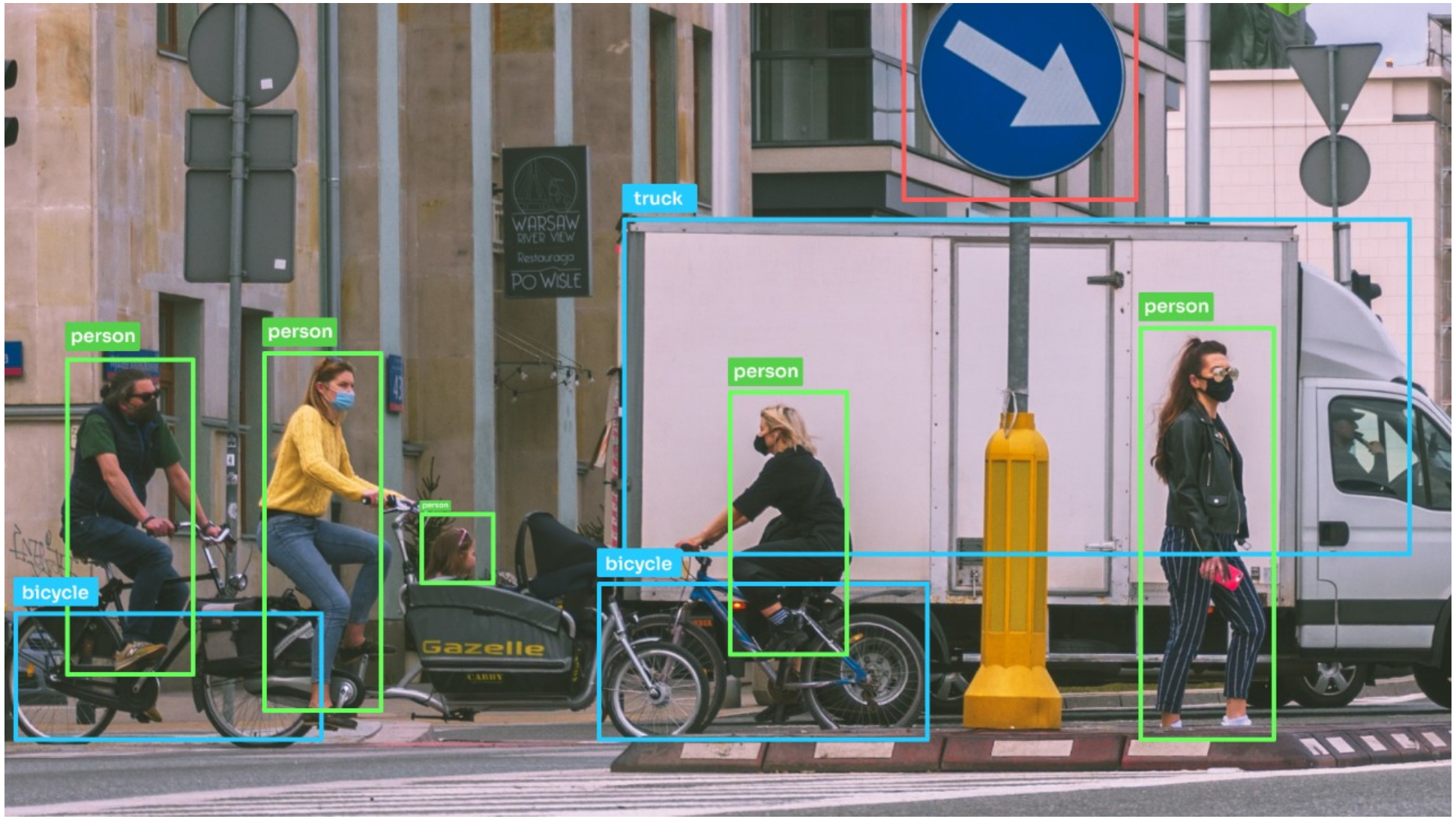
Gradient-Based Adversarial Attacks: Fast Gradient Sign Method (FGSM) and its Extensions

Dr. Franziska Boenisch
SprintML Lab, CISPA Helmholtz Center for Information Security

Machine Learning (ML) models are powerful...



… but they are easily fooled!



~~Panda?~~

Gibbon!



Panda!



~~Cat?~~

Guacamole!

Cat!

Goodfellow et al., ICLR'15

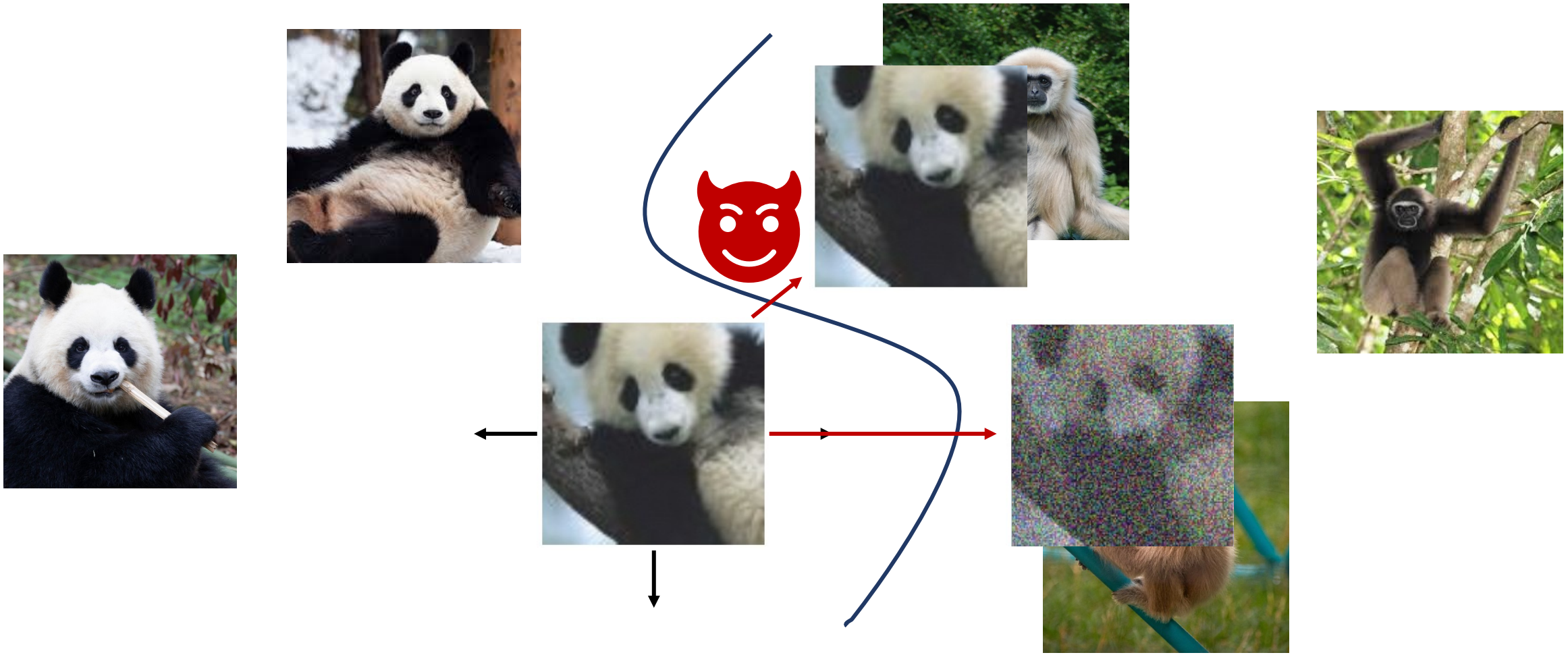
Misclassification through targeted perturbations



Original Image

Panda
57.7% Confidence

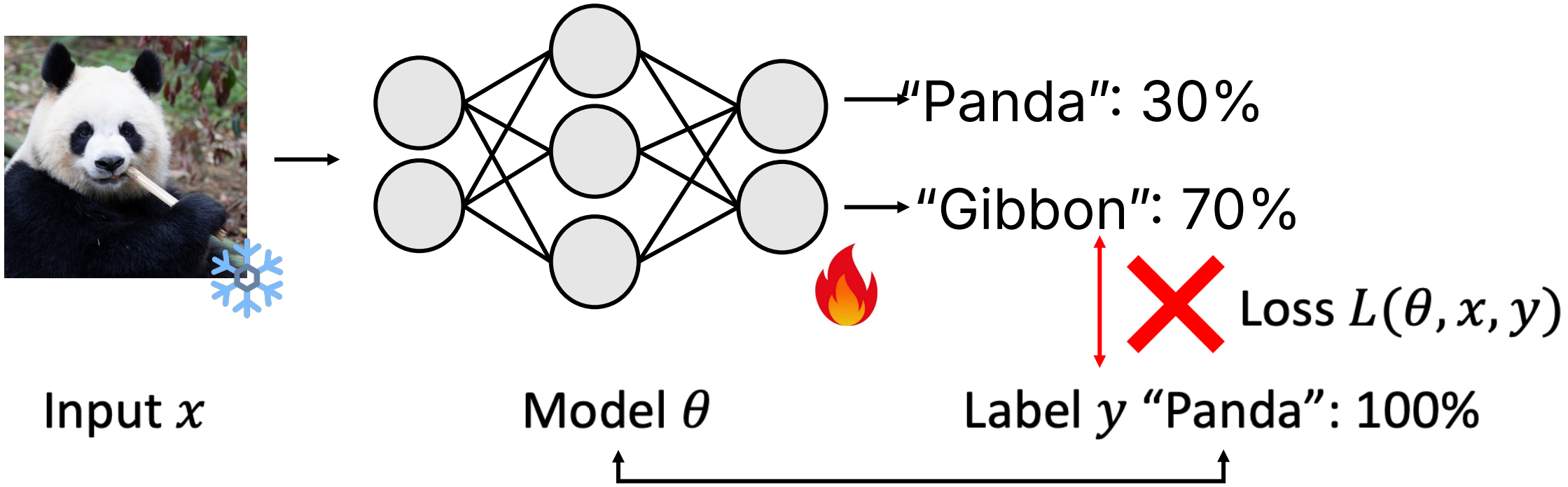
Noise pushes data over the decision boundary



"Panda"

"Gibbon"

How do ML models learn?



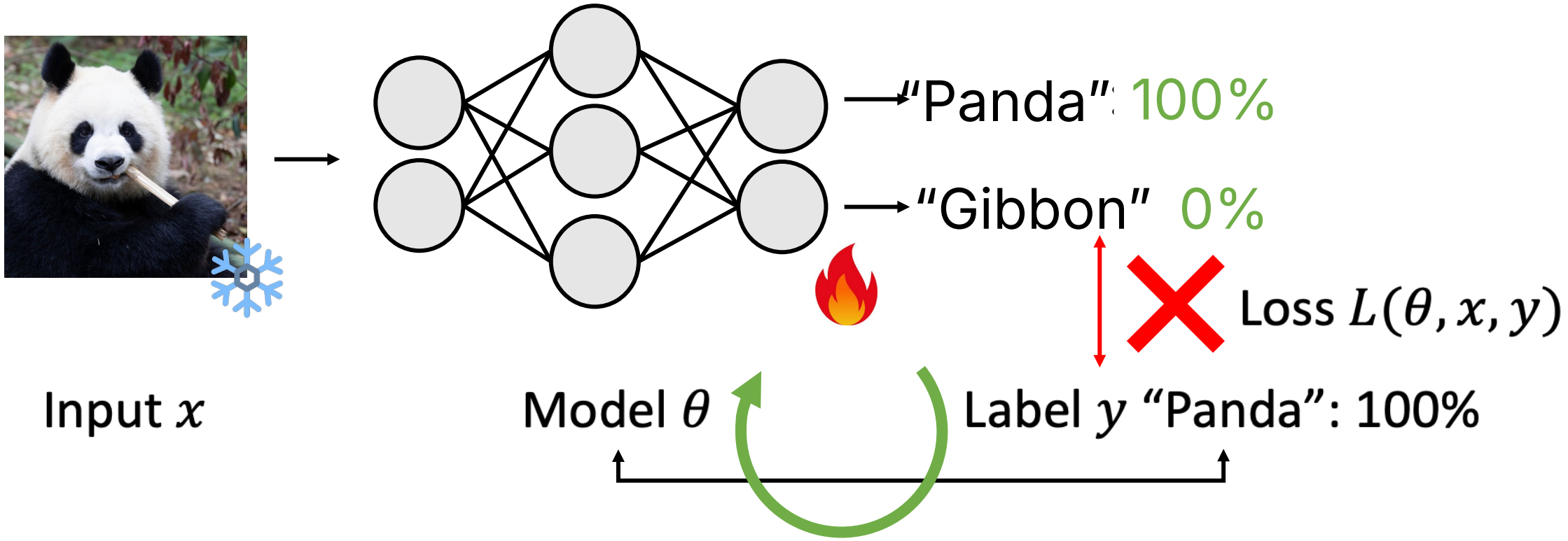
Trainable = can change

Frozen = cannot change

- Gradient $\nabla_\theta L(\theta, x, y)$ **Gradient Descent**

= Direction of the highest increase

How do ML models learn?



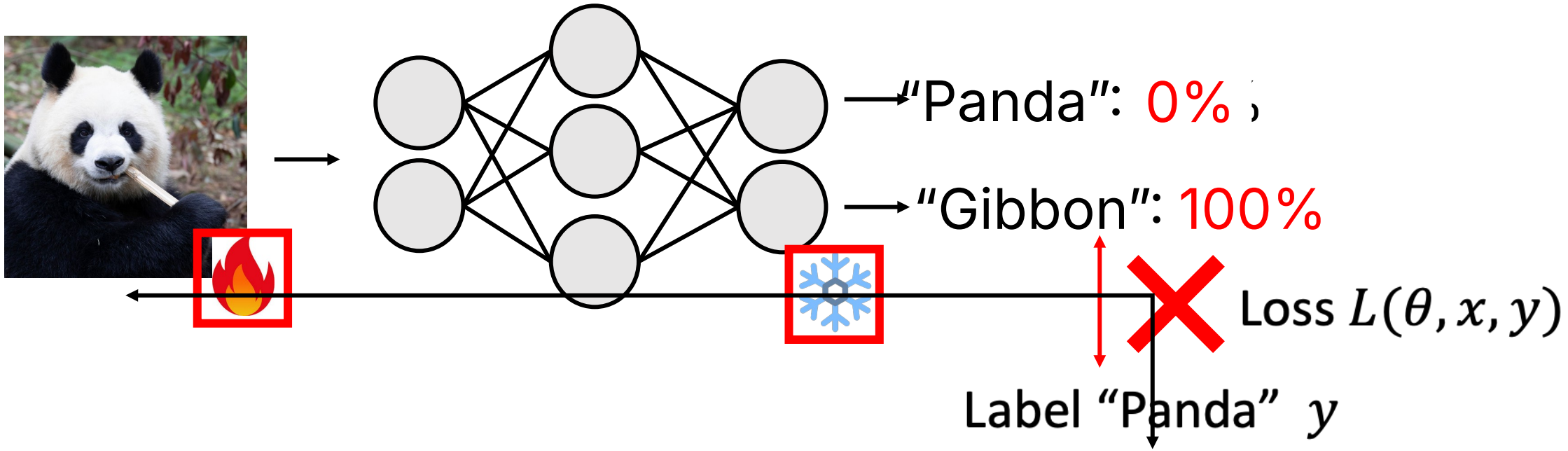
Trainable = can change

Frozen = cannot change

- Gradient $\nabla_\theta L(\theta, x, y)$ **Gradient Descent**

= Direction of the highest increase

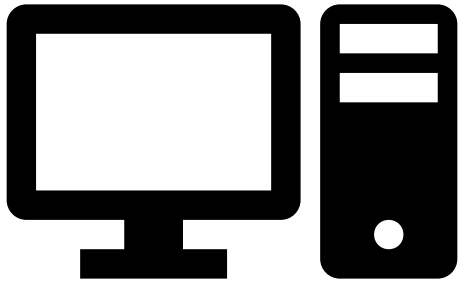
How do we "learn" adversarial noise?



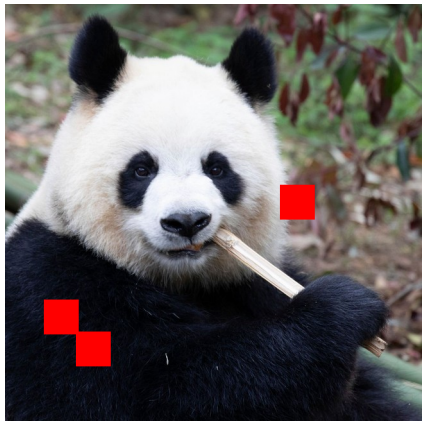


+Gradient $\nabla_x L(\theta, x, y)$ **Gradient Ascent**

What are the drawbacks of gradient ascent?



1. Computes entire gradient



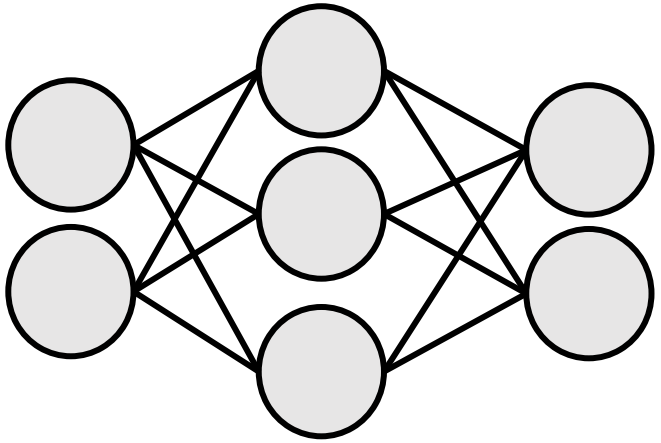
2. Perceptible changes



3. Unlimited changes

The Fast Gradient Sign Method (FGSM)





Loss $L(\theta, x, y)$

-1	-1	1
1	-1	1
-1	-1	1

-2.1	-5.3	1.0
3.5	-4.5	2.3
-4.3	-3.3	1.7

sign(Gradient $\nabla_x L(\theta, x, y)$)

Goodfellow et al., ICLR'15

The Fast Gradient Sign Method (FGSM)

Perturbation Budget



=



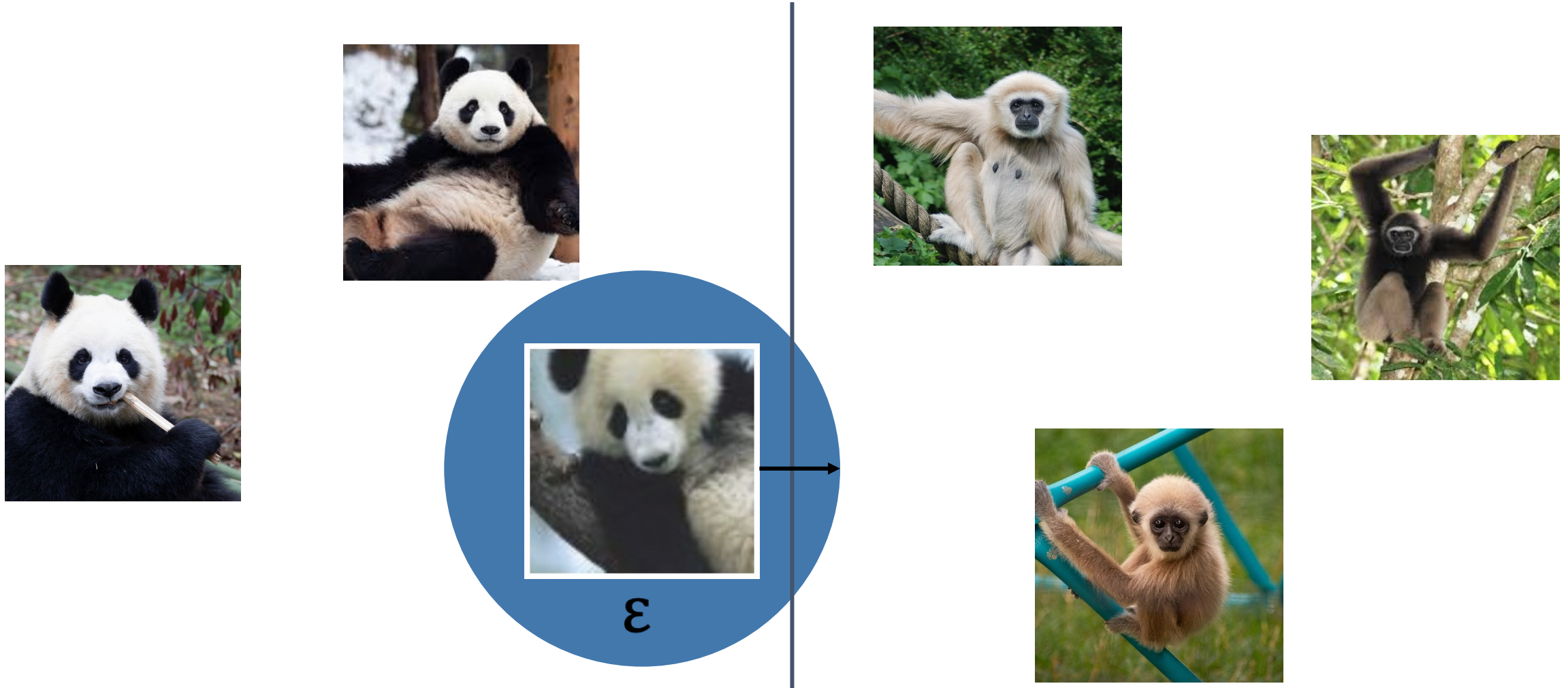
+ $\varepsilon \cdot$

-1	-1	1
1	-1	1
-1	-1	1

$$x' = x + \varepsilon \cdot sign\left(\nabla_x L(\theta, x, y)\right)$$

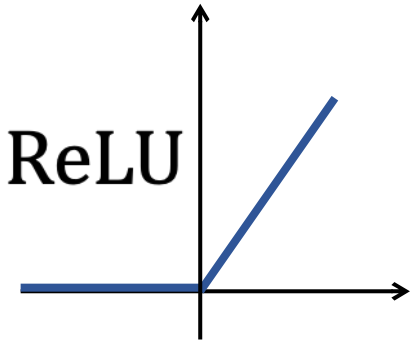
Goodfellow et al., ICLR'15

Intuition: Why is the sign sufficient?



Goodfellow et al., ICLR’15

Intuition: Why is the sign sufficient?













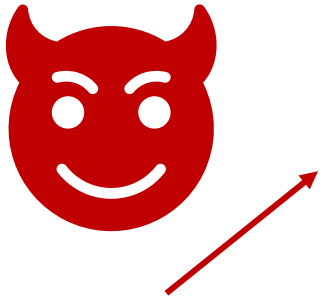


"Panda"

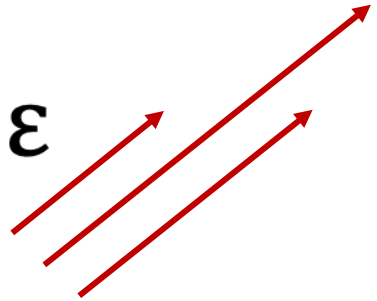
"Gibbon"

Goodfellow et al., ICLR'15

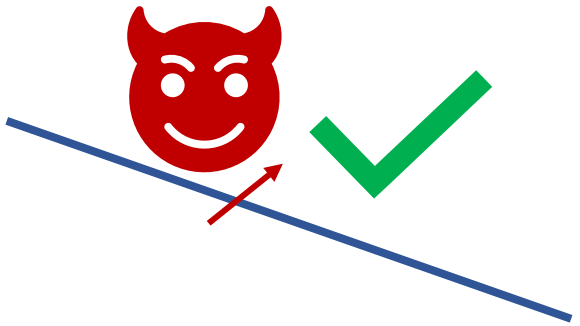
Limitations of FGSM



1. One step only

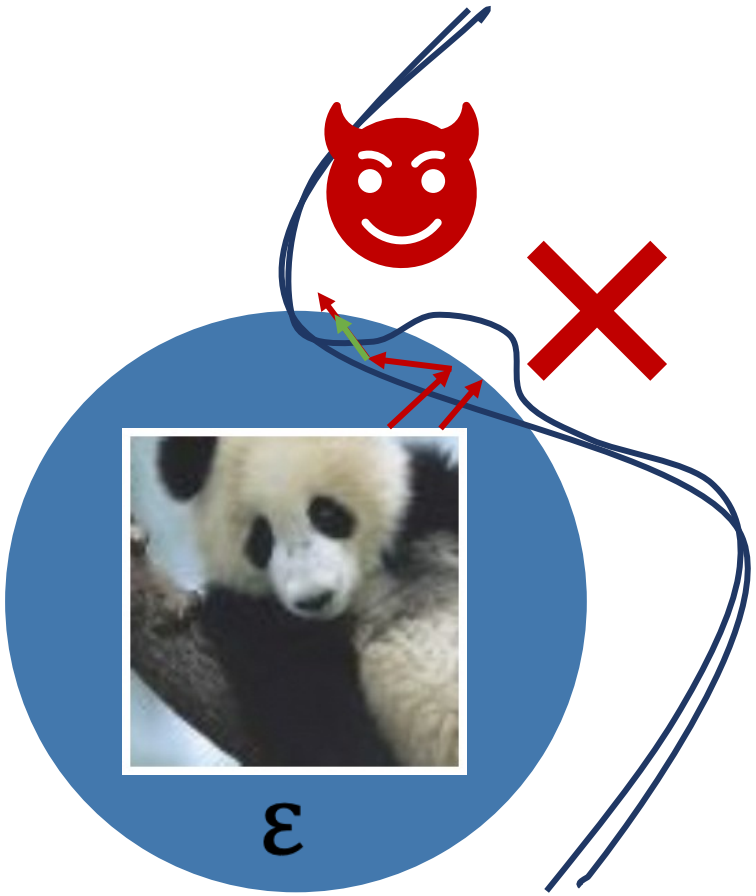


2. Sensitive to step size



3. No adaptation

Extending FGSM: Basic Iterative Method (BIM)



$$x'_0 = x$$

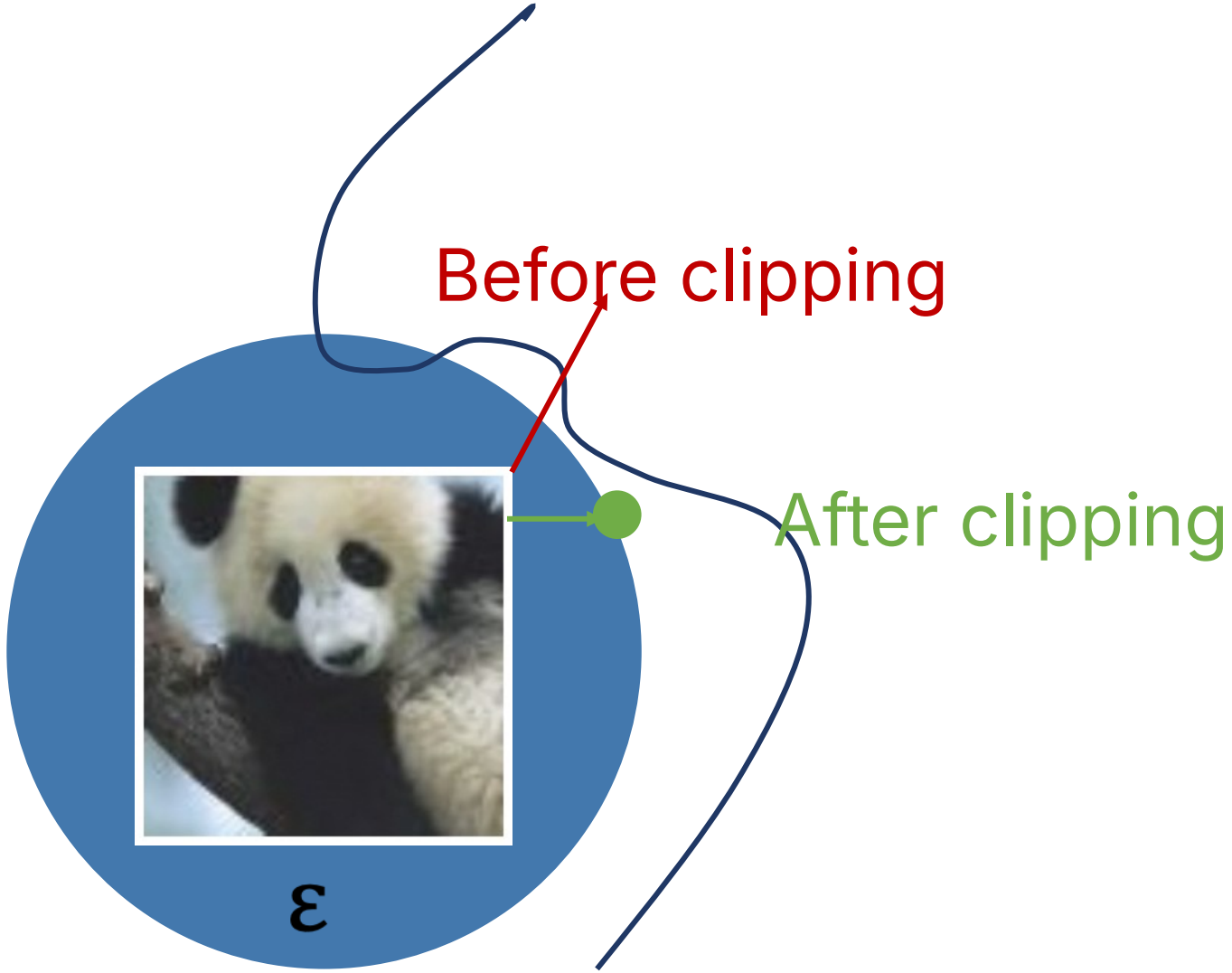
For i in range(1,N):

$$x'_i = clip_{,\varepsilon}\{x'_{i-1} + \alpha \cdot sign\left(\nabla_x L(\theta, x, y)\right)\}$$

Smaller steps than ε

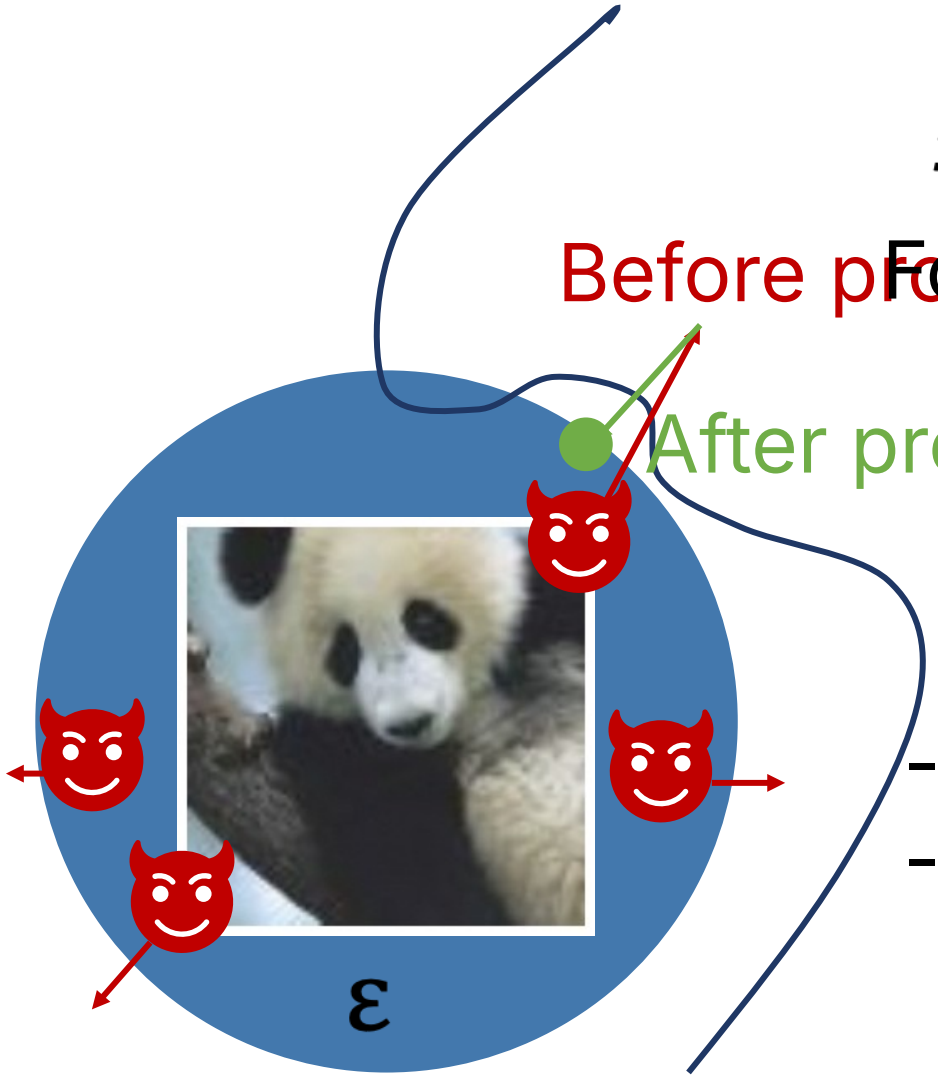
Kurakin et al., 2017

BIM is sensitive to the step size and clipping



Madry et al., ICLR'18

Extending BIM: Projected Gradient Descent (PGD)



$$x'_0 = x + \text{Uniform}(-\varepsilon, \varepsilon)$$

For i in range(1,N):

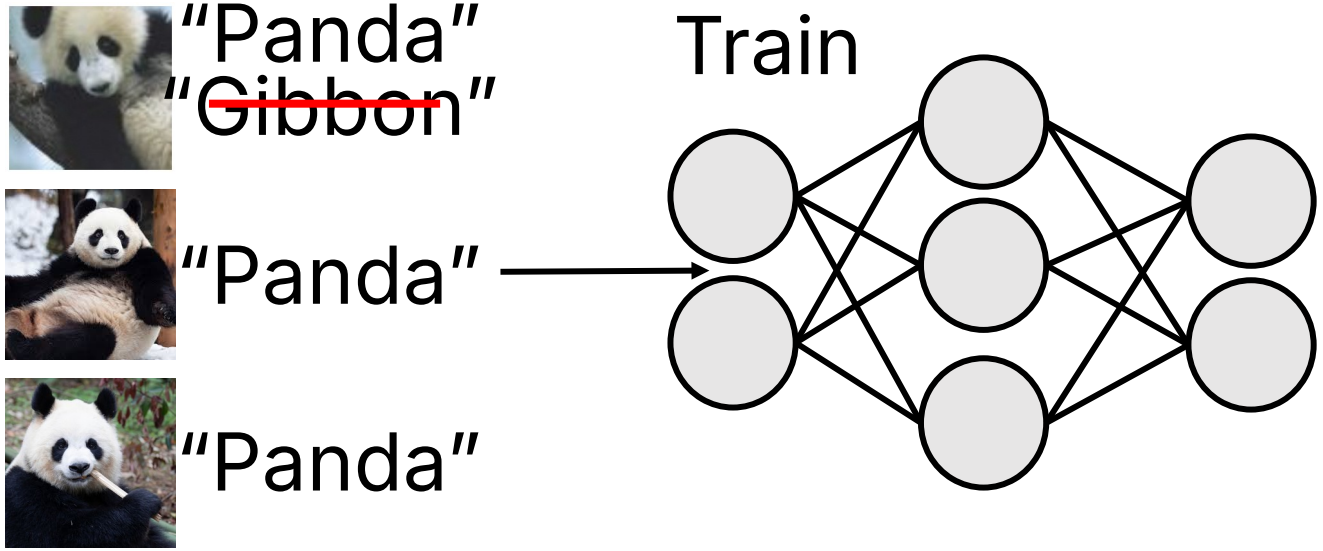
$$x'_i = proj_{,\varepsilon}\{x'_{i-1} + \alpha \cdot sign\left(\nabla_x L(\theta, x, y)\right)\}$$

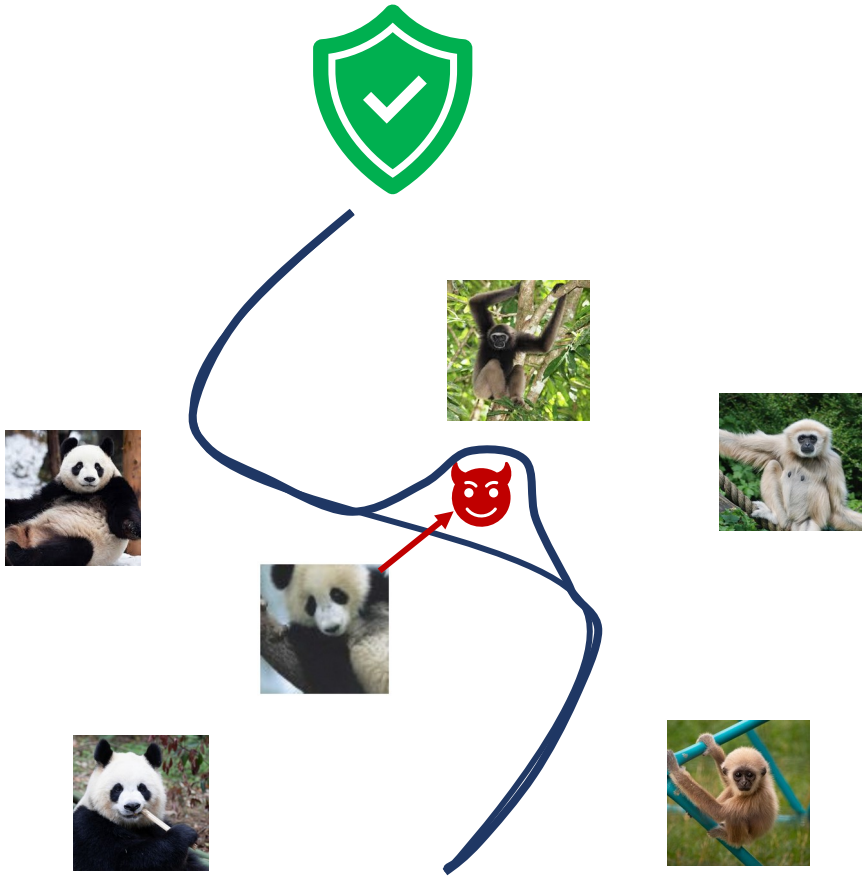
Random starts:

- Start attack at random point next to input
- Keep most effective adversarial example

Madry et al., ICLR'18

Adversarial training as a defense





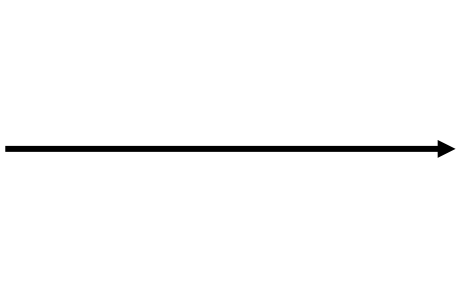
Summary on Adversarial Attacks

Method	Steps	Initialization	Adaptive	Strength
FGSM	1	Data	✗	😈
BIM	Many	Data	✓	😈😈
PGD	Many	Random	✓	😈😈😈



Gibbon

😈



19

Lecture Materials and Discussion

Do you think FGSM and its variants are realistic threats in real-world systems? Why or why not?

SCAN ME

Further Reading

SCAN ME

Video Lecture

SCAN ME

BETA: AI Podcast

More Questions and more discussions!

Is it better to use fast attacks like FGSM in practice (e.g., during training), or stronger attacks like PGD? Why?

Do real attackers really compute gradients?

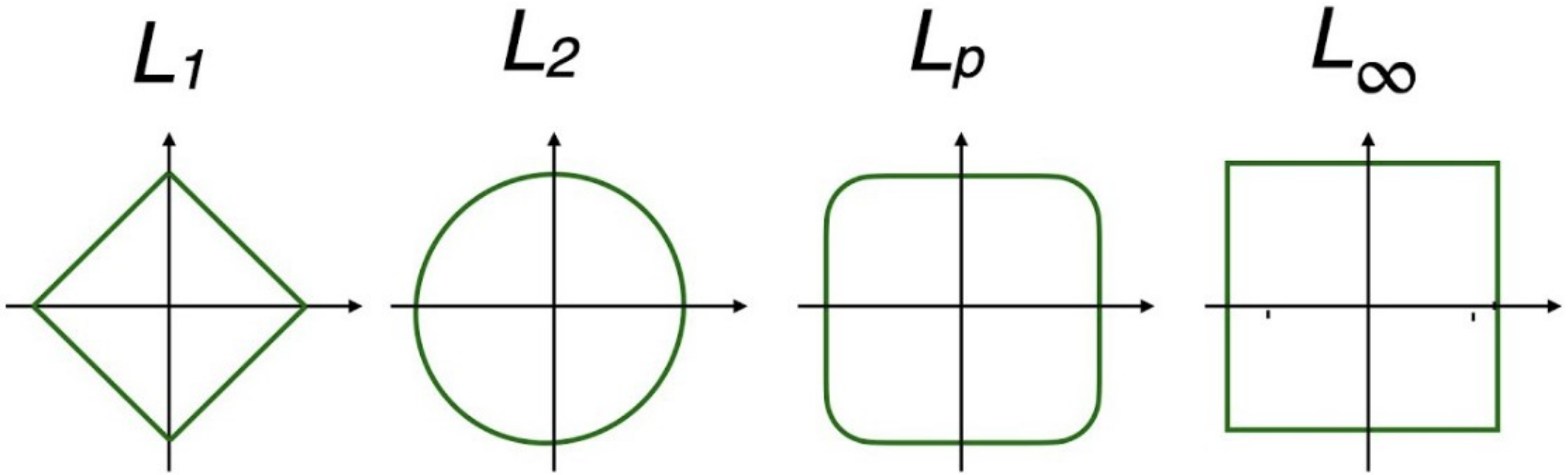
First-order Taylor approximation

- Intuition: *Near a point x, a complicated function behaves like a straight line — that line is determined by the gradient at x.*
- Suppose you have function f(x), and you want to approximate f(x+δ) for a small δ. The first-order Taylor approximation is
  - $f(x+\delta) \approx f(x) + \nabla f(x)^T \delta$
    - $\nabla f(x)$ is gradient of f at x (steepest ascent)
    - $\nabla f(x)^T \delta$ how much function changes if we move by delta
- In the context of FGSM:
  - $L(x+\delta, y) \approx L(x,y) + \nabla_x L(x,y)^T \delta$
- E.g. f(x) = x^2 and we are at x=3. We want to estimate f(3.1)
  - $f'(x) = 2x \Rightarrow f(3) = 6$. First order approximation: $f(3.1) = f(3) + f'(3) \cdot 0.1 = 9 + 6 \cdot 0.1 = \mathbf{9.6}$
  - True $f(3.1) = \mathbf{9.61}$ → pretty close

Mathematical Derivation of FGSM and Its Optimality Under ℓ_∞

- Our starting point:
  - $\max\big(L(x+\delta, y)\big), such\ that\ \big|\big|\delta\big|\big|_\infty \leq \varepsilon$
- First order Taylor approximation
  - $L(x+\delta, y) \approx L(x, y) + \nabla_x L(x, y)^T \delta$
- So our objective is simplified to
  - $\max(\nabla_x L(x, y)^T \delta), such\ that\ \big|\big|\delta\big|\big|_\infty \leq \varepsilon$
- Now the optimization problem is linear in δ and we choose δ in box
  - Maximize dot product: $g^T \delta$, where $g = \nabla_x L(x, y)$
- Solution is $\delta^* = \varepsilon \cdot sign\ (g)$
  - Because each coordinate δ_i should be $+\varepsilon$ or -ε depending on whether g_i is positive or negative to "get the most out of the dot product"

The L_p Norms



Source: https://www.youtube.com/watch?v=NKuLYRui-NU&pp=0gcJCfcAhR29_xXO

FGSM under Different Norms

🌪️ If you want FGSM-style for ℓ_2 norm:

You solve:

$$\max_{\|\delta\|_2 \leq \epsilon} \nabla_x \ell(x, y)^\top \delta$$

The solution is:

$$\delta = \epsilon \cdot \frac{\nabla_x \ell(x, y)}{\|\nabla_x \ell(x, y)\|_2}$$

→ It's like FGSM but instead of taking the **sign**, you **normalize the gradient vector** to unit ℓ_2-norm and scale it by ϵ.

This is sometimes called **"FGM"** (Fast Gradient Method) instead of FGSM.

PGD under Different Norms

🔷 ℓ_∞-PGD (most common)

- Uses **sign(gradient)**
- After each step: clip each pixel independently to stay in the ϵ-box.
- Very effective and commonly used to test robustness.

🔶 ℓ_2-PGD

- Uses **normalized gradient**:

$$x^{(t+1)} = x^{(t)} + \alpha \cdot \frac{\nabla_x \ell(x^{(t)}, y)}{\|\nabla_x \ell(x^{(t)}, y)\|_2}$$

- After each step: project back to ℓ_2-ball → more computationally involved (involves rescaling the perturbation).
- Often produces smoother, more human-like perturbations.

Summary on Adversarial Attacks

Method	Steps	Initialization	Adaptive	Strength
FGSM	1	Data	✗	😈
BIM	Many	Data	✓	😈😈
PGD	Many	Random	✓	😈😈😈



Gibbon

😈

