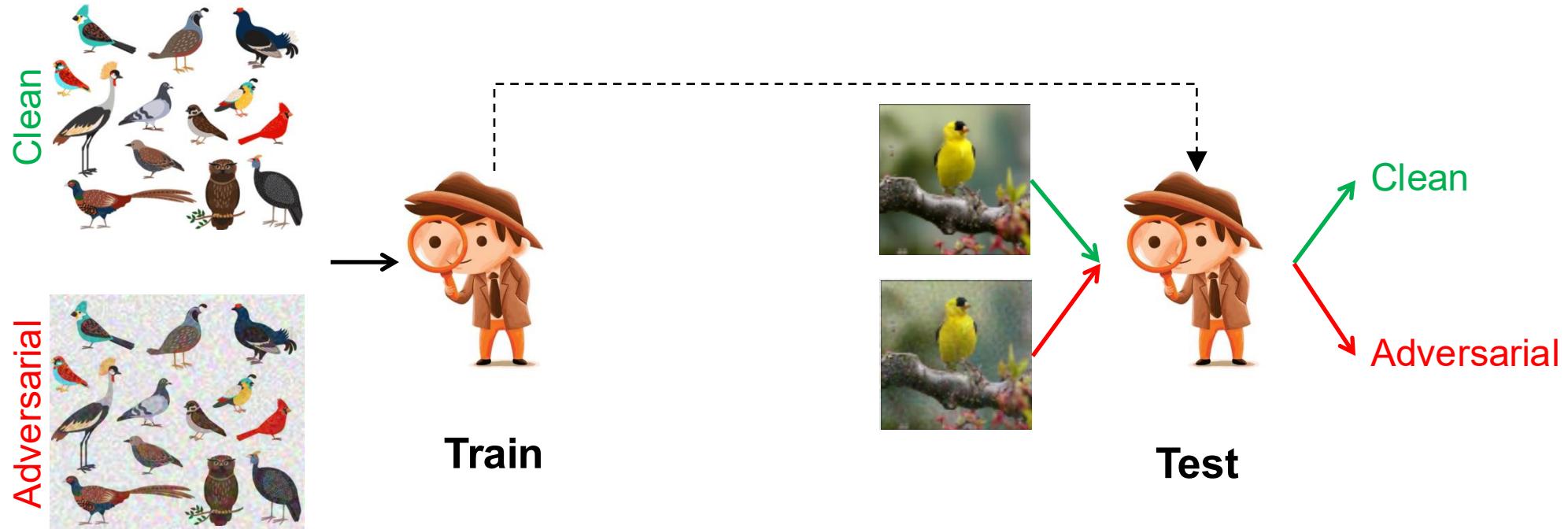


3. (train) Augmentation

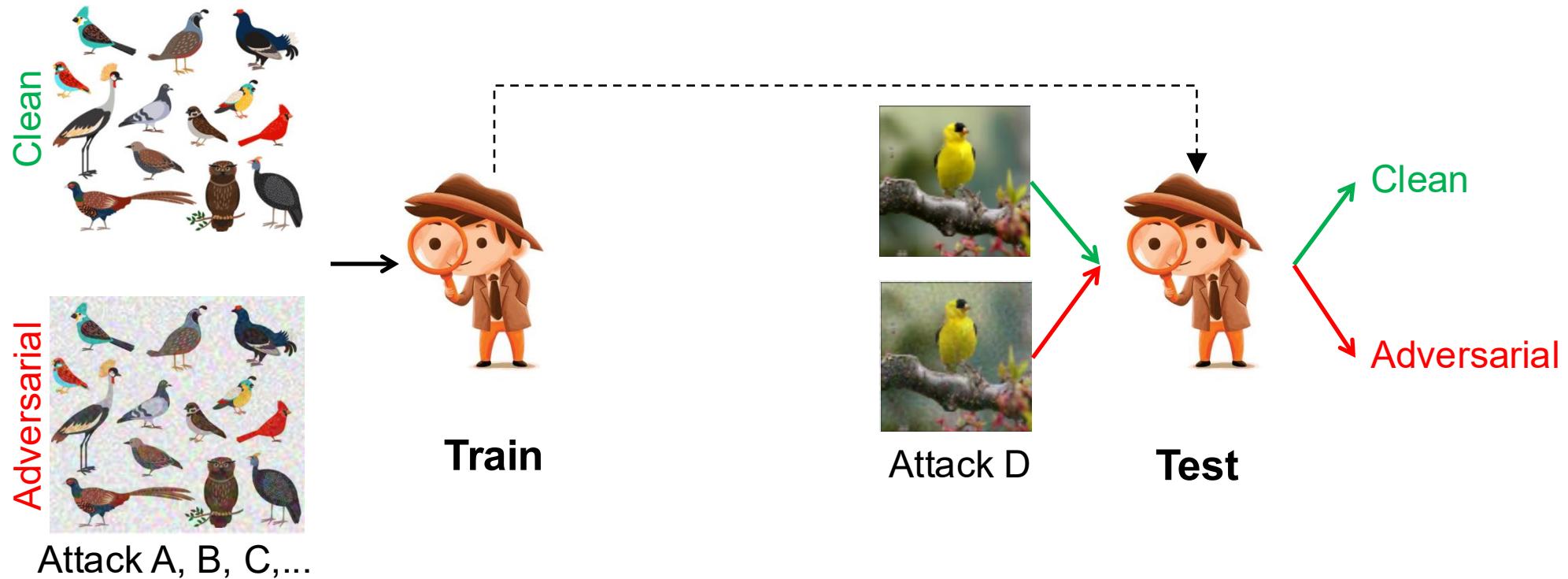
Black-box Defense



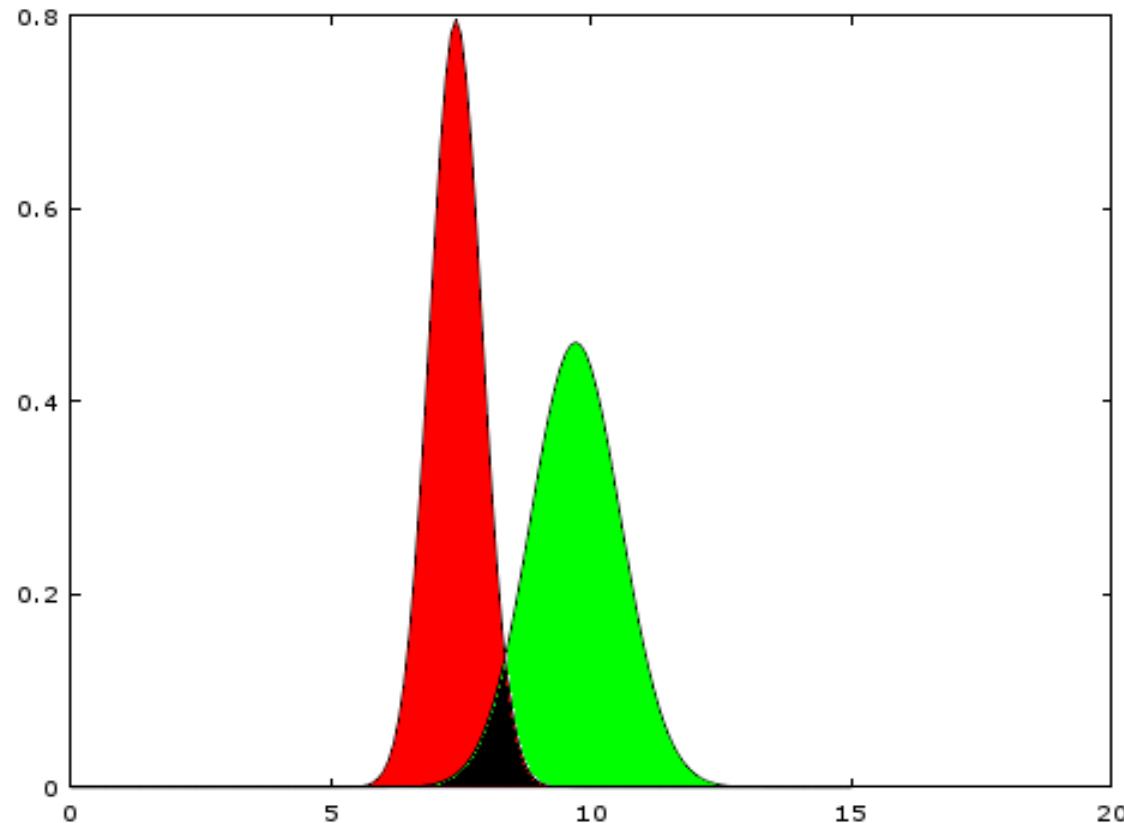
Black-box Defense: Detection



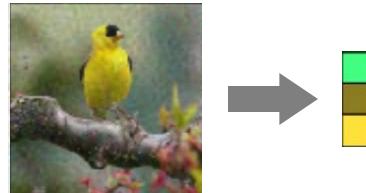
Black-box Defense: Detection



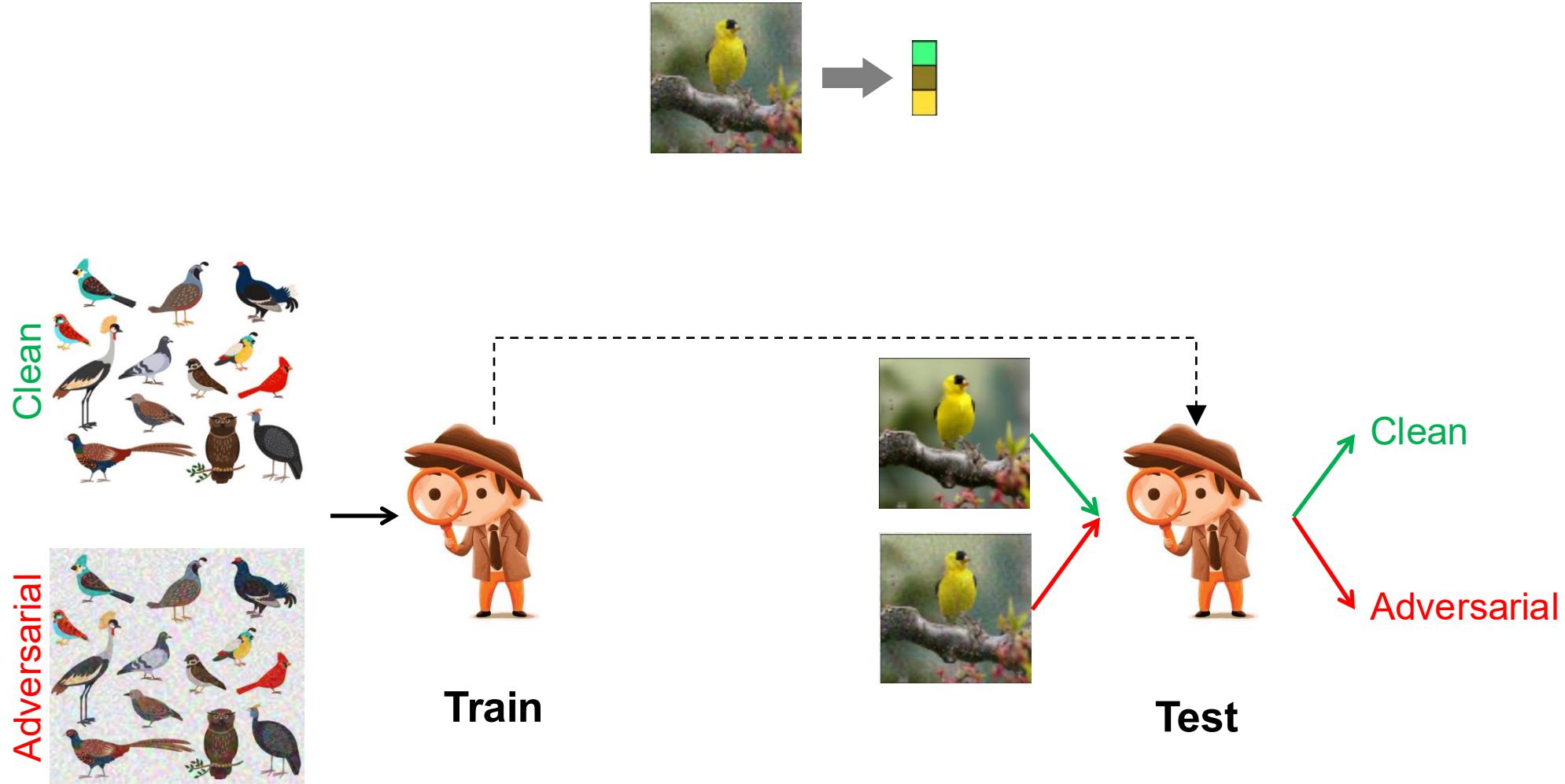
Black-box Defense: Detection: Statistical Features



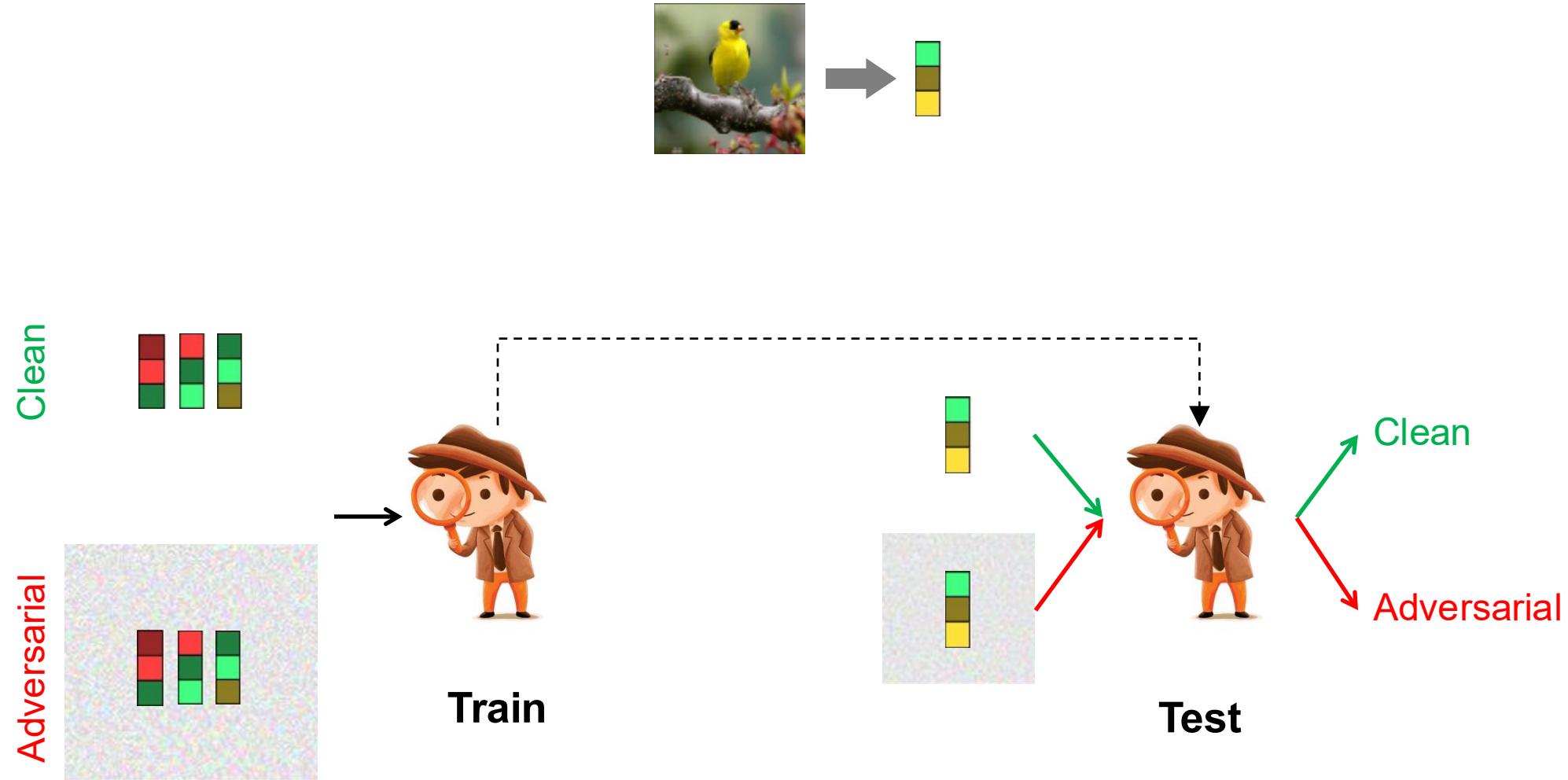
Black-box Defense: Detection: Statistical Features



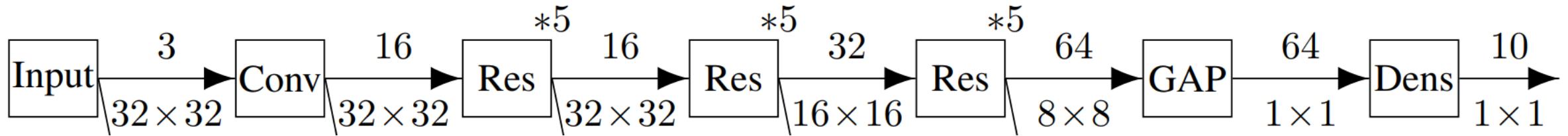
Black-box Defense: Detection: Statistical Features



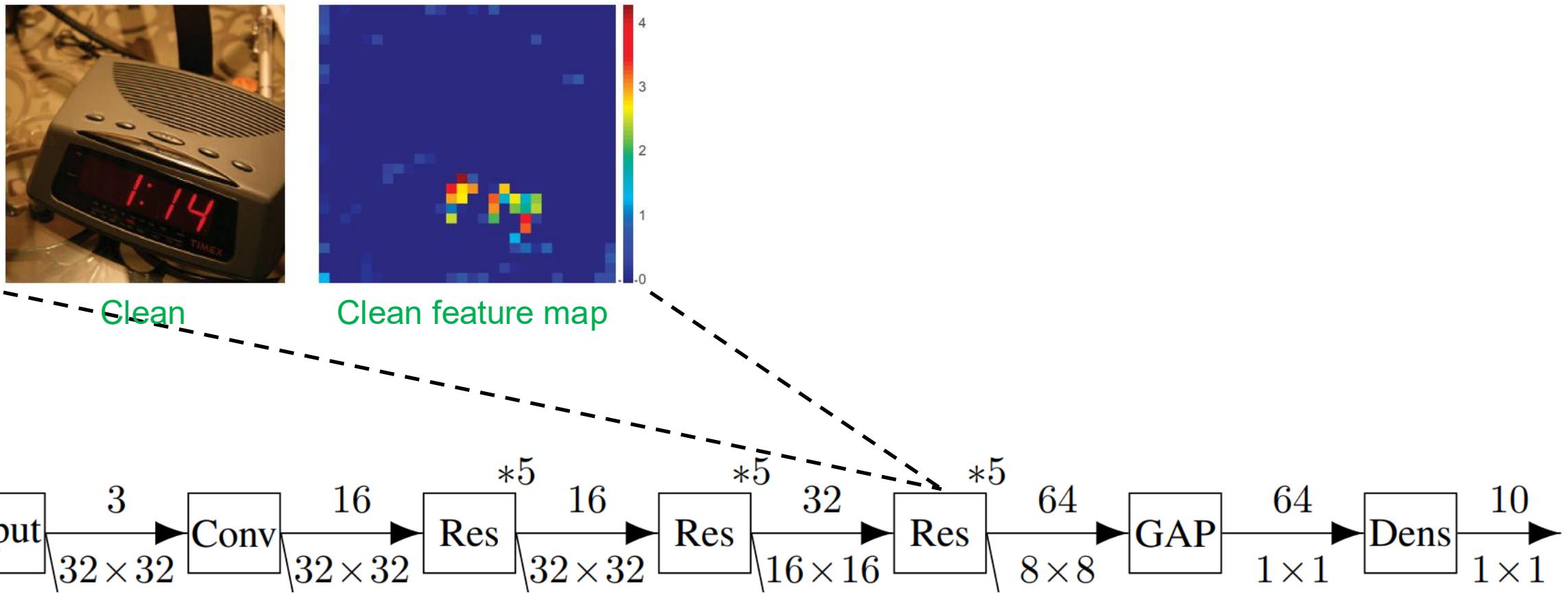
Black-box Defense: Detection: Statistical Features



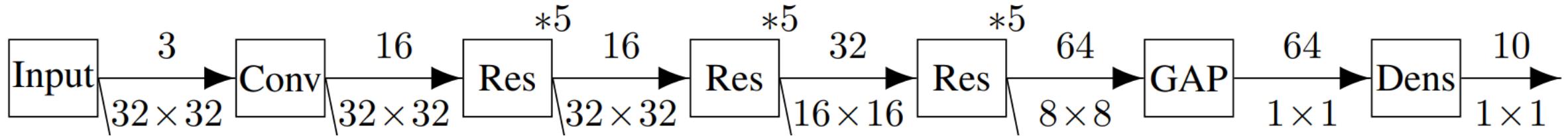
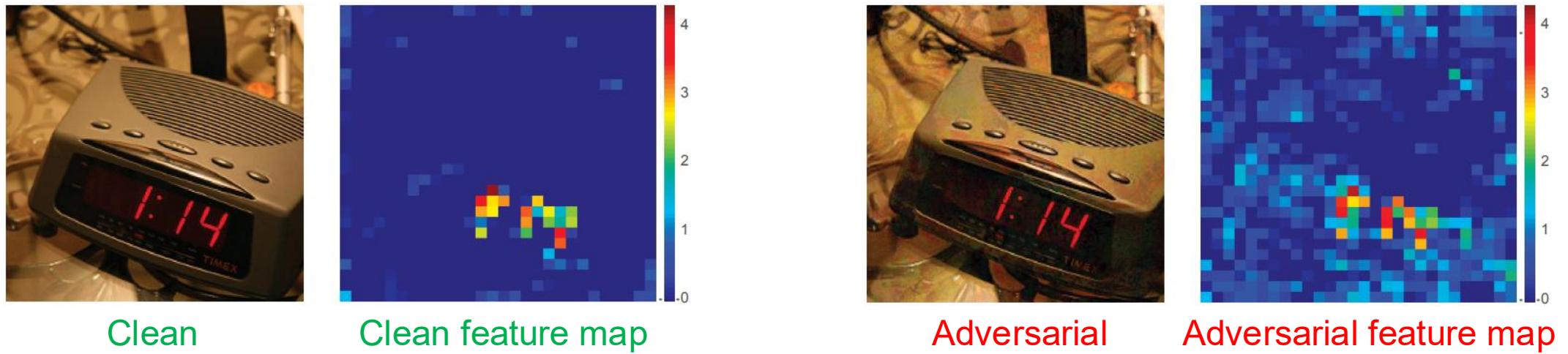
Black-box Defense: Detection: Statistical Features



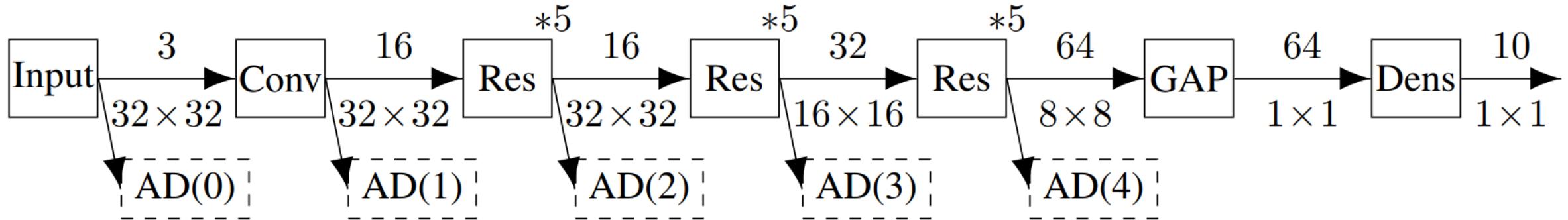
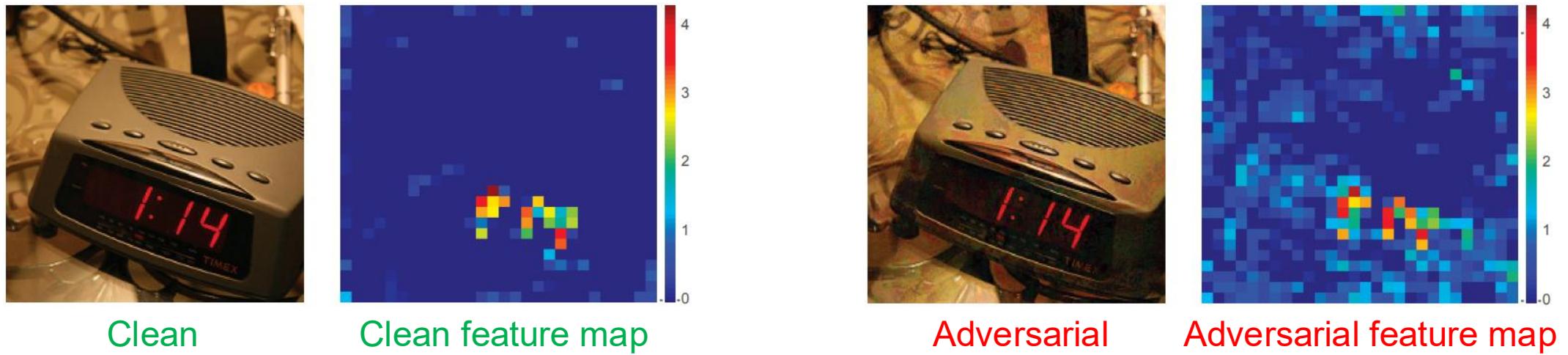
Black-box Defense: Detection: Statistical Features



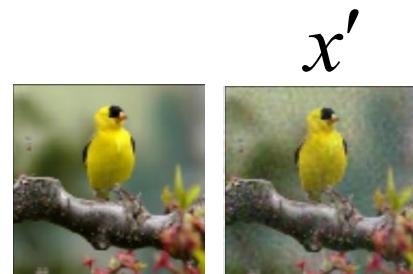
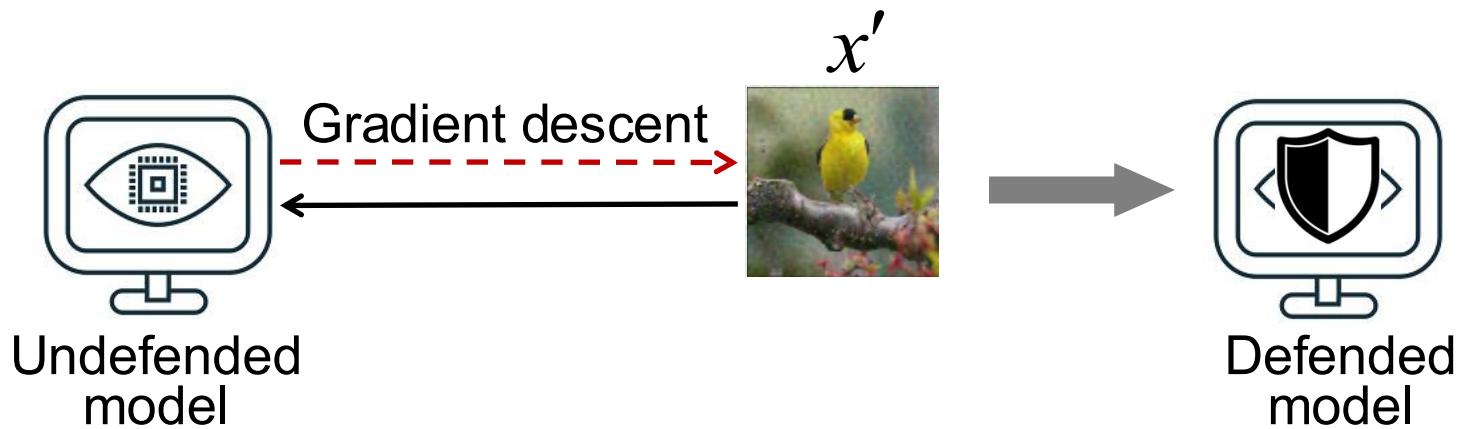
Black-box Defense: Detection: Statistical Features



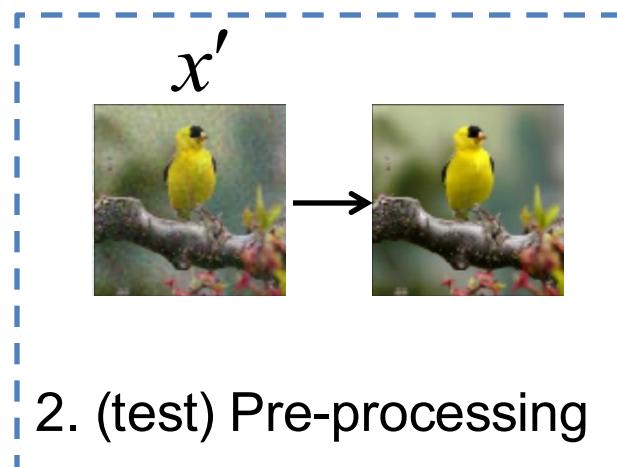
Black-box Defense: Detection: Statistical Features



Black-box Defense

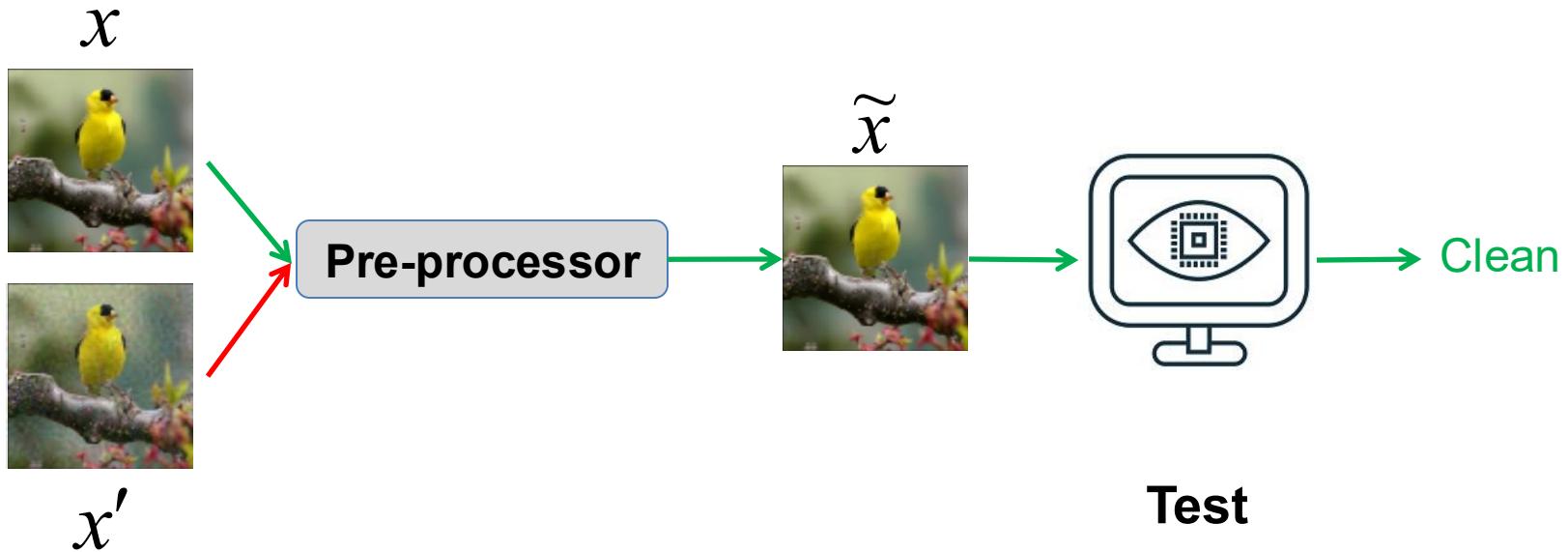


1. (test) Detection

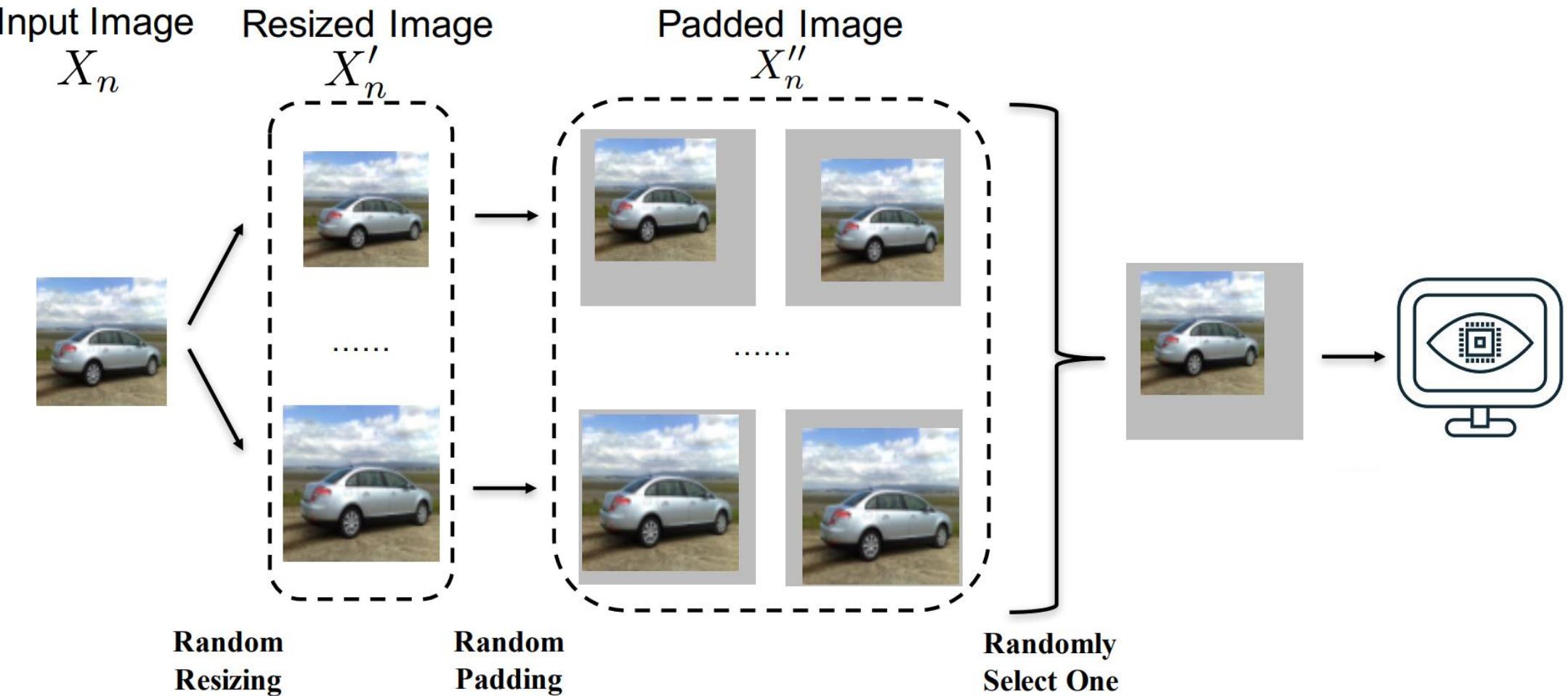


3. (train) Augmentation

Black-box Defense: Pre-processing



Black-box Defense: Pre-processing: Randomization



Black-box Defense: Pre-processing: Denoising

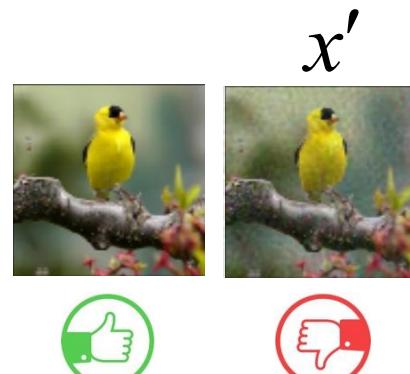
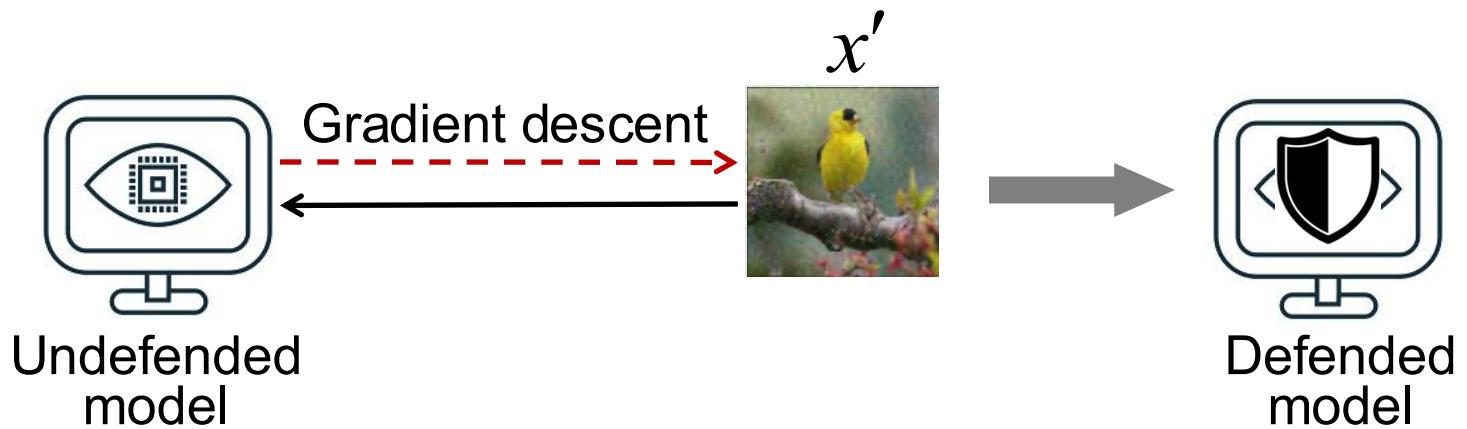


Black-box Defense: Pre-processing: Denoising

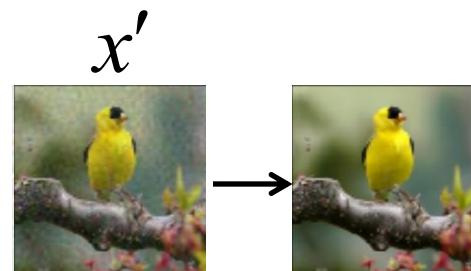


Defense	No Attack $ L_2 = 0$	CW-L2 ($\kappa = 0$) $ L_2 = .0025$
	75.59	10.29
No Defense	75.59	10.29
JPEG [quality=100]	74.95	74.37
JPEG [quality=90]	74.83	74.43
JPEG [quality=80]	74.23	73.92
JPEG [quality=70]	73.61	73.11
JPEG [quality=60]	72.97	72.46
JPEG [quality=50]	72.32	71.86
JPEG [quality=40]	71.48	71.03
JPEG [quality=30]	70.08	69.63
JPEG [quality=20]	67.72	67.32

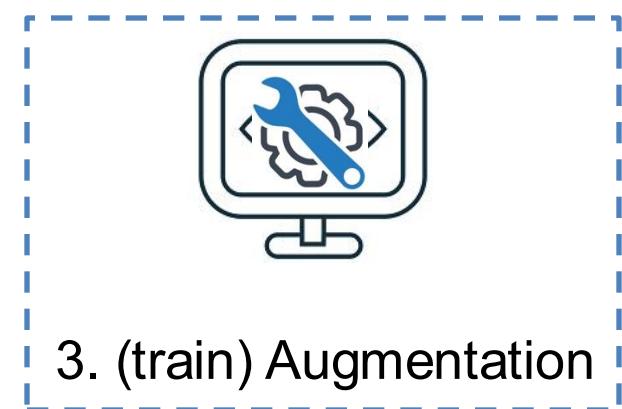
Black-box Defense



1. (test) Detection

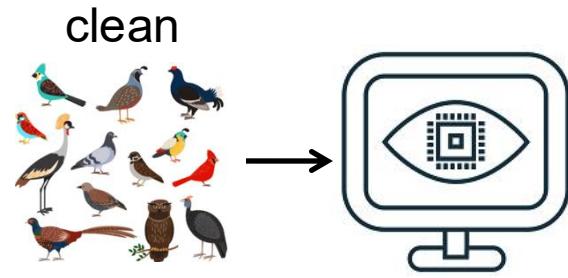


2. (test) Pre-processing

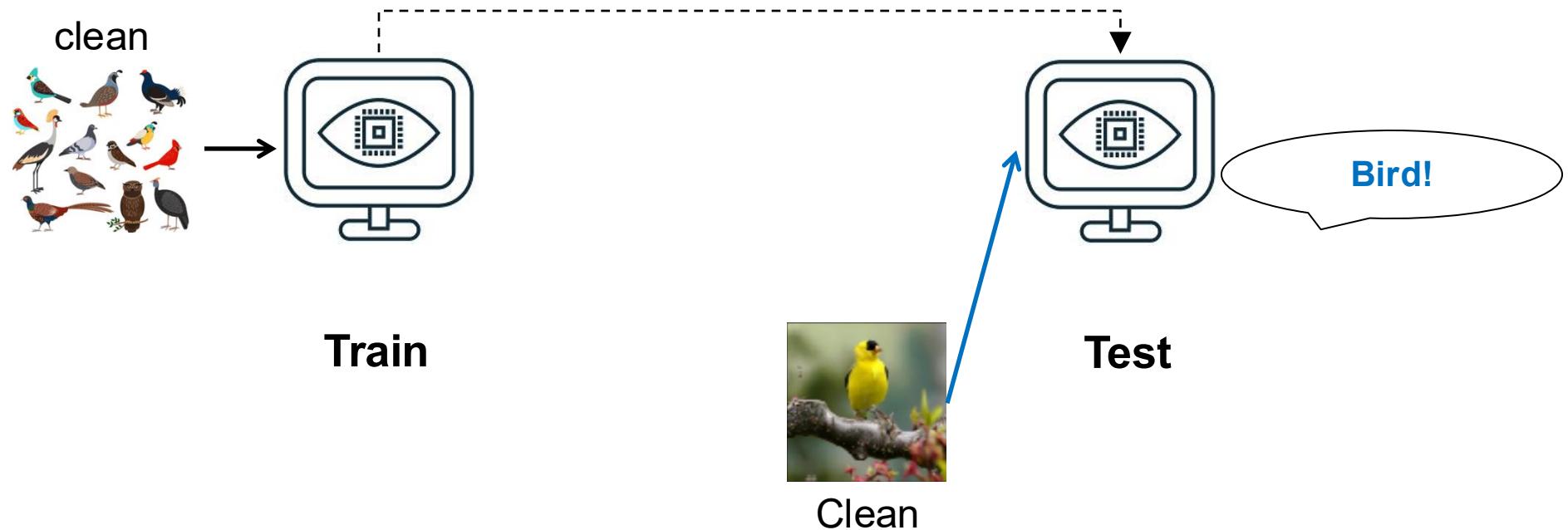


3. (train) Augmentation

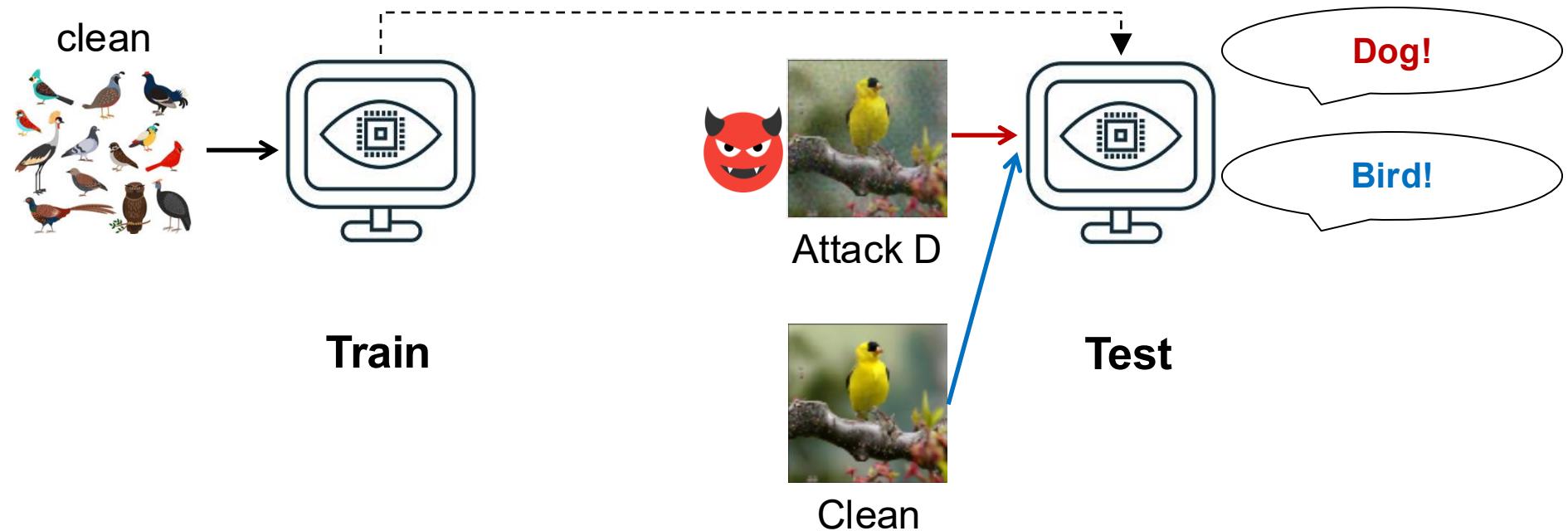
Black-box Defense: Augmentation



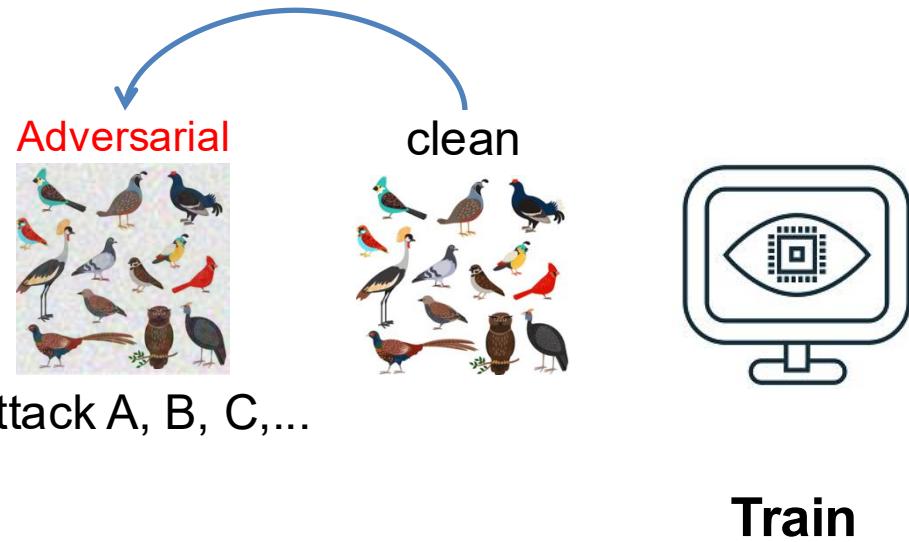
Black-box Defense: Augmentation



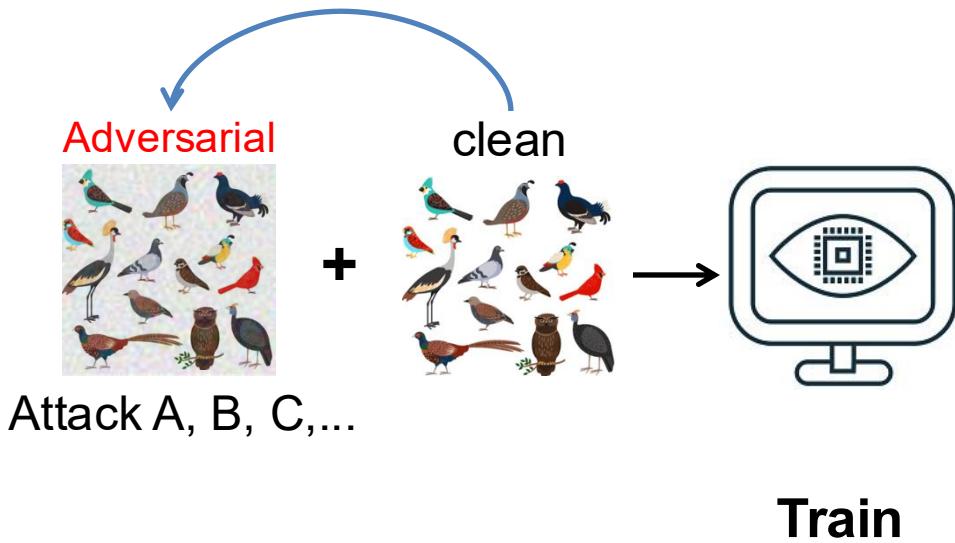
Black-box Defense: Augmentation



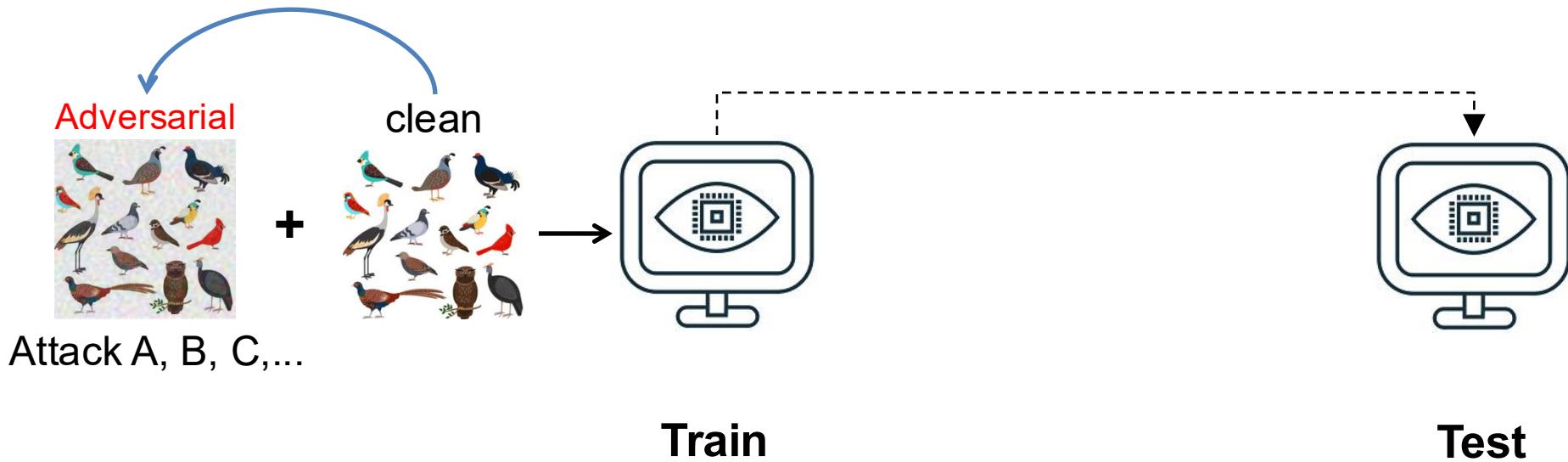
Black-box Defense: Augmentation



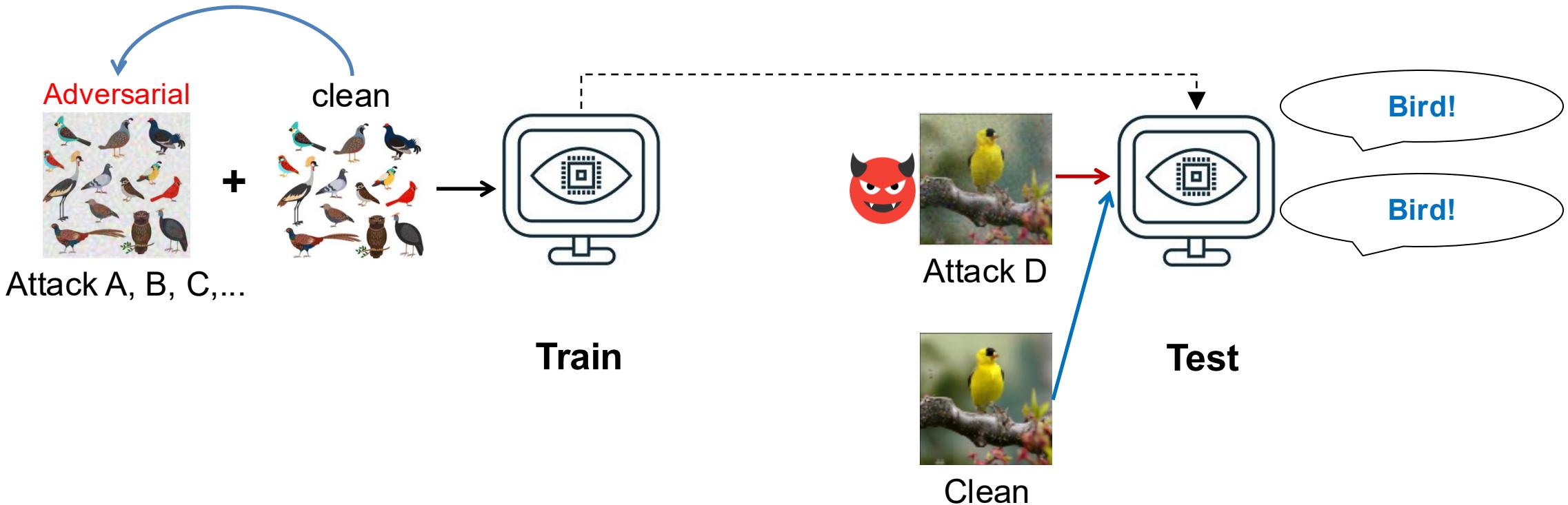
Black-box Defense: Augmentation



Black-box Defense: Augmentation



Black-box Defense: Augmentation



(Empirical) Defenses

- **Black-box:**
Attack doesn't know defense and keeps fixed
- **White-box:**
Attack knows and can be adapted to defense



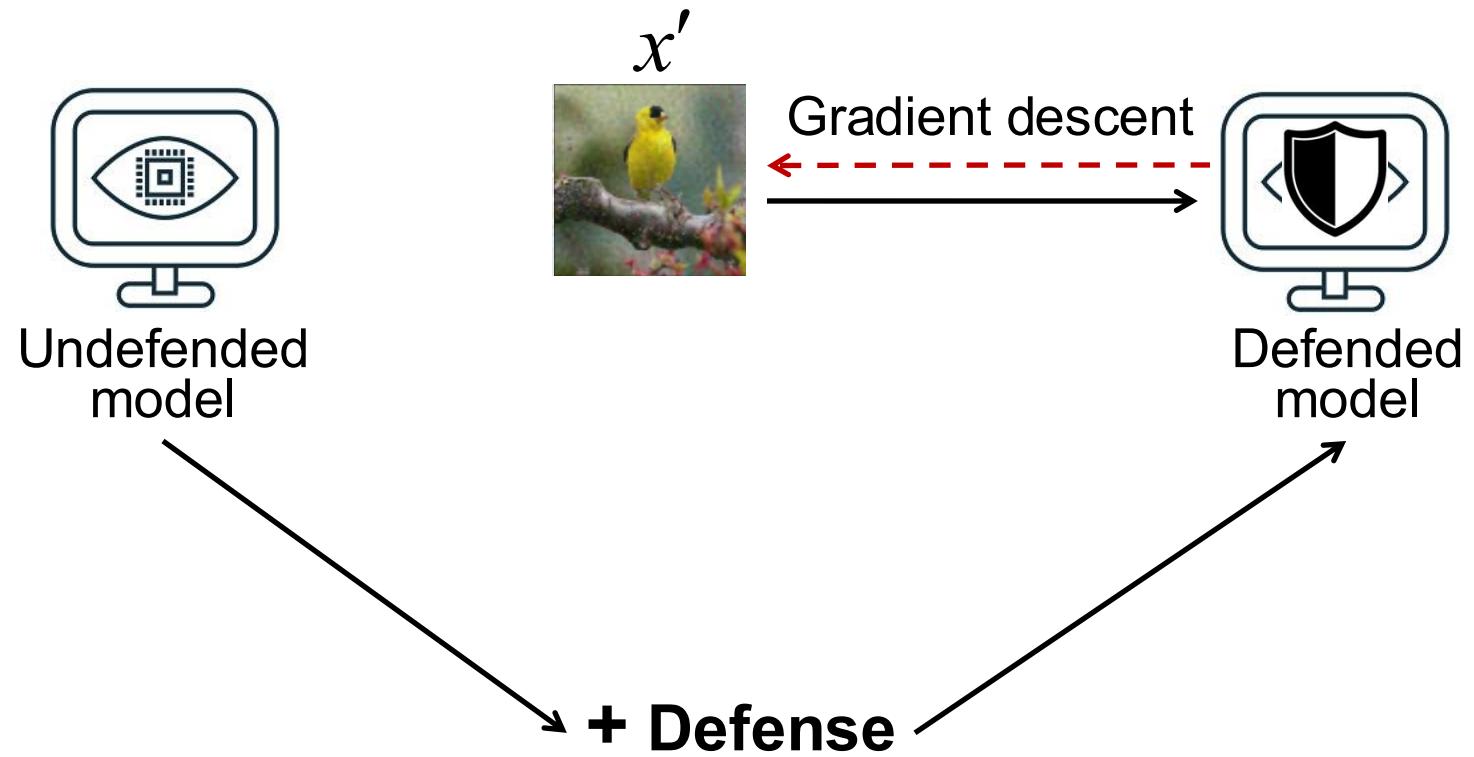
Undefended
model



Defended
model

+ Defense

White-box Defense



Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods

MagNet and “Efficient Defenses Against Adversarial Examples Are Not Robust to Adversarial Examples

Abstract

MagNet and proposed as a defense we can const

1 INTRO

Recent years have shown that neural networks driving force has been demonstrated [38], to beat cars [6].

In this paper, we propose a defense that we can const

1 Introd

It is an open problem whether we can const

nearly all domain

The research proposing ma

correctly, v

the data

classific

the whi

MagNet

adversa

the para

ment

permis

classroo

for profi

commu

on the firs

author(s) m

be republis

, post

and/or a fee.

Requ

AlSeC'17, Novem

© 2017 Copyright

for Computin

ACM ISBN 978-1-

<https://doi.org/10>

ABSTRACT

Neural networks: inputs that are more robust to adversarial examples. In order to better compare their efficiency, we propose ten new loss functions that significantly highlight properties of the network that are not fully understood. Finally, we propose a future proposal.

1 INTRO

Recent years have shown that neural networks driving force has been demonstrated [38], to beat cars [6].

In this paper, we propose a defense that we can const

nearly all domain

The research proposing ma

correctly, v

the data

classific

the whi

MagNet

adversa

the para

ment

permis

classroo

for profi

commu

on the firs

author(s) m

be republis

, post

and/or a fee.

Requ

AlSeC'17, Novem

© 2017 Copyright

for Computin

ACM ISBN 978-1-

<https://doi.org/10>

Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples

On the Robustness of the CVPR 2018 Winner

Is AmI (A) Robust

Abstract—No.

I. ATTACKING “ATTACKS MEET INTENT”

AmI (Attacks meet Interpretability) is a defense [3] to detect [1] adversarial examples from recognition models. By applying interpretation to a pre-trained neural network, AmI identifies the most important neurons. It then creates a second augment with the same parameters but increases the number of important neurons. AmI rejects inputs with augmented neural network disagree.

We find that this defense (presented at a spotlight paper—the top 3% of submissions) is ineffective, and even *defense-oblivious*¹ detection rate to 0% on untargeted attacks more robust to untargeted attacks than the network. Figure 1 contains examples of a defense that fool the AmI defense. We are grateful to the authors for releasing their source code² and hope that future work will continue to by publication time to accelerate progress

1. Introduction

Training neural networks to recognize adversarial examples is a well-known problem: “The

Deflection” (Prakash et al., 2017) and “Adversarial Attacks Denoiser” (Liao et al., 2017)

In response to this challenge, there have been defenses to increase the progress of the defense against adversarial attacks. In this note, we propose a defense that is robust to the adversary solution has not been considered in the literature. As benchmarks, we consider tasks (e.g., Keras, Carlini & Wagner, 2017).

A. Evaluation

Comment on *Biologically inspired protection of deep networks from adversarial attacks*

ON THE LIMITATION OF LOCAL INTRINSIC DIMENSIONALITY FOR CHARACTERIZING THE STABILITIES OF

A

Adversarial Risk and the Dangers of Evaluating Against Weak Attacks

The Efficacy of SHIELD under Different Threat Models

Paper Type: Appraisal Paper of Existing Method

Cory Cornelius
cory.cornelius@intel.com

Nilaksh Das
nilakshdas@gatech.edu

Shang-Tse Chen
schchen351@gatech.edu

Evaluating and Understanding the Robustness of Adversarial Logit Pairing

Logan Engstrom*, Andrew Ilyas*, Anish Athalye*

Massachusetts Institute of Technology
tengstrom@mit.edu, ilyas@mit.edu, athalye@mit.edu

Abstract

We evaluate the robustness of Adversarial Logit Pairing, a recently proposed defense against adversarial examples. We find that a network trained with Adversarial Logit Pairing achieves 0.6% correct classification rate under targeted adversarial attack, the threat model in which the defense is considered. We provide a brief overview of the defense and the threat models/claims considered, as well as a discussion of the methodology and results of our attack. Our results offer insights into the reasons underlying the vulnerability of ALP to adversarial attack, and are of general interest in evaluating and understanding adversarial defenses.

1 Contributions

For summary, the contributions of this note are as follows:

1. **Robustness:** Under the white-box targeted attack threat model specified in Kannan et al., we upper bound the correct classification rate of the defense to 0.6% (Table 1). We also perform targeted and untargeted attacks and show that the attacker can reach success rates of 98.6% and 99.9% respectively (Figures 1, 2).

Towards Evaluating the Robustness of Neural Networks

IEEE Symposium on Security and Privacy, 2017. **Best Student Paper.**

Nicholas Carlini and David Wagner

Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods

ACM Workshop on Artificial Intelligence and Security, 2017. **Finalist, Best Paper.**

Nicholas Carlini and David Wagner

On the Robustness of the CVPR 2018 White-Box Adversarial Example Defenses

Computer Vision: Challenges and Opportunities for Privacy and Security, 2018.

Anish Athalye and **Nicholas Carlini**

Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples

International Conference on Machine Learning, 2018. **Best Paper.**

Anish Athalye*, **Nicholas Carlini***, and David Wagner

On Adaptive Attacks to Adversarial Example Defenses

NeurIPS, 2020.

Florian Tramer, **Nicholas Carlini**, Wieland Brendel, Aleksander Madry

Evading Adversarial Example Detection Defenses with Orthogonal Projected Gradient Descent

ICLR, 2022.

Oliver Bryniarski, Nabeel Hingun, Pedro Pachuca, Vincent Wang, **Nicholas Carlini**



Nicholas Carlini

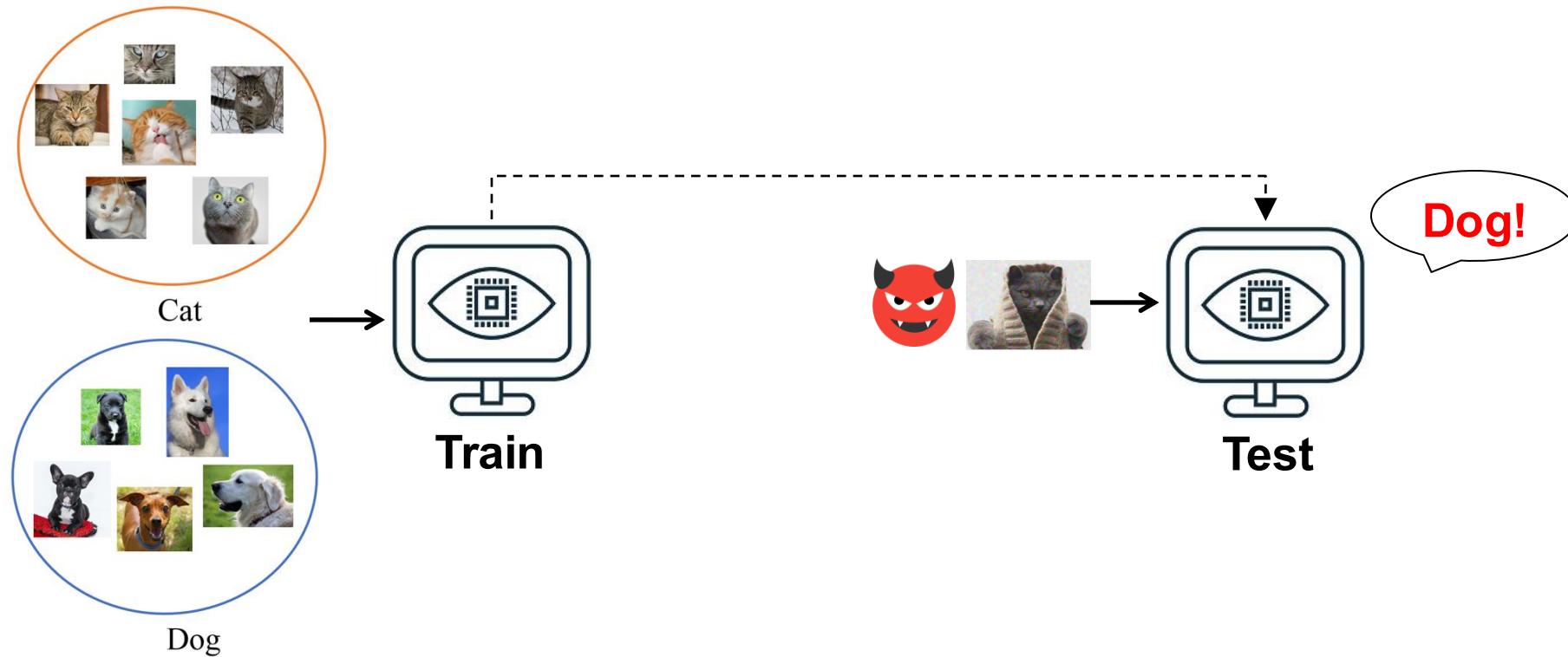
Research Scientist, Google

DeepMind

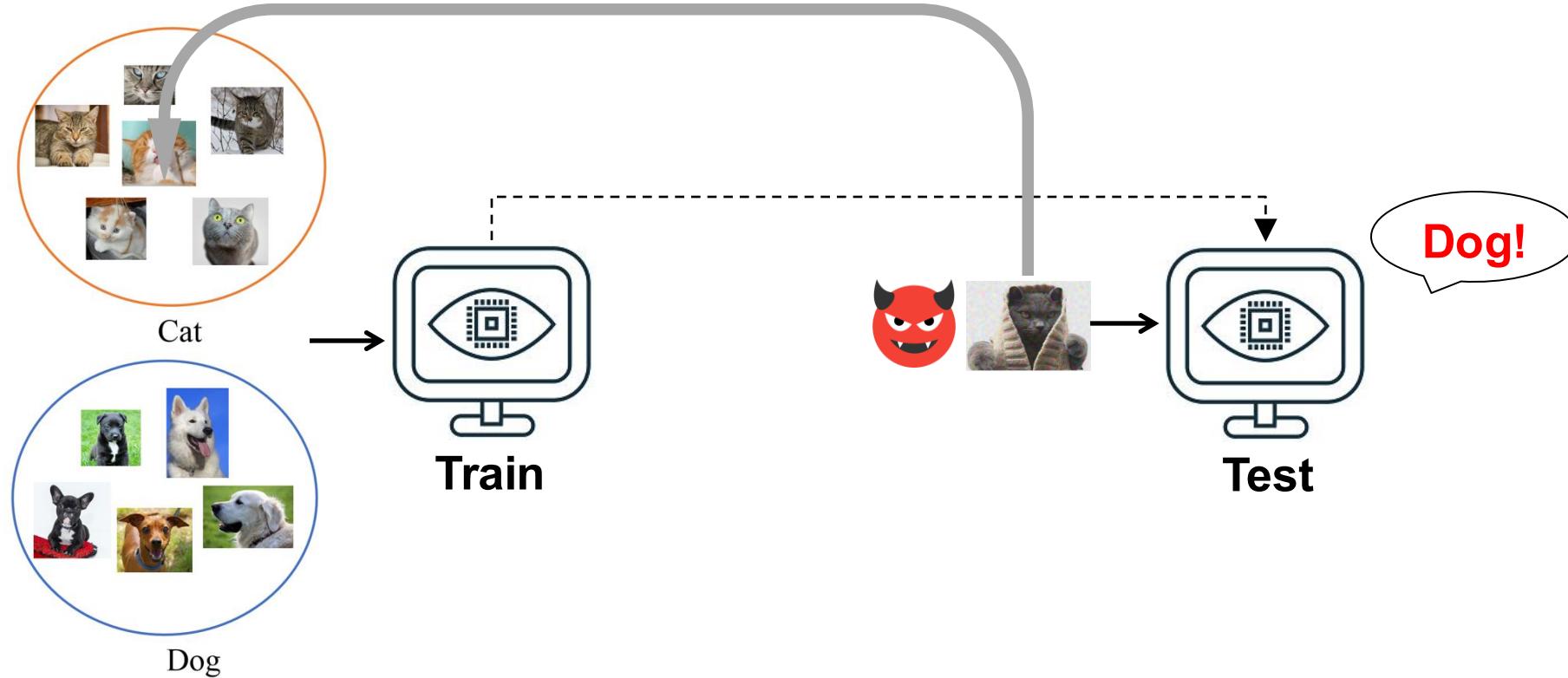
nicholas [at] carlini [dot] com

[GitHub](#) | [Google Scholar](#)

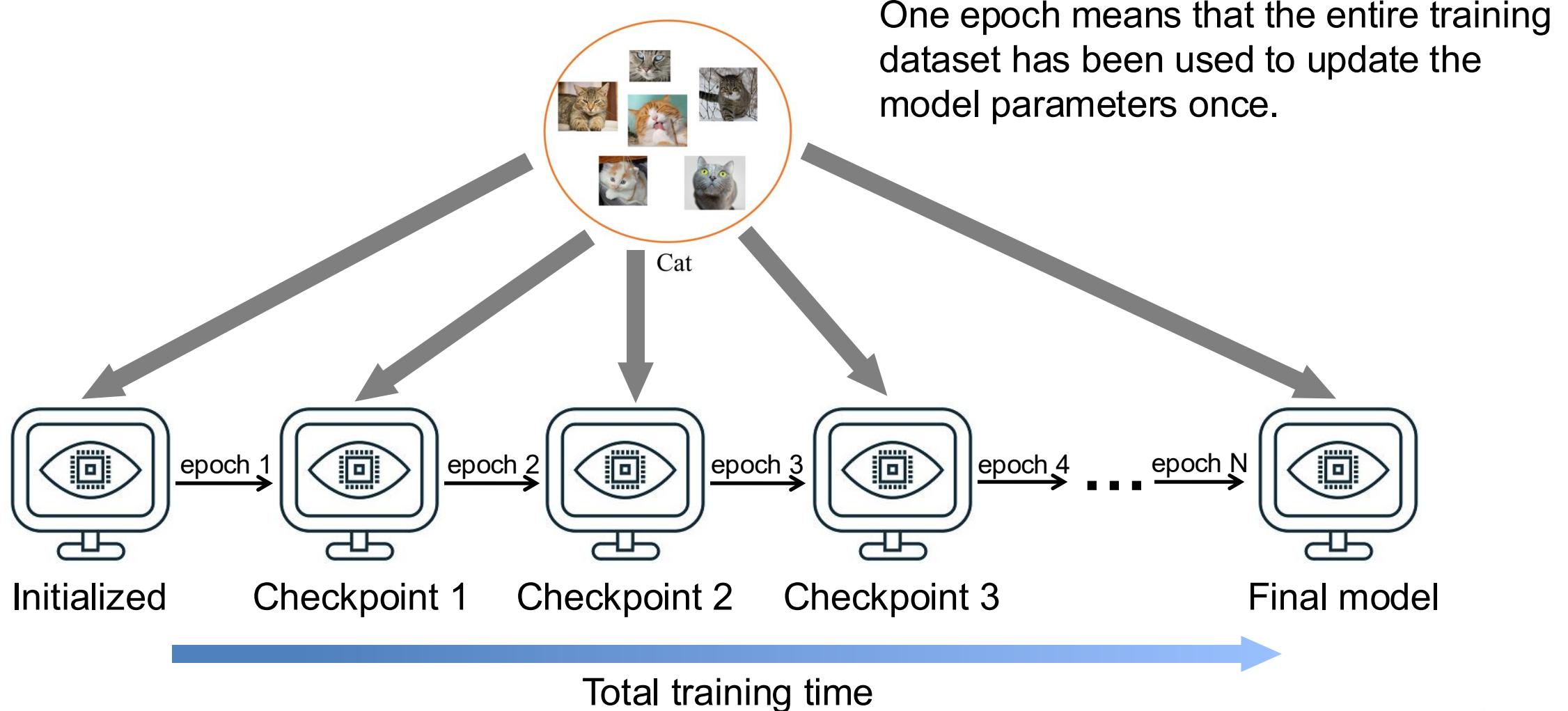
That One Defense: Adversarial Training



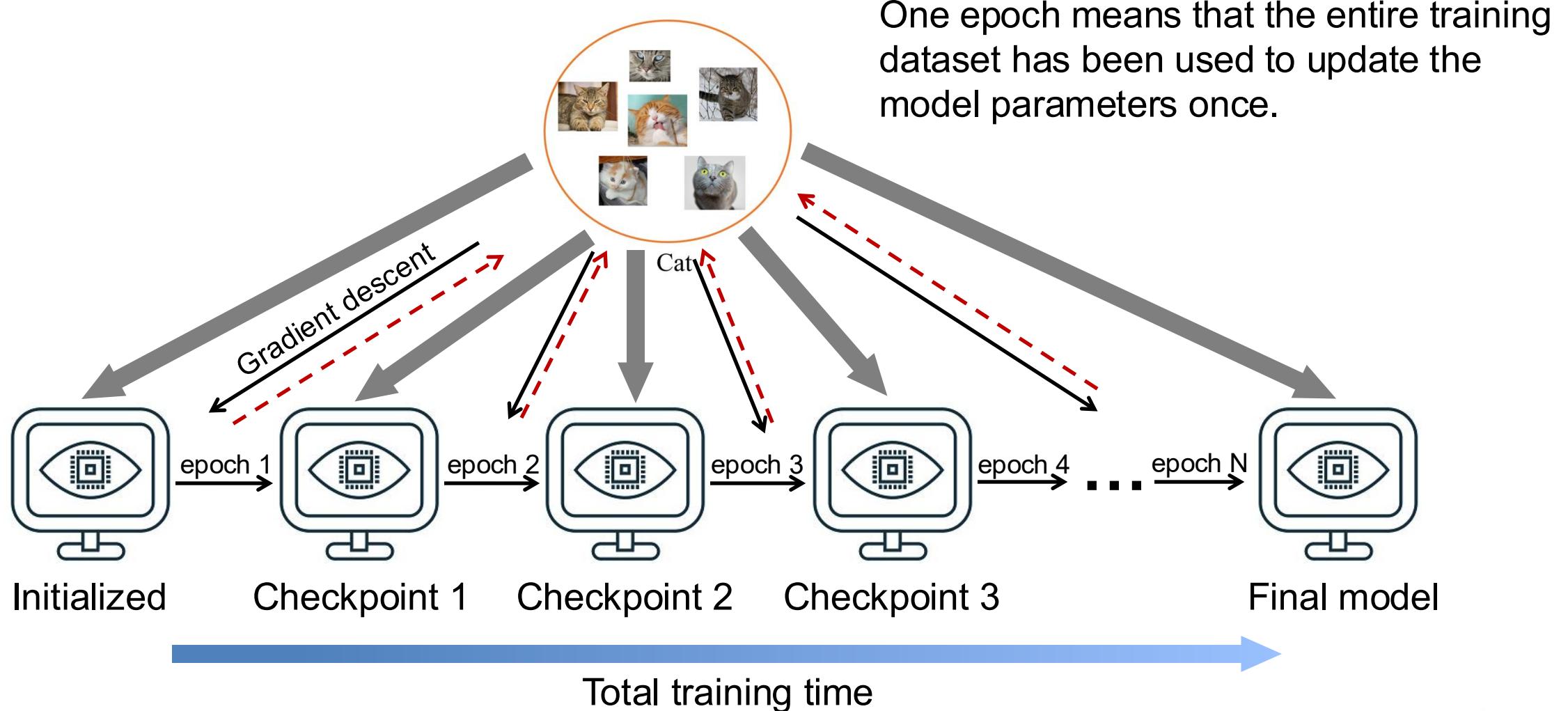
That One Defense: Adversarial Training



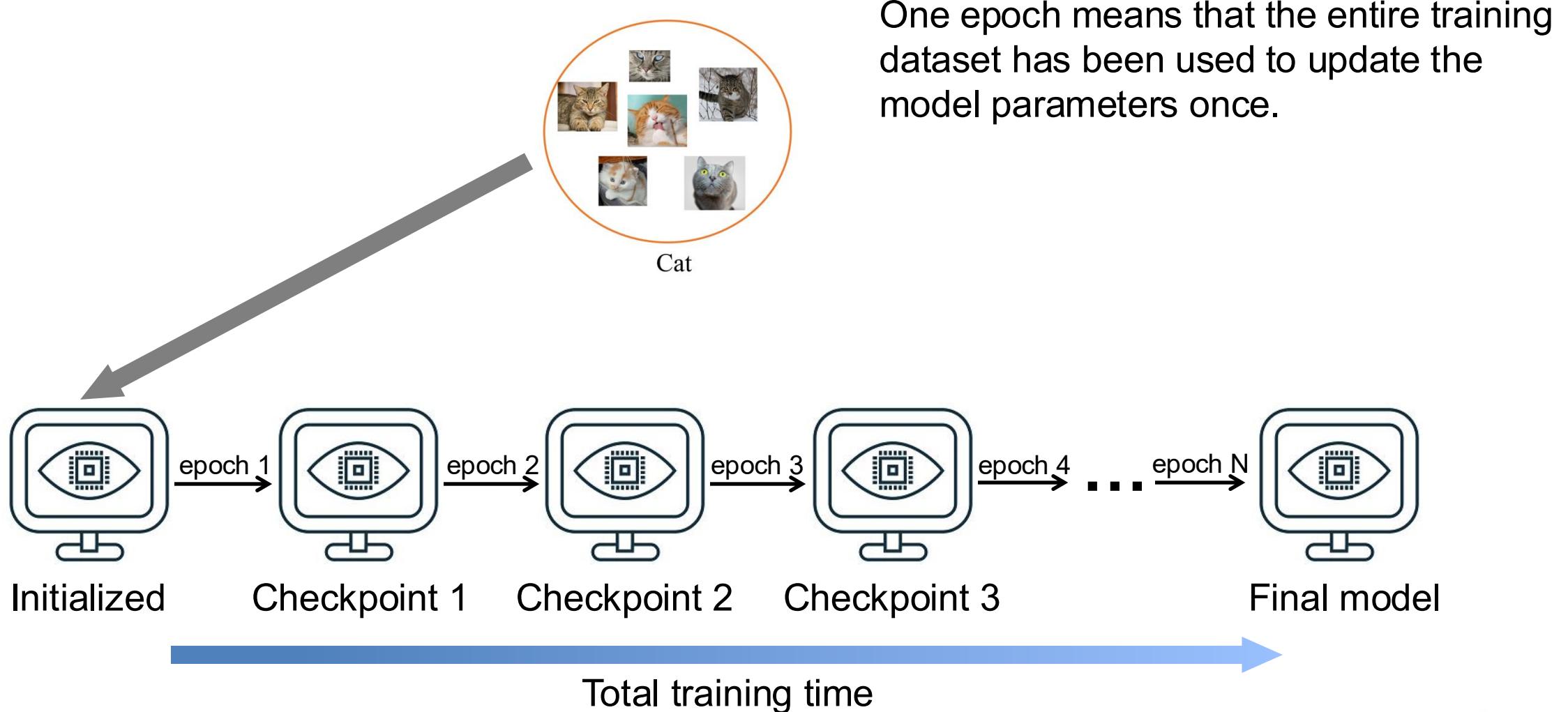
That One Defense: Adversarial Training



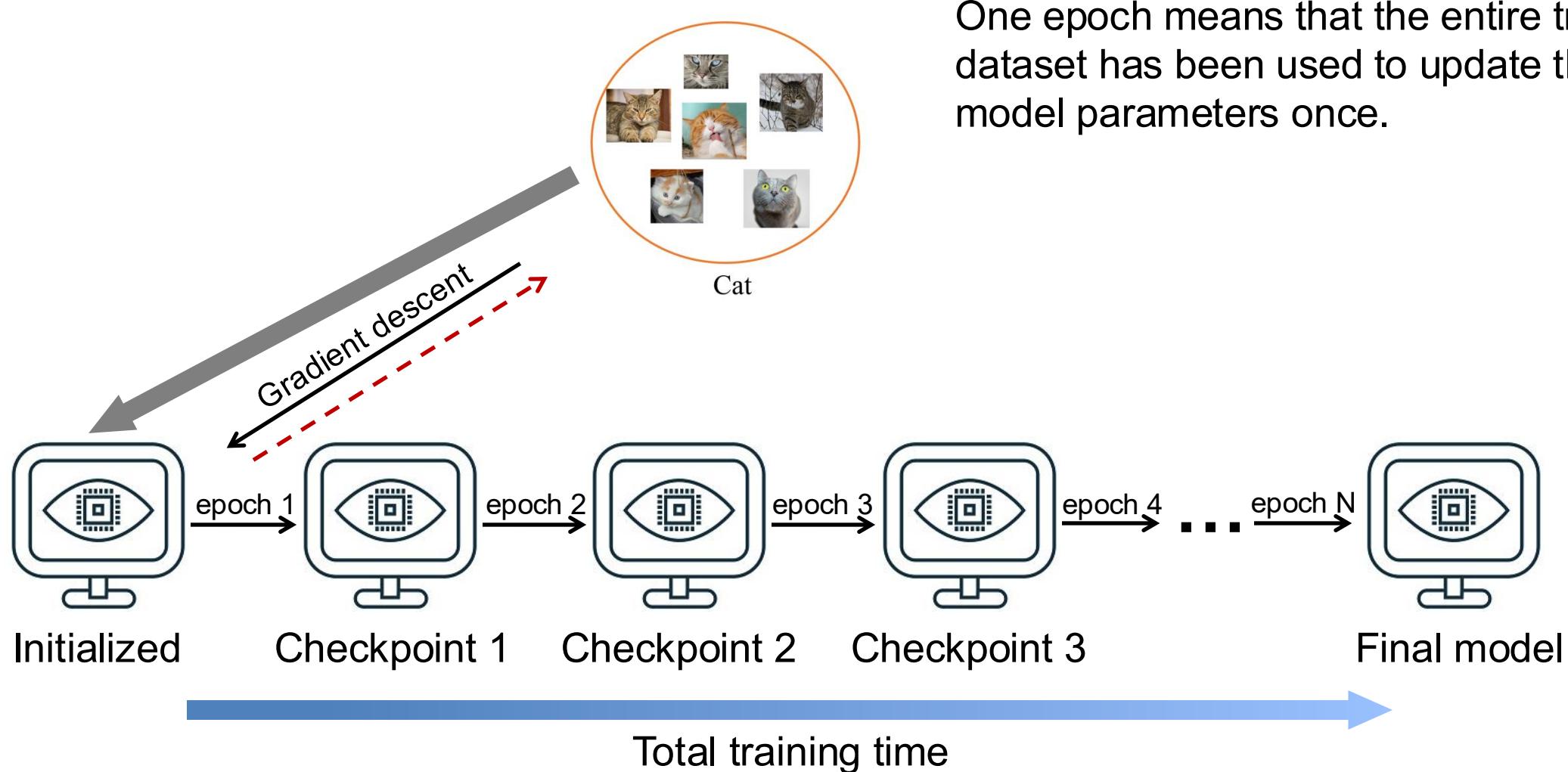
That One Defense: Adversarial Training



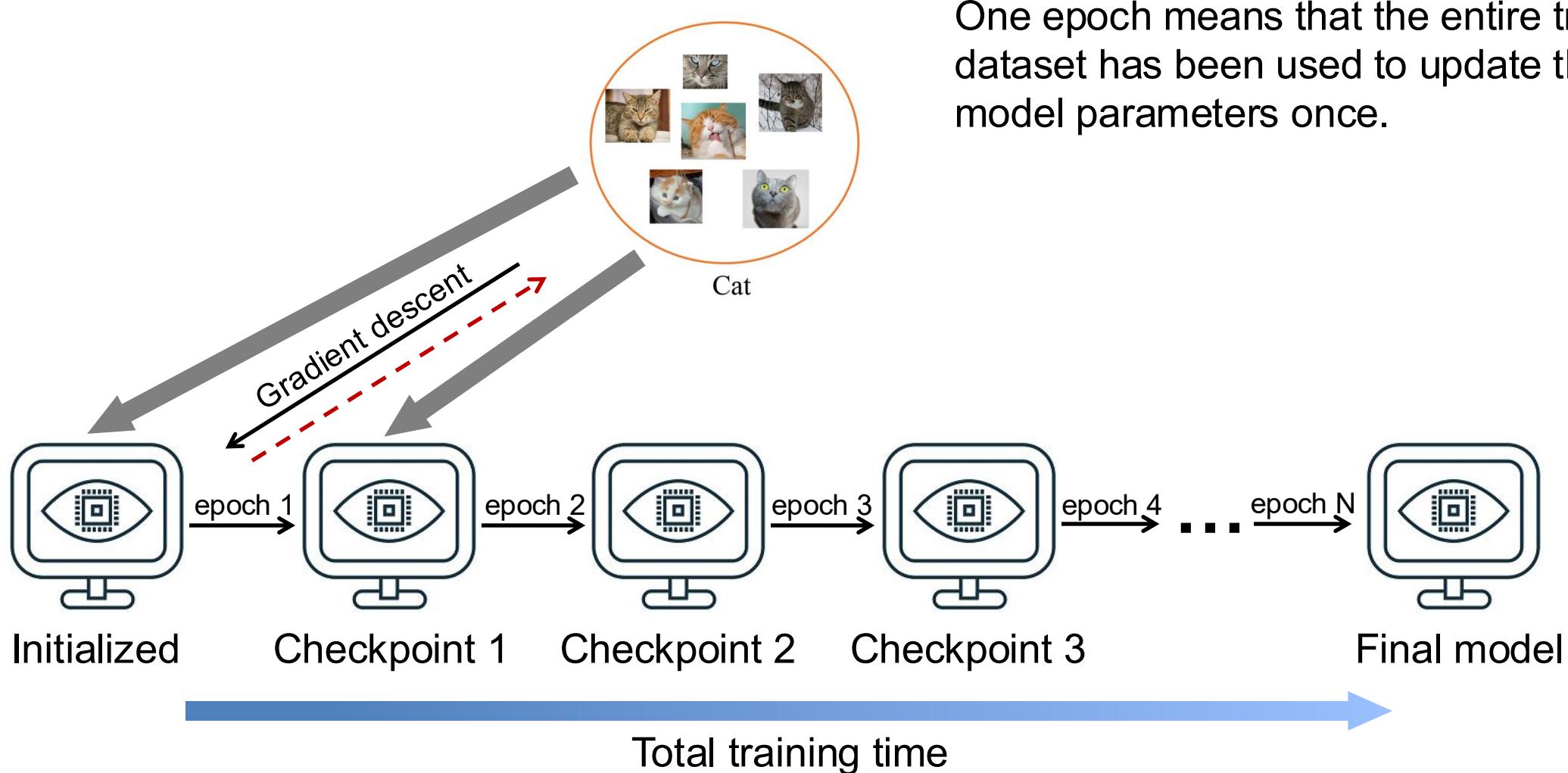
That One Defense: Adversarial Training



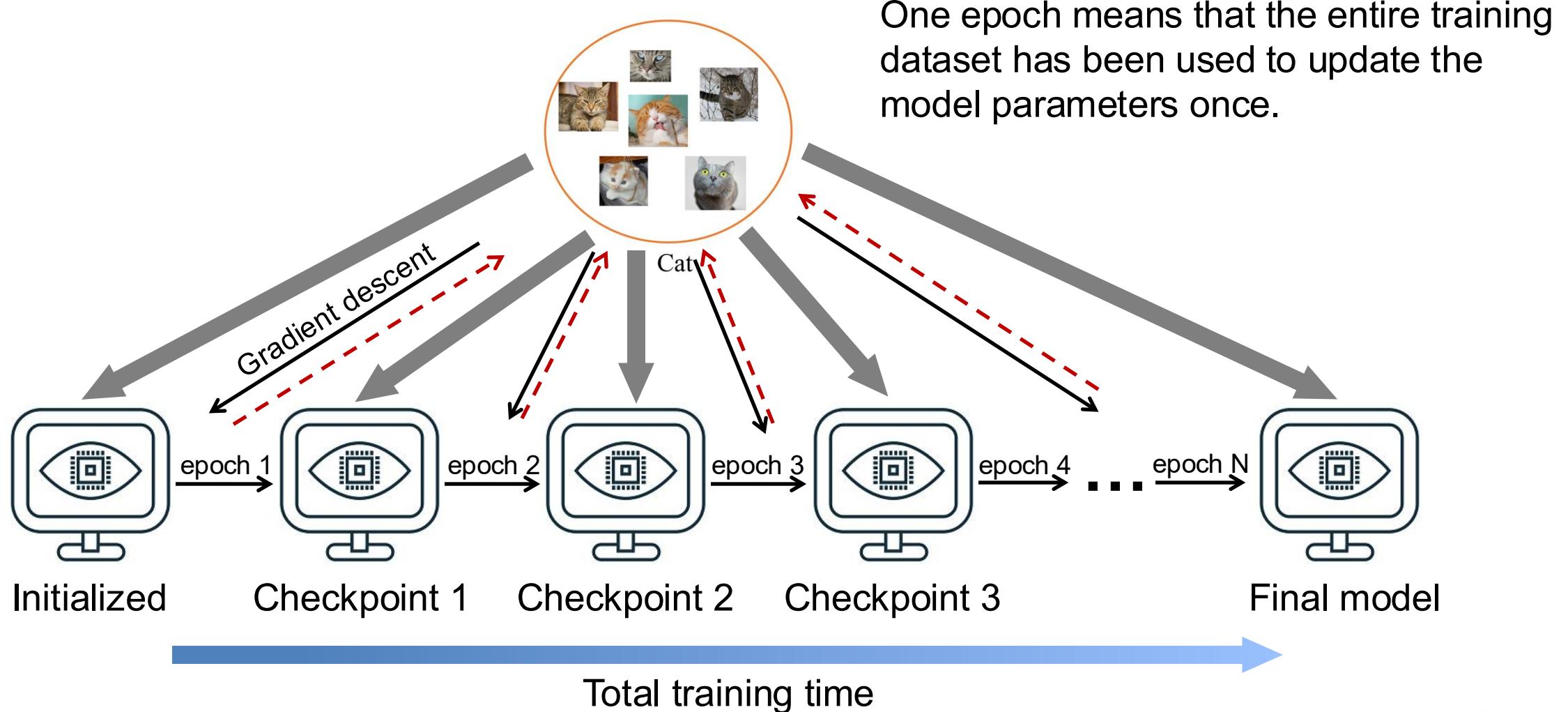
That One Defense: Adversarial Training



That One Defense: Adversarial Training



That One Defense: Adversarial Training



Method 2: Projected Gradient Descent (PGD)

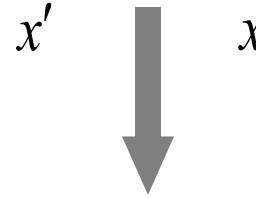
Objective: $x' = \arg \max_x d(y, y_{\text{bird}})$



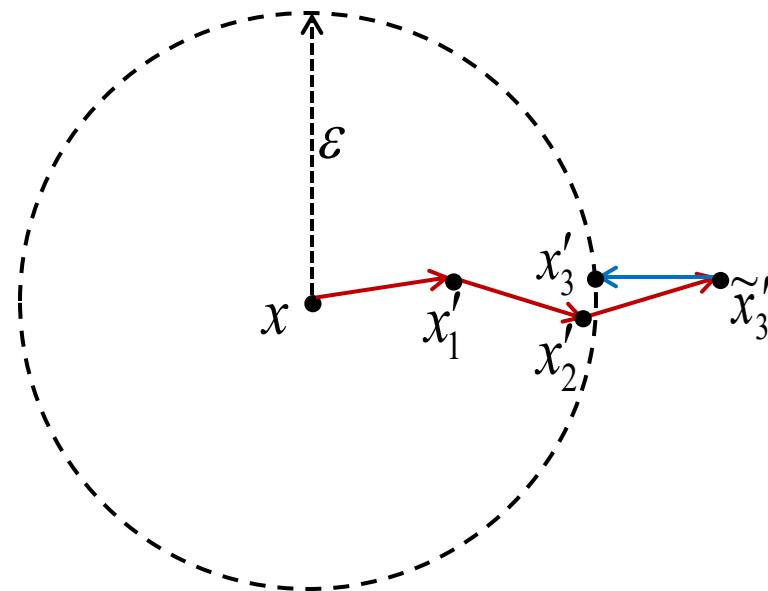
$$x'_0 = x,$$

$$x'_{i+1} = x'_i - \alpha \cdot \text{sign}(\nabla_x d(y, y_{\text{bird}}))$$

s.t. $\| \begin{array}{c} \text{[Image of a bird]} \\ - \end{array} \|_p \leq \varepsilon$



$$x' \leftarrow \text{project}(x' - x, -\varepsilon, \varepsilon)$$



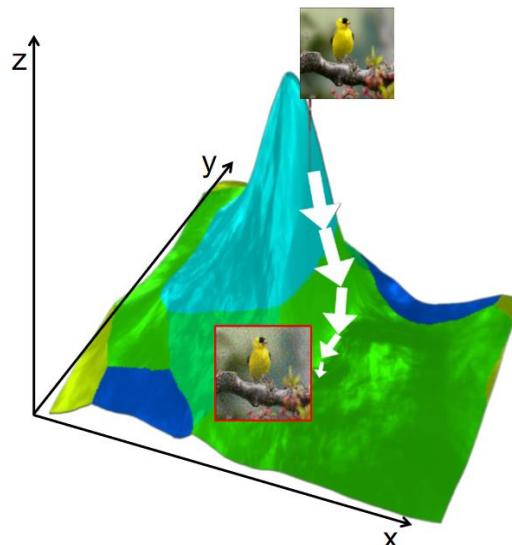
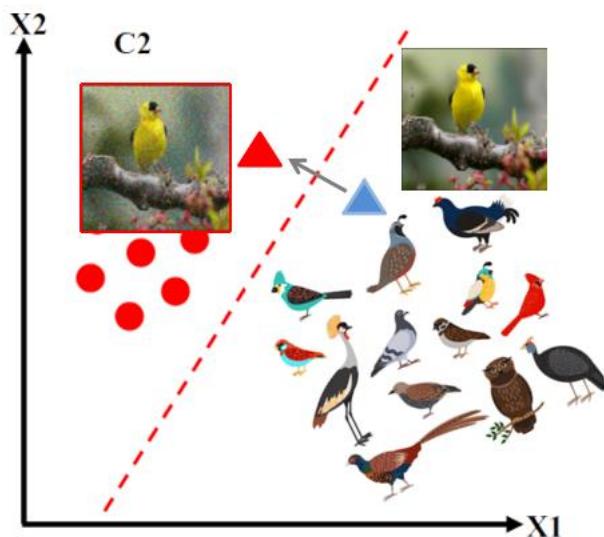
Summary

Success of computer vision

↳ Failures Against Abnormal Examples

↳ Adversarial Examples: Definition

↳ Attacks: Illustration, Formulation, Optimization



Summary

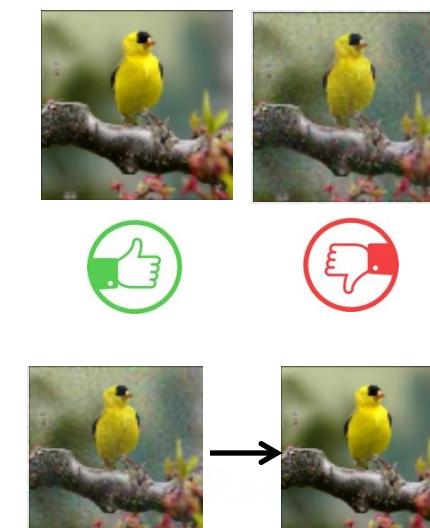
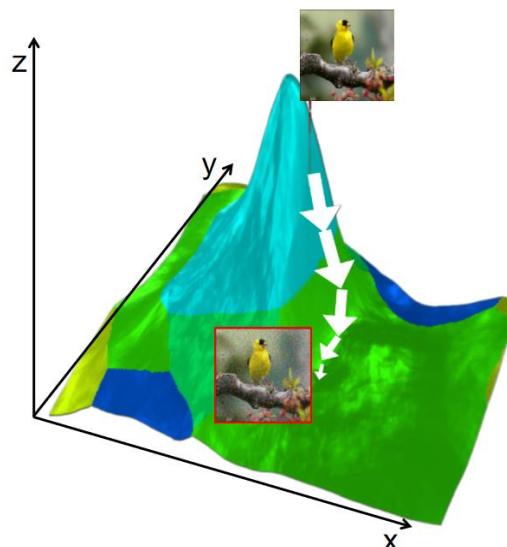
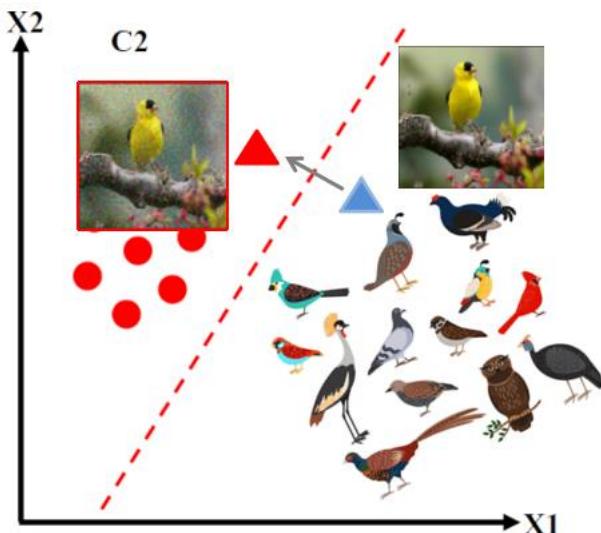
Success of computer vision

↳ Failures Against Abnormal Examples

↳ Adversarial Examples: Definition

↳ Attacks: Illustration, Formulation, Optimization

↳ Defenses: (test) Detection/Pre-processing, (train) Augmentation



Code Example

Optimize adversarial examples using PyTorch

https://savan77.github.io/blog/imagenet_adv_examples.html