Machine Learning and Security

Security of ML Systems

Dr. Thorsten Eisenhofer

BIFOLD

TECHNISCHE UNIVERSITÄT BERLIN

Machine learning and security

Machine learning → security and privacy

- Use learning algorithms to *help* conventional approaches
- Sometimes possible to entirely learn a task

Security and privacy → machine learning ← **Focus for today**

- Attacks on and defenses for machine learning
- New approaches to secure and private learning

Outline

Adversarial machine learning

- Overview over different attack vectors
- Robust optimization

Security of ML systems

- Overview of attack surface
- Example: Differential Testing of Linear Algebra Systems

Outline

Adversarial machine learning

- Overview over different attack vectors
- Robust optimization

Security of ML systems

- Overview of attack surface
- Example: Differential Testing of Linear Algebra Systems

Our Focus: Supervised Machine Learning

Parameterized function

$$f_\theta : \mathbb{X} \rightarrow \mathbb{Y}$$

Space of inputs

Space of outputs

Examples

Malware → benign/malicious

Image → car/human/...

Training

Minimize expected generalization error

$$\underbrace{\mathbb{E}_{(\mathbf{x},y)\sim\mathbb{D}}}_{\text{Data distribution}}[\underbrace{l(f_\theta(\mathbf{x}), y))}_{\text{Loss function}}]$$

Empirical risk minimization

$$\underset{\theta}{\text{minimize}} \frac{1}{|D|} \underbrace{\sum_{(\mathbf{x},y)\in D}}_{\text{Finite dataset}} l(f_\theta(\mathbf{x}), y)$$

Minibatch gradient descent

Repeat:

Select random batch $B \subseteq D$

$$\theta := \theta - \alpha \frac{1}{|B|} \sum_{(\mathbf{x},y)\in B} \nabla_\theta l(f_\theta(x), y)$$

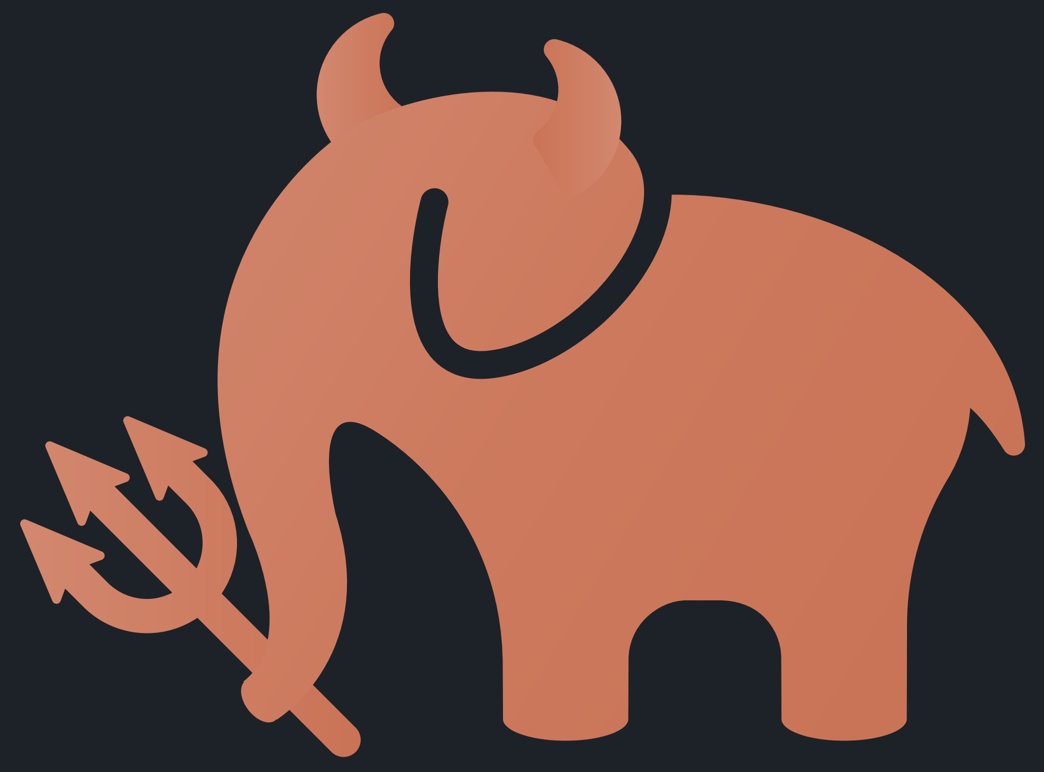
Adversarial Environments

Standard training

- Optimize for expected loss on the training set
- No guarantees for edge cases

Adversarial machine learning

- Can this be exploited by an adversary?
- Study worst-case behavior



Adversary

Threat model

Make claims with regard to the threat model

Goals

- Objective of the attack
- Example: evasion attacks, membership inference, data reconstruction

Knowledge

- White-box with full access, black-box with no access, or grey-box for in between
- Example: access to model parameters or training data

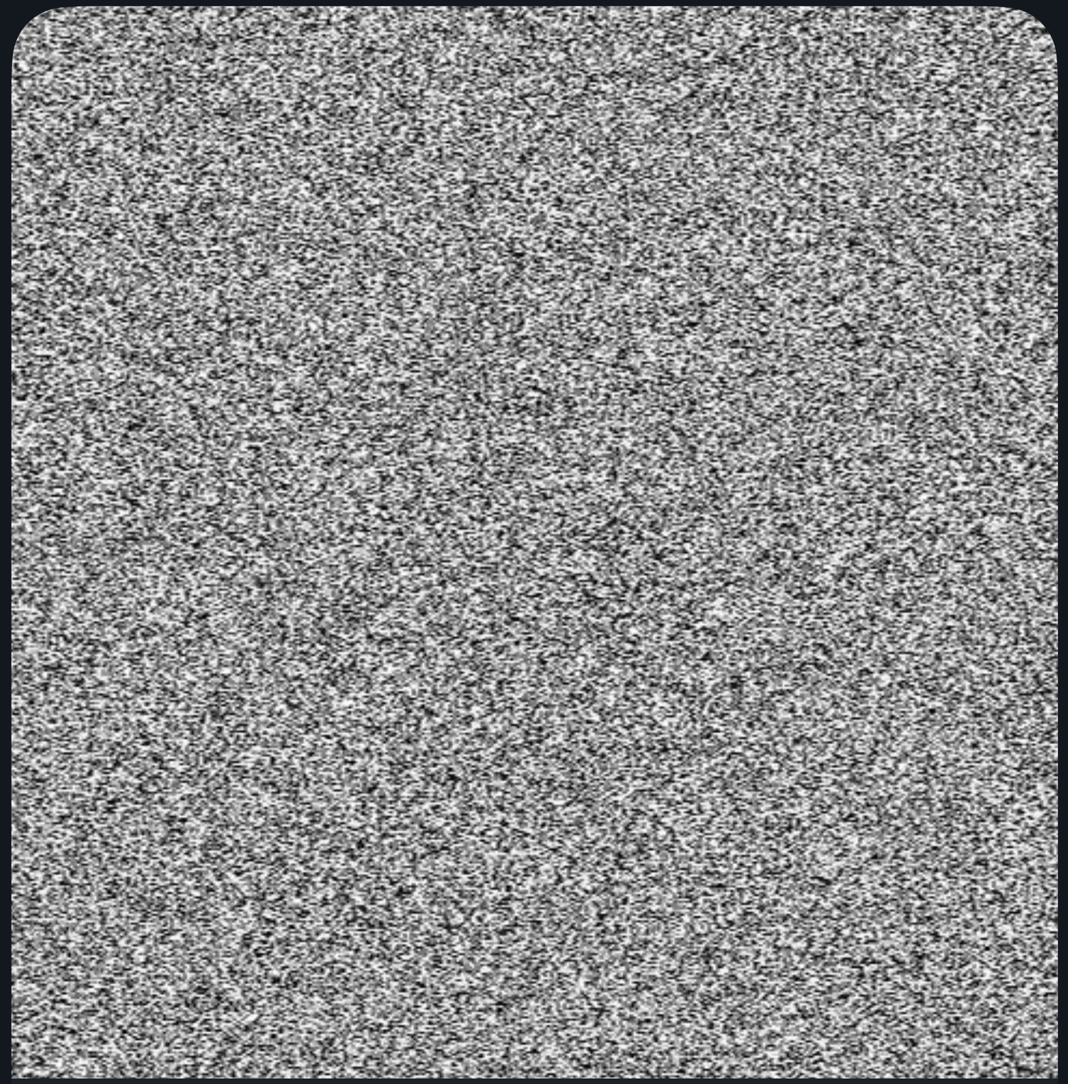
Capabilities

- Training-time attacks vs. deployment-time attacks
- Example: allowed modification to data samples or model weights

Evasion Attacks: Adversarial Examples

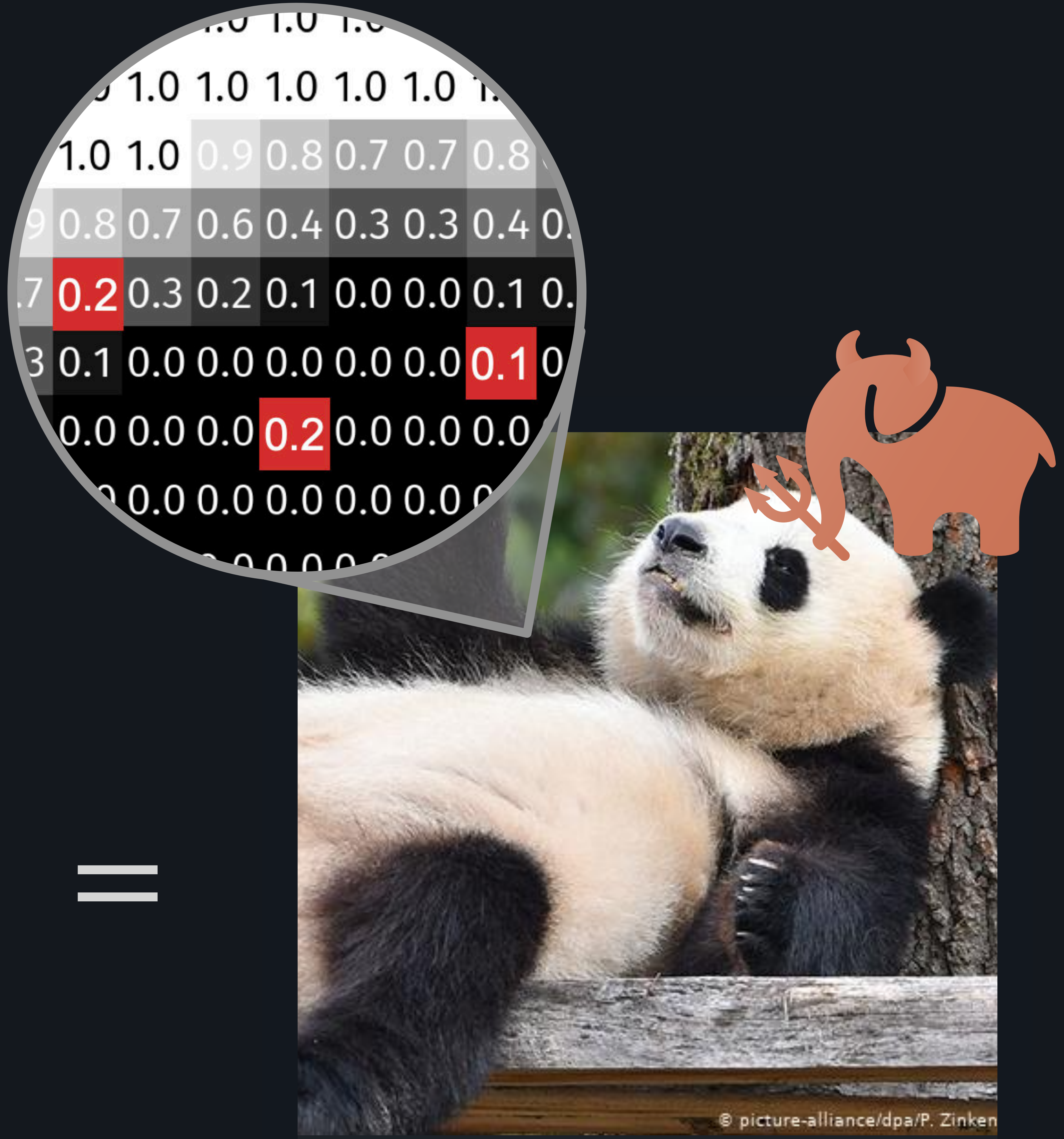
Panda

$+ \varepsilon \cdot$



Perturbations

$=$



Elephant

Goal: Manipulate input to force model into an arbitrary output

How does this work?

Adversarial loss

$$l_{adv}(f_\theta(\mathbf{x}+\delta), y, y_{target}) := \underbrace{l(f_\theta(\mathbf{x}+\delta), y)}_{\text{Increase distance to true class}} - \underbrace{l(f_\theta(\mathbf{x}+\delta), y_{target})}_{\text{Decrease distance to target class}}$$

Perturbation set Δ

e.g., l_∞- ball

$$\Delta := \left\{\delta : ||\delta||_\infty \le \epsilon\right\}$$

Adversarial examples

$$\underset{\delta\in\Delta}{\text{maximize}}\ l_{adv}(f_\theta(\mathbf{x}+\delta), y, y_{target})$$

Instantiations

Goodfellow et al. *"Explaining and Harnessing Adversarial Examples"*, ICLR'15

Fast Gradient Sign Method (FGSM)

$$\delta := \epsilon \cdot \underbrace{\mathrm{sign}}_{\text{Direction only}}(\underbrace{\nabla_\delta}_{\text{Derive to delta}} l_{adv}(f_\theta(\mathbf{x} + \delta), y, y_{target}))$$

Projected gradient descent (PGD)

Repeat:

$$\delta := \underbrace{\mathscr{P}}_{\text{Projection into } \mathbb{X}}(\delta + \alpha \cdot \mathrm{sign}(\nabla_\delta l_{adv}(f_\theta(\mathbf{x} + \delta), y, y_{target})))$$

Outline

Adversarial machine learning

- Overview over different attack vectors
- Robust optimization

Security of ML systems

- Overview of attack surface
- Example: Differential Testing of Linear Algebra Systems

Min-max optimization

Madry et al. *"Towards Deep Learning Models Resistant to Adversarial Attacks"*, ICLR'18

$$\underset{\theta}{\text{minimize}} \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \underset{\delta\in\Delta}{\text{maximize}} \; l(f_\theta(\mathbf{x}+\delta), y)$$

Minibatch gradient descent

Repeat:

Select random batch $B \subseteq D$

$$\theta := \theta - \alpha \frac{1}{|B|} \sum_{(\mathbf{x},y)\in B} \nabla_\theta \underset{\delta\in\Delta}{\text{maximize}} \; l(f_\theta(\mathbf{x}+\delta), y)$$

How can we compute ∇_θ?

- Danskin's theorem
- Gradient at the inner maximization problem is the gradient evaluated at the maximum

Min-max optimization

Madry et al. *"Towards Deep Learning Models Resistant to Adversarial Attacks"*, ICLR'18

Minibatch gradient descent

Repeat:

Select random batch $B \subseteq D$

For $(\mathbf{x}, y) \in B$ **:**

$$\delta^* = \underset{\delta \in \Delta}{\operatorname{argmax}}\ l(m_\Theta(\mathbf{x} + \delta), y)$$

$$\theta := \theta - \alpha \frac{1}{|B|} \sum_{(\mathbf{x},y) \in B} \nabla_\theta\, l(f_\theta(\mathbf{x} + \delta^*), y)$$

In practice:
Training both on normal points and adversarial examples

Adversarial Training

- Adversarial examples give lower bound for δ^*
- Current state-of-the-art but no guarantees

Certified robustness

- Exact solution through combinatorial problem solving
- Upper bound through relaxation's
- So far: not scalable

Outline

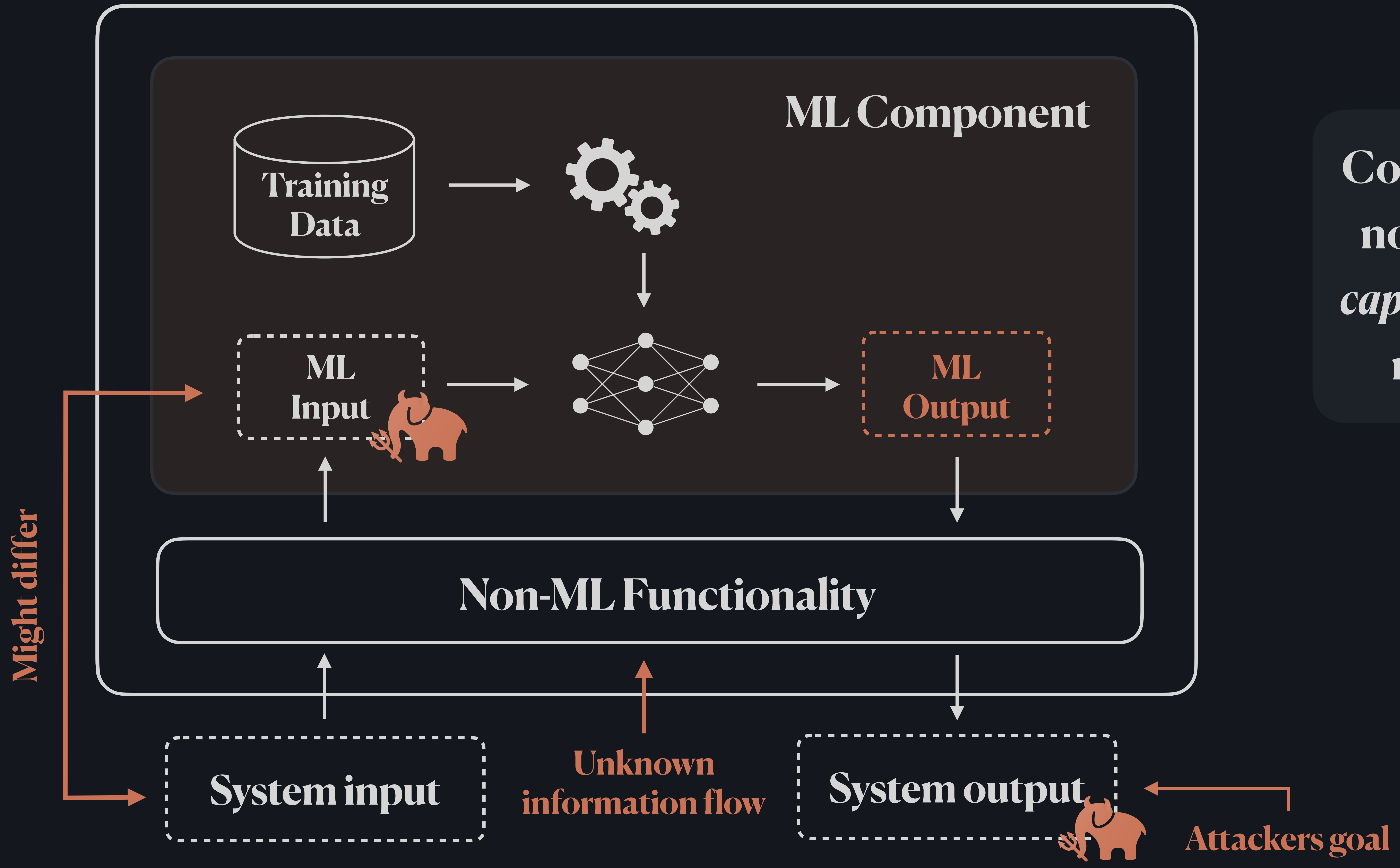
Adversarial machine learning

- Overview over different attack vectors
- Robust optimization

Security of ML systems

- Overview of attack surface
- Example: Differential Testing of Linear Algebra Systems

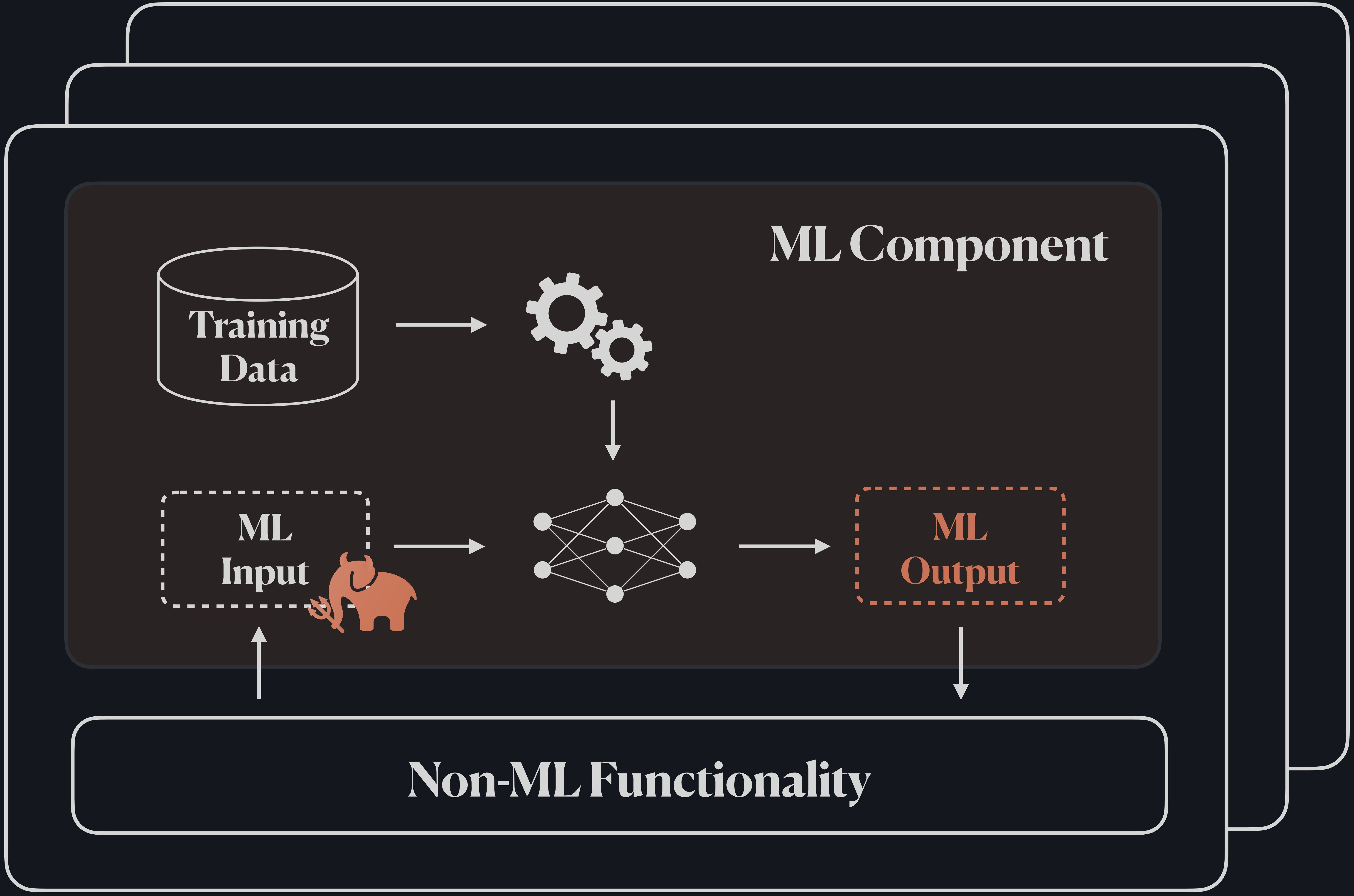
ML Systems



Common threat models do not express well the *goals*, *capabilities* and *knowledge* of real-world adversaries

Is this all?

ML Systems



Common threat models do not express well the *goals*, *capabilities* and *knowledge* of real-world adversaries

Need to also consider the system *vertically*

Outline

Adversarial machine learning

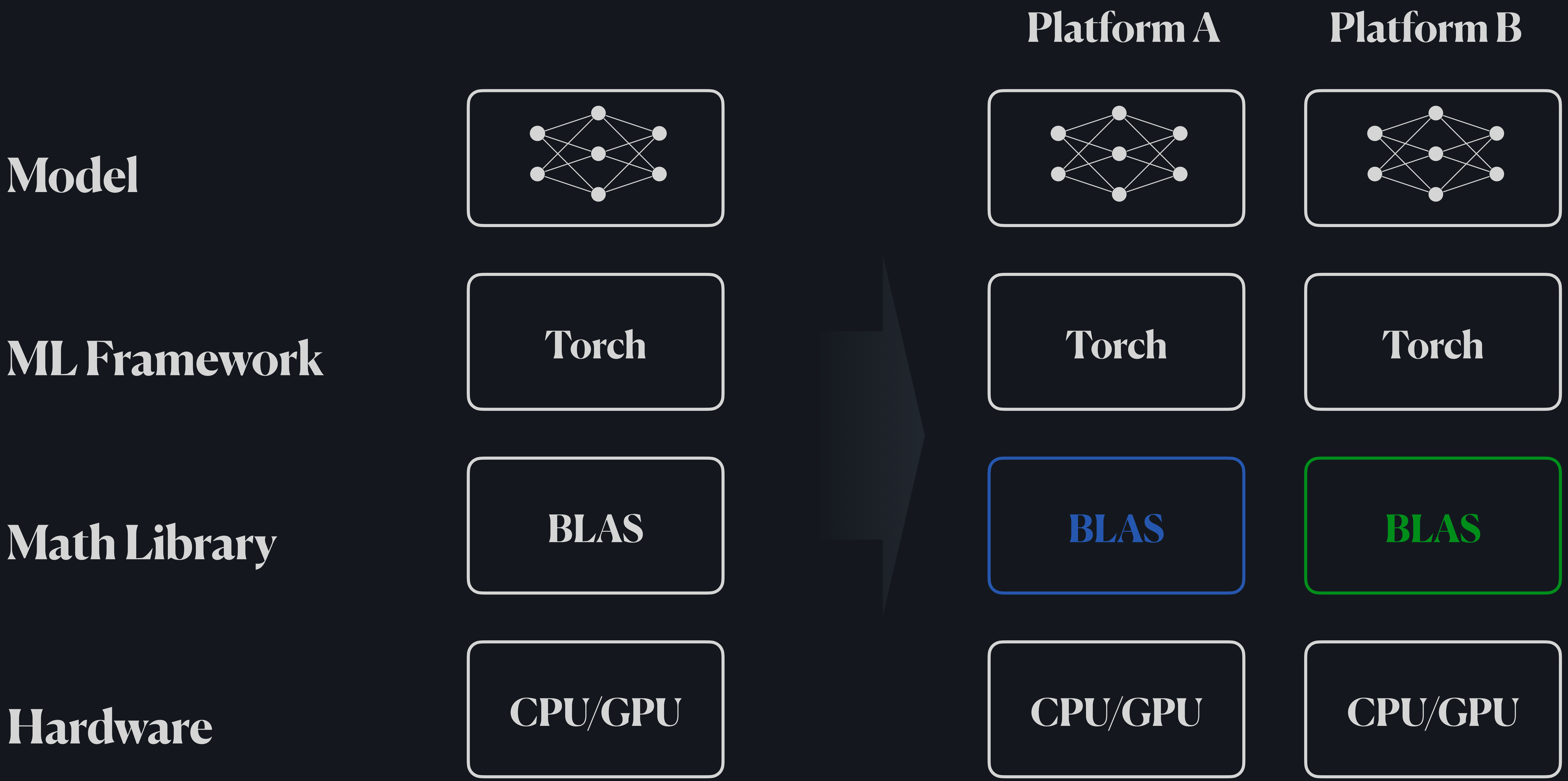
- Overview over different attack vectors
- Robust optimization

Security of ML systems

- Overview of attack surface
- Example: Differential Testing of Linear Algebra Systems

Attacking ML vertically

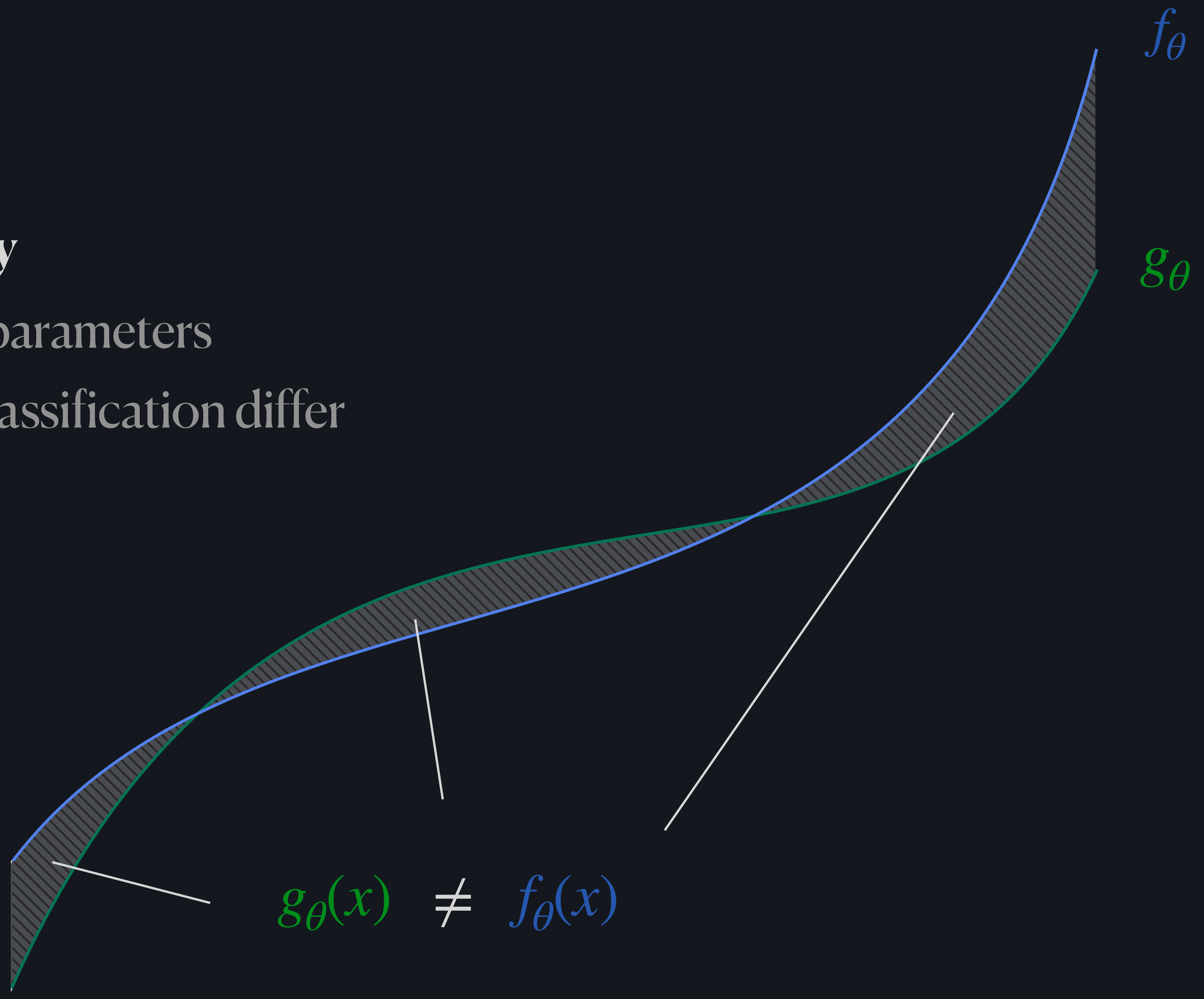
Möller et al. *“Differential Testing of Linear Algebra Systems”*, WiP



Attacking ML vertically

Decision boundaries vary

- Even with identical model parameters
- Tiny pockets exist where classification differ



Detour: Floating Points

Sign | Exponent (8 bits) | Mantissa (23 bits)

0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$$+ \, (1 + 0) * 2^{(126 \, - \, 127)} = 0.5$$

0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

$$+ \, (1 + 0.00000011920928955078125) * 2^{(126 \, - \, 127)} \approx 0.50000006$$

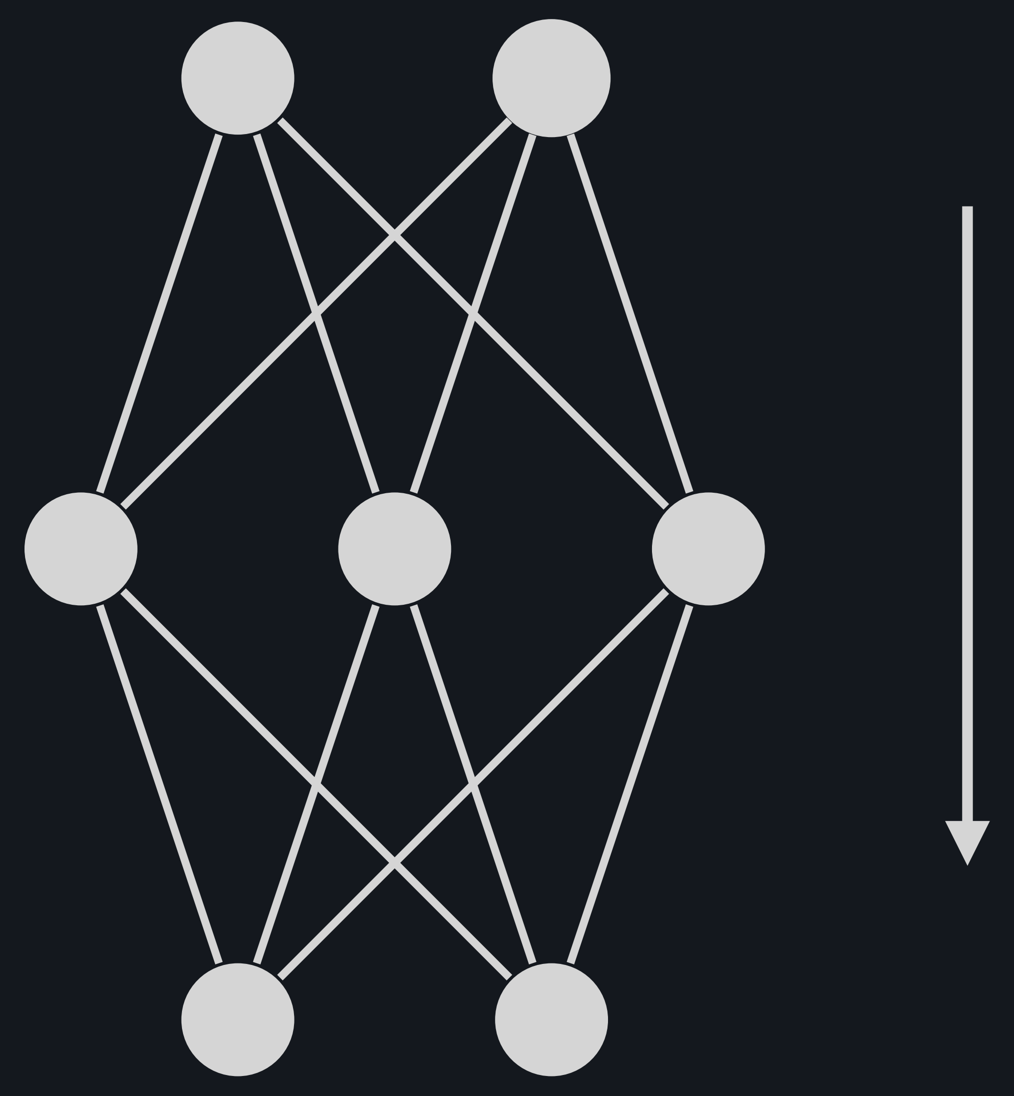
Detour: Floating Points

Gaps create rounding errors

Problem statement

Floating point arithmetics used in neural networks

- Each layer = series of matrix multiplication
- Matrix multiplication = series of floating point operations



Basic Linear Algebra Subprograms (BLAS)

- Implement (efficient) matrix multiplication
- Different implementations can be used

Output can differ between BLAS libraries!

Boundary Samples

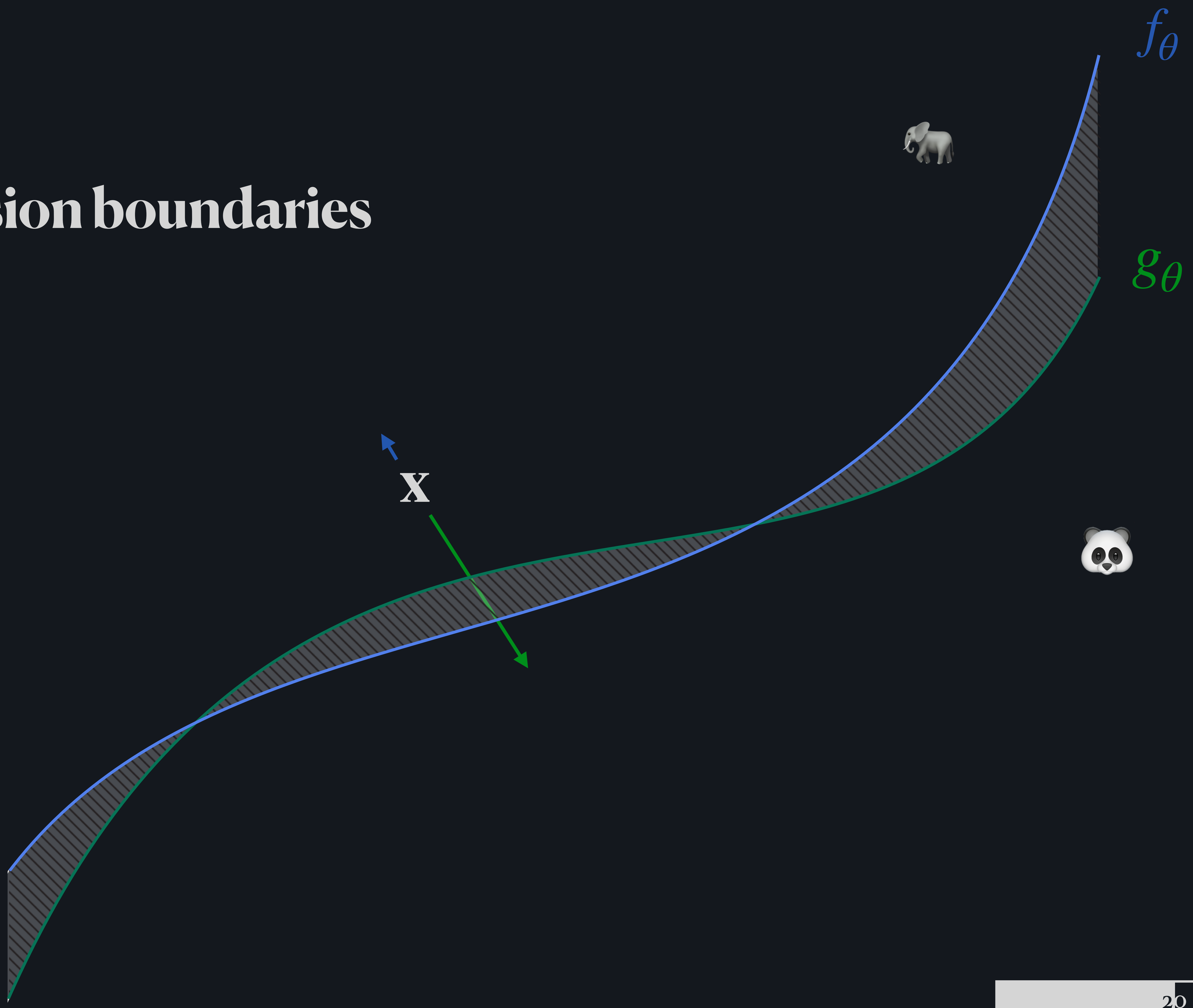
Move input sample $\mathbf{x}$ between decision boundaries

Find δ s.t. $f_\theta(\mathbf{x} + \delta) = $ 🐘

$g_\theta(\mathbf{x} + \delta) = $ 🐼

Formulate as optimization problem

$$\underset{\delta \in \Delta}{\text{minimize}} \quad l(f_\theta(\mathbf{x} + \delta), 🐘) + l(g_\theta(\mathbf{x} + \delta), 🐼)$$

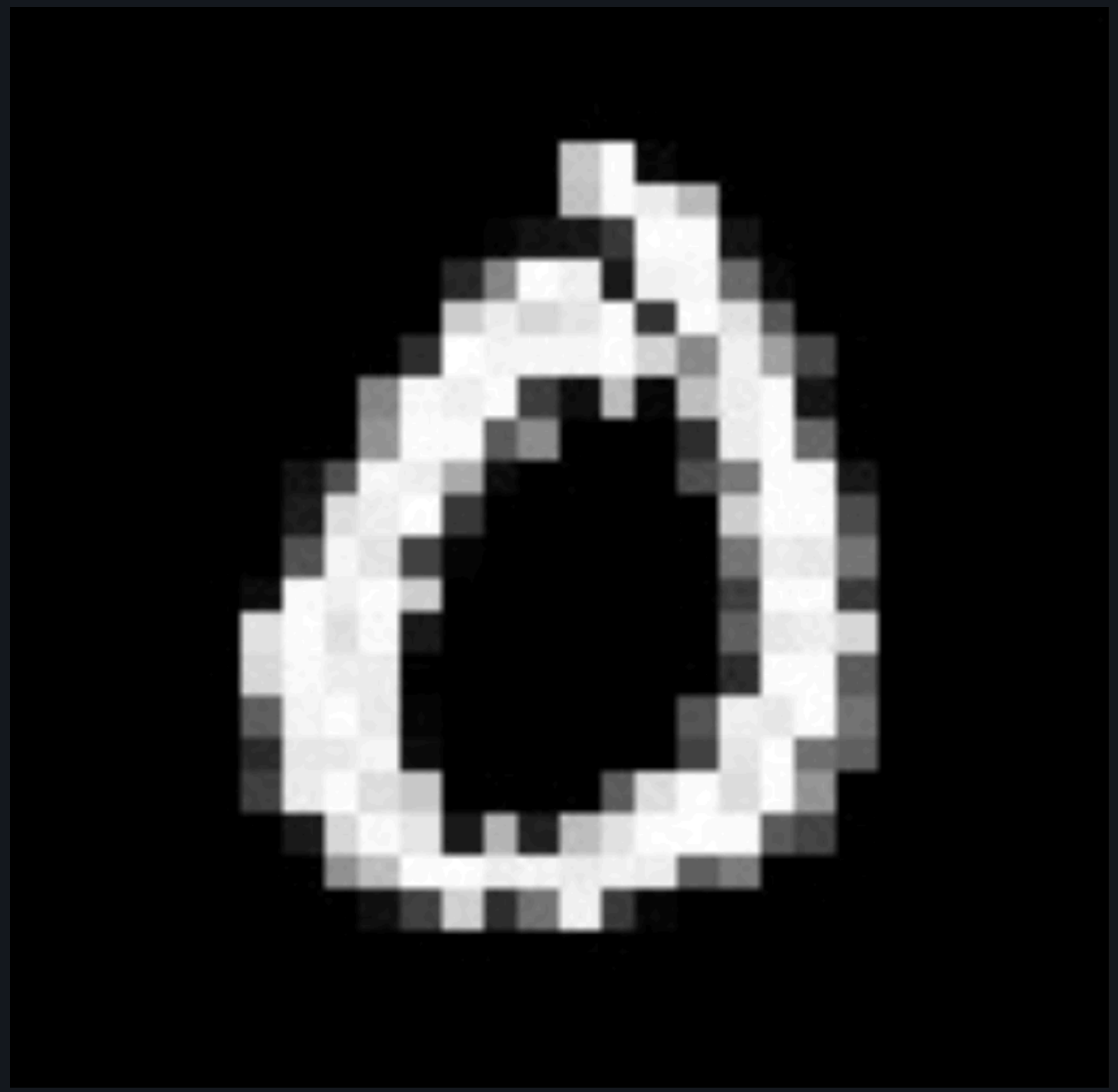


Boundary Samples

Chimera Samples

Generalize optimization for multiple backends

- Each backend targets a different class
- One sample → multiple classifications



"Default" → 3

Openblas → 2

Intel MKL → 0

Flexiblas → 1

Take Aways & Future Work

Adversarial machine learning

- Need to consider the deployment of a model
- Increased attack surface

Countermeasures beyond the model

- Defenses are hard in the general setting
- Use domain expertise to improve model robustness
- Safeguard the model

Thank you!

Fin