Yale

Verification and Robust Reinforcement Learning

CPSC680: Trustworthy Deep Learning

Rex Ying

Readings

- Readings are updated on the website (syllabus page)
- **Readings:**
  - [Adversarial Robustness of Deep Neural Networks: A Survey from a Formal Verification Perspective]

- Credit for helping with this lecture: [Huan Zhang] (faculty at UIUC)

Content

- Introduction to Formal Verification in Machine Learning
- Verification for Deep Neural Networks
- Adversarial Robustness for Reinforcement Learning

Content

- **Introduction to Formal Verification in Machine Learning**
- Verification for Deep Neural Networks
- Adversarial Robustness for Reinforcement Learning

How to Build Trustworthy Mission-Critical Systems?

Prove trustworthiness

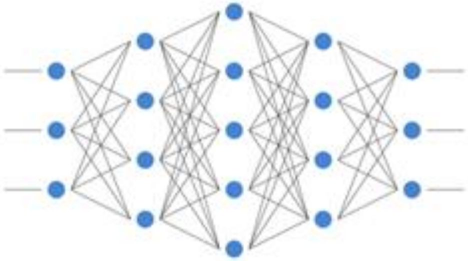
Verifying correctness using formal **verification**

Disprove trustworthiness

Testing with edge cases to find bugs

Translating to Trustworthy Deep Learning

Enabling trustworthy AI via **formal verification** and **adversarial testing**



Prove trustworthiness

How to formally prove the trustworthiness of AI?

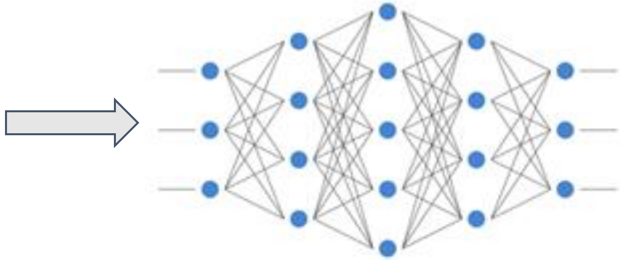
Disprove trustworthiness

We have talked about this

How to find edge cases and bugs of AI?

Approach: efficient and **specialized** verification and testing methods for AI

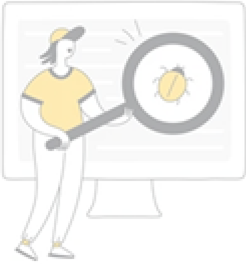
Formal Verification of AI



Prove trustworthiness

How to formally prove the trustworthiness of AI?

Disprove trustworthiness



How to find edge cases and bugs of AI?

Goal of Formal Verification of AI: an Example

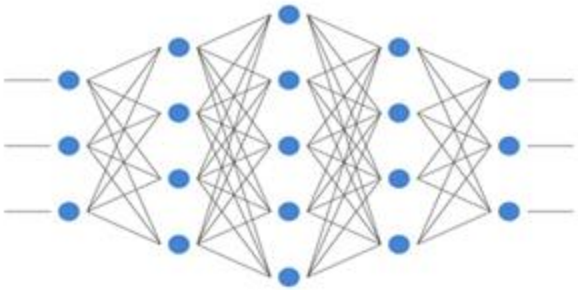
Eykholt et al. 2018

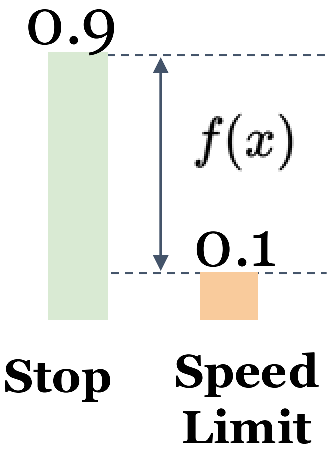
STOP 😃

Speed ~~Limit~~ 😈

Goal: prove that adversarial examples do *not* exist!

A Simplified Example of the Verification Problem





Score for each label

$f(x)$: Difference in predicted probability

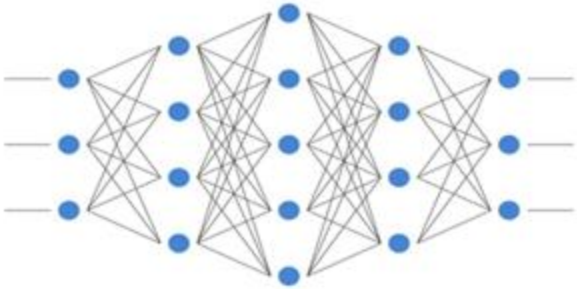
Attacker may put *anything* here

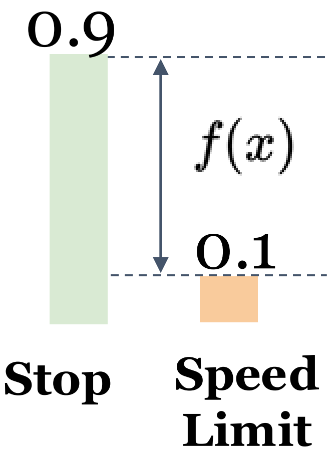
What's this perturbation constraint?

$$f(x) > 0 \Rightarrow \text{ No adversarial examples}$$

The Canonical Form of Verification Problem

Attacker may put *anything* here





Score for each label

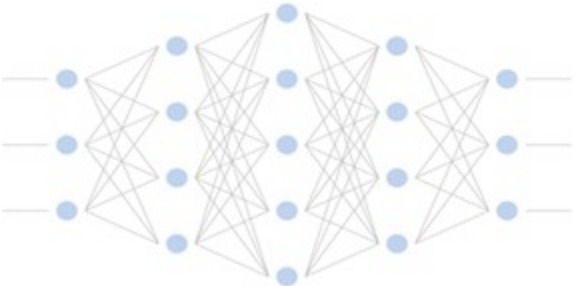
$\mathcal{S}$ = all possible pixel perturbations

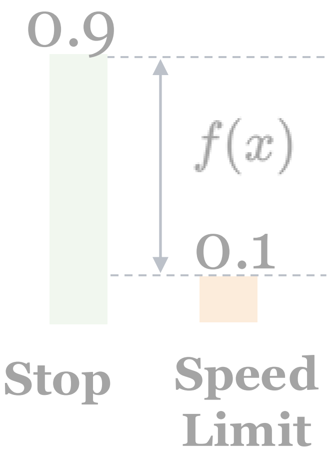
$x_1 \in \mathcal{S}$ $x_2 \in \mathcal{S}$ $x_3 \in \mathcal{S}$

Prove: $\forall x \in \mathcal{S},\ f(x) > 0$

The Canonical Form of Verification Problem

Attacker may put *anything* here





Score for each label

$\mathcal{S}$ = all possible pixel perturbations

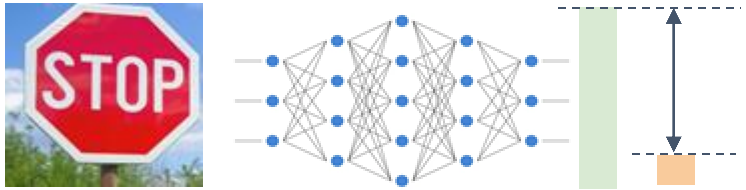
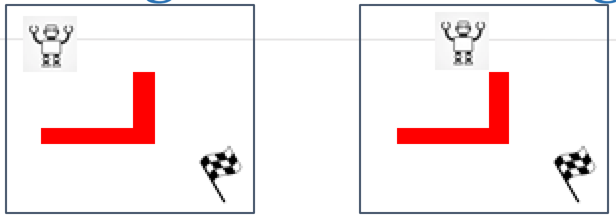
$x_1 \in \mathcal{S}$ $x_2 \in \mathcal{S}$ $x_3 \in \mathcal{S}$ …

That was just an example of a verification problem

Prove: $\forall x \in \mathcal{S},\ f(x) > 0$

The Canonical Form of Verification is General

Prove: $\forall x \in \mathcal{S},\ f(x) > 0$

Application	Specification $f(x)$	Sets of inputs $\mathcal{S}$
	Attack cannot succeed	 Perturbed pixel values
		 Drone starting at different angles
	Robot does not reach a obstacle (reachability)	 Different starting points

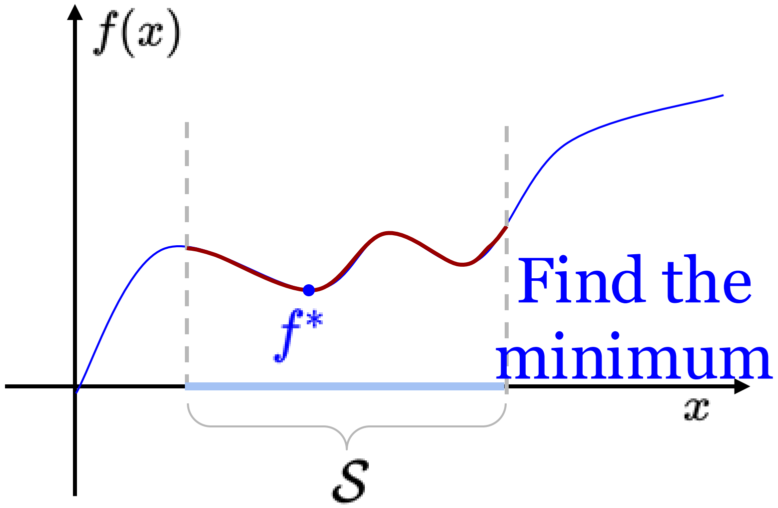
What is the general approach to prove this?

Content

- Introduction to Formal Verification in Machine Learning
- **Verification for Deep Neural Networks**
- Adversarial Robustness for Reinforcement Learning

How Hard is the Problem?

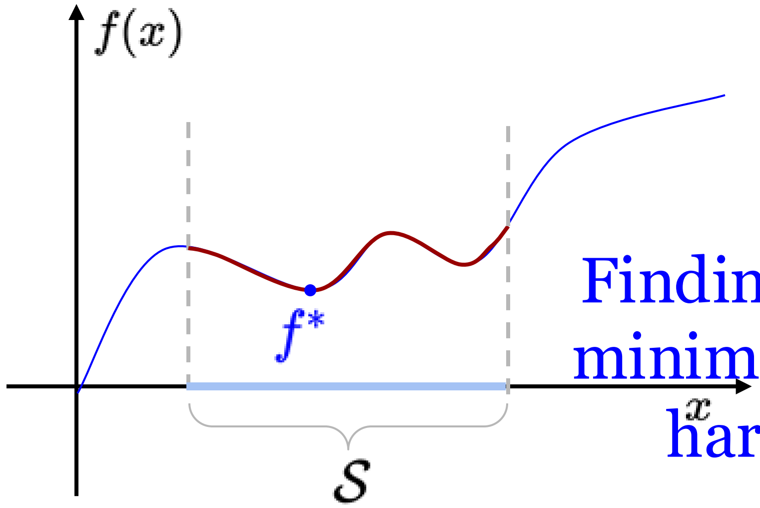
Prove: $\forall x \in \mathcal{S},\ f(x) > 0$



NP-hard for general neural networks (Katz et al., 2017)
(non-convex optimization)

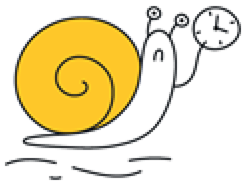
Traditional Approach: Using Optimization Solvers

Prove: $\forall x \in \mathcal{S},\ f(x) > 0$

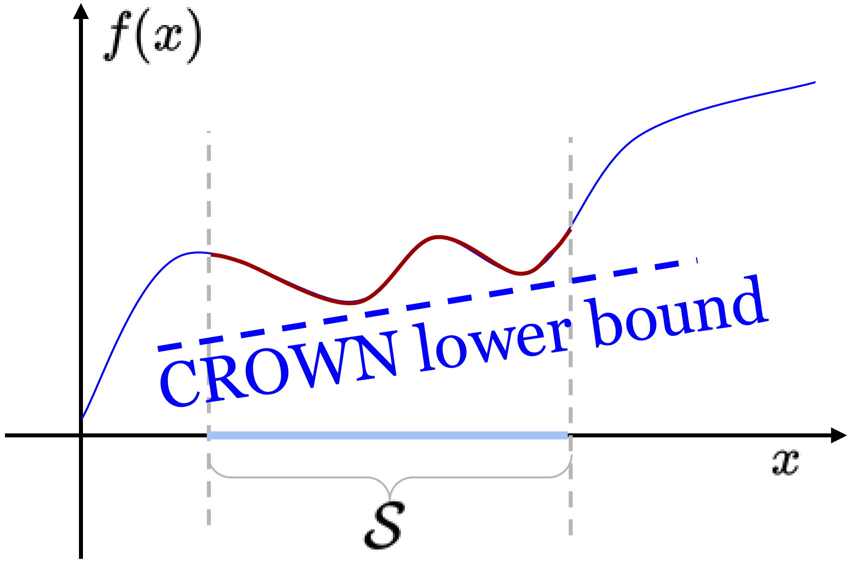


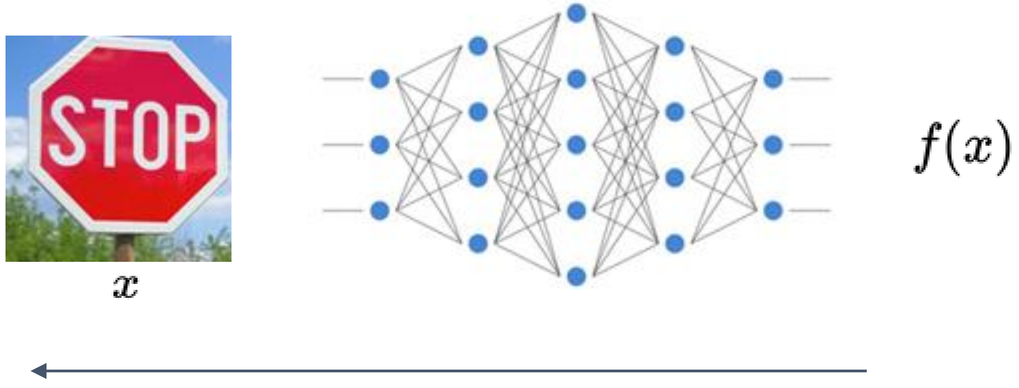
Finding the minimum is hard!

Linear programming (LP), semidefinite programming (SDP), mixed integer programming (MIP), or Satisfiability modulo theories (SMT)



CROWN: a **Linear Bound Propagation** Algorithm

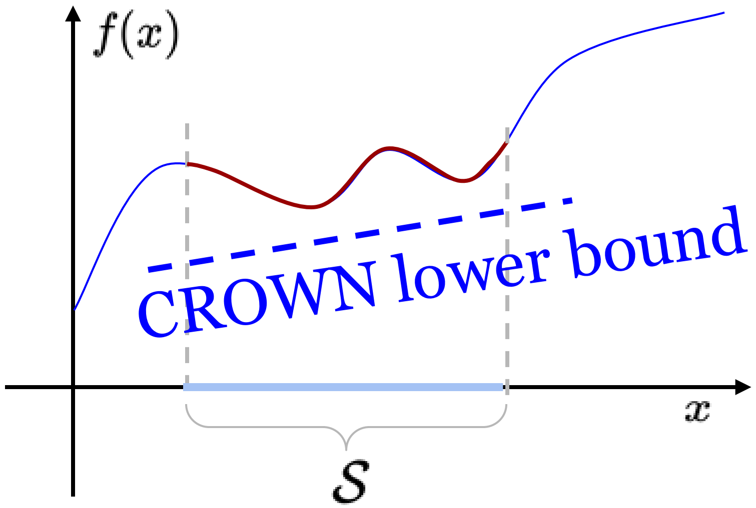




Propagate linear inequalities w.r.t. $f(x)$

- Find a provable linear lower bound for neural networks
- Bound by efficiently propagating linear inequalities (GPU accelerated)

Prove the verification problem with CROWN

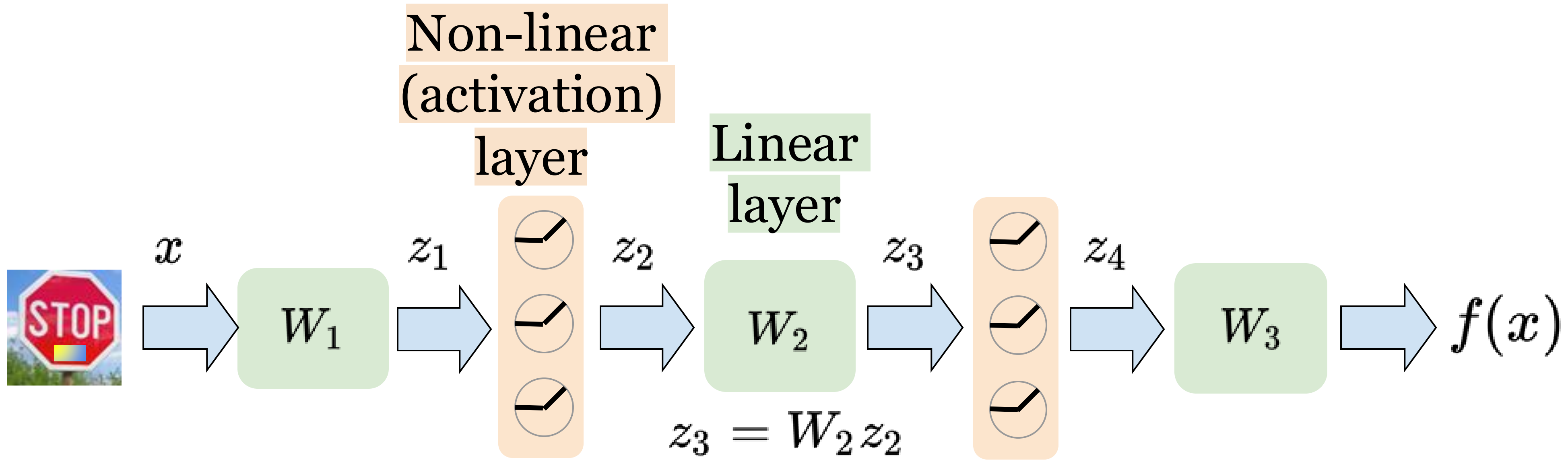


Prove: $\forall x \in \mathcal{S},\ f(x) > 0$

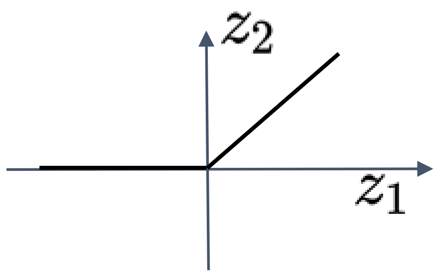
$x_1 \in \mathcal{S}$ $x_2 \in \mathcal{S}$ $x_3 \in \mathcal{S}$ ...

Lower bound > 0 ⇨ $f(x) > 0$ ⇨ verified (always a stop sign)

How to Propagate the Linear Bounds?



$z_2 = \mathrm{ReLU}(z_1)$

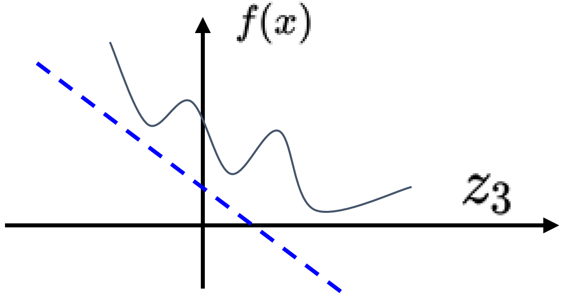


Steps:

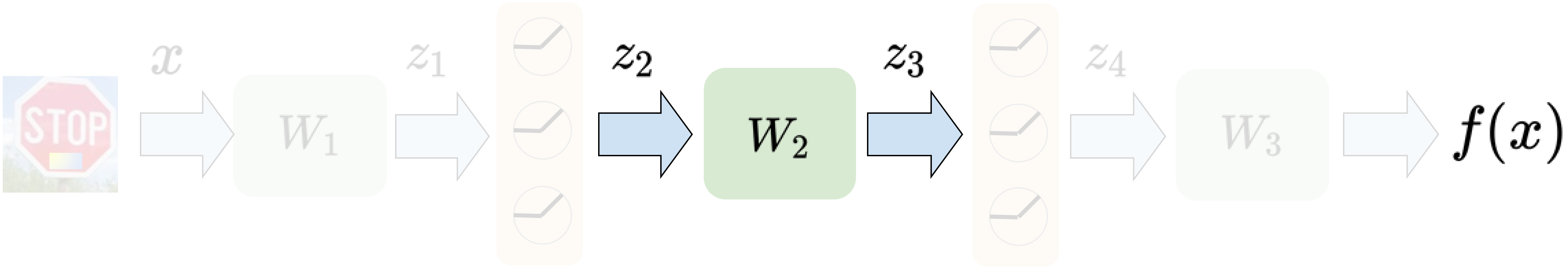
- Propagate bounds through linear layers
- Propagate bounds through non-linear layers

What Linear Inequalities to Propagate?

A linear lower bound for an intermediate layer

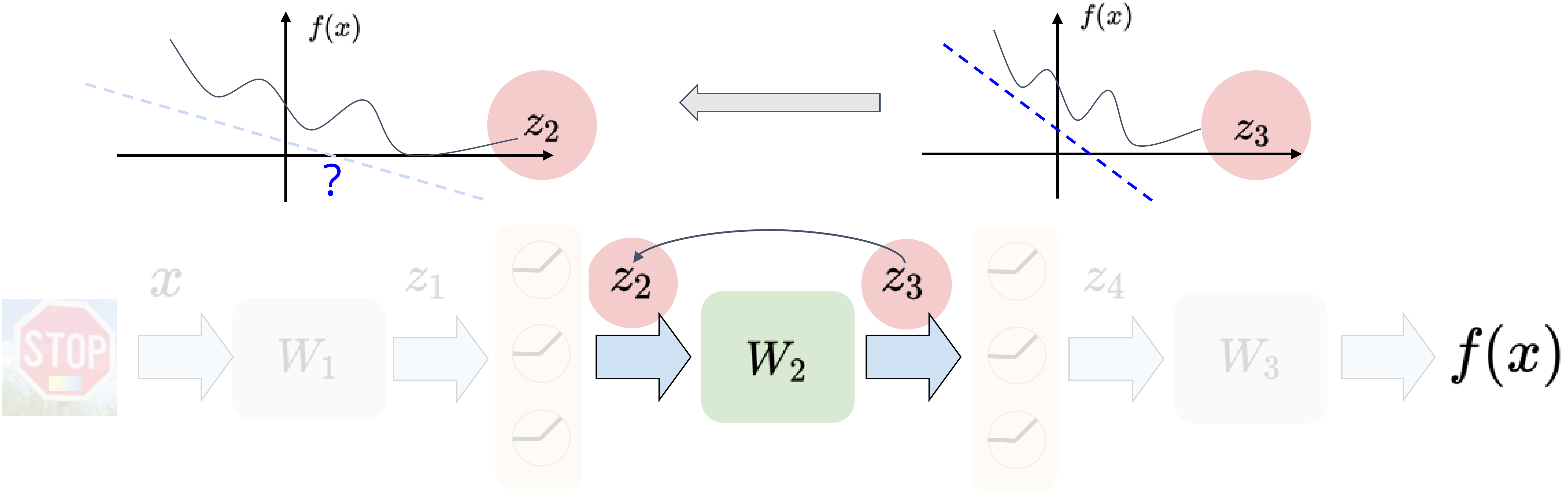


1-D example. Generally it's a linear hyperplane



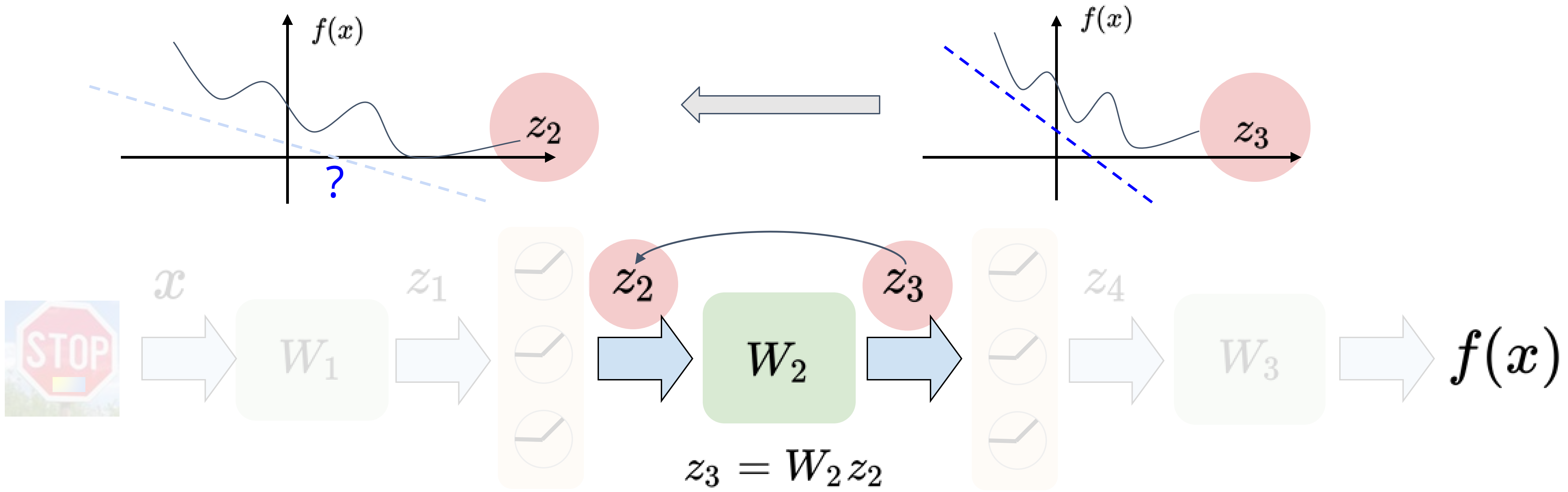
$$f(x) \geq a^\top z_3 + b$$

Propagating Bounds through Linear Layers



Propagate it to one layer before,
while keeping the lower bound valid

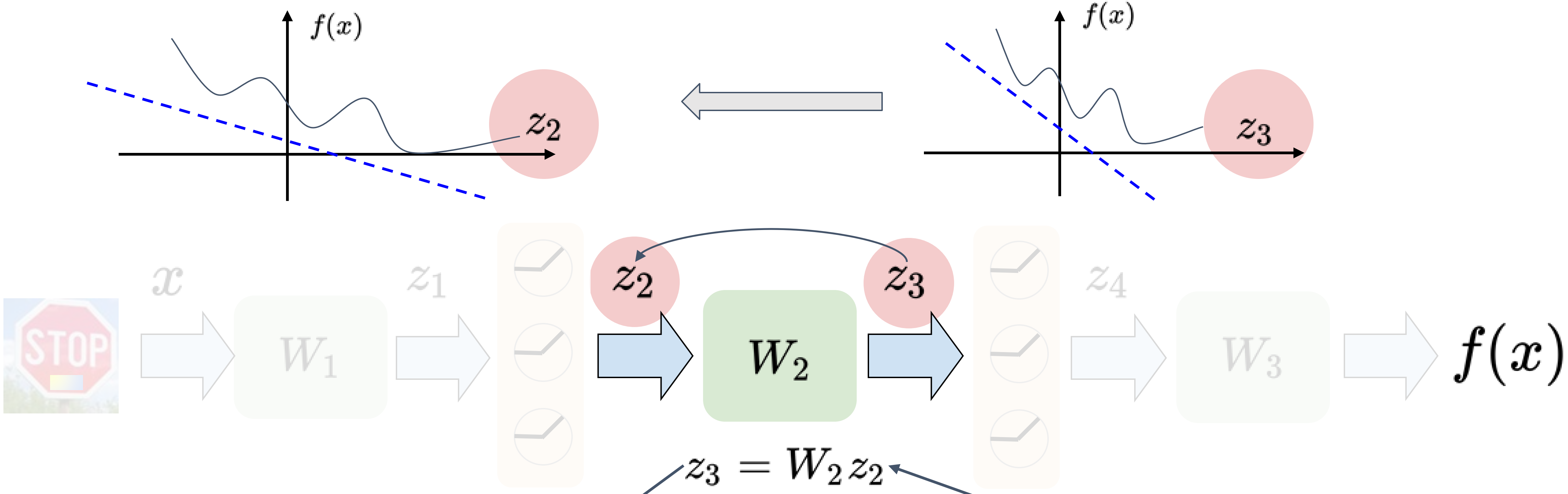
Propagating Bounds through Linear Layers



$$z_3 = W_2 z_2$$

$$f(x) \geq a^\top z_3 + b$$

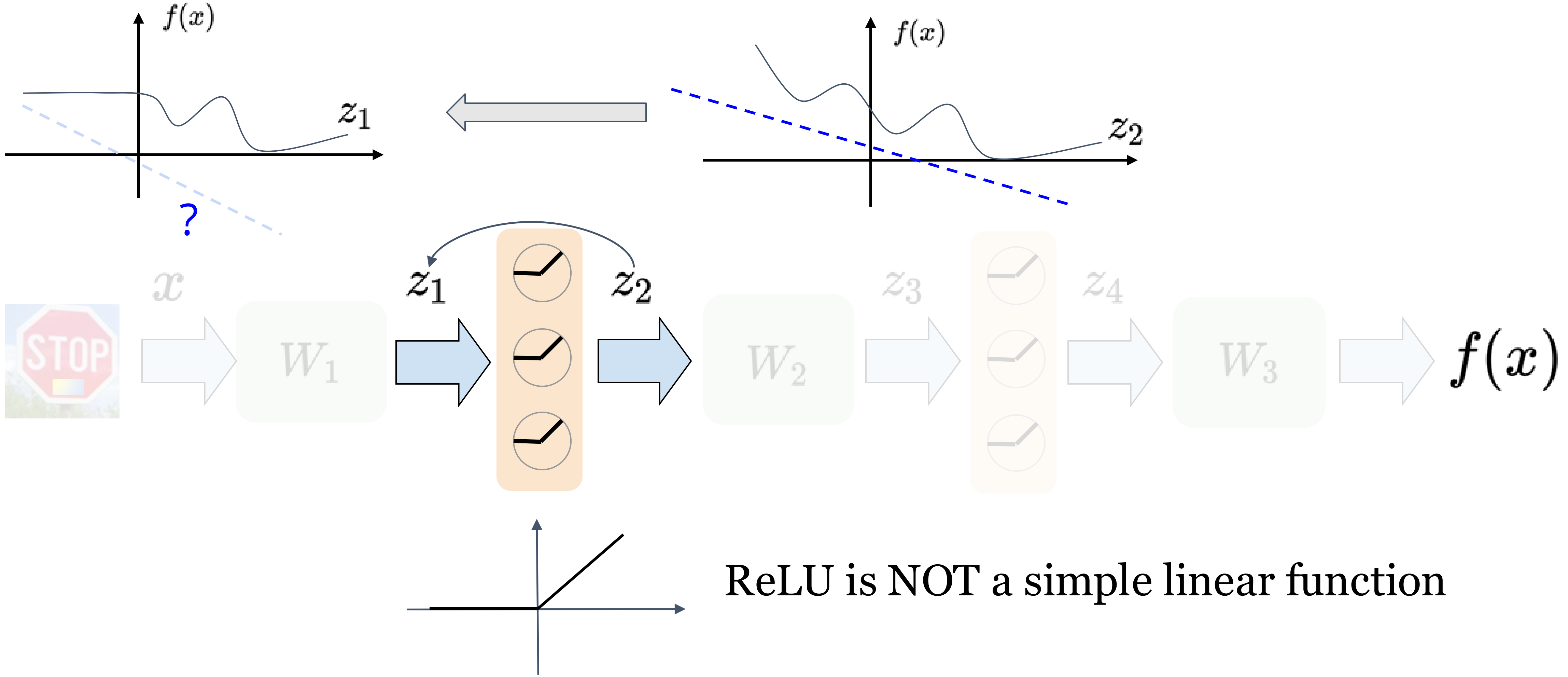
Propagating Bounds through Linear Layers



$$f(x) \geq a^\top W_2 z_2 + b \quad \longleftarrow \quad f(x) \geq a^\top z_3 + b$$

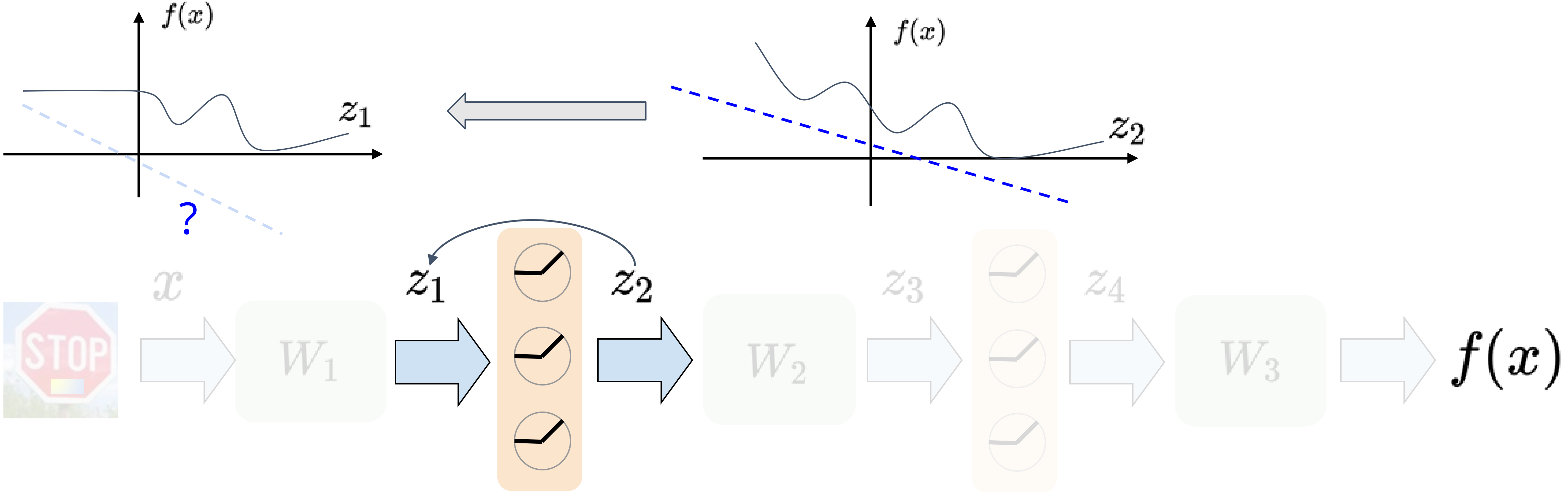
Inequality updated after propagation

Propagating Bounds through non-Linear Layers



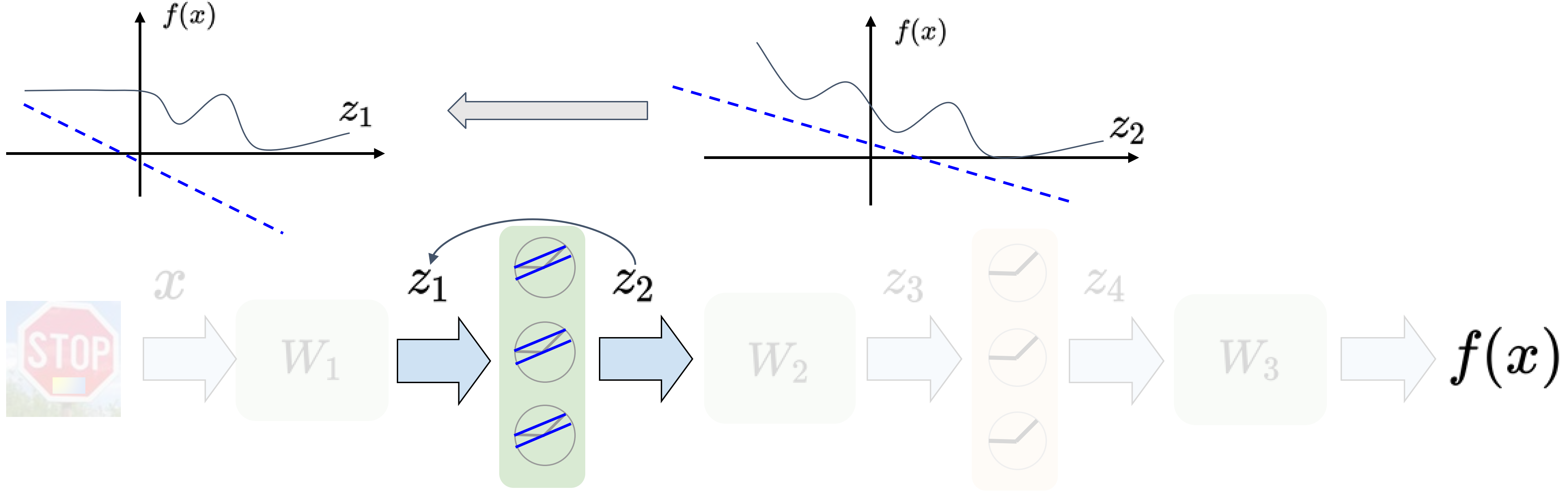
ReLU is NOT a simple linear function

Propagating Bounds through non-Linear Layers



$$f(x) \geq a^\top W_2 z_2 + b \quad \forall x \in \mathcal{S}$$

Propagating Bounds through non-Linear Layers



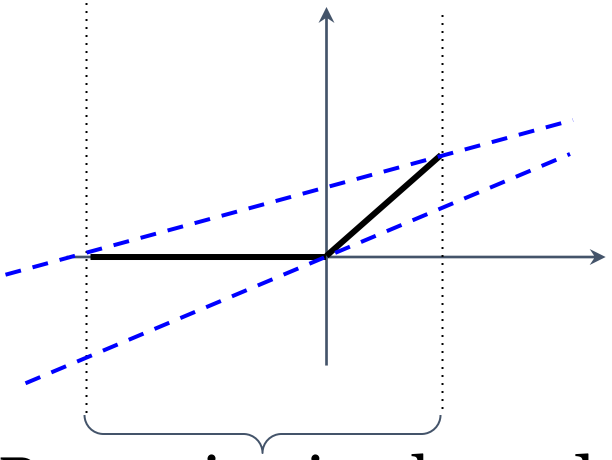
Theorem (informal): we can efficiently find $D,\ b'$ such that:

$$f(x) \geq a^\top W_2 D z_1 + b' \Longleftarrow f(x) \geq a^\top W_2 z_2 + b \quad \forall x \in \mathcal{S}$$

[Efficient Neural Network Robustness Certification with General Activation Functions (NeurIPS 2018)]

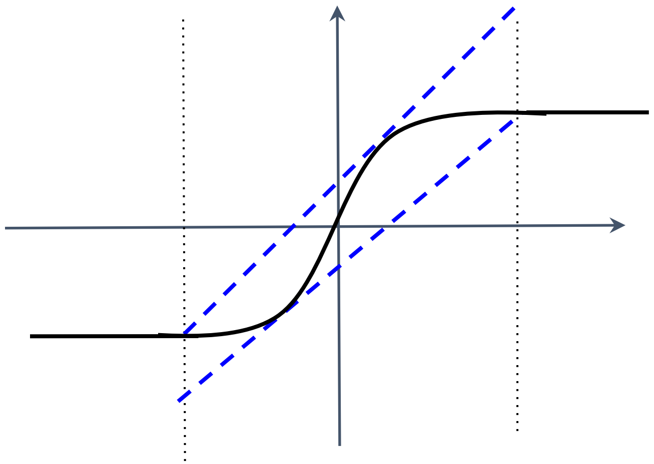
Proof for non-Linear Propagation

Proof sketch: conservatively use linear bounds to replace a non-linear function.



Pre-activation bounds

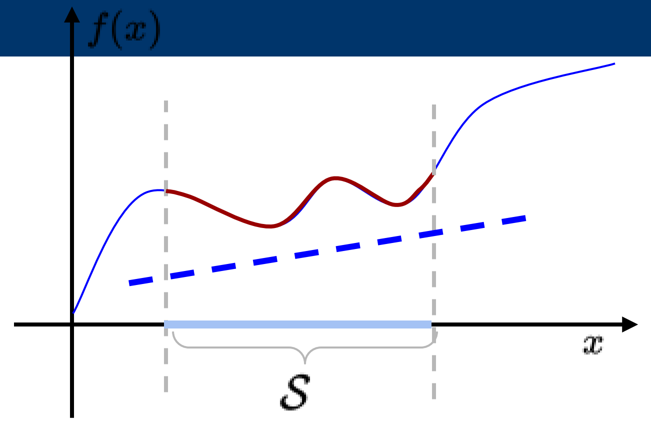
(can be pre-computed using CROWN)

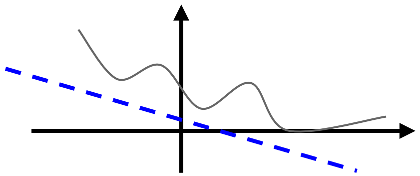


Theorem (informal): we can efficiently find $D,\ b'$ such that:

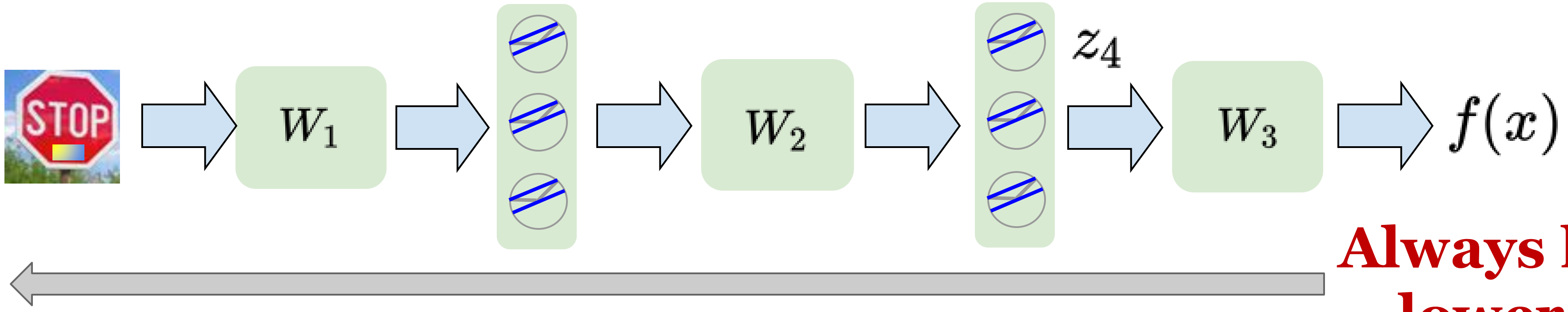
$$f(x) \geq a^\top W_2 D z_1 + b' \Longleftarrow f(x) \geq a^\top W_2 z_2 + b \quad \forall x \in \mathcal{S}$$

CROWN Algorithm





Bounds propagated through simple matrix multiplations! Fast and GPU-friendly



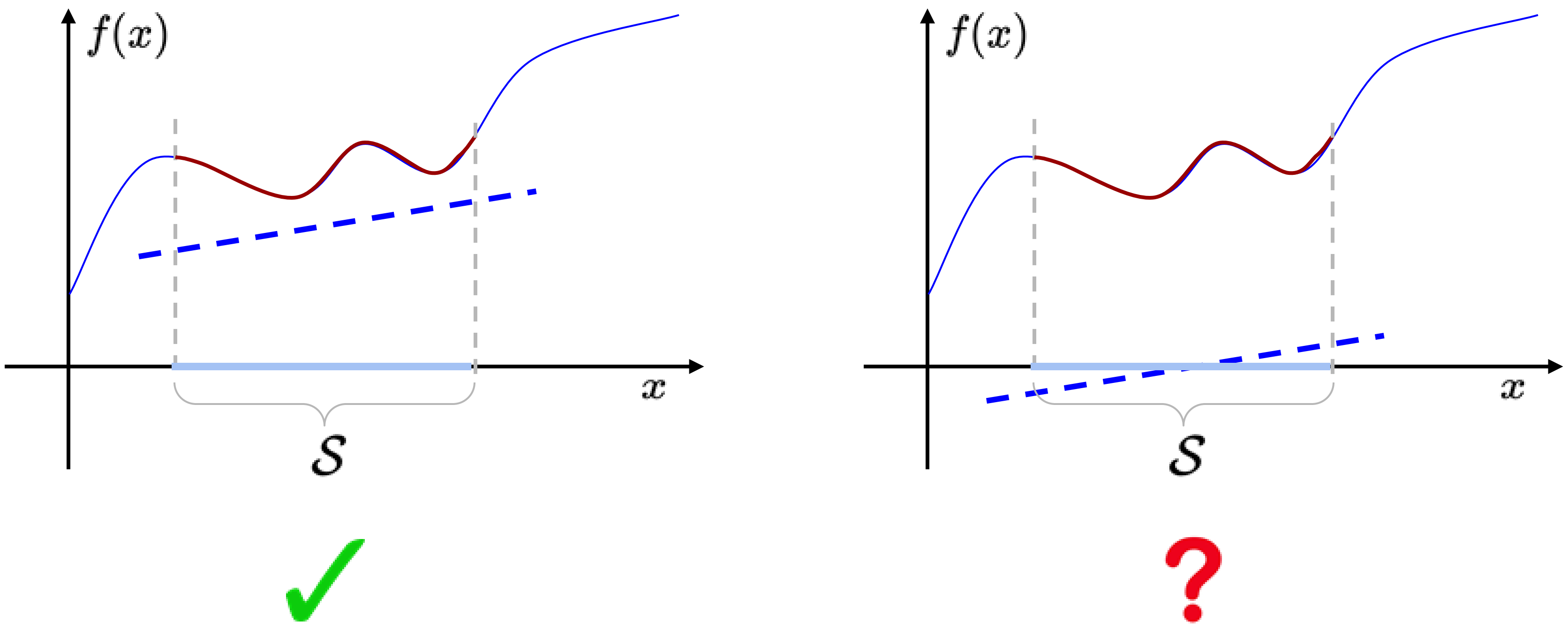
Always keep valid lower bounds

a_{CROWN} has the form $W_3 D_2 W_2 D_1 W_1$

CROWN main theorem (simplified): $f(x) \geq a_{\text{CROWN}}^{\top} x + b_{\text{CROWN}} \quad \forall x \in \mathcal{S}$

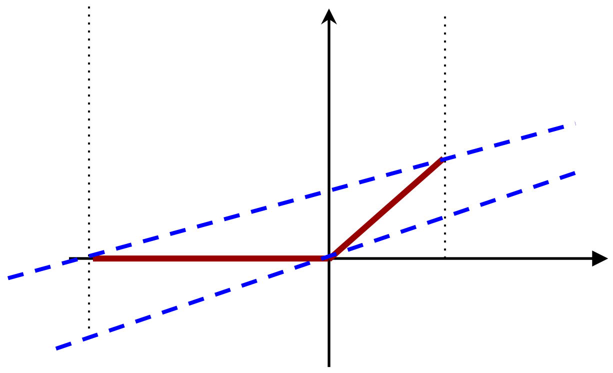
What if the Bounds are too Loose?

Prove: $\forall x \in \mathcal{S},\ f(x) > 0$

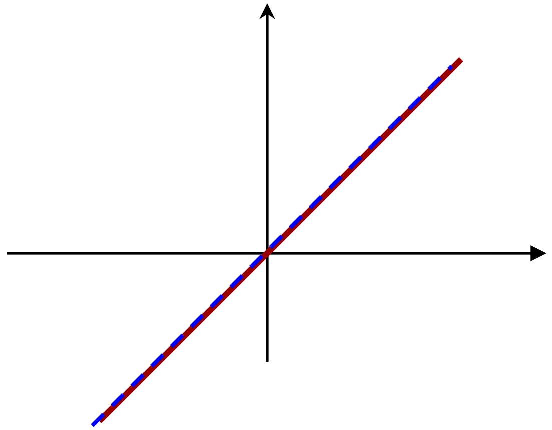


Tighten Bounds with Branch and Bound

- One approach is to reduce the number of ReLU (non-linear) functions!



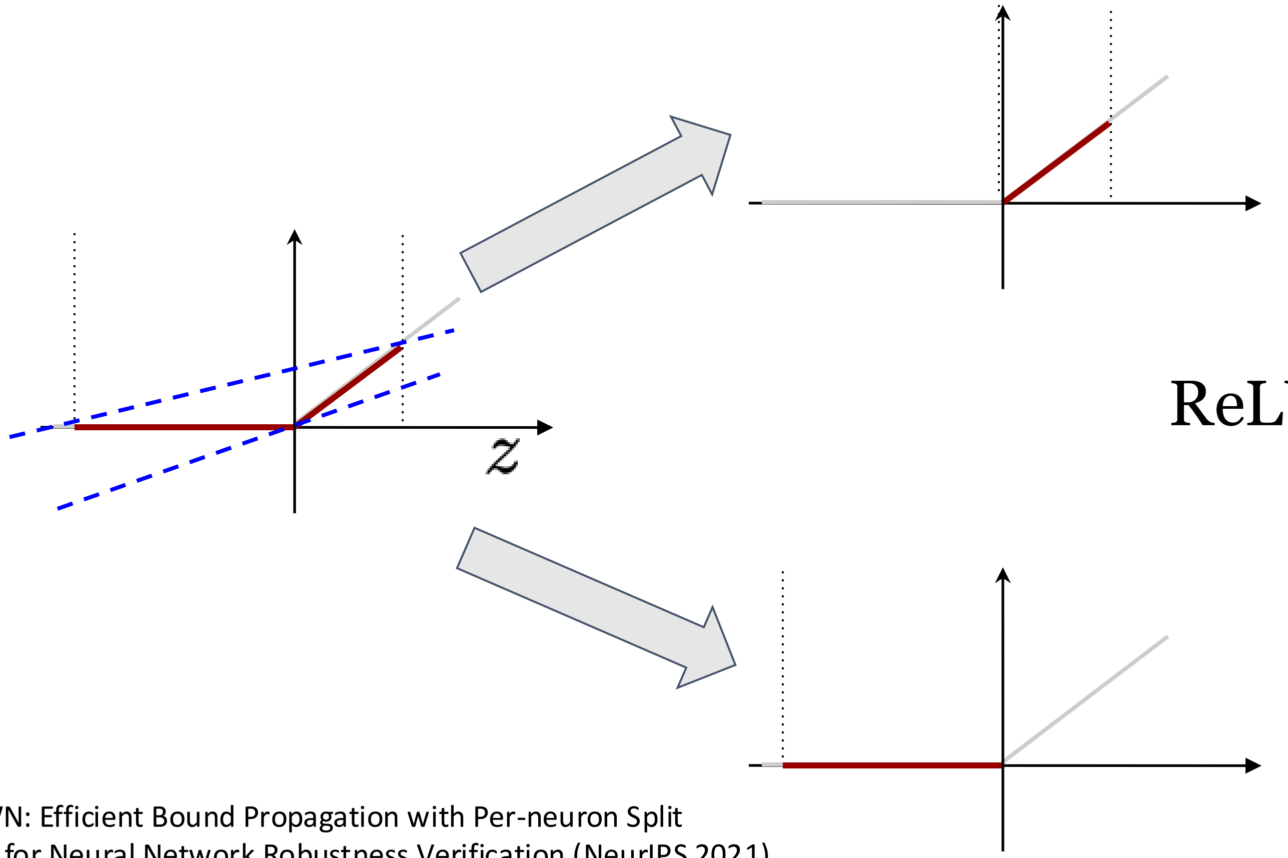
ReLU function:
Bounded by conservative
linear bounds



Linear function:
Bounds are tight

Tighten Bounds with Branch and Bound

- Choose a ReLU neuron, and split it into two subproblems:



$z \geq 0$

ReLU becomes linear in both subproblems

$z \leq 0$

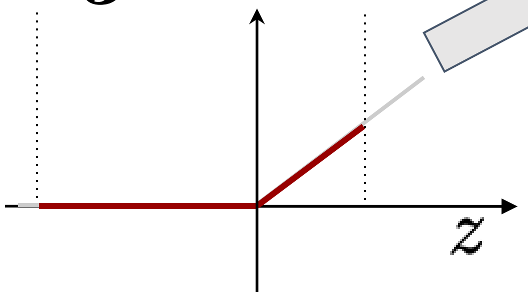
Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification (NeurIPS 2021)

Tighten Bounds with Branch and Bound

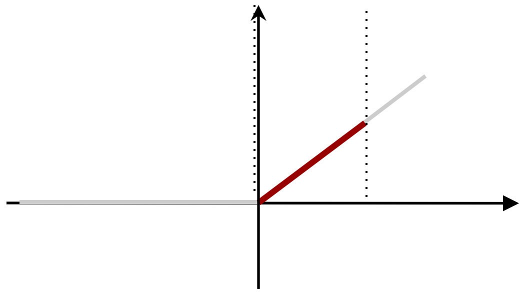
Verification problem:

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

lower bound = **-3.0**



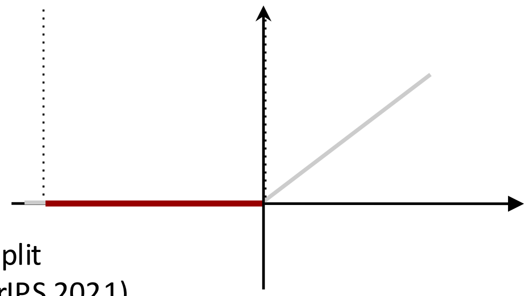
$\forall x \in \mathcal{S}, z \geq 0, f(x) > 0\,?$



lower bound = **0.5**

Subproblem Verified ✓

$\forall x \in \mathcal{S}, z \leq 0, f(x) > 0\,?$

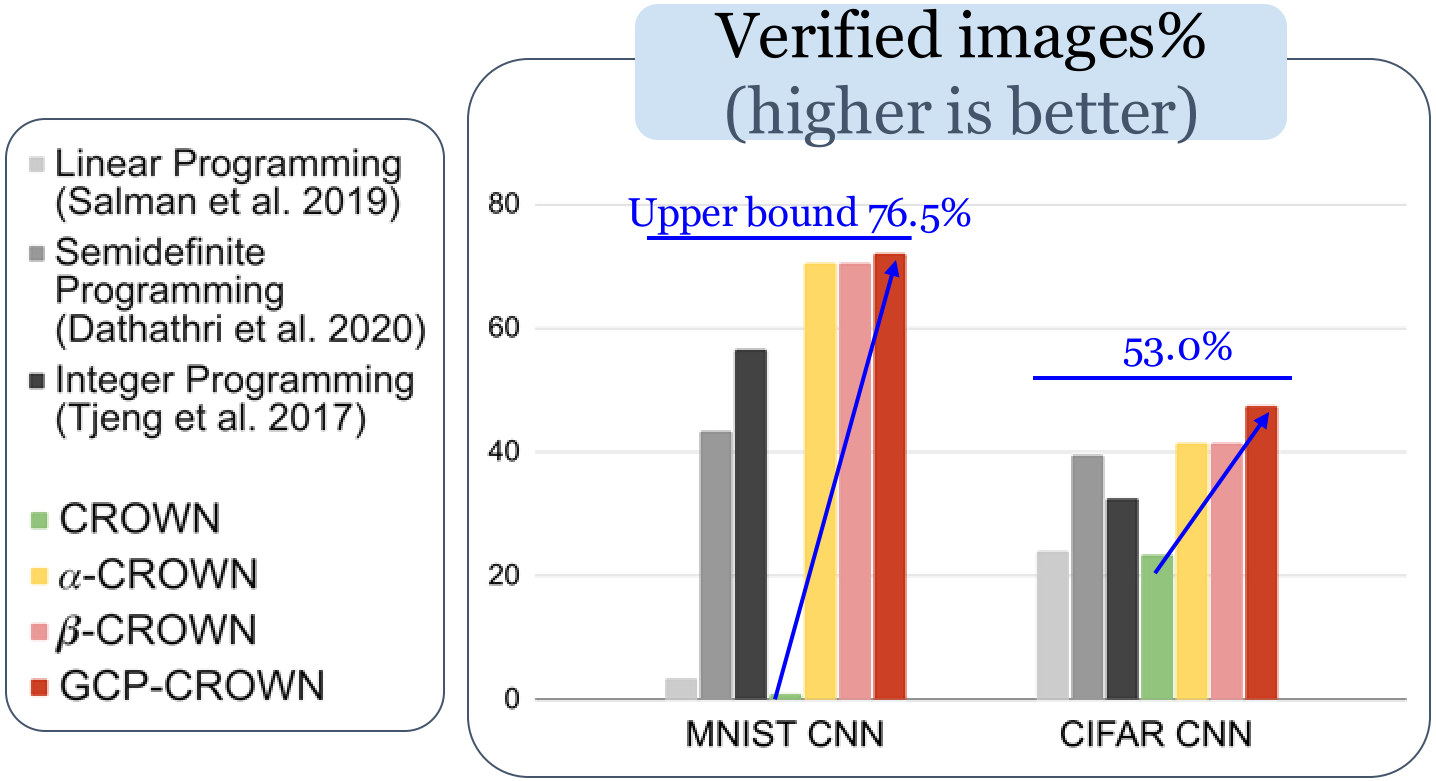


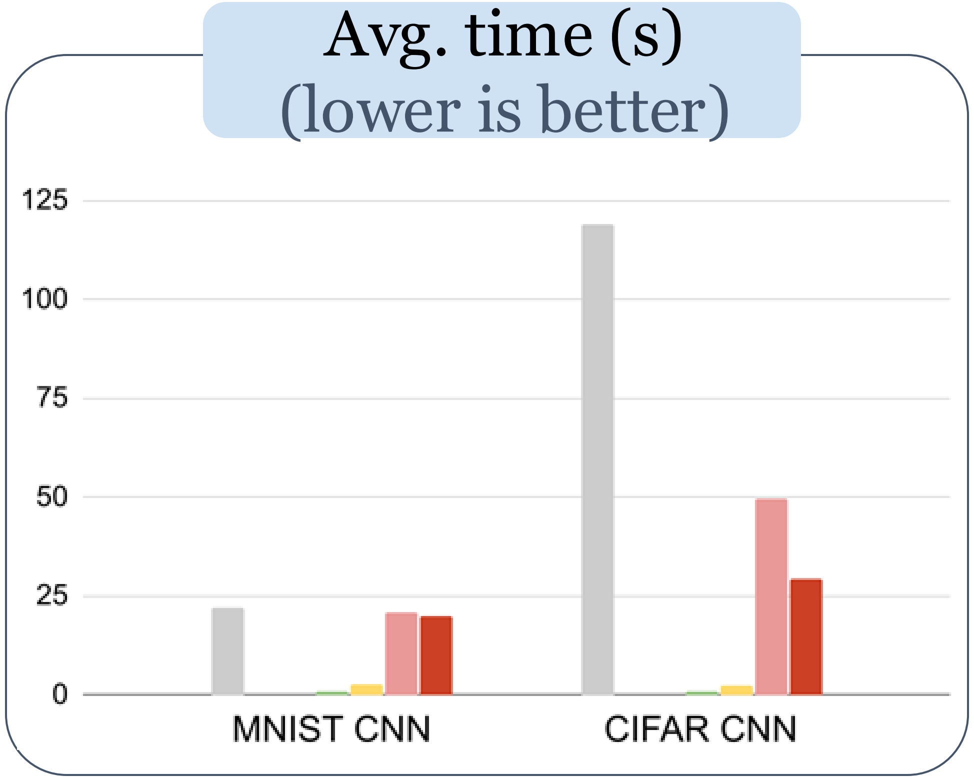
lower bound = **-2.0**

Choose another ReLU neuron to split

Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification (NeurIPS 2021)

Benchmarks: CROWN Verification





Model size: ~5k neurons

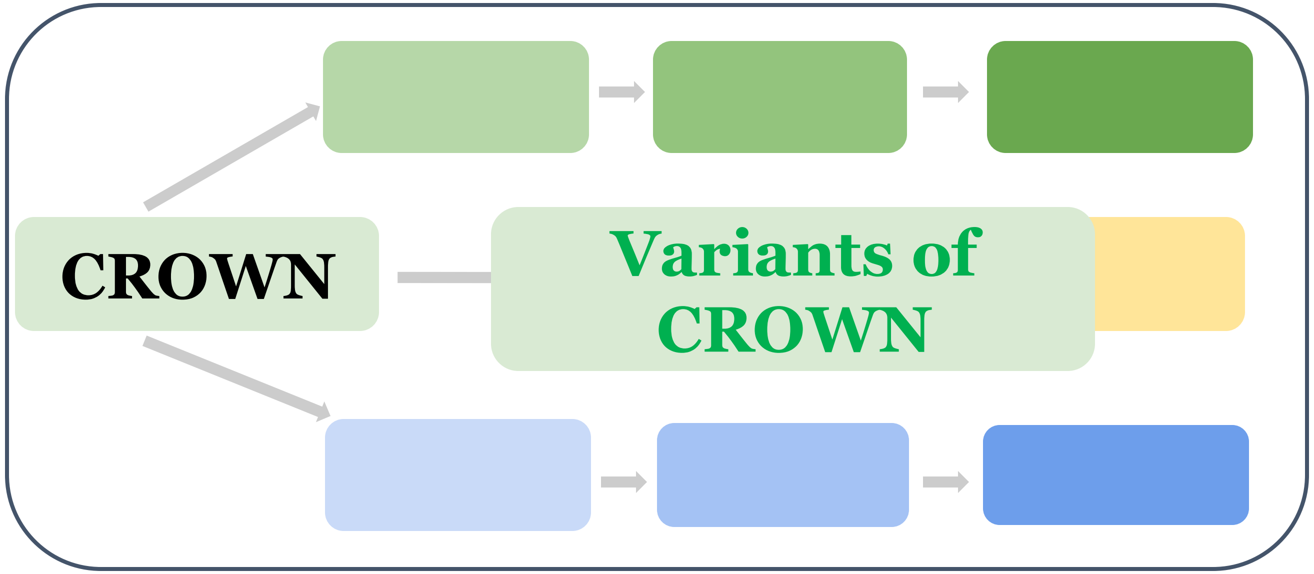
Integer programming and semidefinite programming **not plotted** (**~1 day**)

1000x faster (1-day ⇒ 1-min), enabling practical verification!

Practical Tool: α,β-CROWN

Theoretical framework

Practical verifier: **$\boldsymbol{\alpha,\beta}$-CROWN**



GCP-CROWN: cutting plane method (NeurIPS 2022a*)

Jacobian/Gradients (AAAI 2019a*, NeurIPS 2022b)

Faster verifiable training (NeurIPS 2021b)

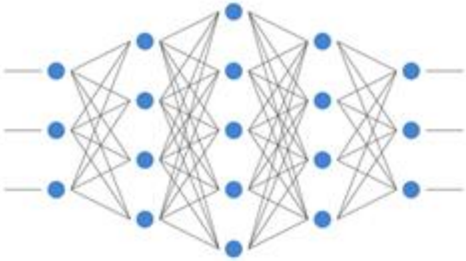
Winner of VNN-COMP 2021, 2022, 2023

https://abcrown.org

Content

- Introduction to Formal Verification in Machine Learning
- Verification for Deep Neural Networks
- **Adversarial Robustness for Reinforcement Learning**

Adversarial Testing

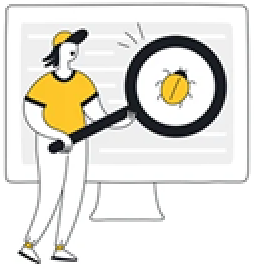


Prove trustworthiness

CROWN

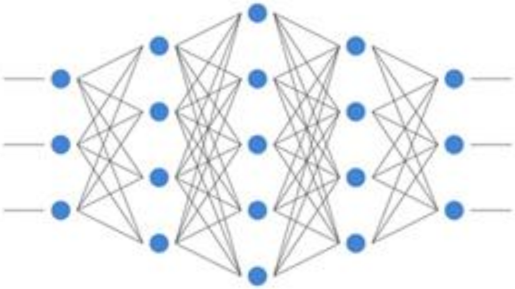
How to formally prove the trustworthiness of AI?

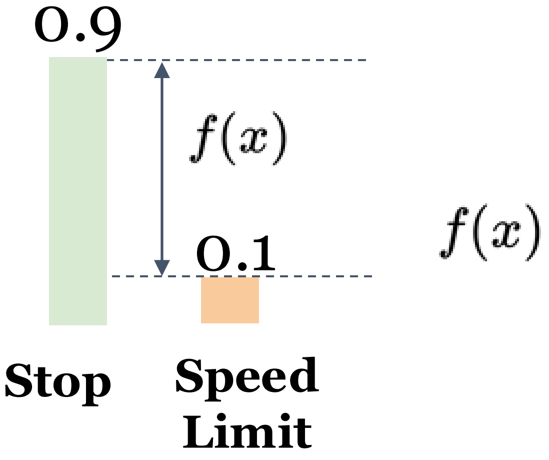
Disprove trustworthiness



How to find edge cases and bugs of AI?

Disprove by Finding Examples (1)





Set of inputs $\mathcal{S}$

Verification: **prove** $\forall x \in \mathcal{S},\ f(x) > 0$

Adversarial attack: **disprove** by showing

$$\exists x \in \mathcal{S}, \text{s.t. } f(x) \leq 0$$

…

$x_1 \in \mathcal{S}$	$x_2 \in \mathcal{S}$	$x_3 \in \mathcal{S}$
$f(x_1) > 0$	$f(x_2) > 0$	$f(x_3) \leq 0$

Disprove by Finding Examples (2)

- This is just another name for adversarial attacks
  - We talked about FGSM, PGD, C-W, DeepFool, GEA, BPDA …
- There are even more sophisticated attacks

Case study 1: **No gradients**

ZOO, Obfuscated Gradients (BPDA)

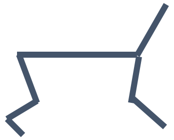
Case study 2: **complex objective**

Deep reinforcement learning!

SA-MDP, GoAttack

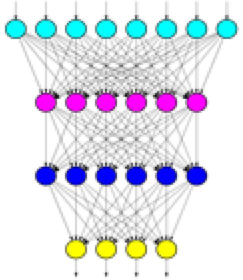
Deep Reinforcement Learning (DRL)

Reward



Agent states (positions, velocities…)

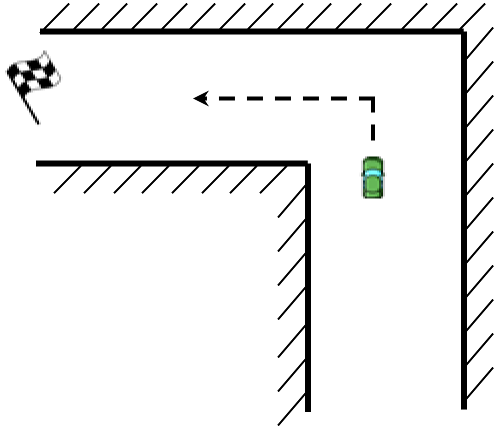
s

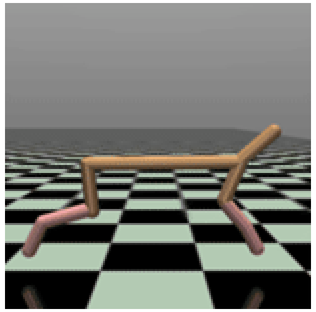


Neural network

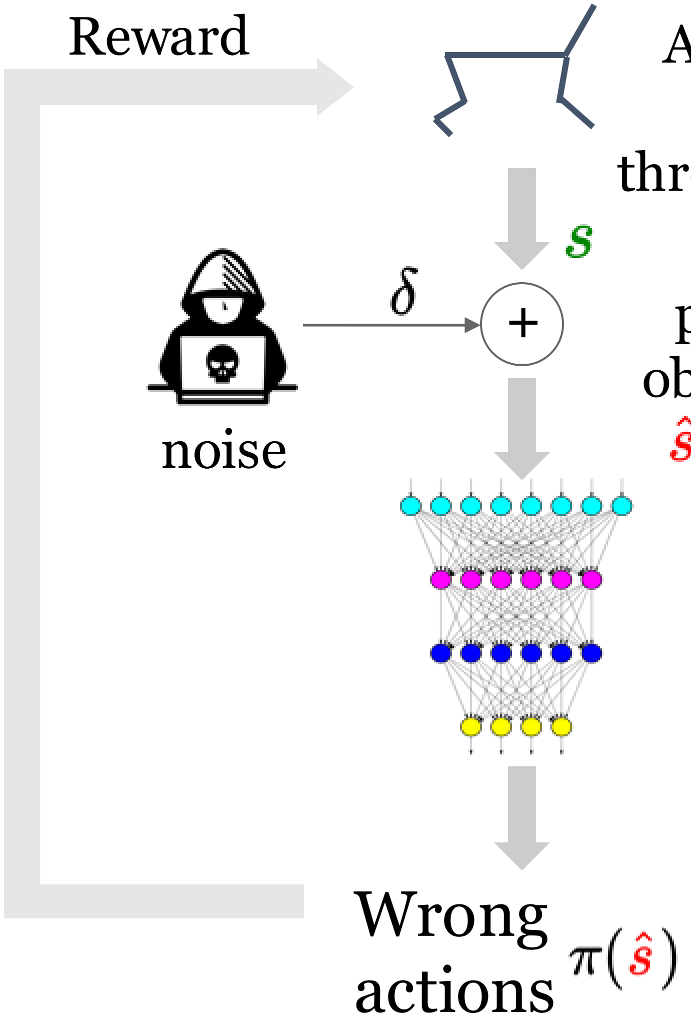
Agent actions $\pi(s)$

Agent's goal: maximize reward

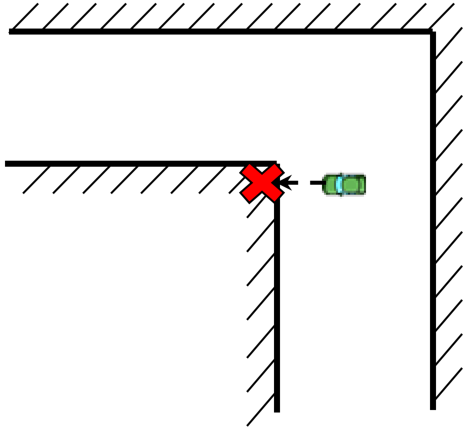


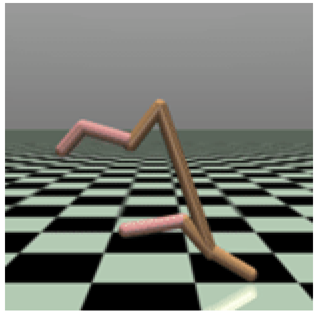


Attack Observations in DRL

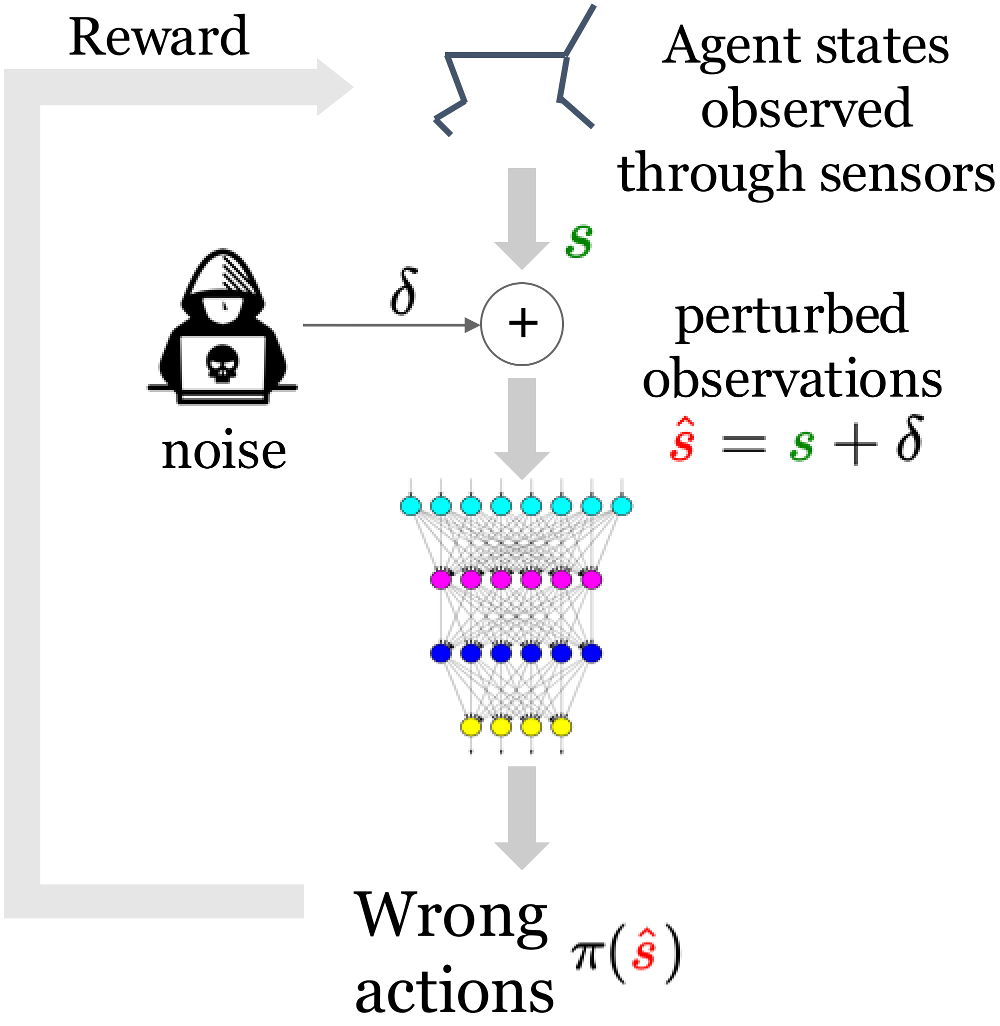


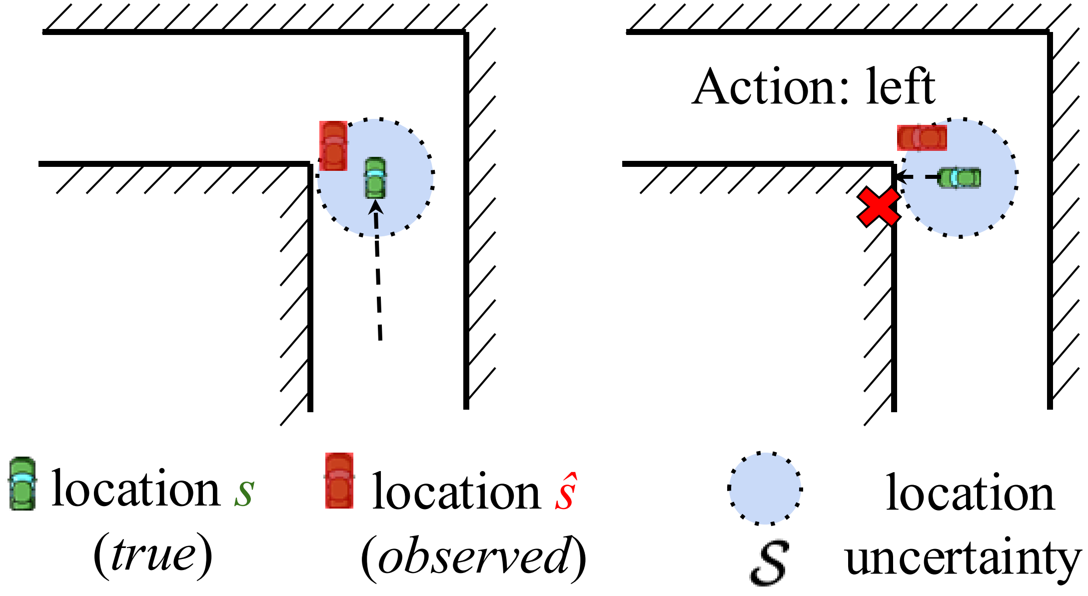
Adversary's goal: minimize reward while keeping attack stealthy



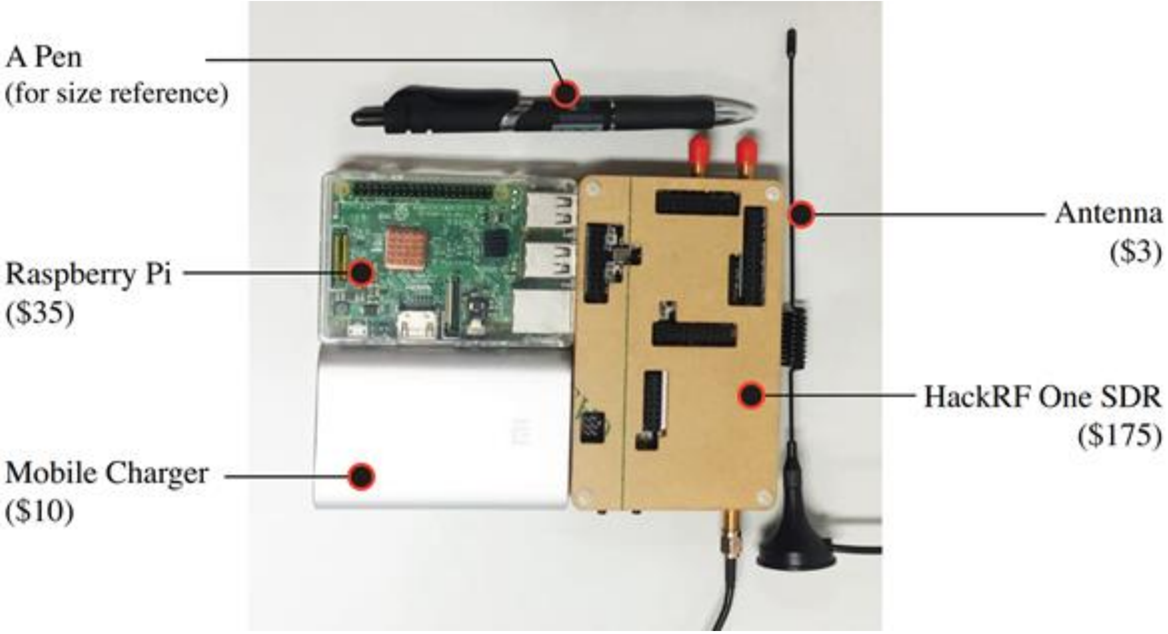


Attack Observations in DRL

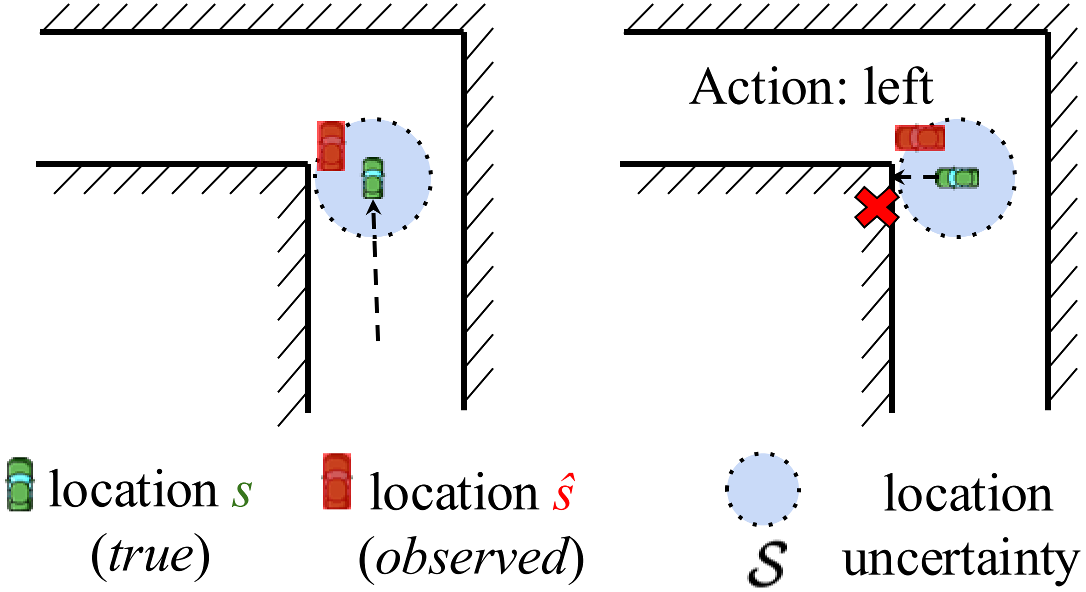




Such Attack can be Realistic



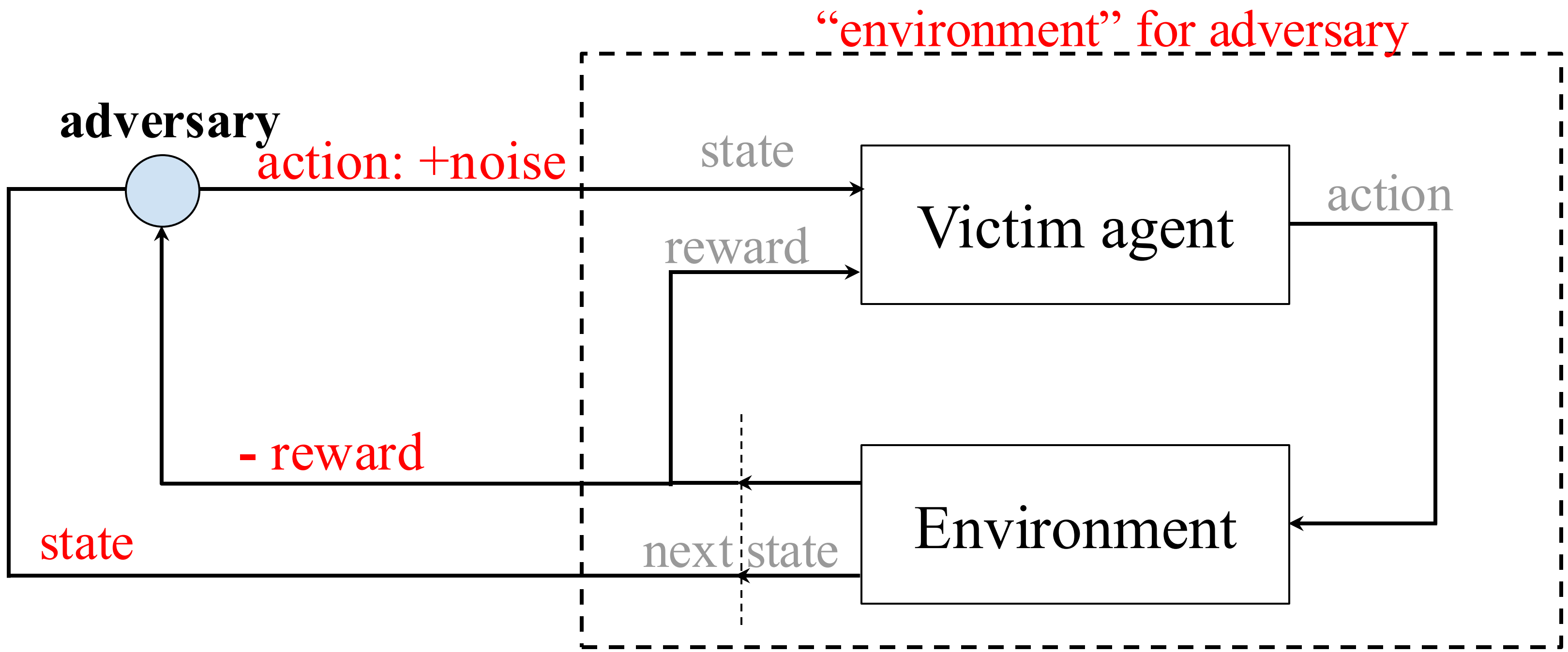
GPS spoofing
Zeng et al., USENIX 2018



Optimal Adversarial Attack

Theorem (informal): The optimal (strongest) adversarial attack on RL is another RL problem, defined as a Markov decision process (MDP).

Solve the RL attack problem by learning an adversary using RL

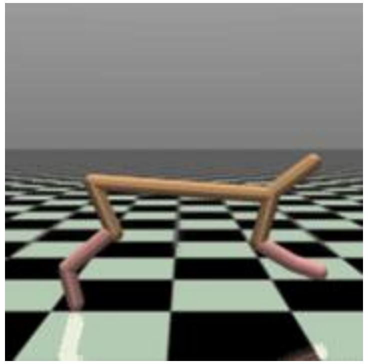


Robust Reinforcement Learning on State Observations With Learned Optimal Adversary, ICLR 2021

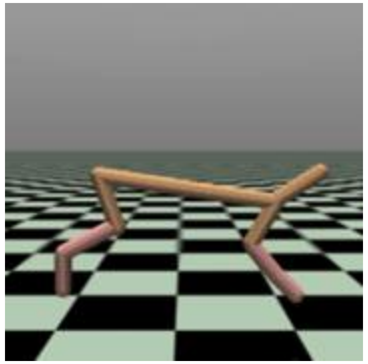
Optimal Attack is a Strong Adversary!

- Agents don't just fail; they move to the *opposite* direction!

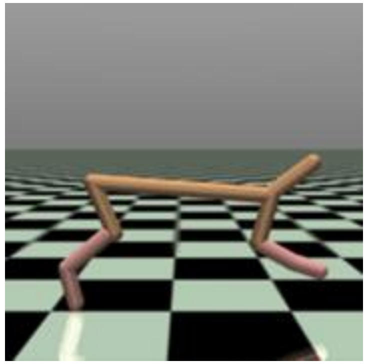
HalfCheetah



Episode rewards: +7094

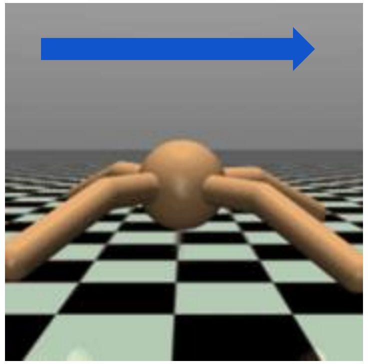


+85

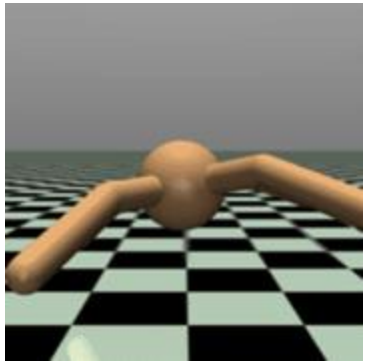


-743

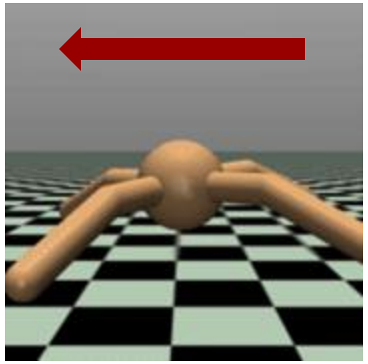
Ant



+5851



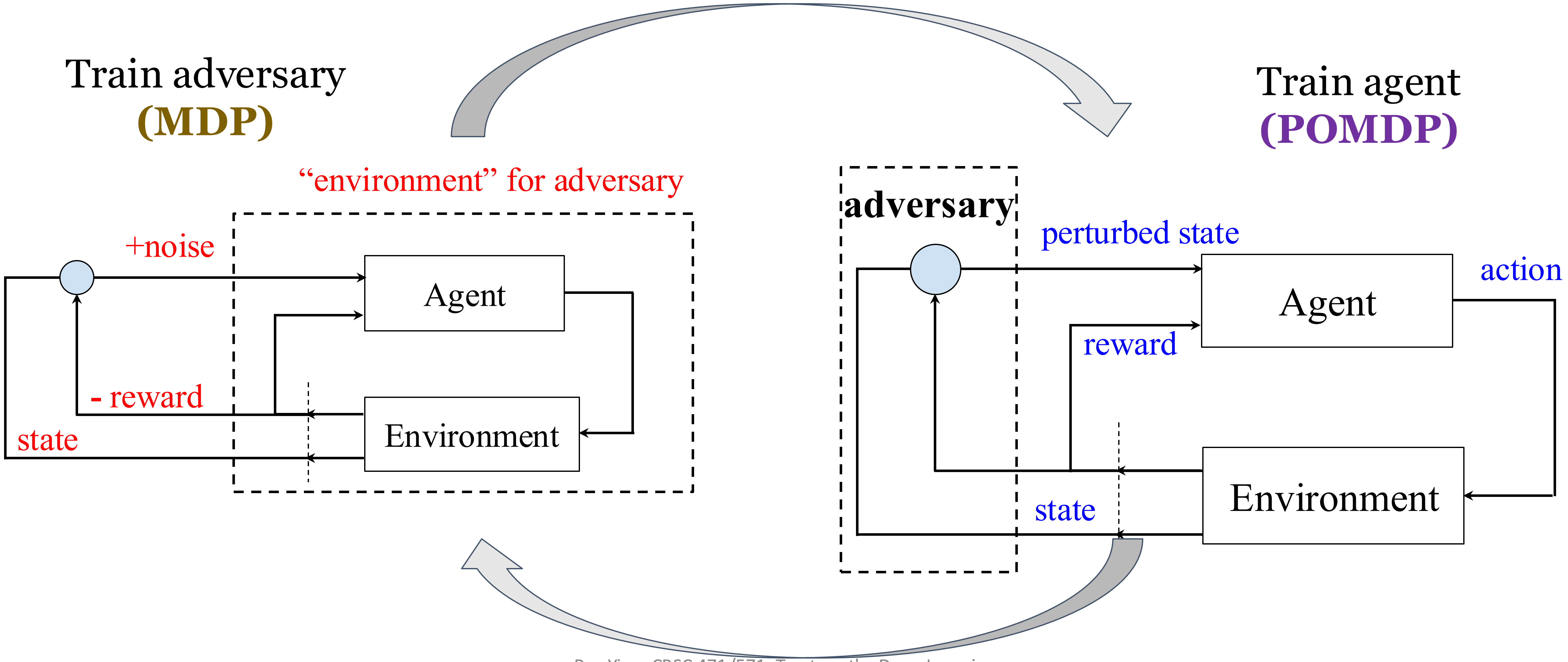
+63



-1141

No attack | MAD attack | Optimal attack

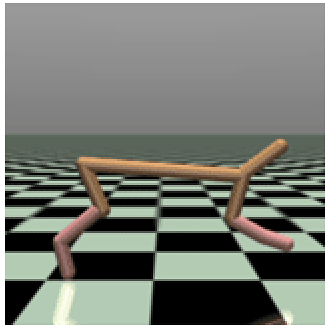
Train robust deep reinforcement learning agents



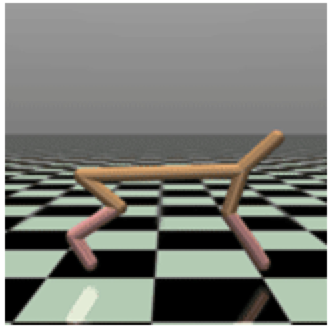
Robust Deep Reinforcement Learning Agents

Episode rewards:

HalfCheetah

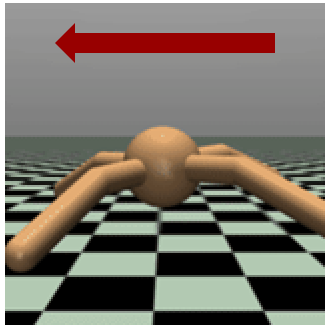


-743

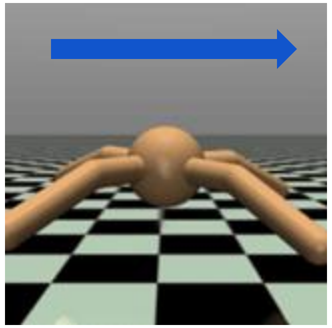


+5250

Ant



-1141



+3835

Optimal attack on vanilla RL agents

Optimal attack on robust RL agents