

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)

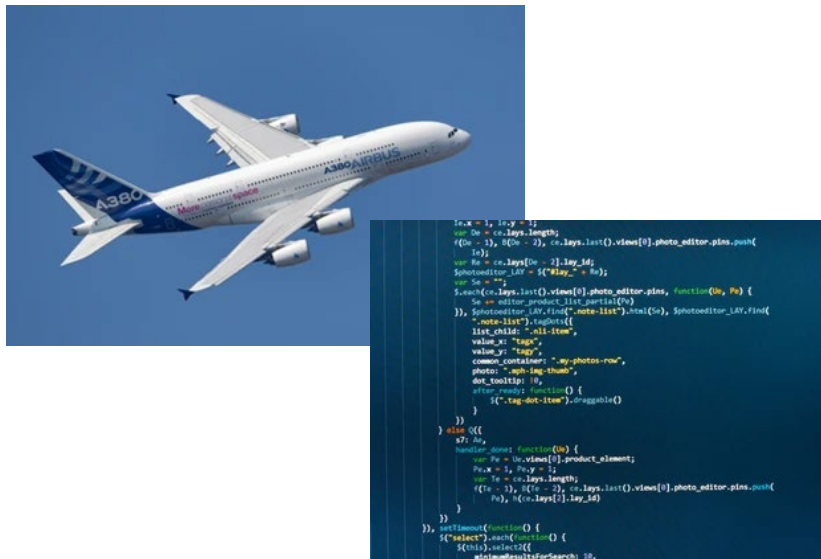
Content

- 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 did we build trustworthy mission critical systems?



Prove
trustworthiness
SS



Verifying correctness using
formal **verification**

Disprove
trustworthiness
SS



Testing with edge
cases to find bugs

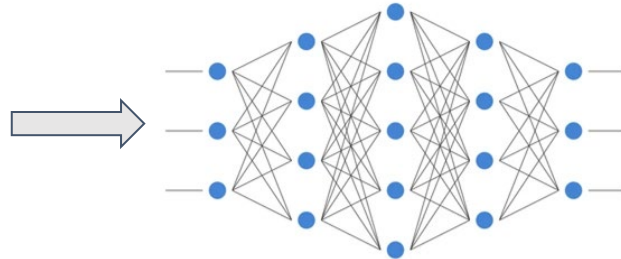
Translating to Trustworthy Deep Learning

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

Prove
trustworthine
ss



How to formally prove the trustworthiness of AI?



Approach: efficient and specialized verification and testing methods for AI

Disprove
trustworthiness

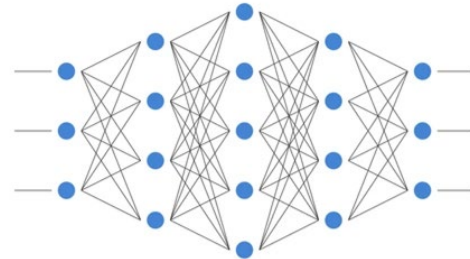


We have talked about this without rigorous proofs

cases and bugs of AI?

Formal Verification of AI

```
1: x = 1, y = 1;
2: var l0 = ce.layers.length;
3: if (l0 > 2) { (l0 - 2), ce.layers.last().views[0].photo_editor.pins.push(
4:   l0);
5: var l0 = ce.layers[l0 - 2].lay_id;
6: photoeditor_lay = {"lay_id": l0};
7: var l0 = l0 + 1;
8: l0.each((ce.layers.last().views[0].photo_editor.pins, function(l0, p0) {
9:   ce ce editor_product_list.push(l0);
10:   photoeditor_lay.find("note-list").val(l0), photoeditor_lay.find(
11:     "note-list").tagList({
12:       list_child: "all-item",
13:       value_x: "tag",
14:       value_y: "tag",
15:       common_container: "my-photos-row",
16:       photo: "my-tag-thumb",
17:       dot_tooltip: 0,
18:       after_click: function() {
19:         {"tag-dot-list", "tag-dot-list"}
20:       }
21:     })
22:   })
23: } else {
24:   handler_click function(l0) {
25:     var p0 = ce.views[0].product_element;
26:     p0.x = 1, p0.y = 1;
27:     var l0 = ce.layers.length;
28:     if (l0 > 2) { (l0 - 2), ce.layers.last().views[0].photo_editor.pins.push(
```

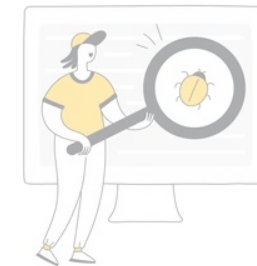


Prove
trustworthiness
SS



How to formally prove the
trustworthiness of AI?

Disprove
trustworthiness
SS

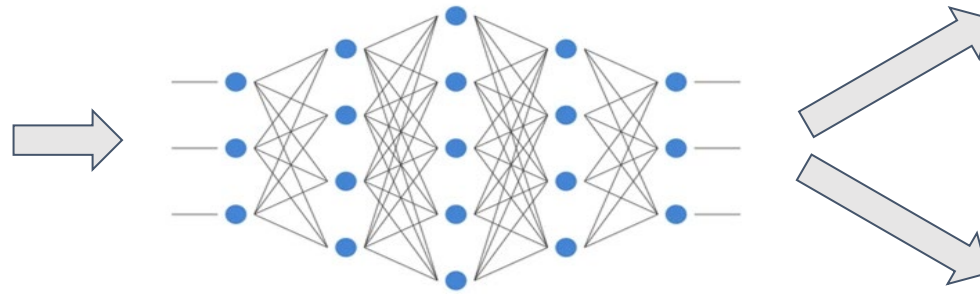


How to find edge
cases and bugs of AI?

Goal of formal verification of AI: an Example



Eykholt et al. 2018



STOP 😊

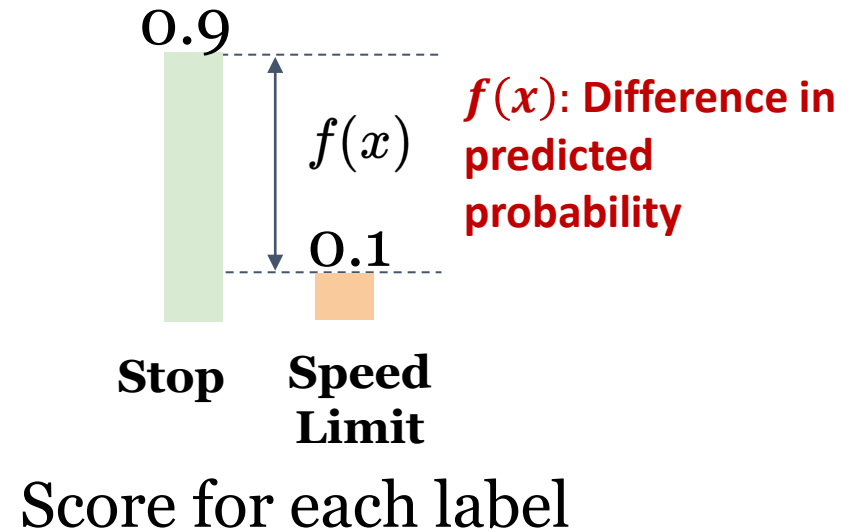
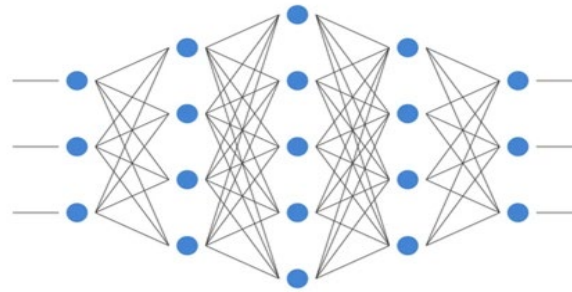
Speed limit 😈

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

A simplified example of the verification problem



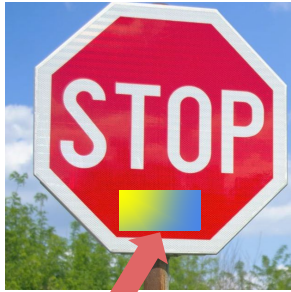
Attacker may put
anything here



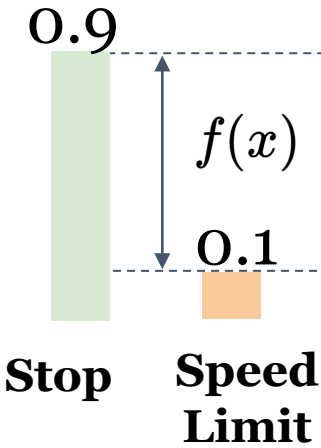
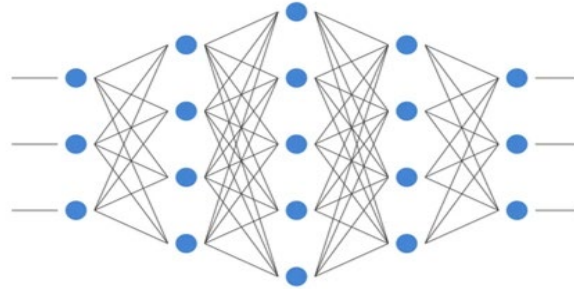
What's this perturbation constraint?

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

The Canonical Form of Verification Problem

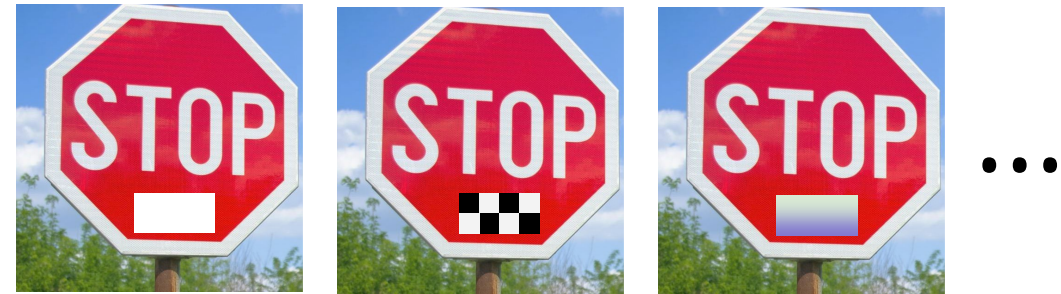


Attacker may put
anything here



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

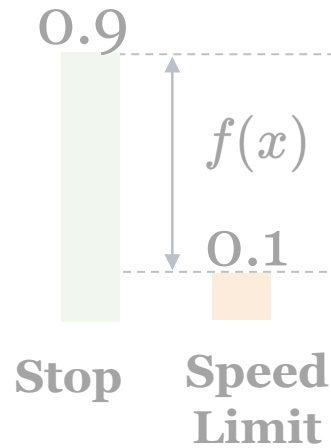
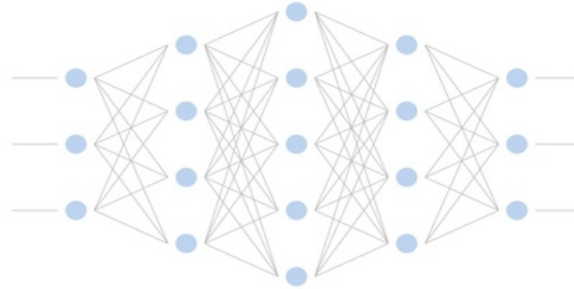
The Canonical Form of Verification Problem



Attacker may put
anything here

That was just an example of a
verification problem

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



Score for each label

\mathcal{S} = all possible pixel perturbations



$x_1 \in \mathcal{S}$



$x_2 \in \mathcal{S}$



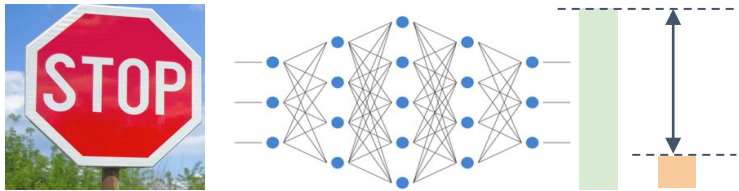
$x_3 \in \mathcal{S}$

...

The Canonical Form of Verification is General

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

Application



Specification $f(x)$

Attack cannot succeed

Sets of inputs \mathcal{S}



Perturbed pixel values



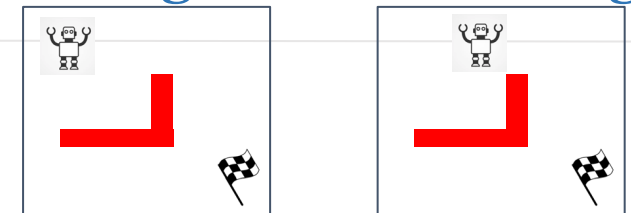
What is the general approach to prove this?



Drone starting at different angles



Robot does not reach
a obstacle
(reachability)



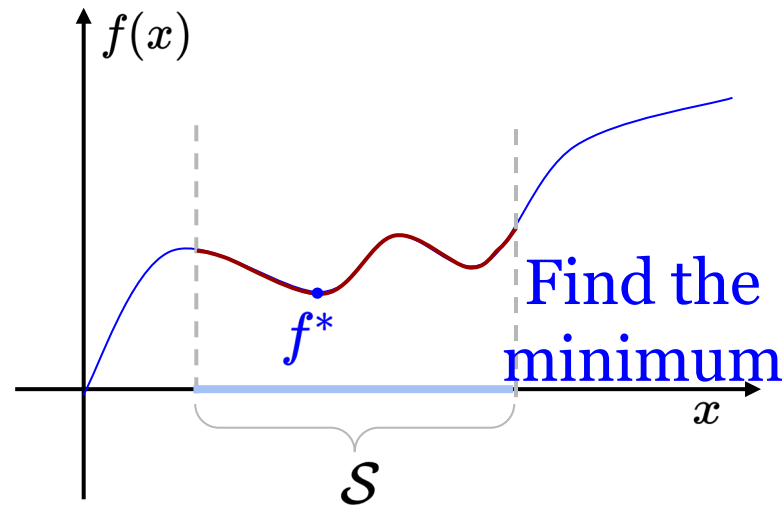
Different starting points

Content

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

How Hard is the Problem?

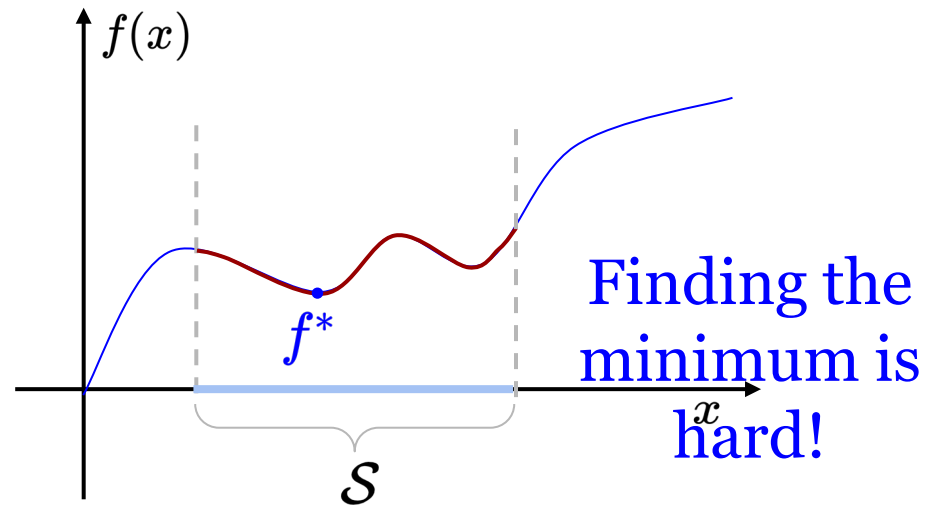
Prove: $\forall x \in \mathcal{S}, f(x) > 0$



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

Traditional Approach: Using Optimization Solvers

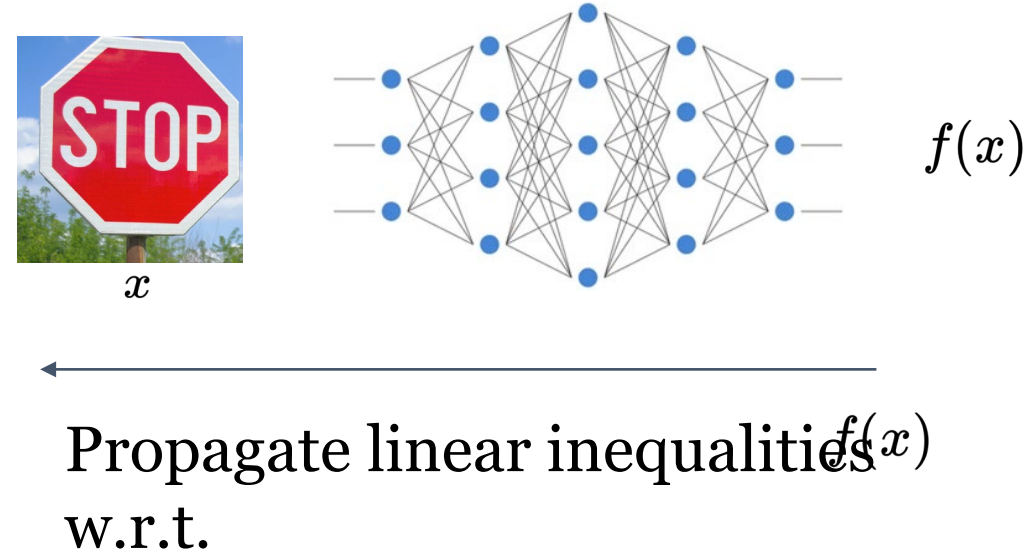
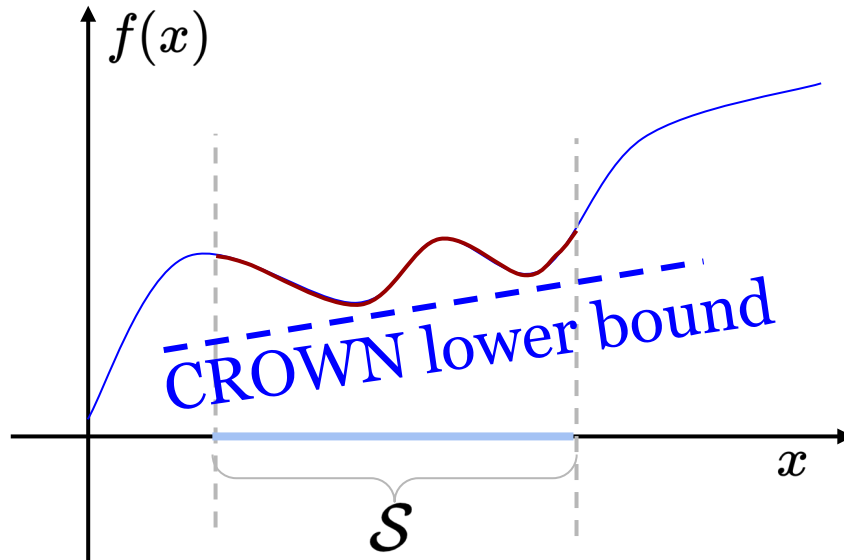
Prove: $\forall x \in \mathcal{S}, f(x) > 0$



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

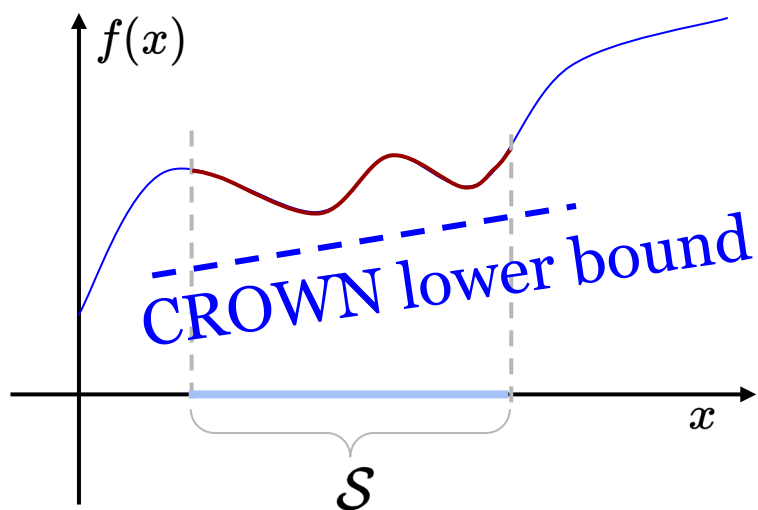


CROWN: a Linear Bound Propagation Algorithm



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

Prove the verification problem with CROWN



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



$x_1 \in \mathcal{S}$



$x_2 \in \mathcal{S}$

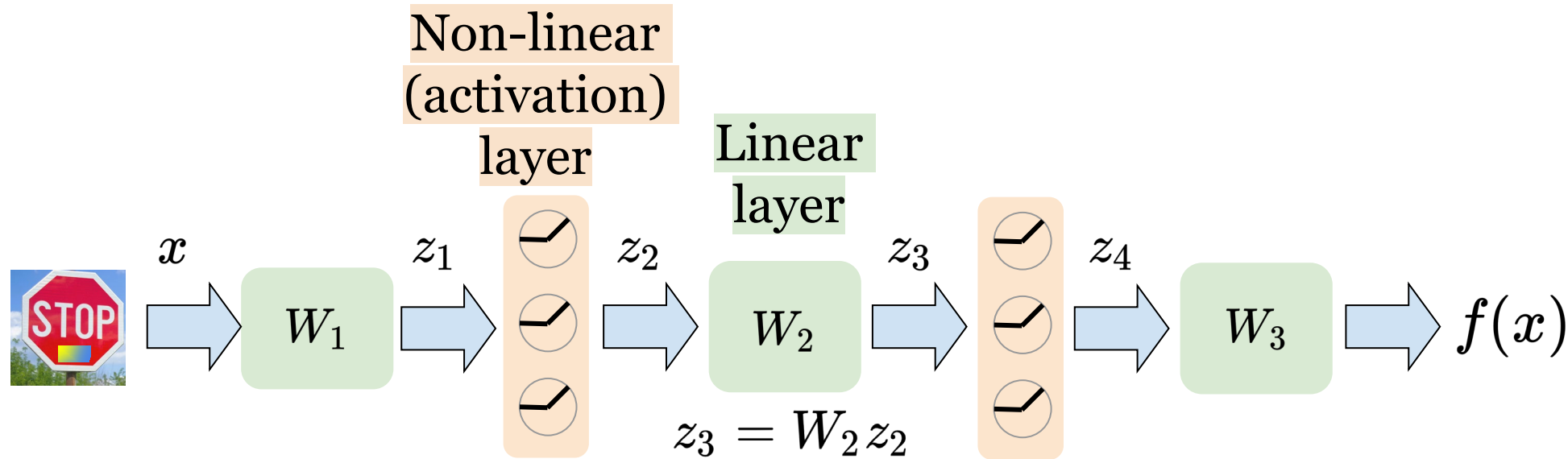


$x_3 \in \mathcal{S}$

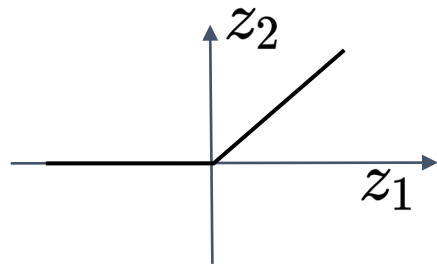
...

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

How to Propagate the Linear Bounds?



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

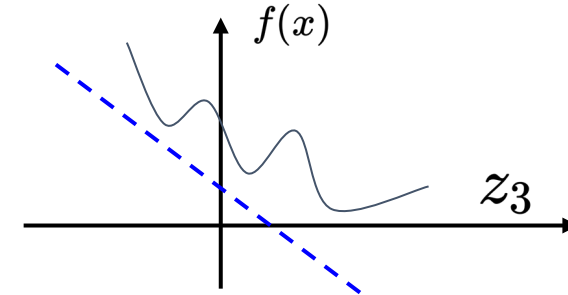


Steps:

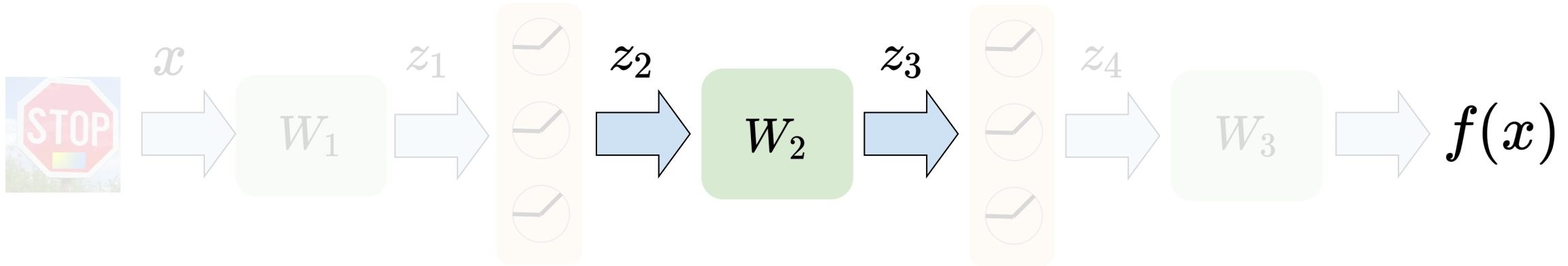
- Propagate bounds through linear layers
- Propagate bounds through non-linear layers

What Linear Inequalities to Propagate?

A linear lower bound for an intermediate layer

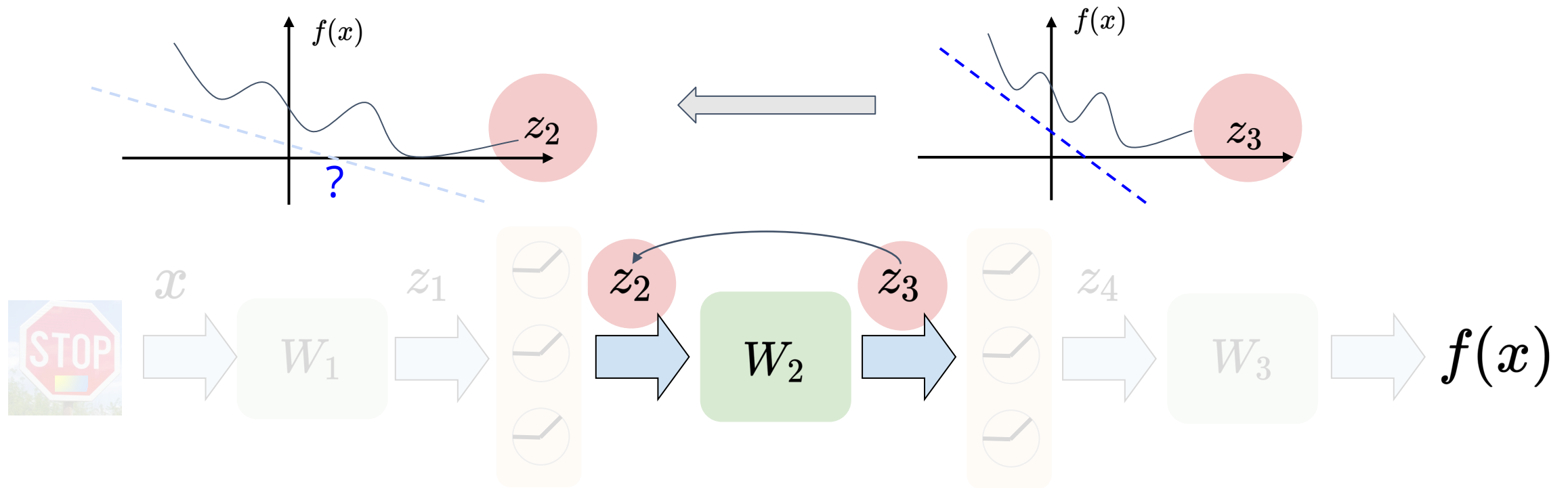


1-D example.
Generally it's a
linear hyperplane



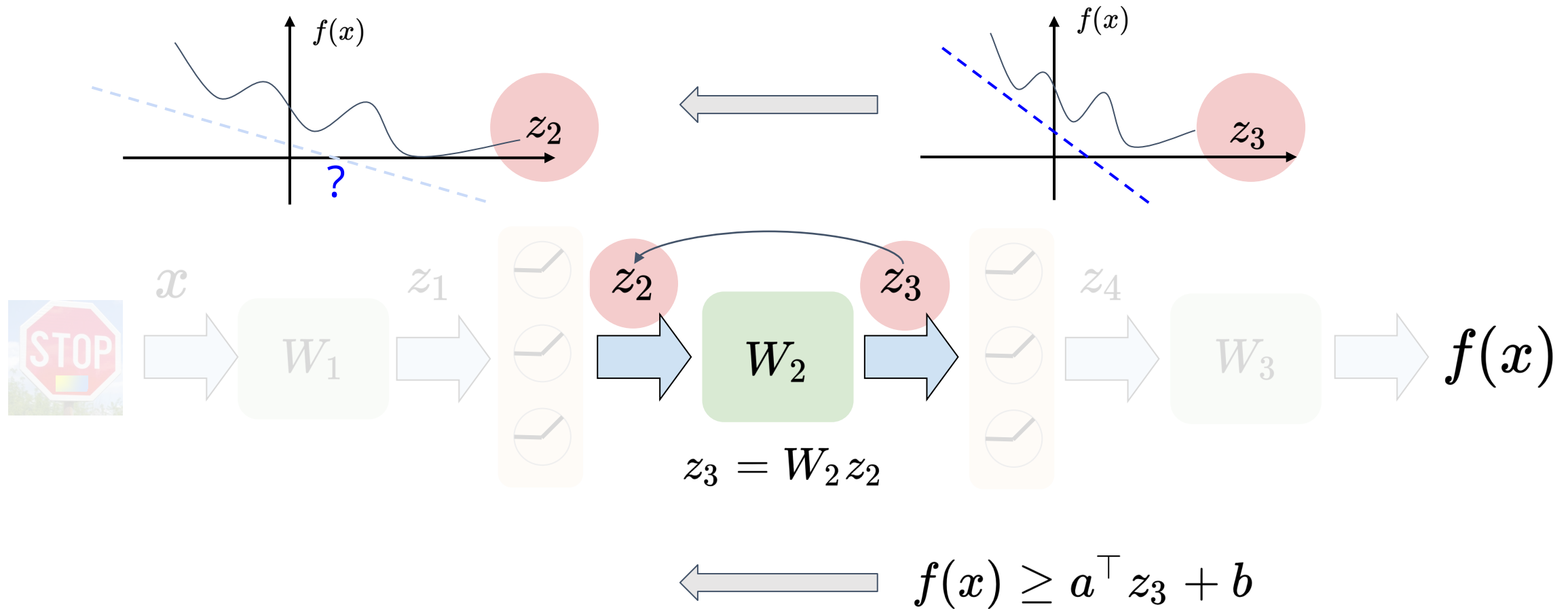
$$f(x) \geq a^\top z_3 + b$$

Propagating Bounds through Linear Layers

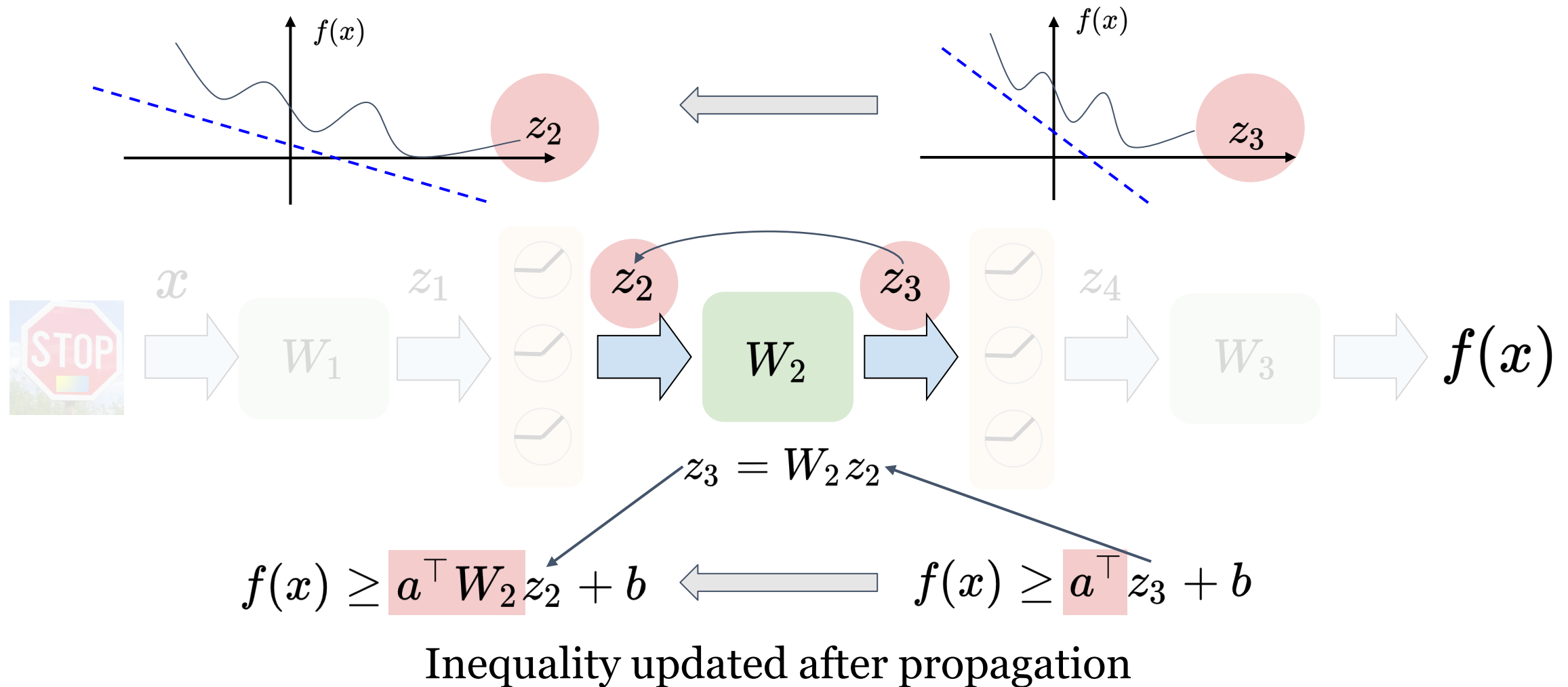


Propagate it to one layer before,
while keeping the lower bound valid

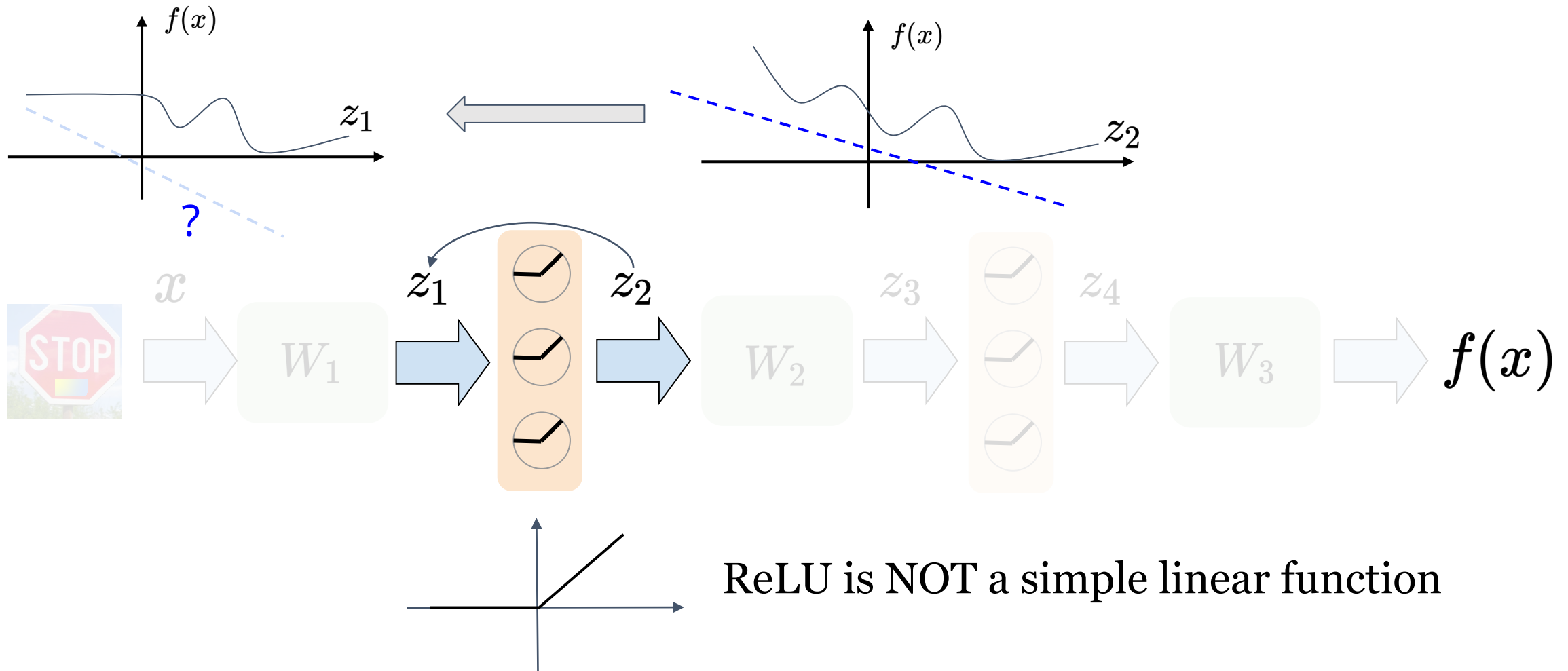
Propagating Bounds through Linear Layers



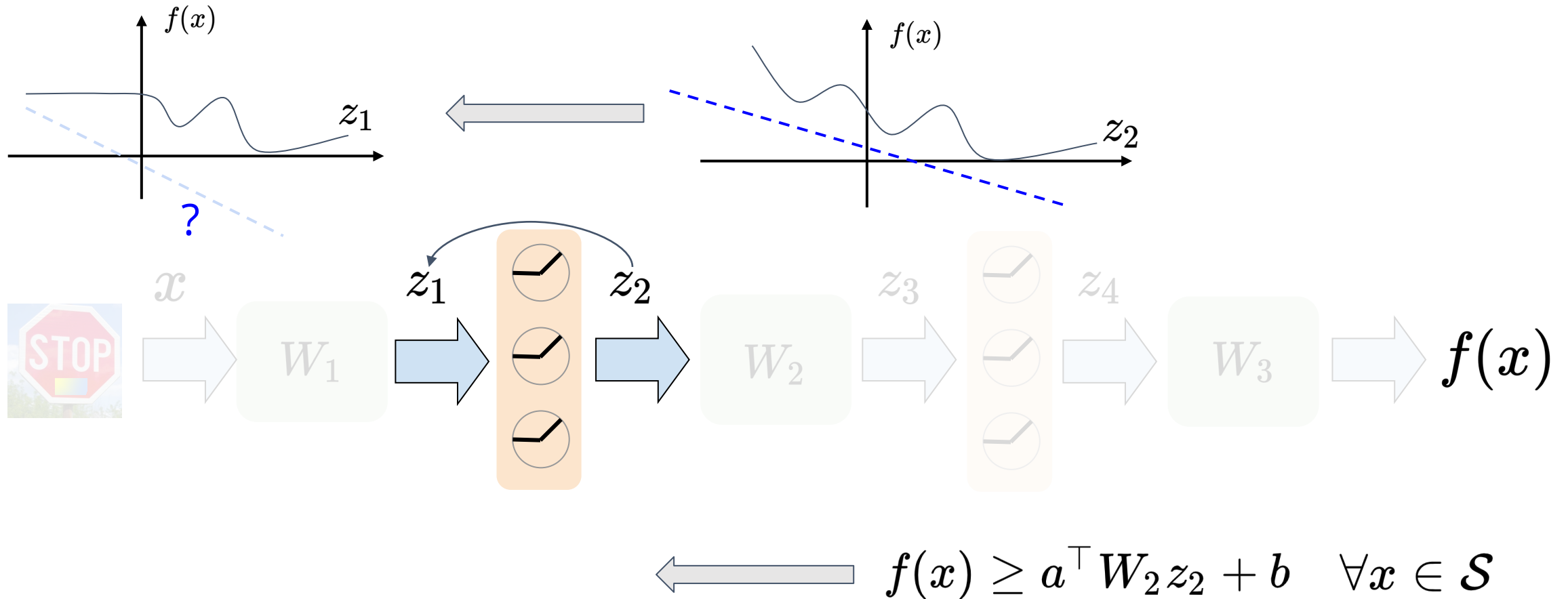
Propagating Bounds through Linear Layers



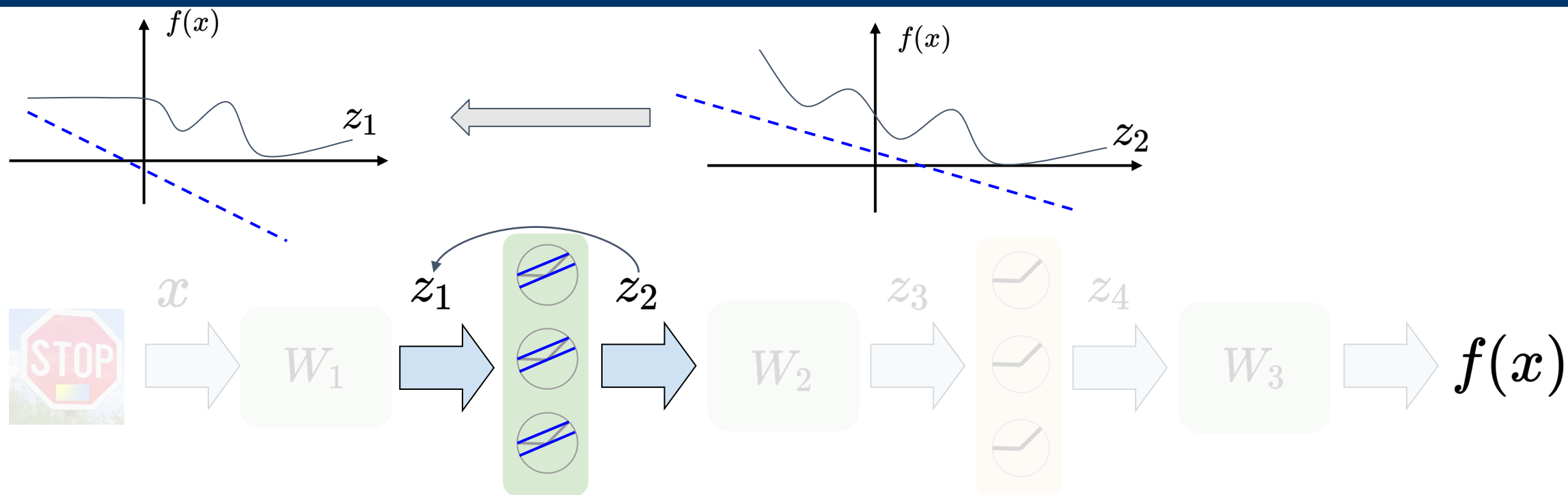
Propagating Bounds through non-Linear Layers



Propagating Bounds through non-Linear Layers



Propagating Bounds through non-Linear Layers

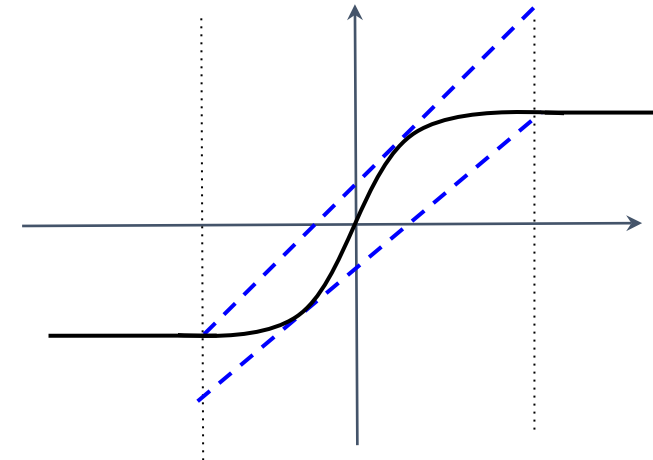
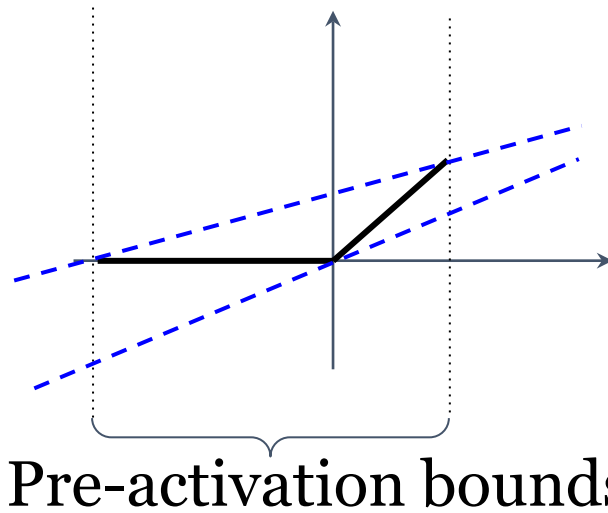


Theorem (informal): we can efficiently find D , b' such that:

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

Proof for non-Linear Propagation

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

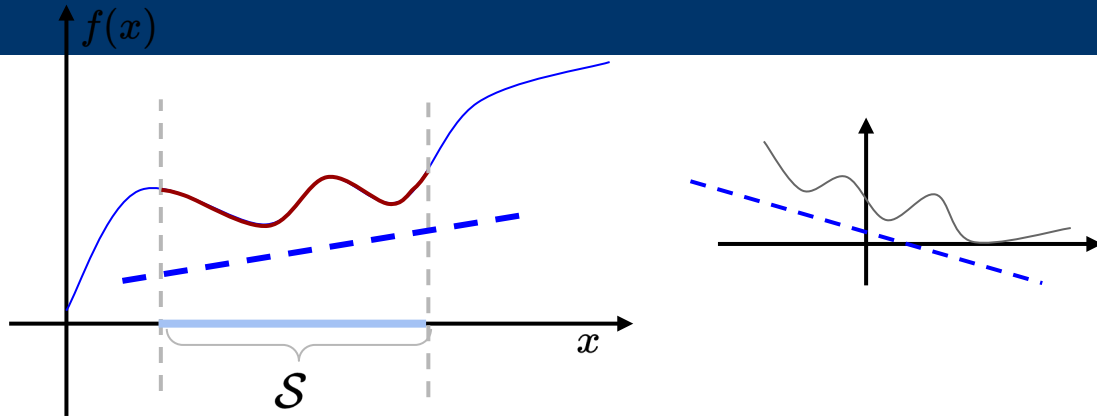


(can be pre-computed using CROWN)

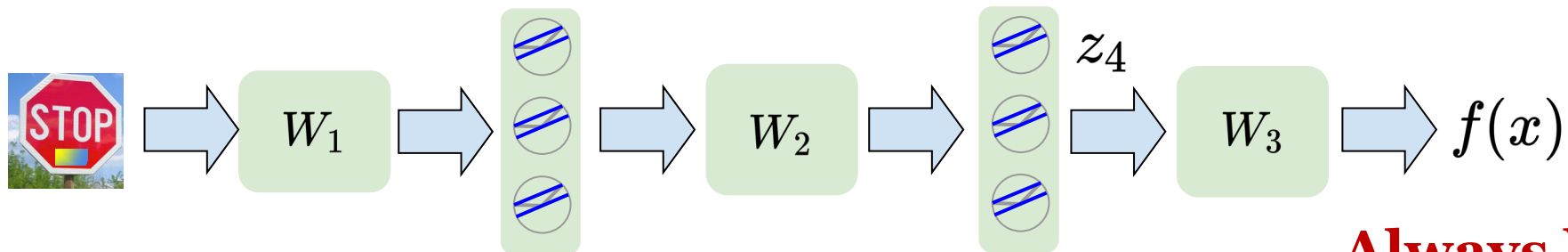
Theorem (informal): we can efficiently find D , b' such that:

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

CROWN Algorithm



**Bounds propagated through
simple matrix multiplications!
Fast and GPU-friendly**



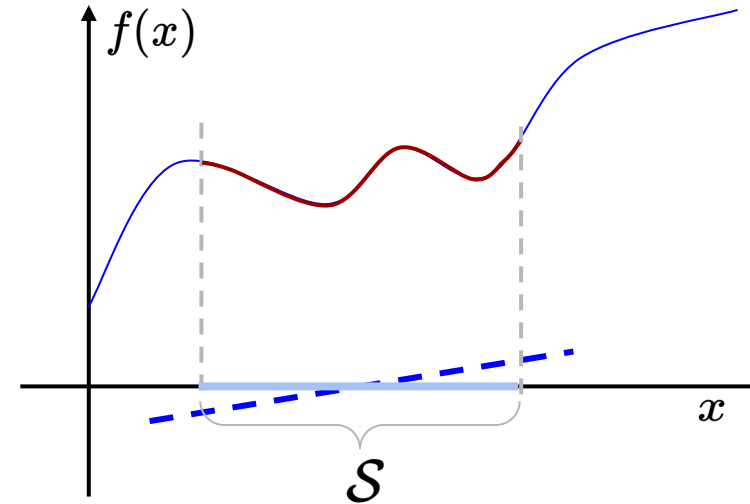
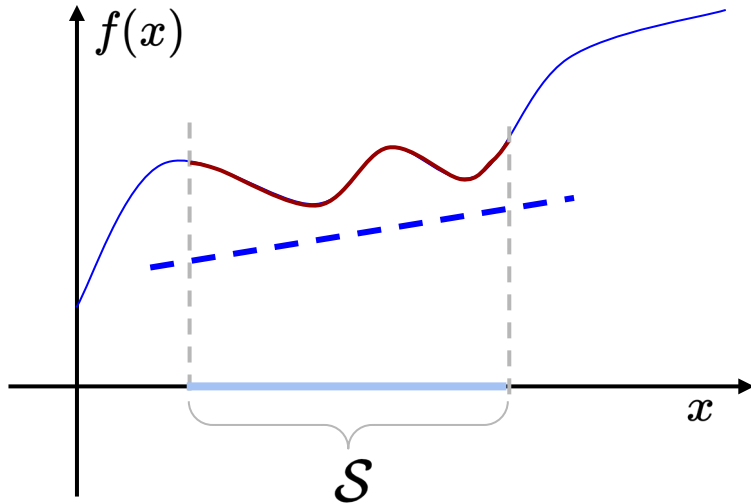
**Always keep valid
lower bounds**

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

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

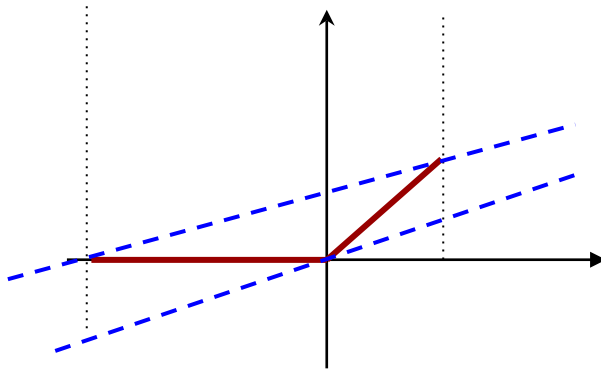
What if the Bounds are too Loose?

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

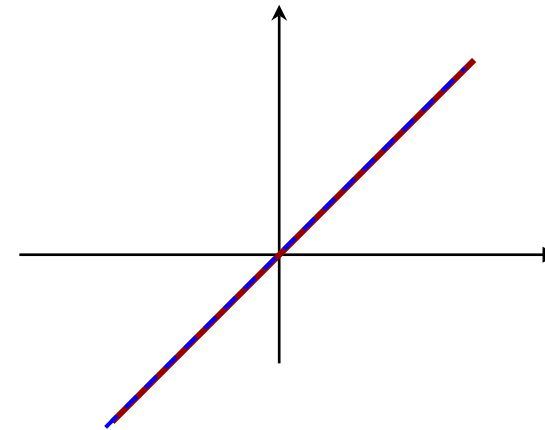


Tighten Bounds with Branch and Bound

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



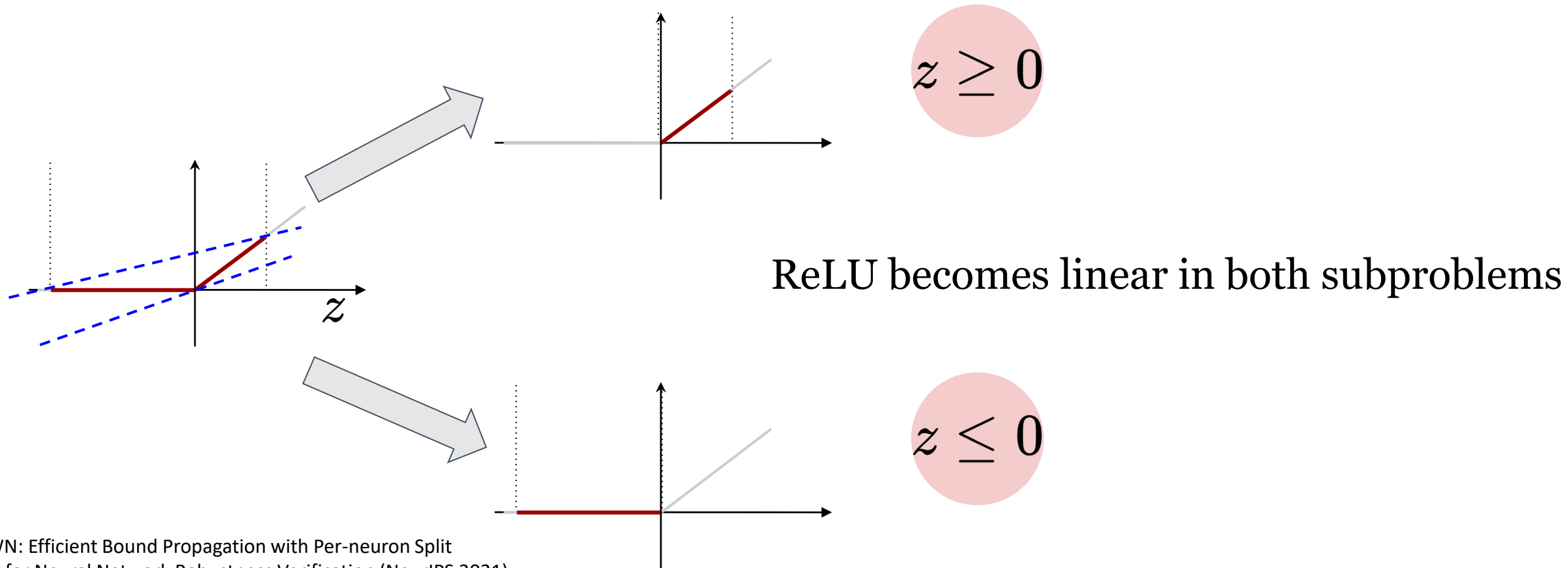
ReLU function:
Bounded by conservative
linear bounds



Linear function:
Bounds are tight

Tighten Bounds with Branch and Bound

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



Tighten Bounds with Branch and Bound

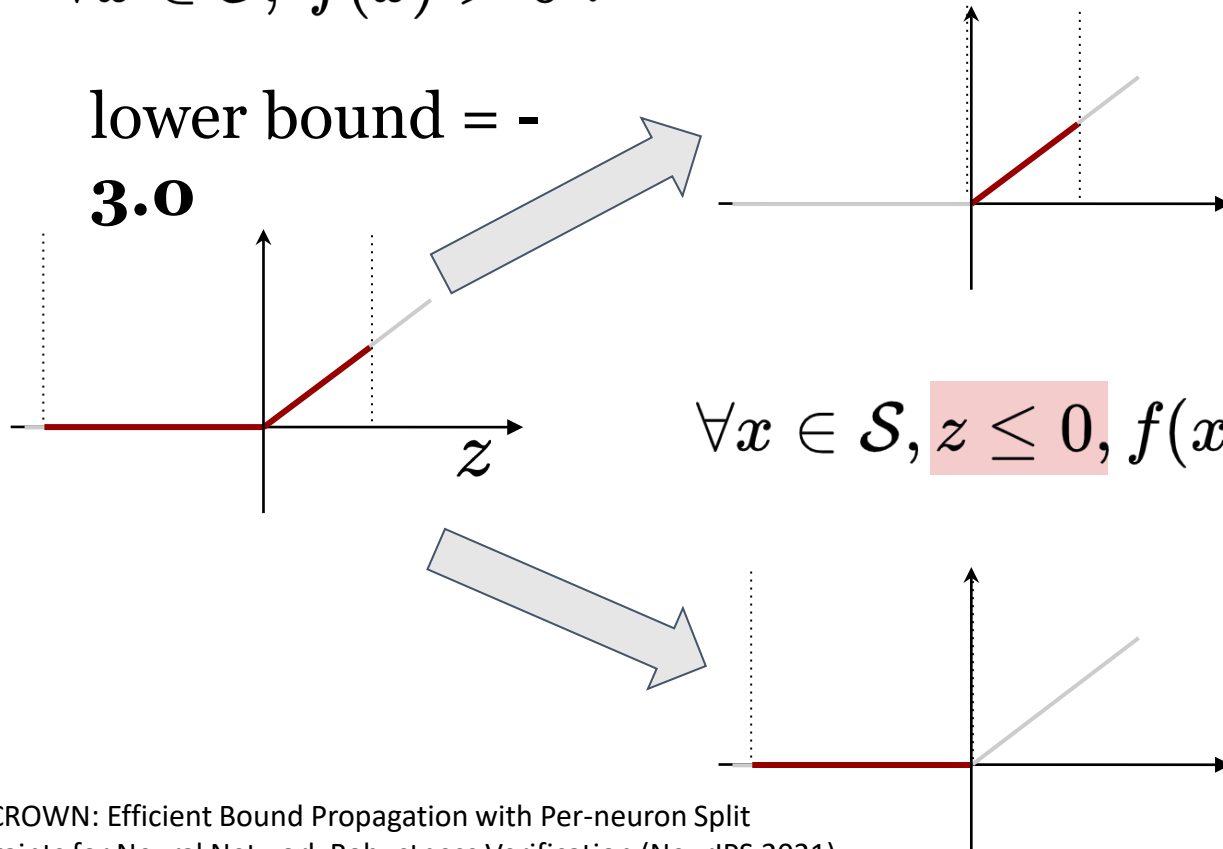
Verification problem: $\forall x \in \mathcal{S}, z \geq 0, f(x) > 0 ?$

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

lower bound = **0.5**

Subproblem Verified ✓

lower bound = -
3.0



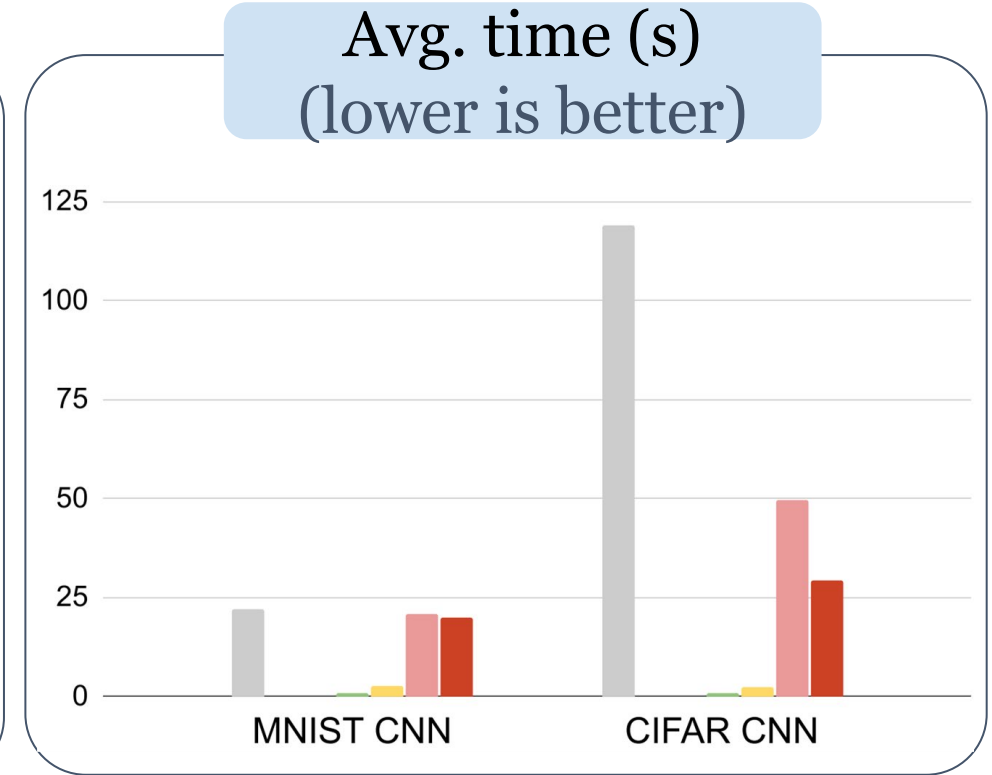
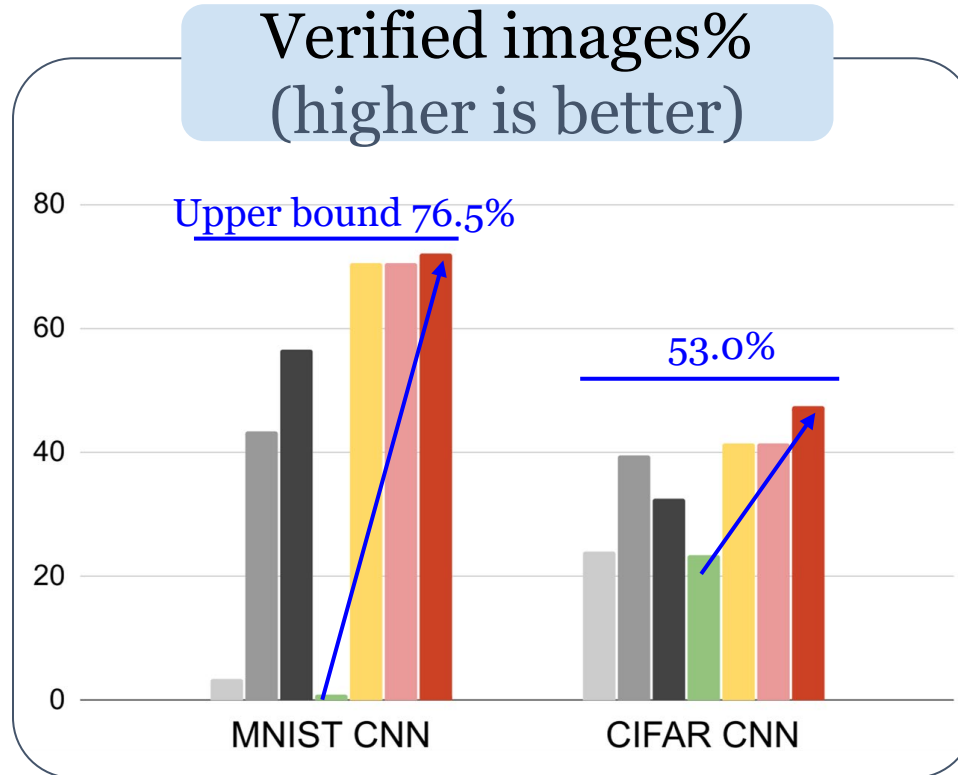
$\forall x \in \mathcal{S}, z \leq 0, f(x) > 0 ?$

lower bound = **-2.0**

Choose another ReLU
neuron to split

Benchmarks: CROWN Verification

- Linear Programming (Salman et al. 2019)
- Semidefinite Programming (Dathathri et al. 2020)
- Integer Programming (Tjeng et al. 2017)
- CROWN
- α -CROWN
- β -CROWN
- GCP-CROWN



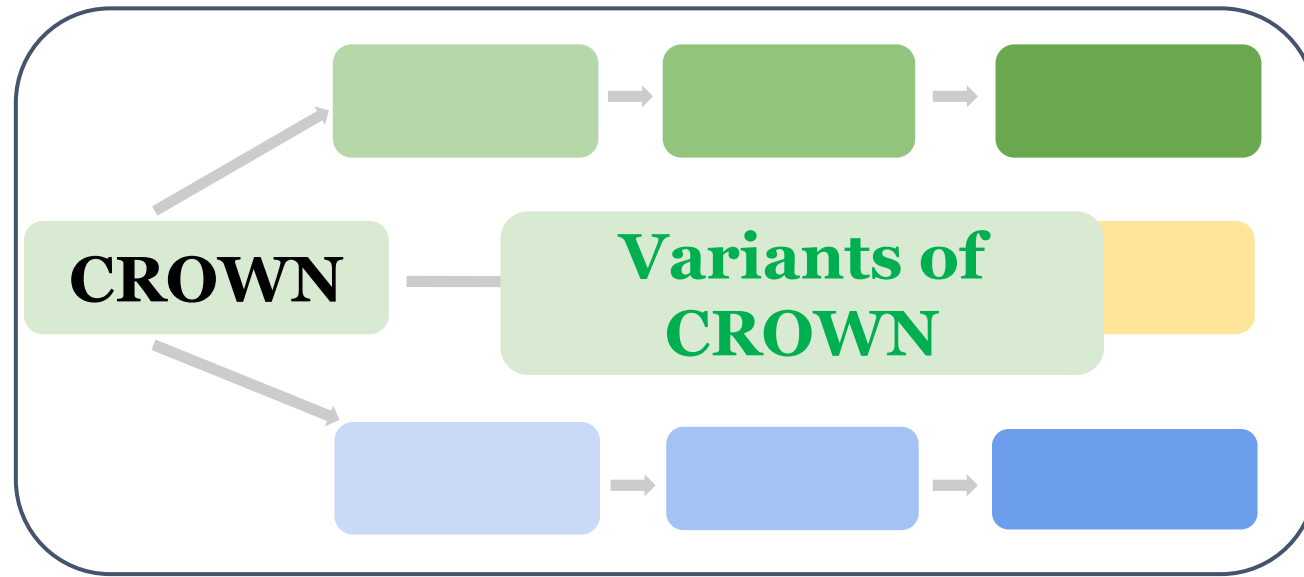
Model size: ~5k
neurons

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

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

Practical Tool: α, β -CROWN

Theoretical framework



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

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

Faster verifiable
training
(NeurIPS 2021b)

Practical verifier: α, β -CROWN



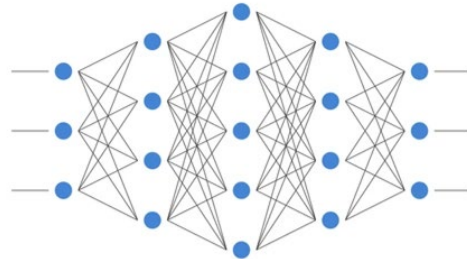
Winner of VNN-COMP
2021, 2022, 2023

<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



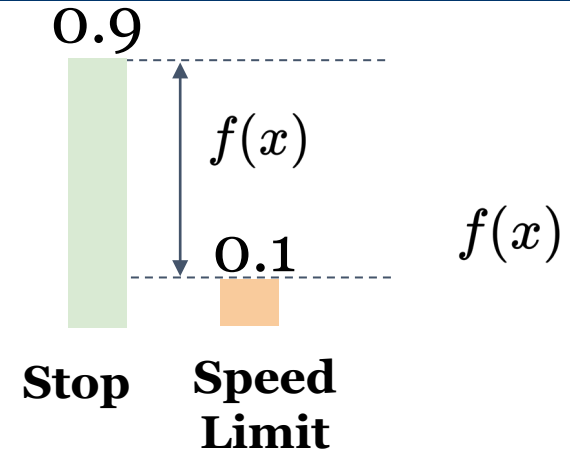
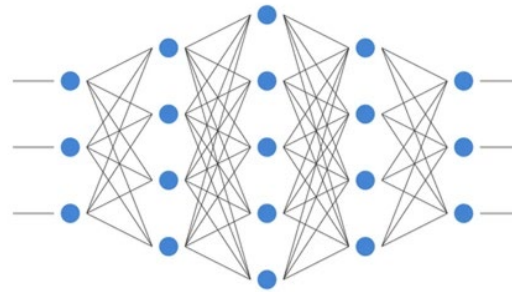
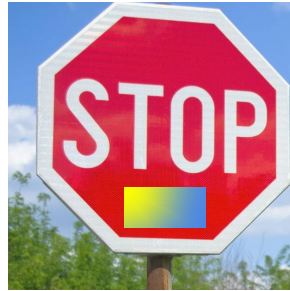
How to formally prove the trustworthiness of AI?



How to find edge cases and bugs of AI?

Disprove
trustworthiness

Disprove by Finding Examples (1)



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

Adversarial attack: **disprove** by showing

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

Set of inputs \mathcal{S}



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



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



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

...

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**

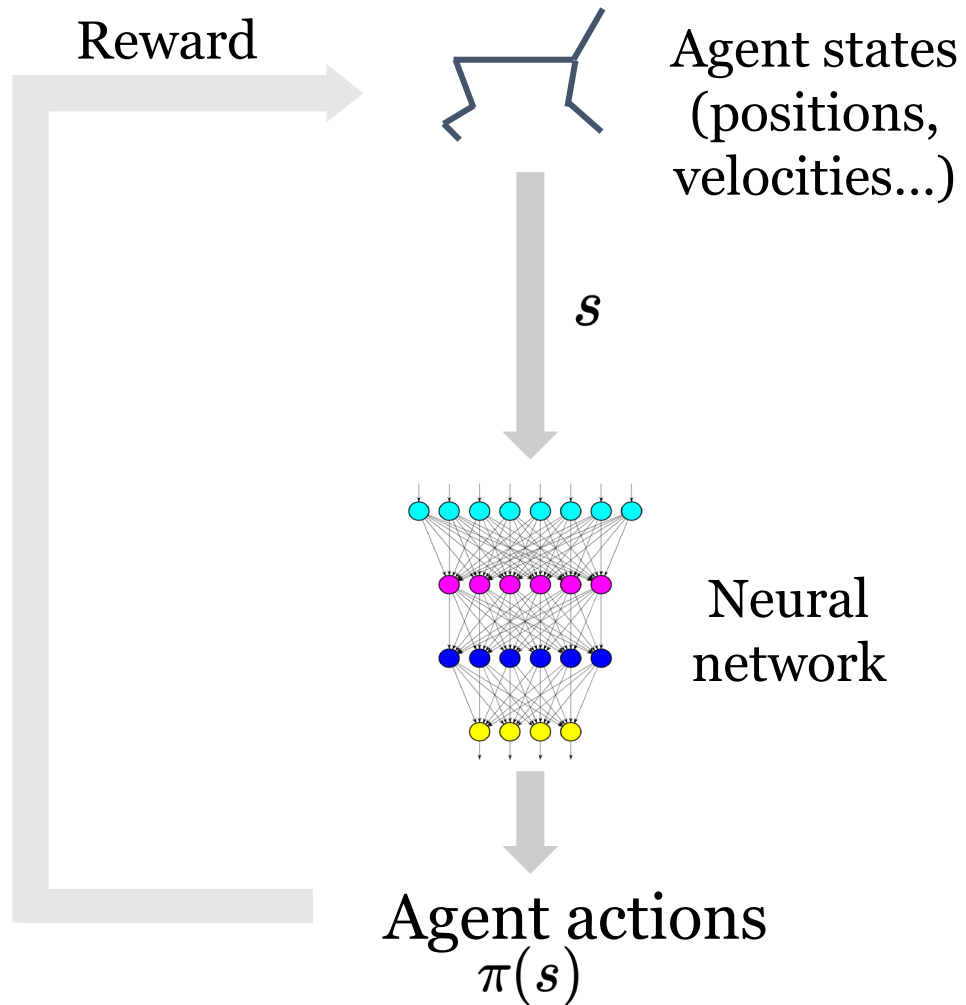
[ZOO](#), [Obfuscated Gradients](#)

Case study 2: **complex objective**

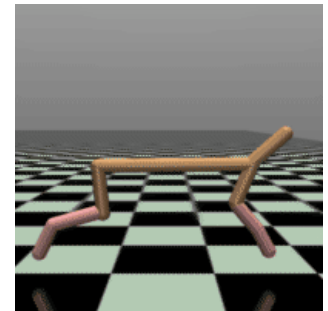
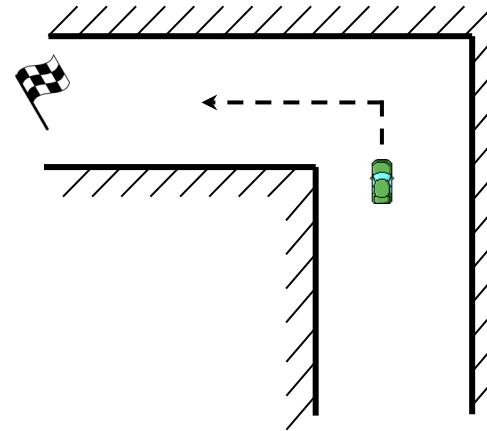
Deep reinforcement learning!

[SA-MDP](#), [GoAttack](#)

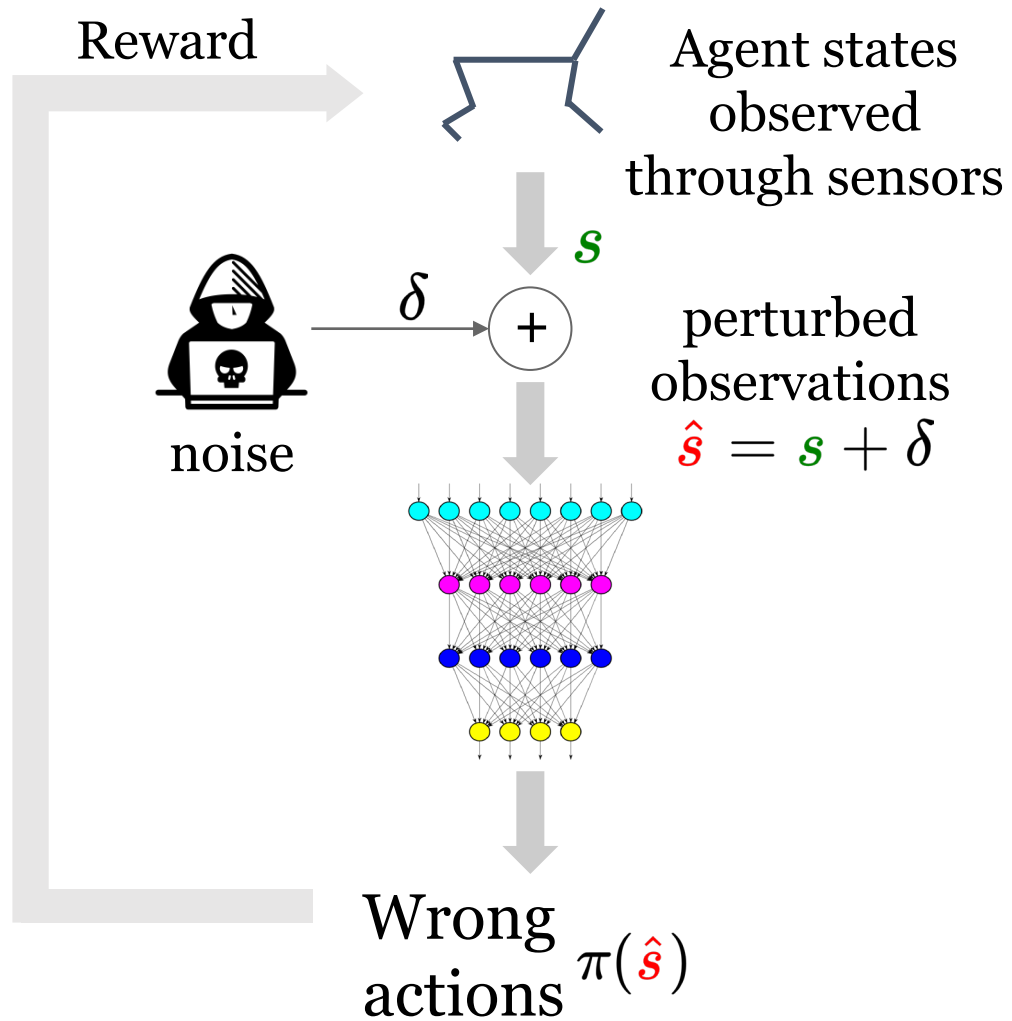
Deep Reinforcement Learning (DRL)



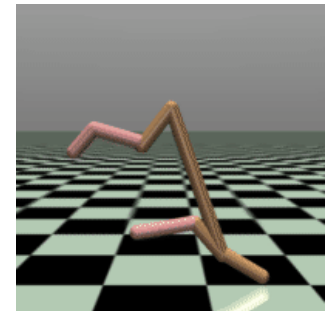
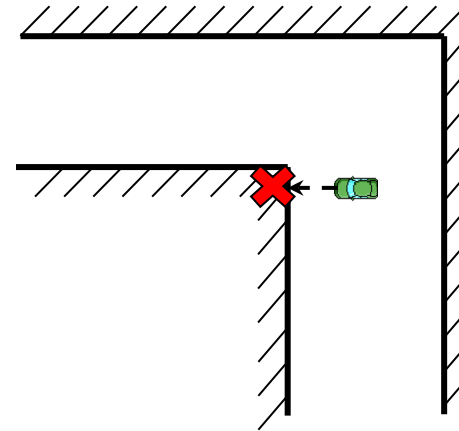
Agent's goal: maximize reward



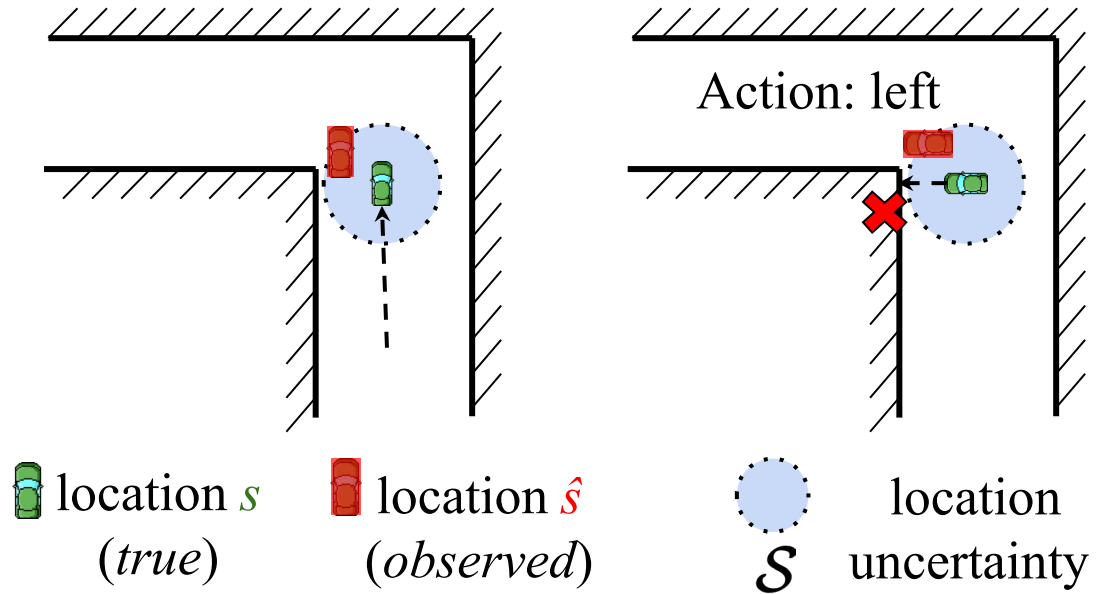
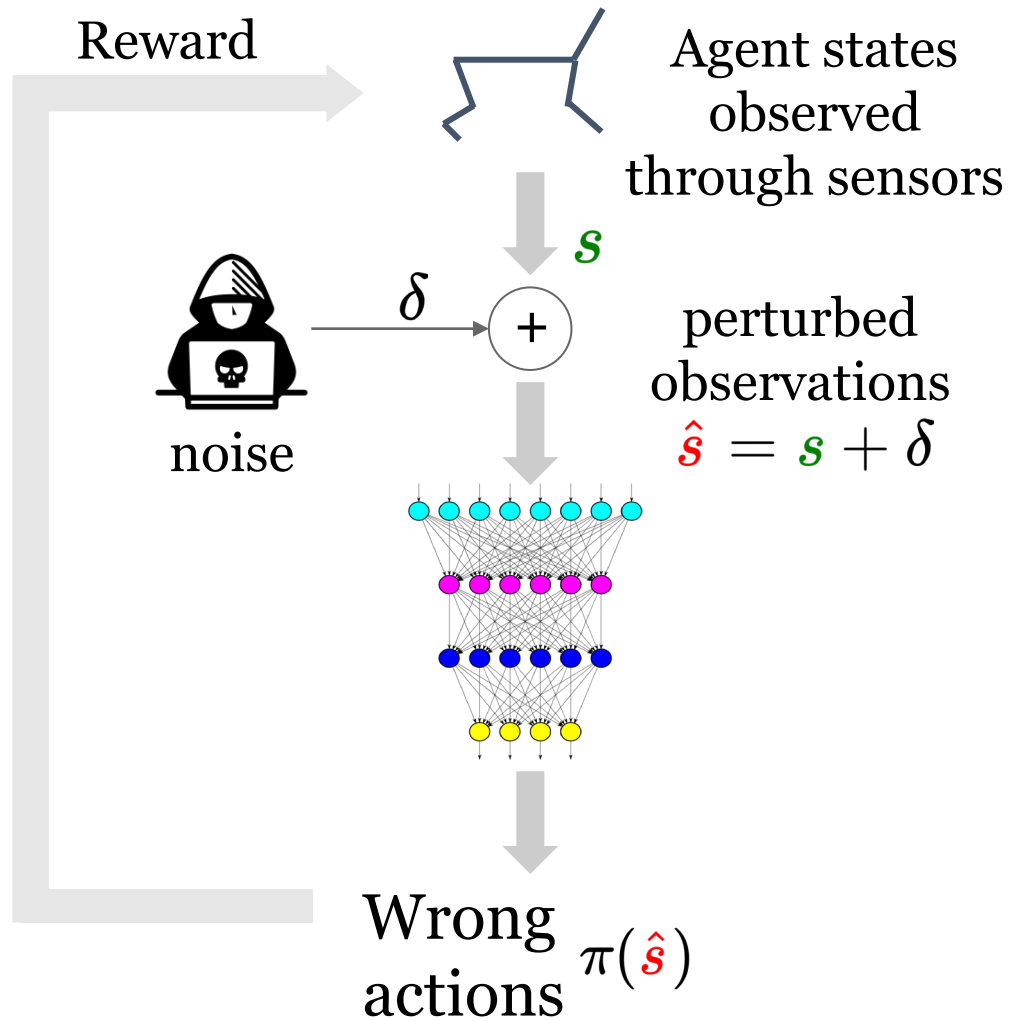
Attack Observations in DRL



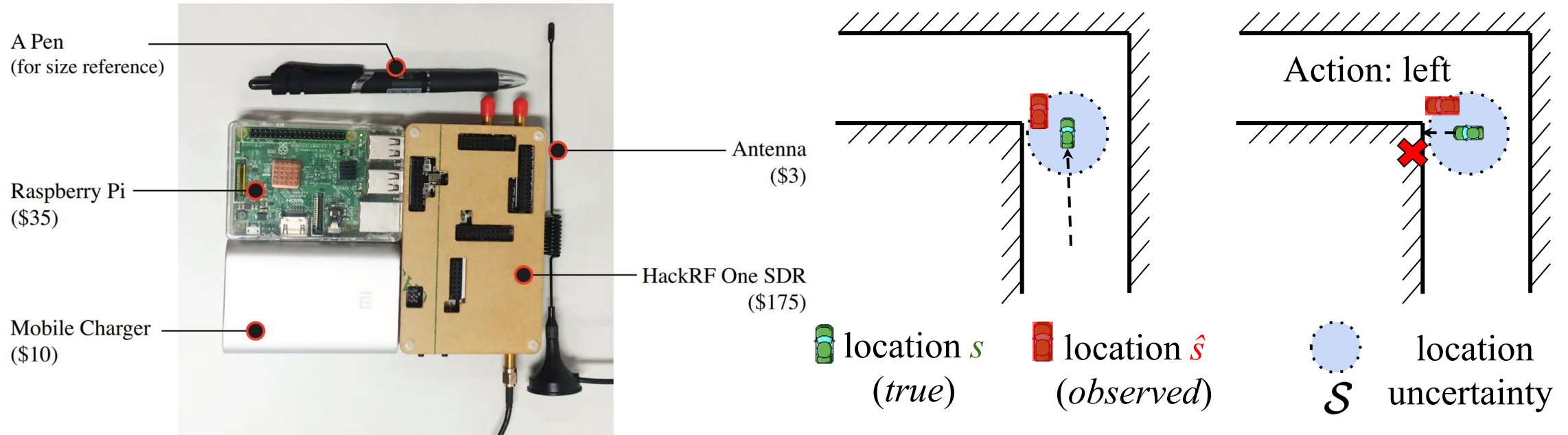
Adversary's goal: minimize reward while keeping attack stealthy



Attack Observations in DRL



Such Attack can be Realistic

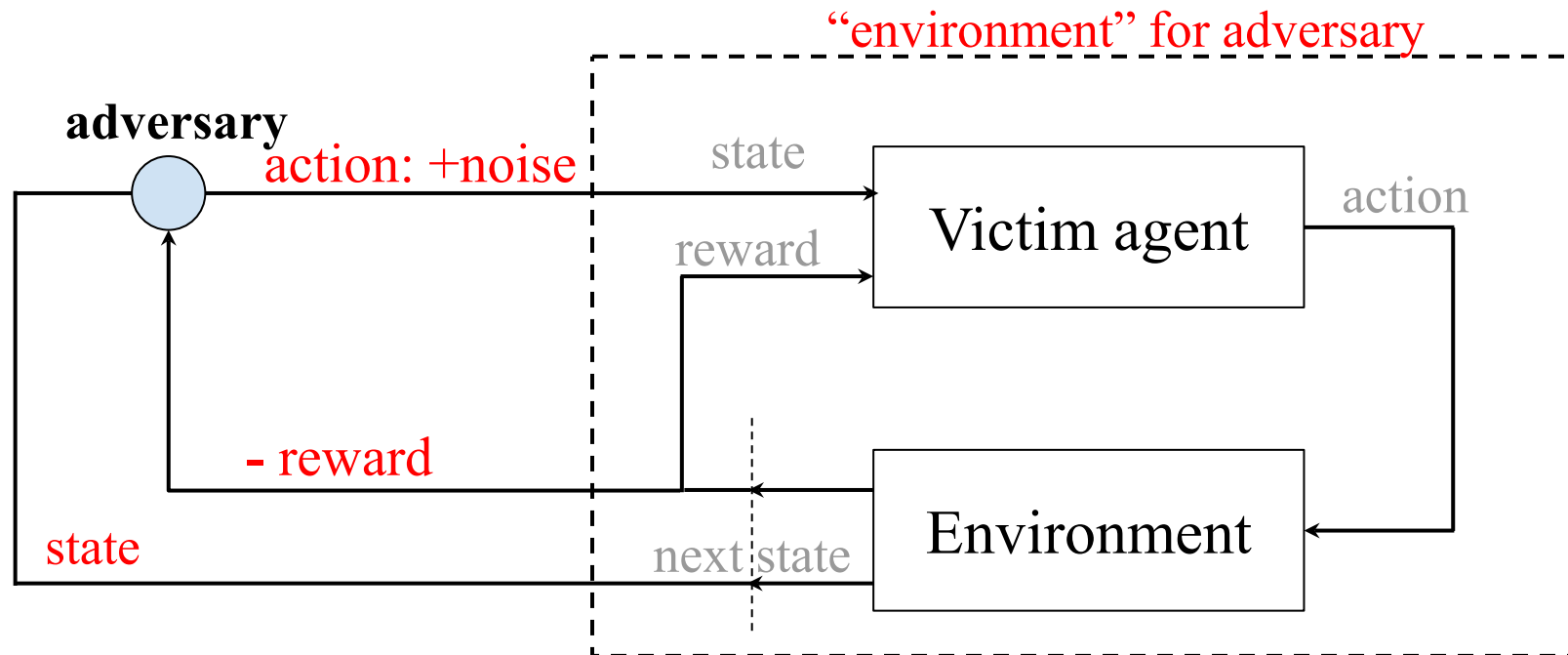


GPS spoofing
Zeng et al., USENIX
2018

Optimal Adversarial Attack

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

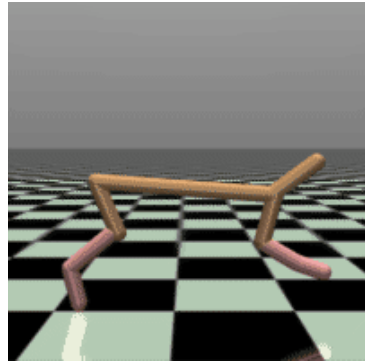
Solve the RL attack problem by learning an adversary using RL



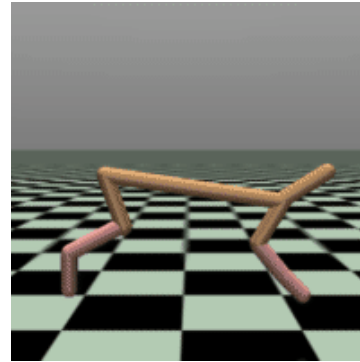
Optimal Attack is a Strong Adversary!

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

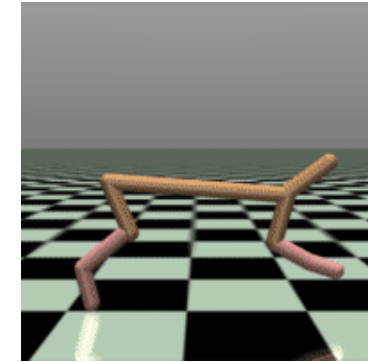
HalfCheetah



Episode
rewards:
+7094

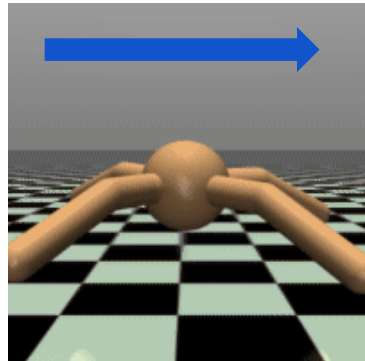


+85

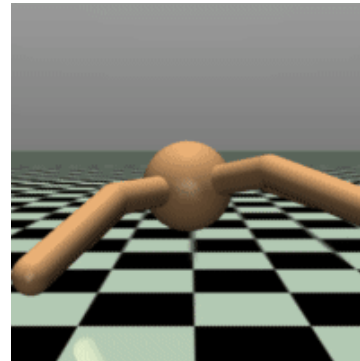


-743

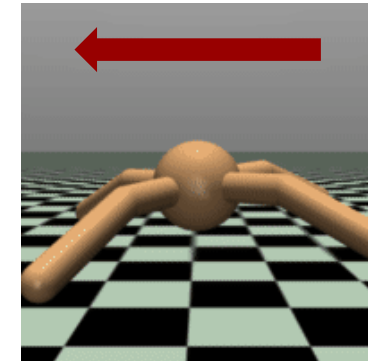
Ant



+5851



+63



-1141

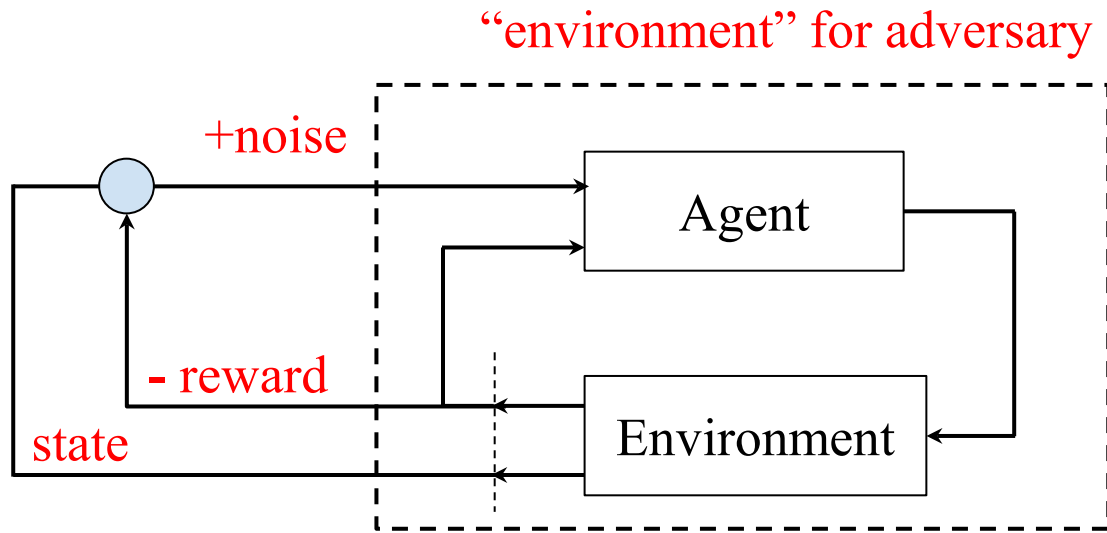
No attack

MAD attack

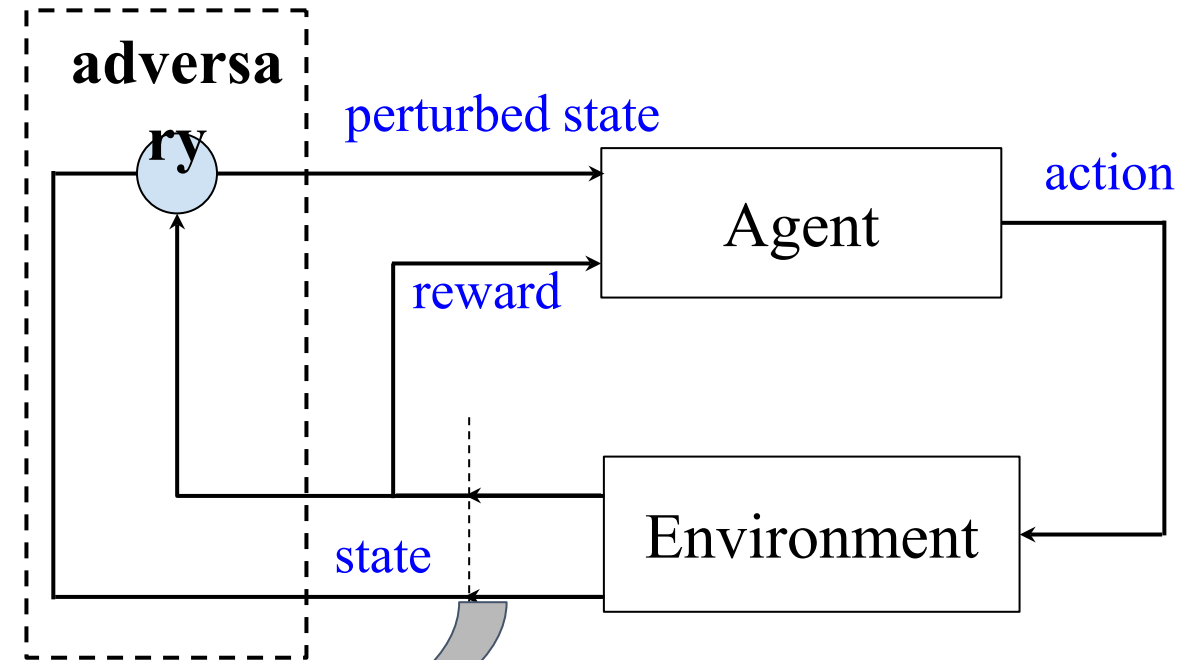
Optimal attack

Train robust deep reinforcement learning agents

Train adversary
(MDP)

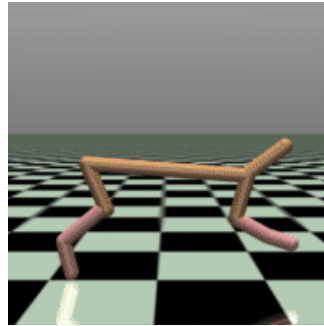


Train agent
(POMDP)



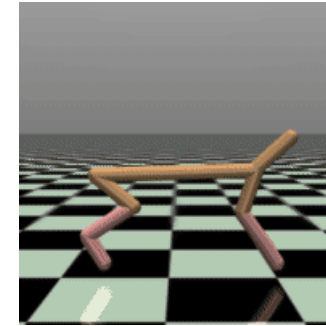
Robust Deep Reinforcement Learning Agents

HalfCheetah



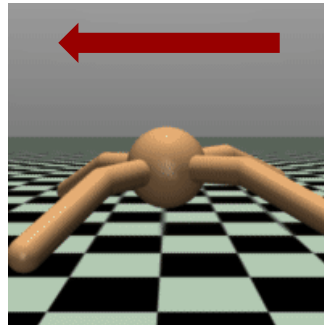
Episode rewards:

-743



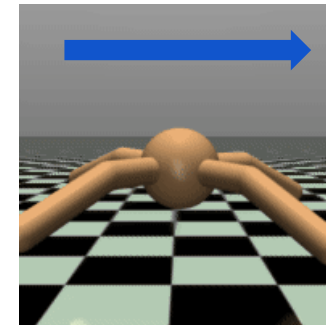
+5250

Ant



-1141

Optimal attack on
vanilla RL agents



+3835

Optimal attack on
robust RL agents