Most slides are adapted from AAAI 2022 tutorial

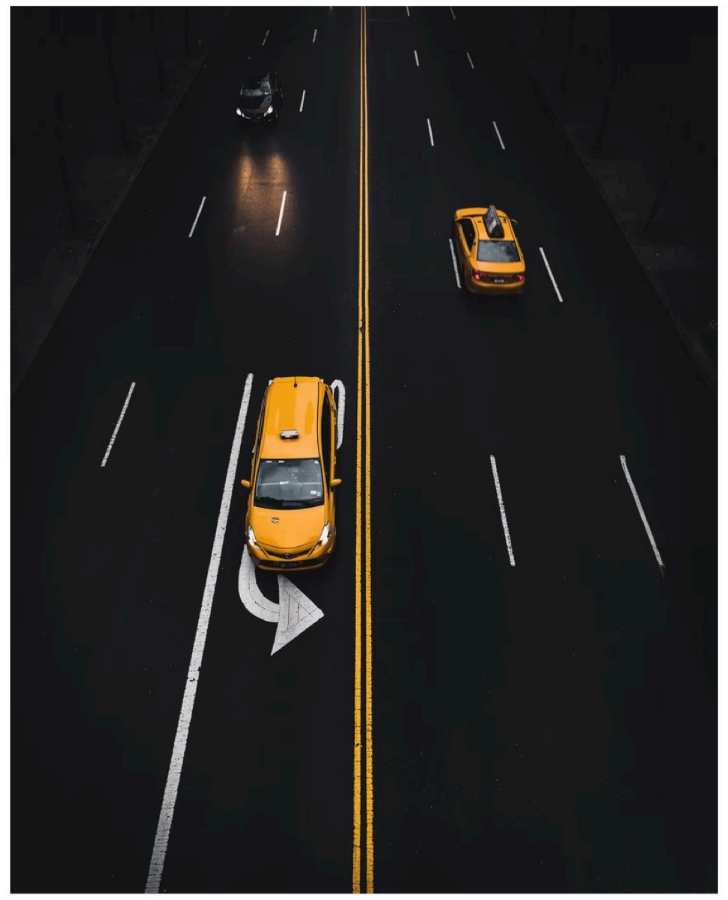
COMP6211I: Trustworthy Machine Learning

Test-time Integrity (verification)

Minhao CHENG

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

Can we trust NNs in safety-critical tasks?



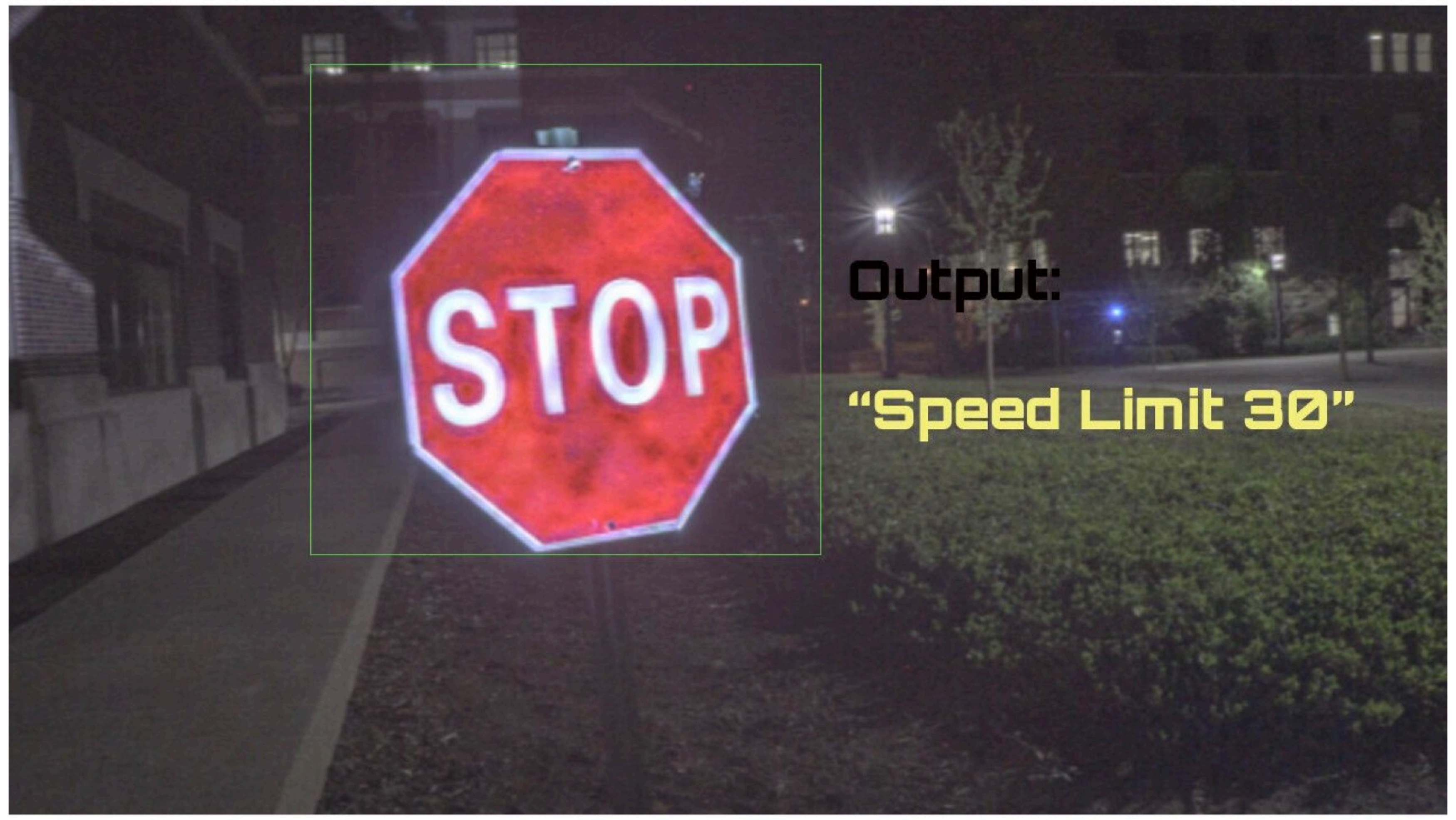
Autonomous Driving
Aircraft Autopiloting

Medical Equipments
AI-based Diagnosis

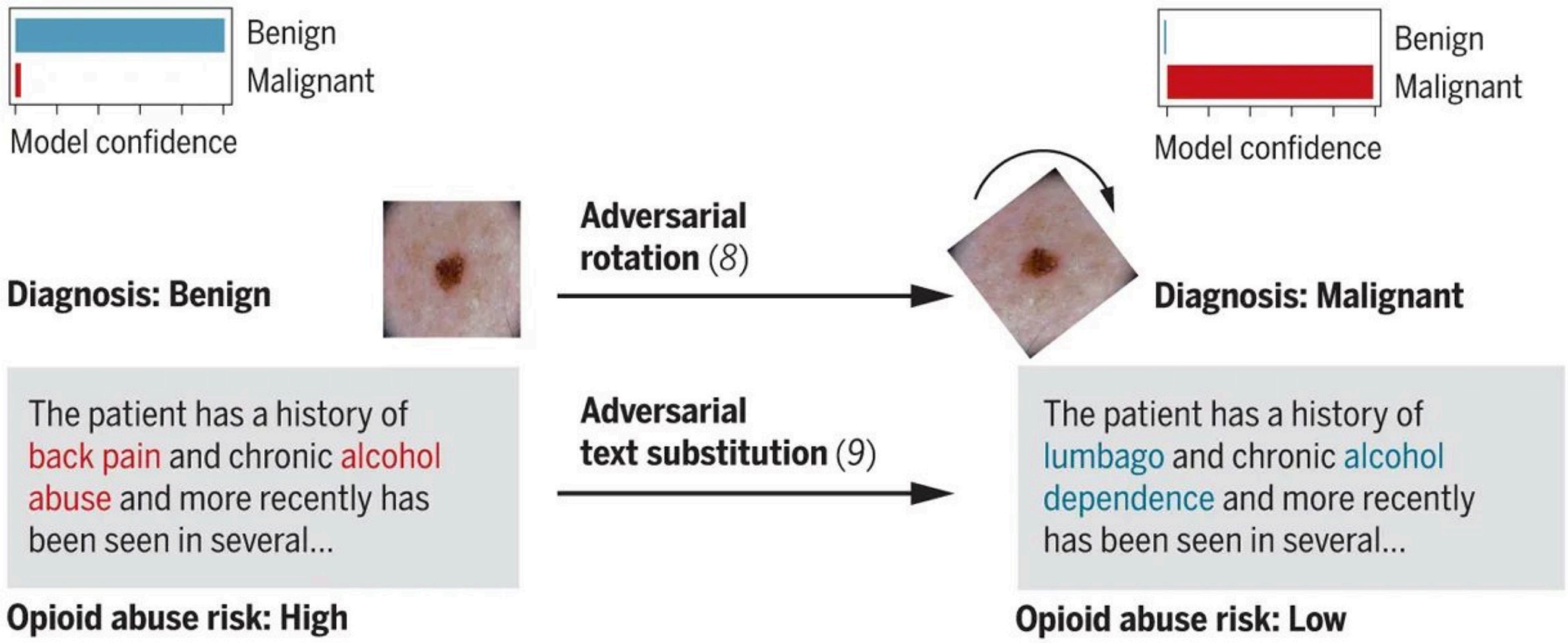
Security/Surveillance
Systems

Can we trust NNs in safety-critical tasks?

- No! As we have seen in train/test time integrity

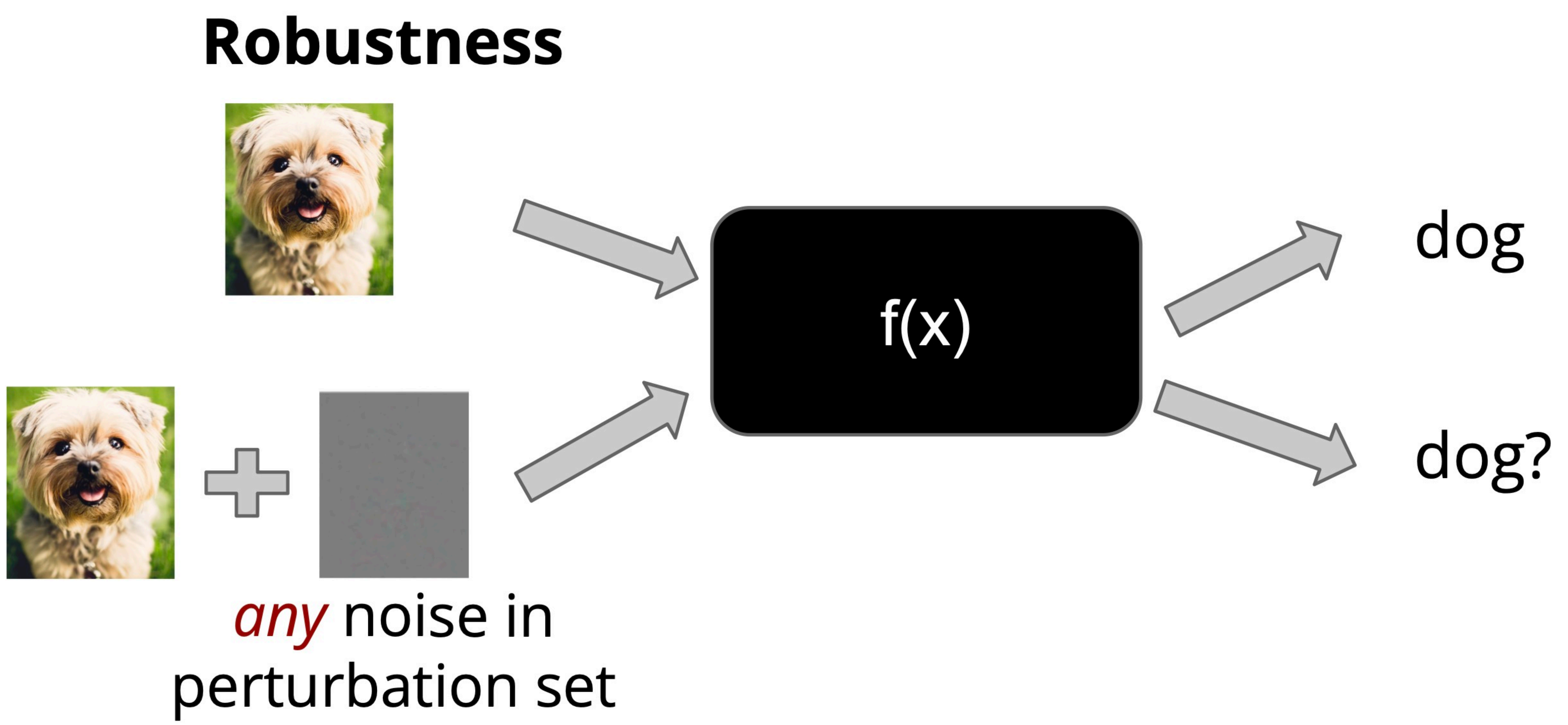


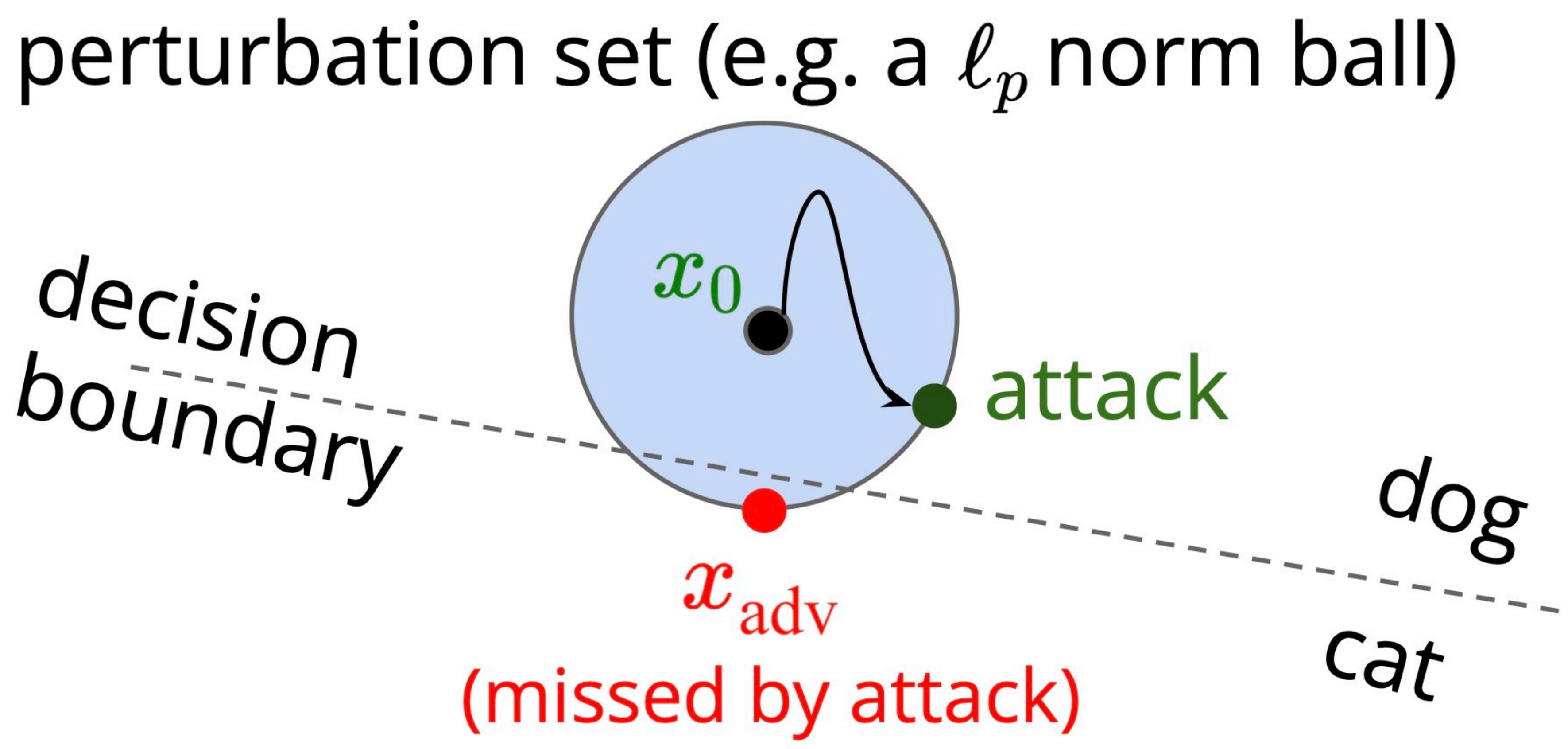
"Optical adversarial attack" by Gnanasambandam et al., ICCV 2021



"Adversarial attacks on medical machine learning" by N. Cary et al., Science

What is neural network verification?





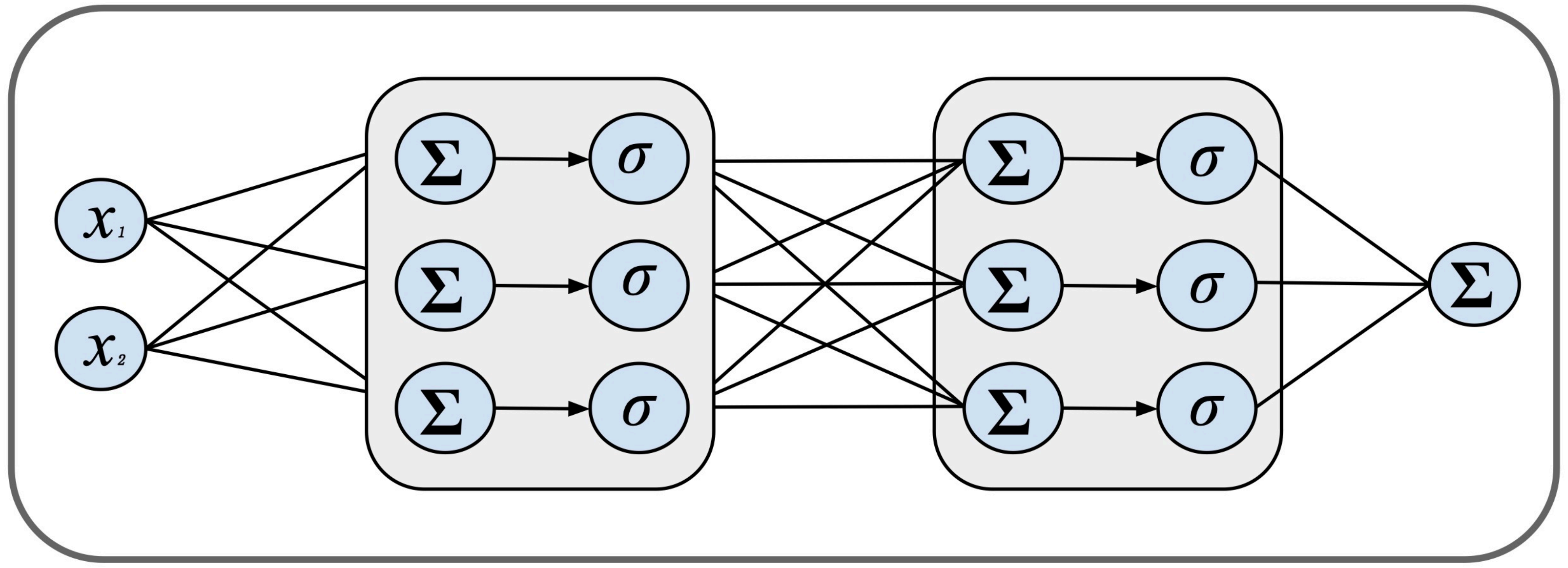
- Verification requires a formal proof to show the property holds
- In the robustness verification setting, a model can't be attack $\neq$ Verified
- Many heuristic defense was broken under stronger attacks
- A verified model cannot be attacked by any attacks (including unforeseen ones)

The Basic Formulation of Robustness Verification

- Consider a simple binary classification case:

x_0

Input is a **point**



Neural Network

$f(x_0) = 1.2$

Output is a score

$f(x_0) > 0$

Positive Example

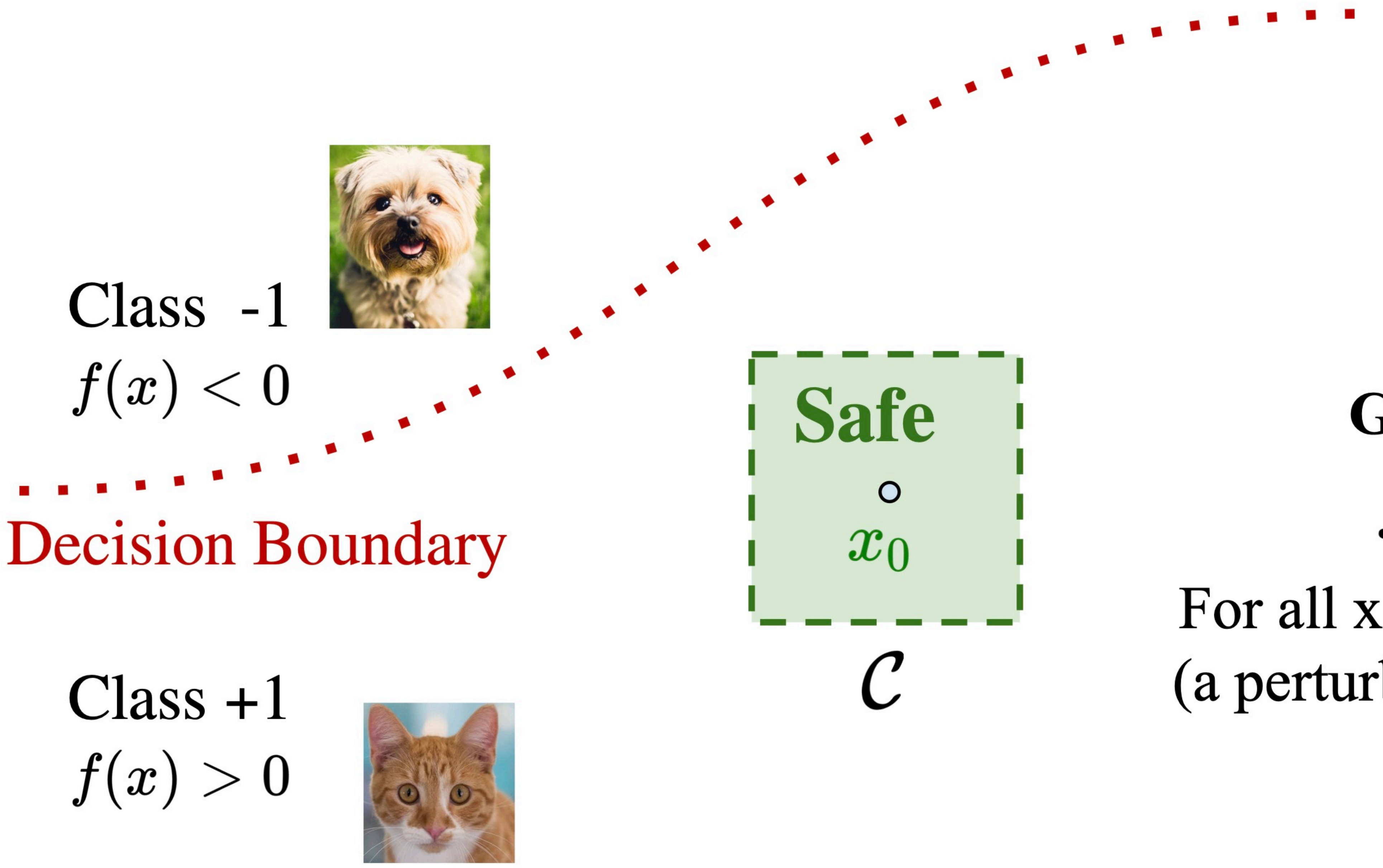
$f(x_0) \leq 0$

Negative Example

The Basic Formulation of Robustness Verification

Suppose $f(x_0) > 0$. Can we verify this property:

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



Goal: Prove

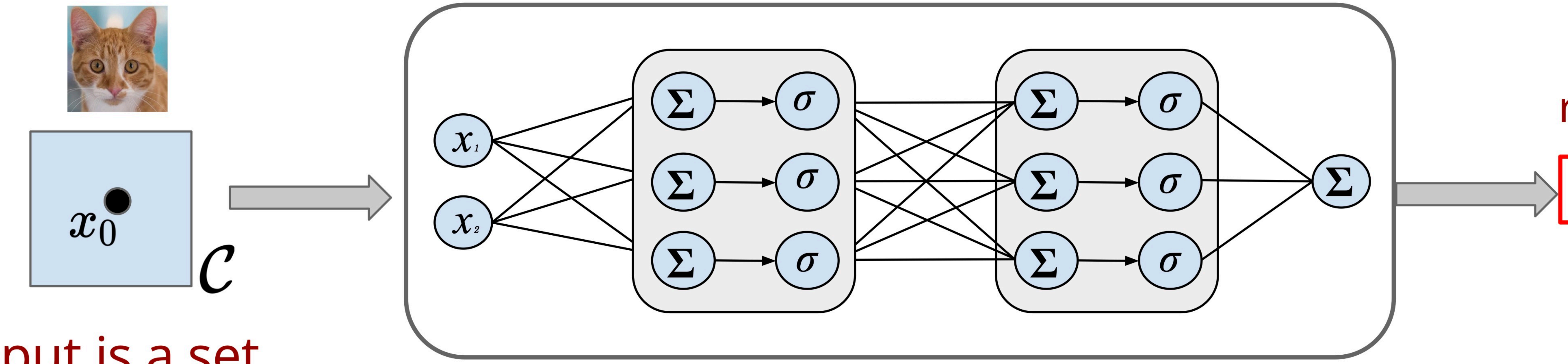
$f(x) > 0$

For all x in the green box
(a perturbation around x_0)

The Basic Formulation of Robustness Verification

Suppose $f(x_0) > 0$. Can we verify this property:

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



Input is a <u>set</u>

Neural Network

Output is a range/set of scores

$$\boxed{0.2} \leq f(x) \leq 2.2$$

"cat" even in the worst case

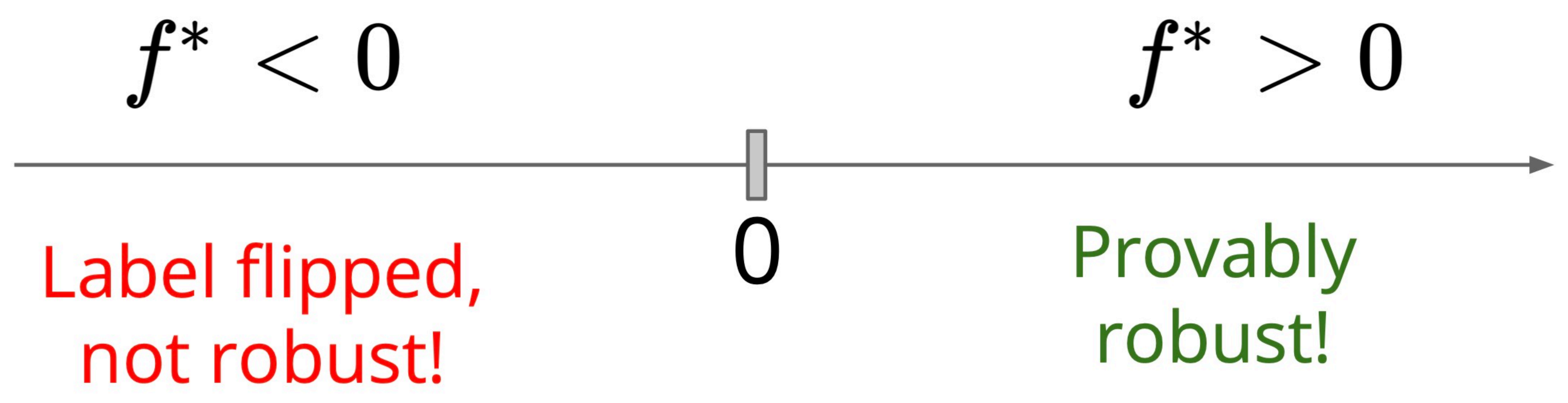
Must consider a set of infinite points as the input of the NN.

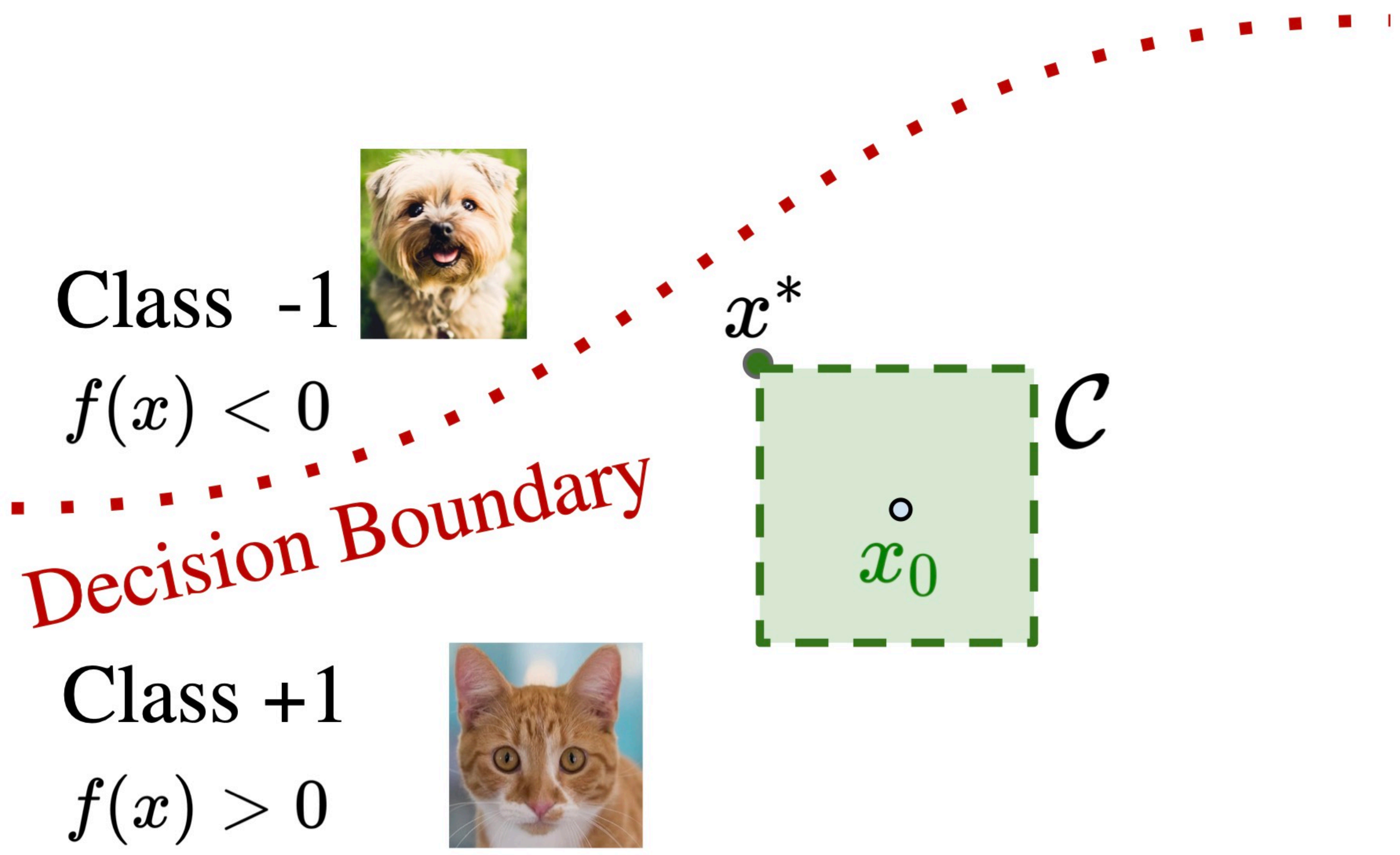
The Basic Formulation of Robustness Verification

Assuming $f(x_0) > 0$, we solve the optimization problem to find the worst case:

$$f^* = \min_{x \in \mathcal{C}} f(x)$$

$\mathcal{C}$ is usually a perturbation set "around" x_0, e.g., $\mathcal{C} := \{x \| \|x - x_0\|_p \le \epsilon\}$

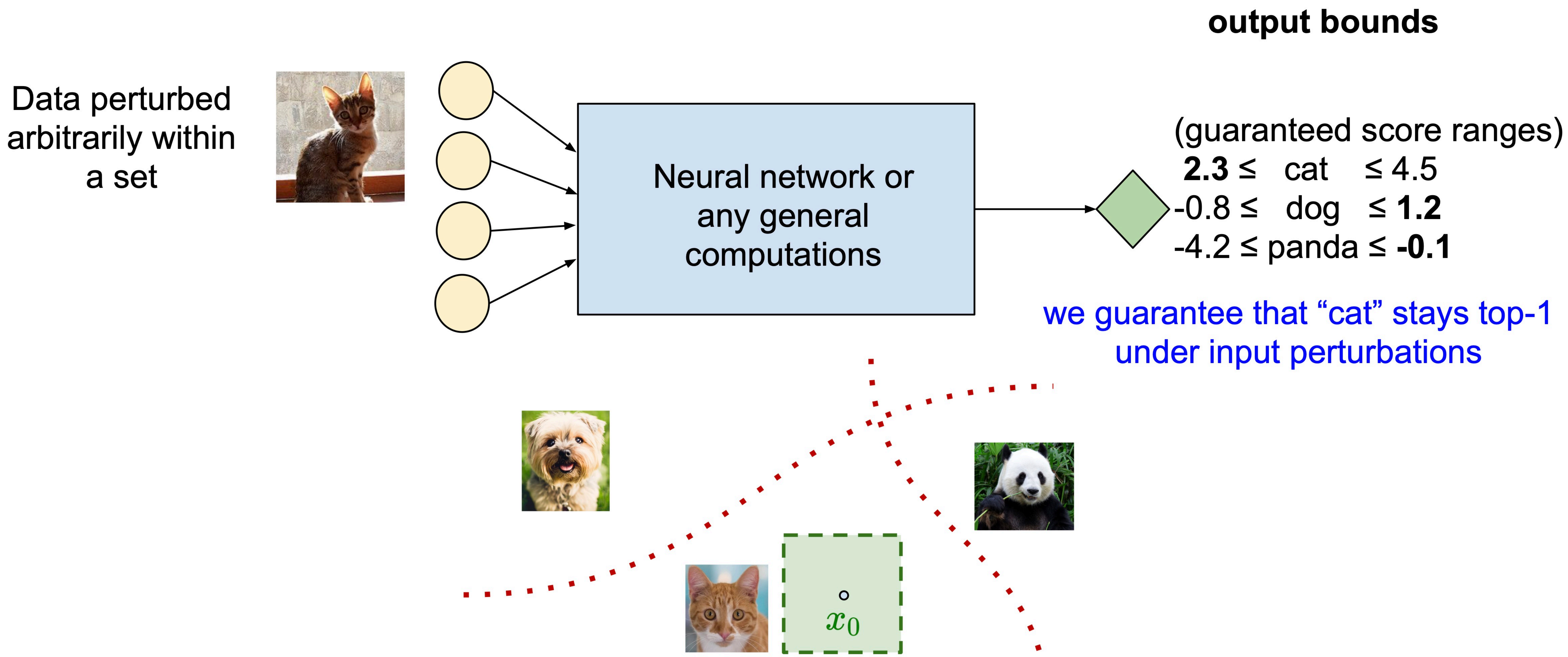




Is it a hard problem?

The Basic Formulation of Robustness Verification

Multi-class case:



Why the verification problem is challenging

This is the fundamental problem we want to solve (Wong & Kolter 2018, Salman et al. 2019):

$$f^* = \min z^{(L)}$$

Last layer output f(x), at layer L

pre-activation

$$\text{s.t.} \quad z^{(i)} = W^{(i)} \hat{z}^{(i-1)} + b^{(i)} \qquad i \in \{1, \cdots, L\}$$

Linear constraints

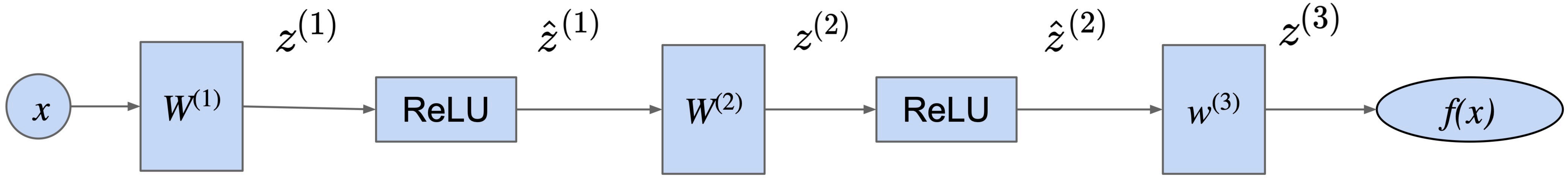
$$\hat{z}^{(i)} = \sigma(z^{(i)}) \quad i \in \{1, \cdots, L-1\}$$

Non-linear, non-convex constraints

post-activation

$$\hat{z}^{(0)} = x, \quad x \in \mathcal{C}$$

Input perturbations

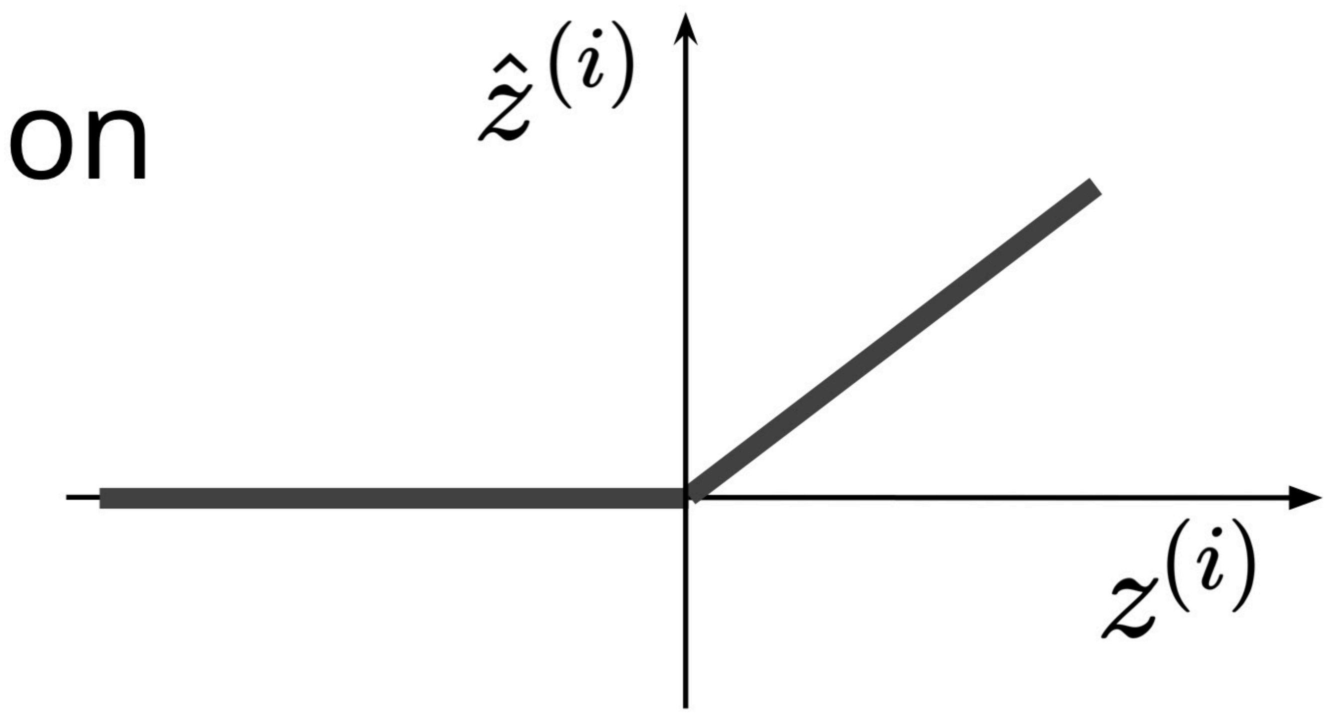


Why the verification problem is challenging

$$\hat{z}^{(i)} = \sigma(z^{(i)}), i \in \{1, \cdots, L-1\}$$

Non-convex constraints

e.g., ReLU function

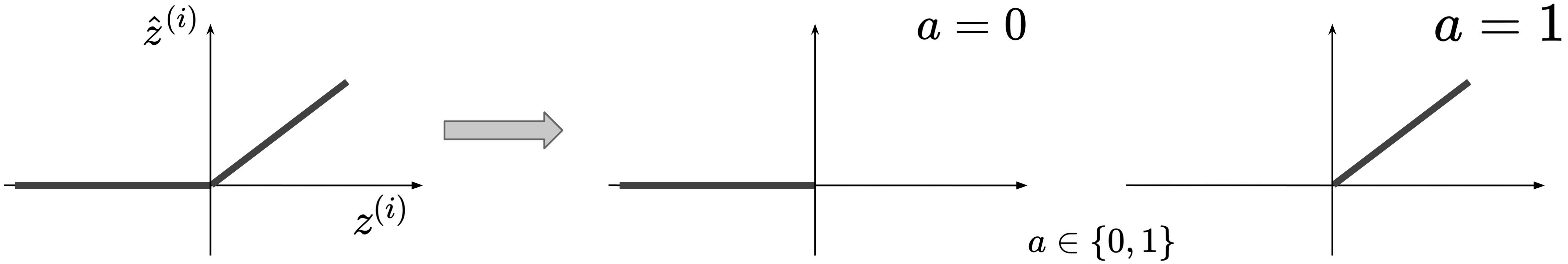


The constraint says that $(\hat{z}^{(i)}, z^{(i)}) \in \mathrm{Graph}(\mathrm{ReLU})$

Generally, NP-complete (Katz et al., 2017)

Why the verification problem is challenging

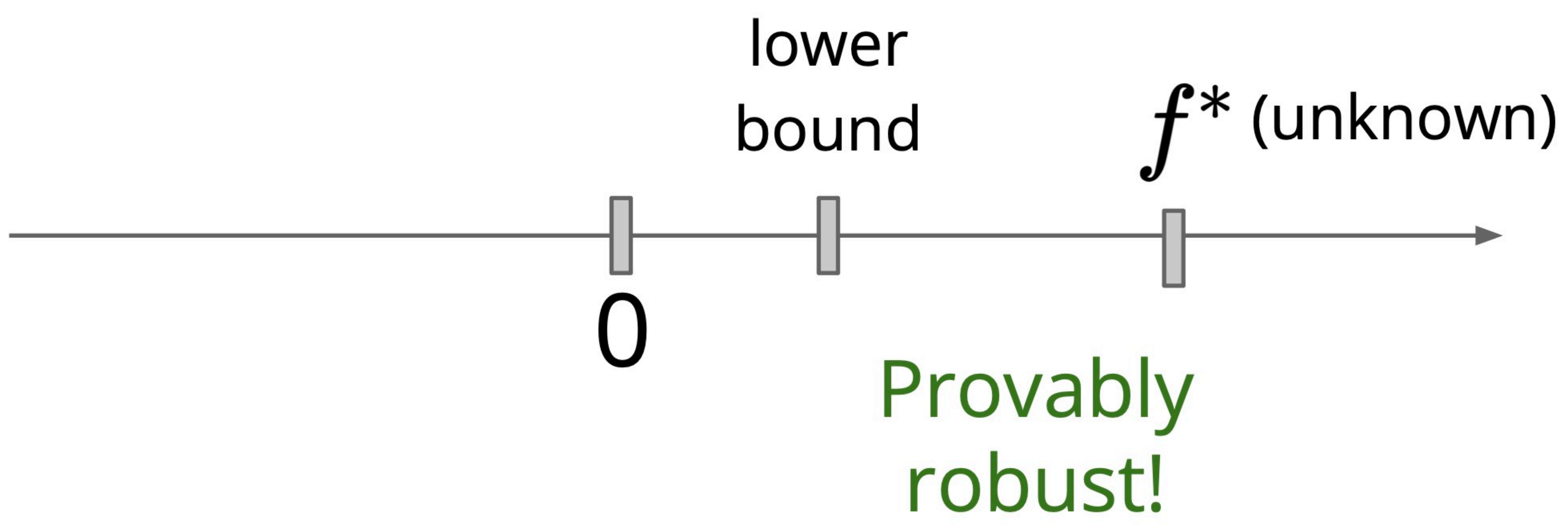
- Approach 1: Using mixed integer programming (MIP) encoding of ReLU neurons (Tjeng et al. 2017) => *Complete* verification which solves the exact f^*

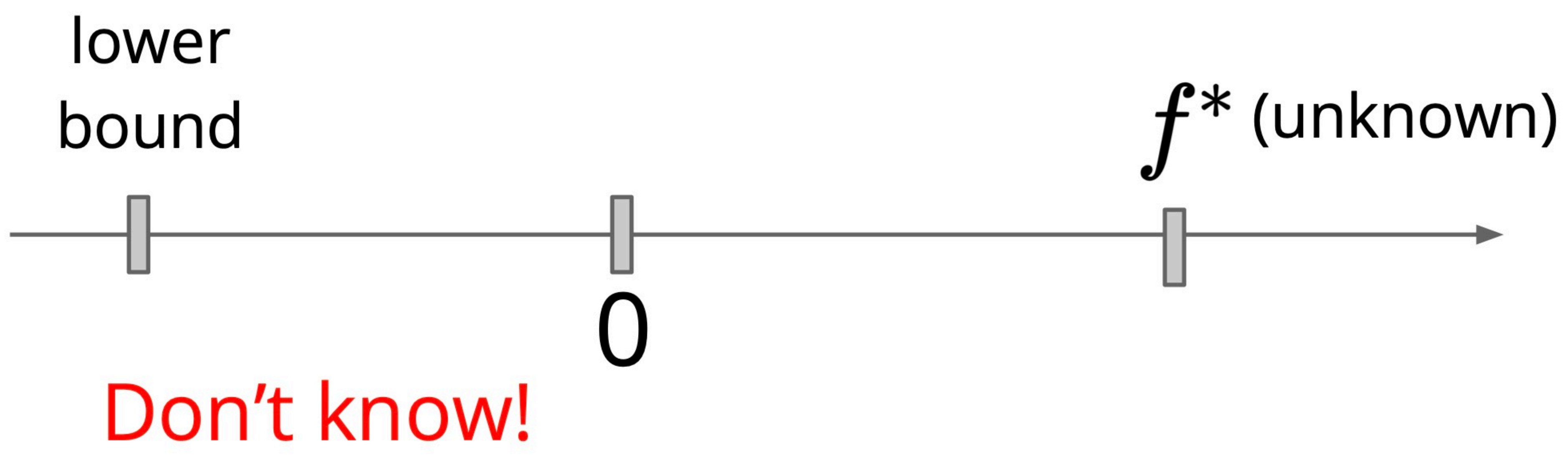


$\hat{z}^{(i)} = \mathrm{ReLU}(z^{(i)})$ $\qquad$ $\hat{z}^{(i)} = 0$ $\qquad$ $\hat{z}^{(i)} = z^{(i)}$

Why the verification problem is challenging

- Approach 2: Relax the MIP to a LP (Salman et al. 2019) => Incomplete verification: find a *lower bound* of f^*. If lower bound >0, the network is verifiably robust
  - Still requires an LP solver, which can still be slow for large networks
  - LP often produces loose bound; if lower bound << 0 it is useless



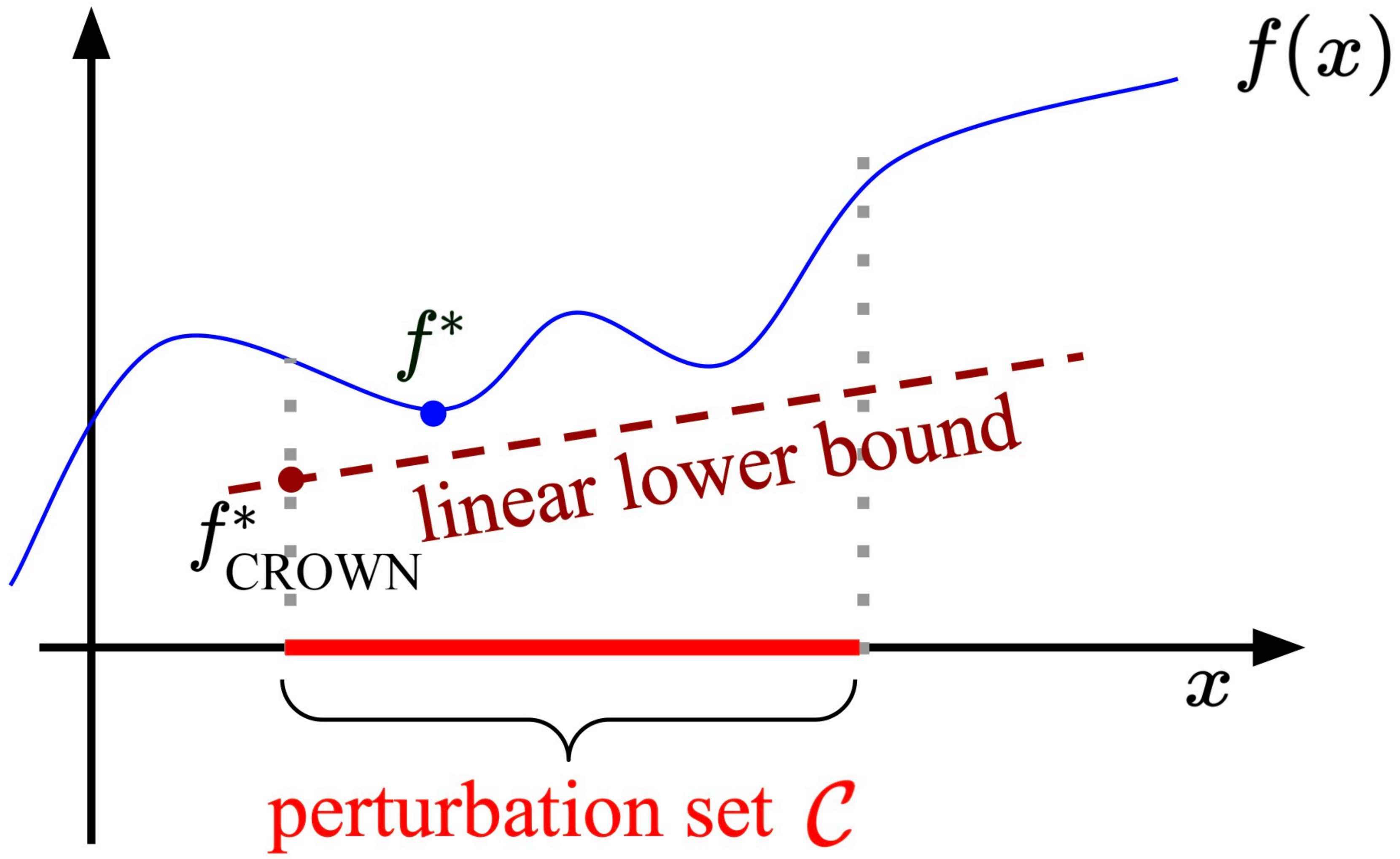


CROWN: Bound Propagation based Verification

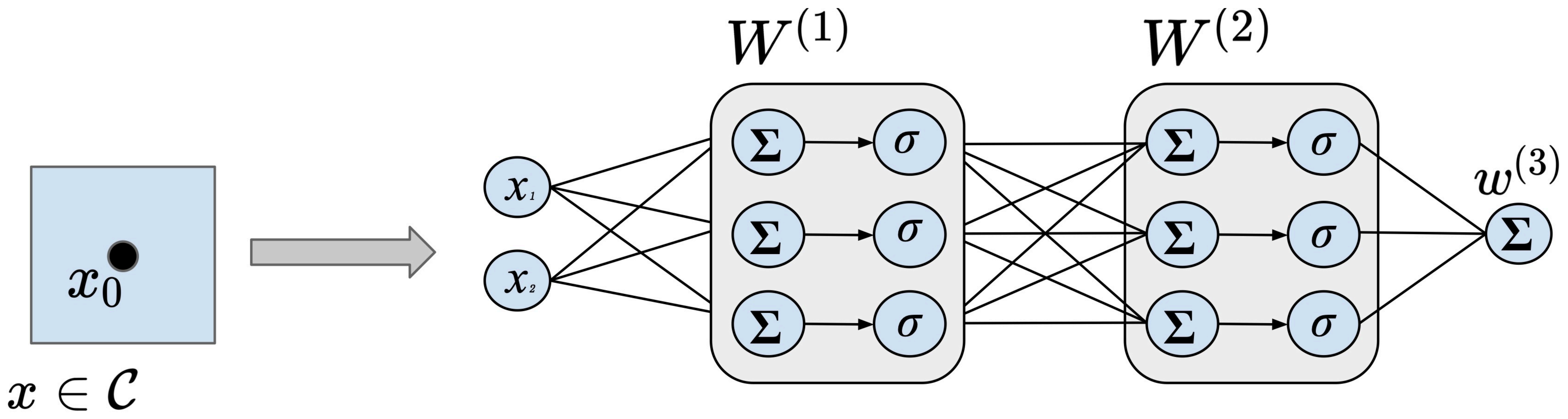
- We want to find a lower bound for this problem efficiently:

$$f^*_{\text{CROWN}} \leq f^* = \min_{x \in \mathcal{C}} f(x)$$

- $f^*_{\text{CROWN}} > 0 \Rightarrow f^* > 0$, so no adversarial example exists if $f^*_{\text{CROWN}} > 0$
- **CROWN** (Zhang et al. 2018) is an efficient linear **bound propagation** based algorithm to find linear lower/upper bounds of NNs
- Equivalent to **DeepPoly** (Singh et al., 2019), another popular verification algorithm



Find the lower bound on feed-forward networks



$$f^* = \min_{x \in \mathcal{C}} f(x)$$

- If there are no non-linear operations (e.g., ReLUs), all weights can be multiplied together

$$f(x) = w^{(3)\top} W^{(2)} W^{(1)} x = a^\top x$$

- Bounds for linear functions are easy (e.g., Hölder's inequality for Lp norm)

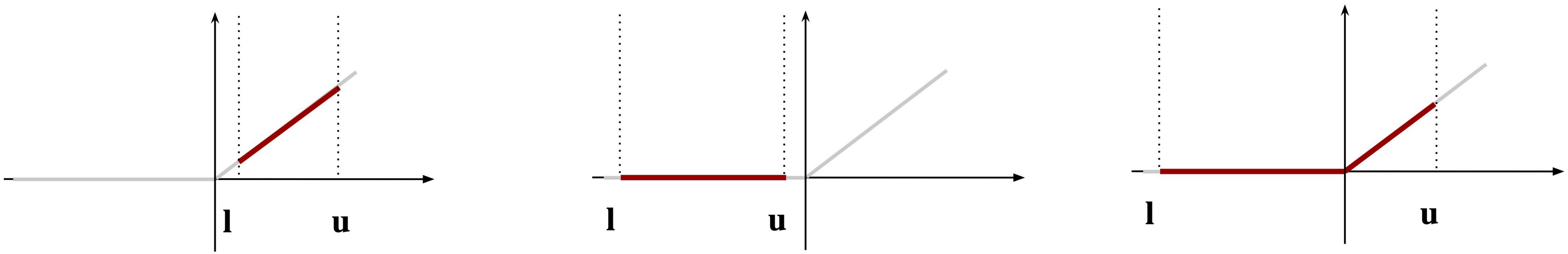
$$f^* := -\epsilon \|a\|_1 + a^\top x_0 \qquad x \in \{x \mid \|x - x_0\|_\infty \leq \epsilon\}$$

How to convert ReLU into a linear function

$$f(x) = w^{(3)}\mathrm{ReLU}(W^{(2)}\mathrm{ReLU}(W^{(1)}x))$$

$$\mathrm{ReLU}(z) = \max(0, z)$$

ReLU neurons have three cases depending on bounds on their inputs:



$\mathbf{l} \geq 0$, always active (linear)

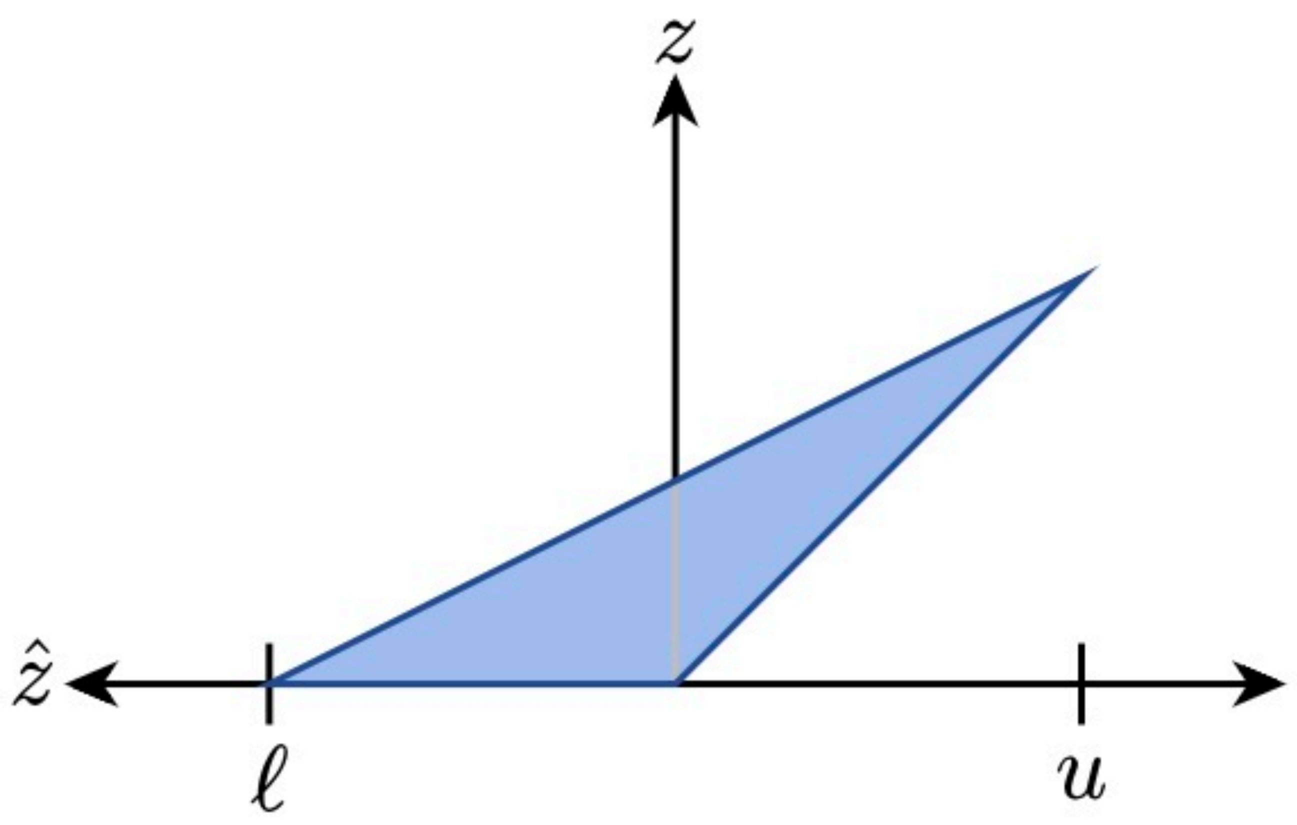
$\mathbf{u} \leq 0$, Always inactive (zero)

$\mathbf{l} \leq 0 \leq \mathbf{u}$
Unstable (non-linear)
Must be relaxed

$\mathbf{l}$ and $\mathbf{u}$ are pre-activation bounds (also called intermediate layer bounds)

Convex envelope

1. If $\ell < 0 < u$, then will take the convex envelope of the ReLU between ℓ and u. Specifically, this is the triangular region formed by the points $(\ell, 0)$, (u, u), and $(0, 0)$. We can express this region as a set of three inequalities: the region below the line connecting $(\ell, 0)$ and (u, u), above the line $z = 0$, and above the line $z = \hat{z}$. Then, we can replace the ReLU activation with the convex set $\mathcal{C}(\ell, u)$ defined by these inequalities: $\mathcal{C}(\ell, u) = \{(\hat{z}, z) : -u\ell \geq z(u - \ell) - u\hat{z},\ z \geq 0,\ z \geq \hat{z}\}$

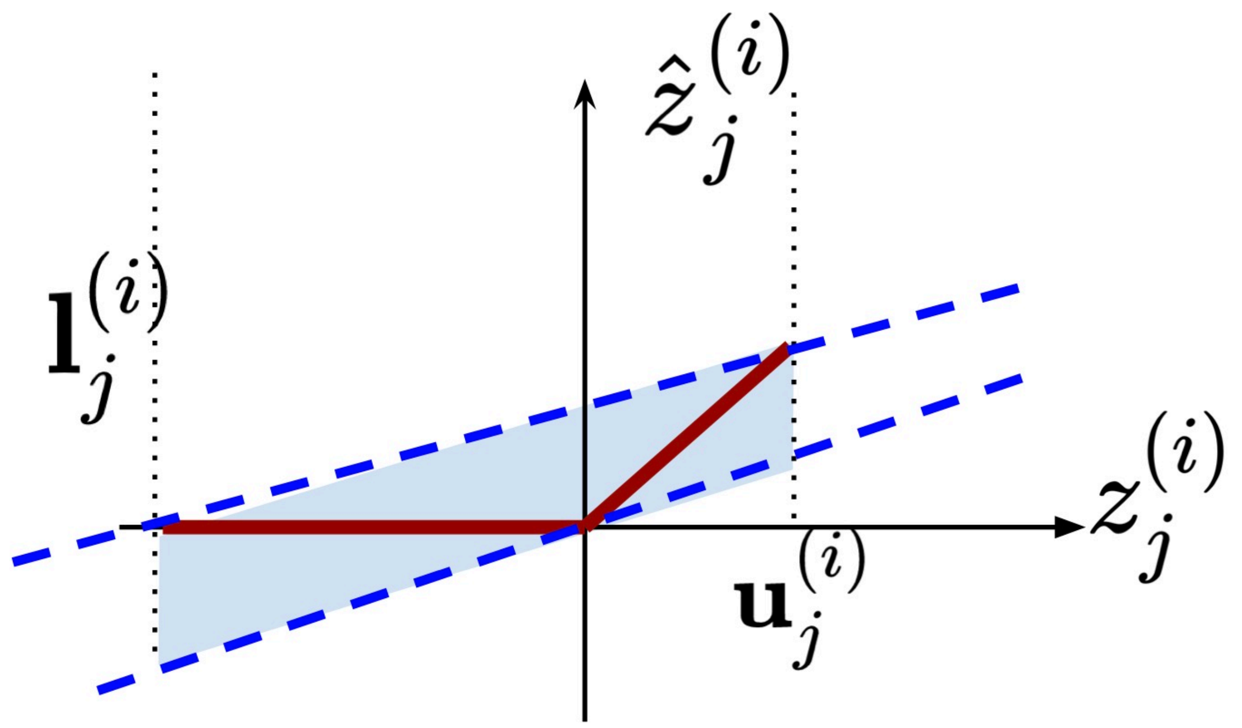


Convex ReLU set

How to convert ReLU into a linear function

For the j-th ReLU neuron in layer i: $\hat{z}_j^{(i)} = \mathrm{ReLU}\left(z_j^{(i)}\right)$

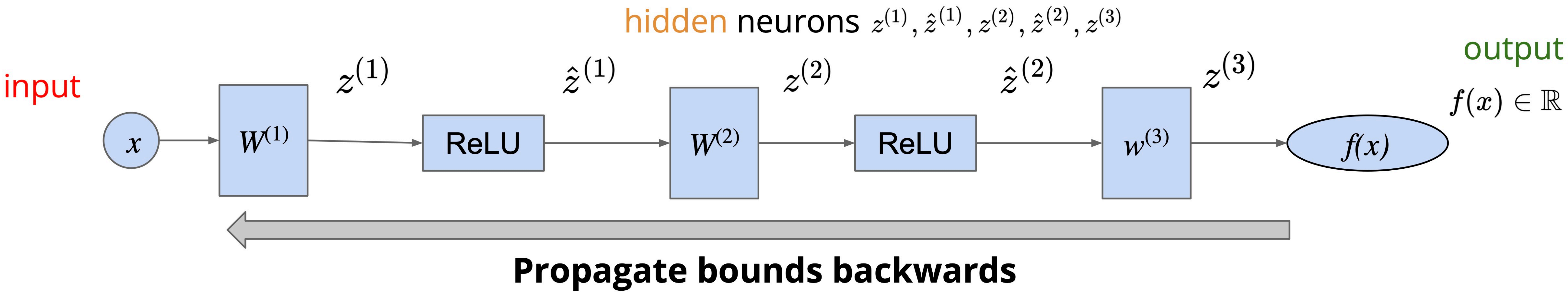
Assuming its input is bounded: $\mathbf{l}_j^{(i)} \leq z^{(i)} \leq \mathbf{u}_j^{(i)}$ and unstable: $\mathbf{l}_j^{(i)} \leq 0 \leq \mathbf{u}_j^{(i)}$



- Idea: use two linear bounds to replace ReLU, to obtain linear bounds for the entire network
- Can also be extended to non-ReLU functions (e.g., tanh, maxpool).

CROWN backward bound propagation

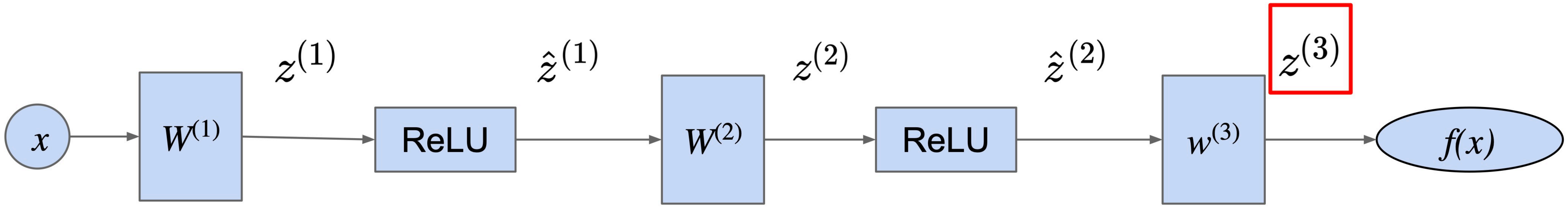
- In CROWN, we **propagate a linear lower bound** for output neuron w.r.t. hidden or input neuron.



- $W^{(1)}, W^{(2)}, w^{(3)}$ are weights of the NN (output dimension is 1 so $w^{(3)}$ is an vector)
- **Goal**: get a lower bound for $f(x), x \in \mathcal{C}$

CROWN backward bound propagation

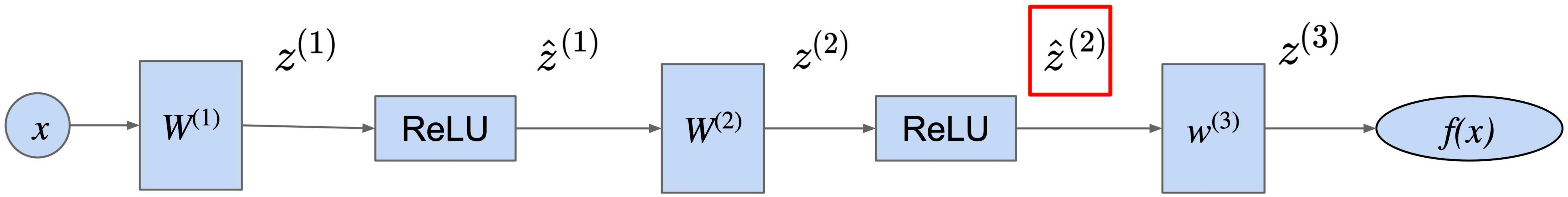
- Goal: find linear relationships between output and every hidden neuron



$$f(x) = z^{(3)}$$

CROWN backward bound propagation

- Goal: find linear relationships between output and every hidden neuron

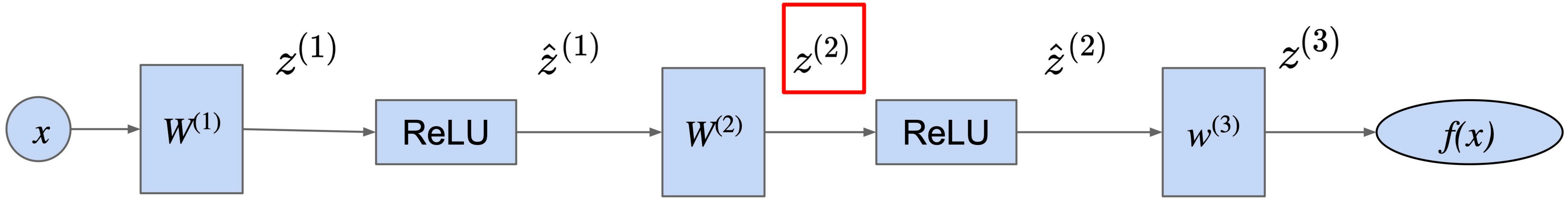


$$f(x) = w^{(3)\top} \hat{z}^{(2)}$$

(By definition)

CROWN backward bound propagation

- Goal: find linear relationships between output and every hidden neuron



$$f(x) \geq w^{(3)\top} D^{(2)} z^{(2)} + \text{const.}$$

Encountered an nonlinear operation, need to maintain this inequality.

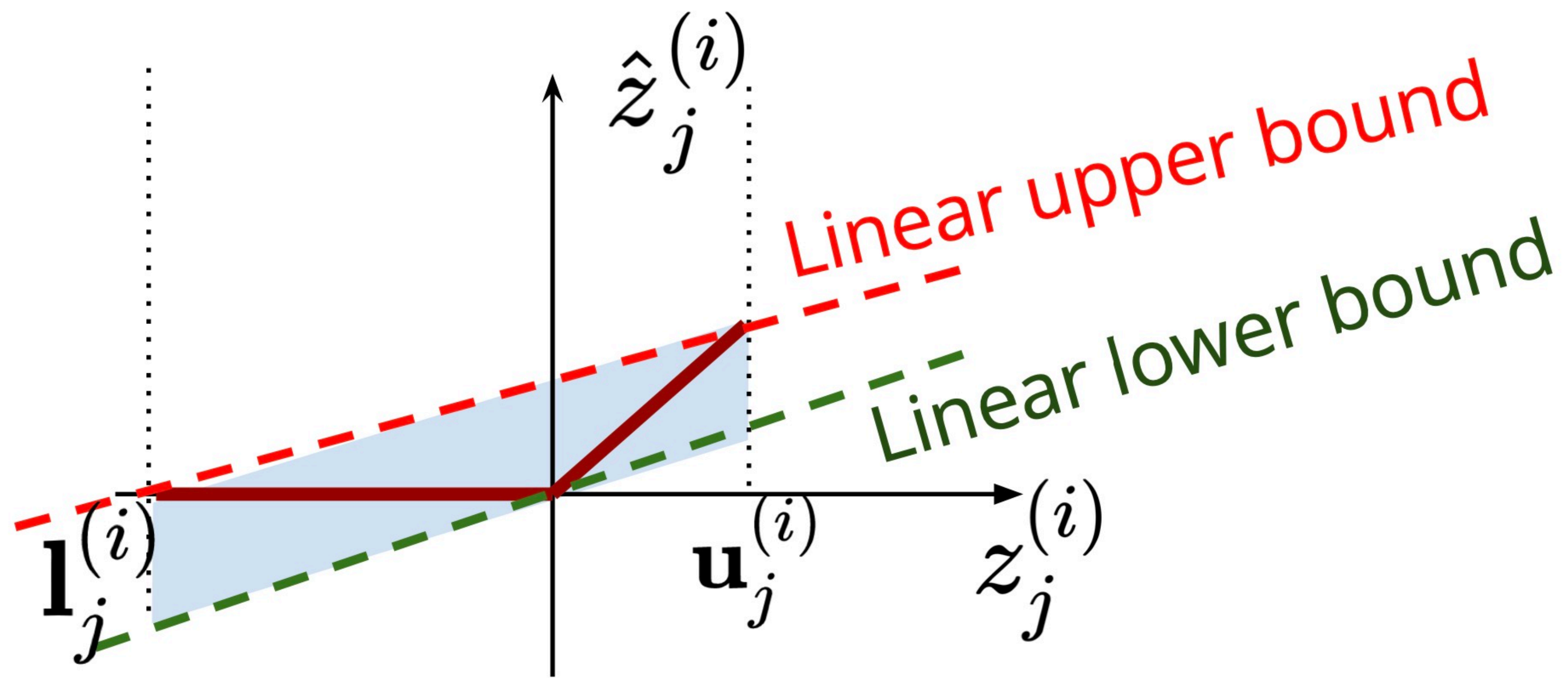
A diagonal matrix $D^{(2)}$ reflects the relaxation of ReLU neurons will be used.

Relaxation during bound propagation

- How to design $D^{(2)}$ so the lower and upper bounds are maintained?
- **First step**: for **each unstable** ReLU neuron, linearly lower and upper bound the non-linear function

pre-activation bounds

$$\mathbf{l}_j^{(i)} \leq z_j^{(i)} \leq \mathbf{u}_j^{(i)}$$



$$\boxed{\underline{a}_j^{(i)} z_j^{(i)} + \underline{b}_j^{(i)}} \leq \hat{z}_j^{(i)} := \mathrm{ReLU}(z_j^{(i)}) \leq \boxed{\overline{a}_j^{(i)} z_j^{(i)} + \overline{b}_j^{(i)}}$$

Relaxation during bound propagation

- **Second step**: Take the lower or upper bound based on the worst-case

Goal: lower bound $f(x) := w^{(3)\top} \mathrm{ReLU}(z^{(2)}) := w^{(3)\top} \hat{z}^{(2)} = \sum_j \boxed{w_j^{(3)} \cdot \hat{z}_j^{(2)}}$

lower bound each term!

- Take the lower bound of $\hat{z}_j^{(2)}$ when $w_j^{(3)}$ is positive
- Take the upper bound of $\hat{z}_j^{(2)}$ when $w_j^{(3)}$ is negative

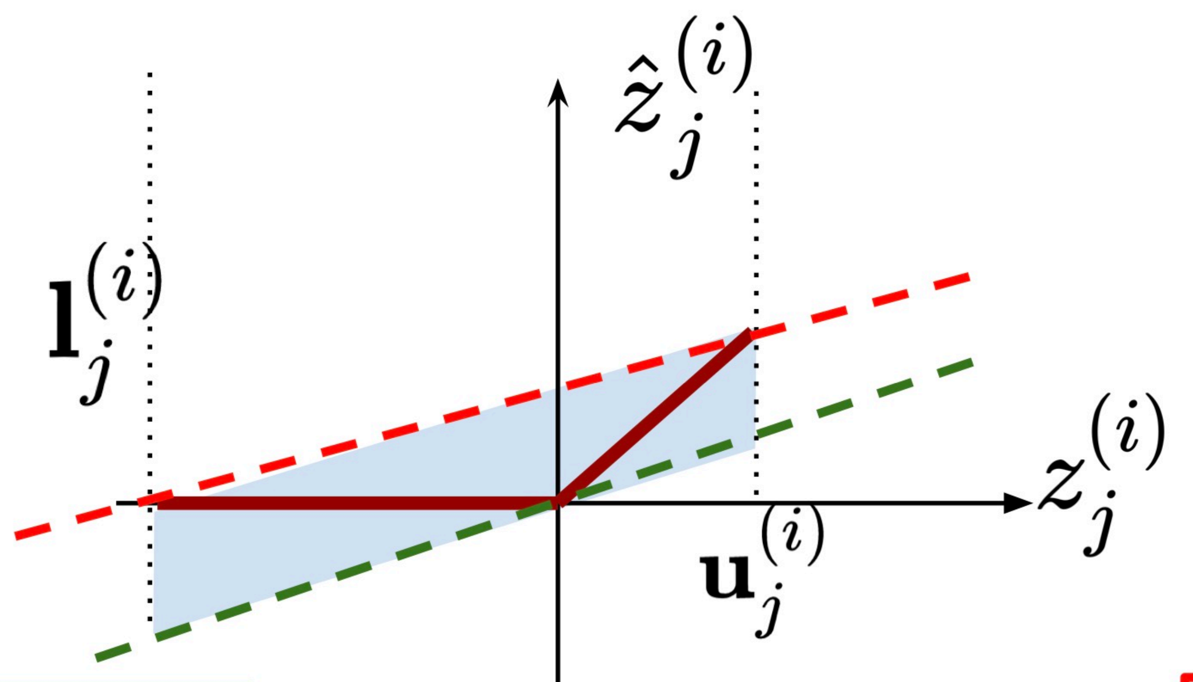
$$\sum_j w_j^{(3)} \cdot \hat{z}_j^{(2)} \geq \sum_{j, w_j^{(3)} \geq 0} w_j^{(3)} \cdot \text{lower bound of } \hat{z}_j^{(2)} + \sum_{j, w_j^{(3)} < 0} w_j^{(3)} \cdot \text{upper bound of } \hat{z}_j^{(2)}$$

Relaxation during bound propagation

- **Second step**: Take the lower or upper bound based on the worst-case

Goal: lower bound $\sum_j w_j^{(3)} \cdot \hat{z}_j^{(2)} \geq \sum_{j, w_j^{(3)} \geq 0} w_j^{(3)} \cdot \boxed{\text{lower bound of } \hat{z}_j^{(2)}} + \sum_{j, w_j^{(3)} < 0} w_j^{(3)} \cdot \boxed{\text{upper bound of } \hat{z}_j^{(2)}}$

replace with linear bounds



$$\boxed{\underline{a}_j^{(i)} z_j^{(i)} + \underline{b}_j^{(i)}} \leq \hat{z}_j^{(i)} := \mathrm{ReLU}(z_j^{(i)}) \leq \boxed{\overline{a}_j^{(i)} z_j^{(i)} + \overline{b}_j^{(i)}}$$

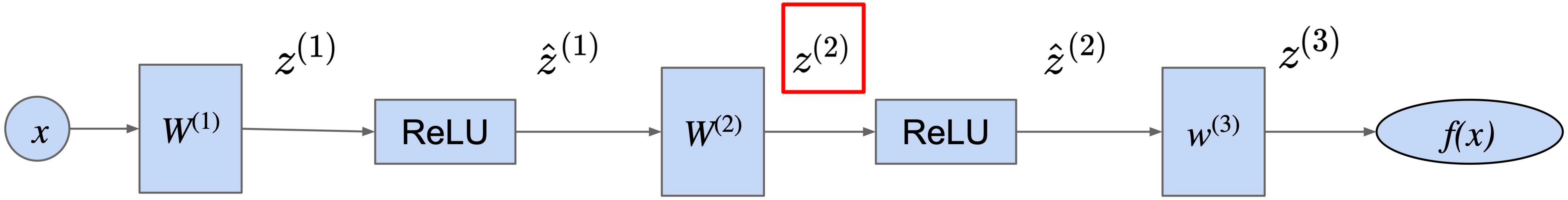
Rearrange (ignore bias terms):

$$w^{(3)\top} \hat{z}^{(2)} \geq w^{(3)\top} D^{(2)} z^{(2)} + \text{bias}$$

Diagonal matrix $D_{j,j}^{(2)} = \begin{cases} \underline{a}_j^{(2)}, w_j^{(3)} \geq 0 \\ \overline{a}_j^{(2)}, w_j^{(3)} < 0 \end{cases}$

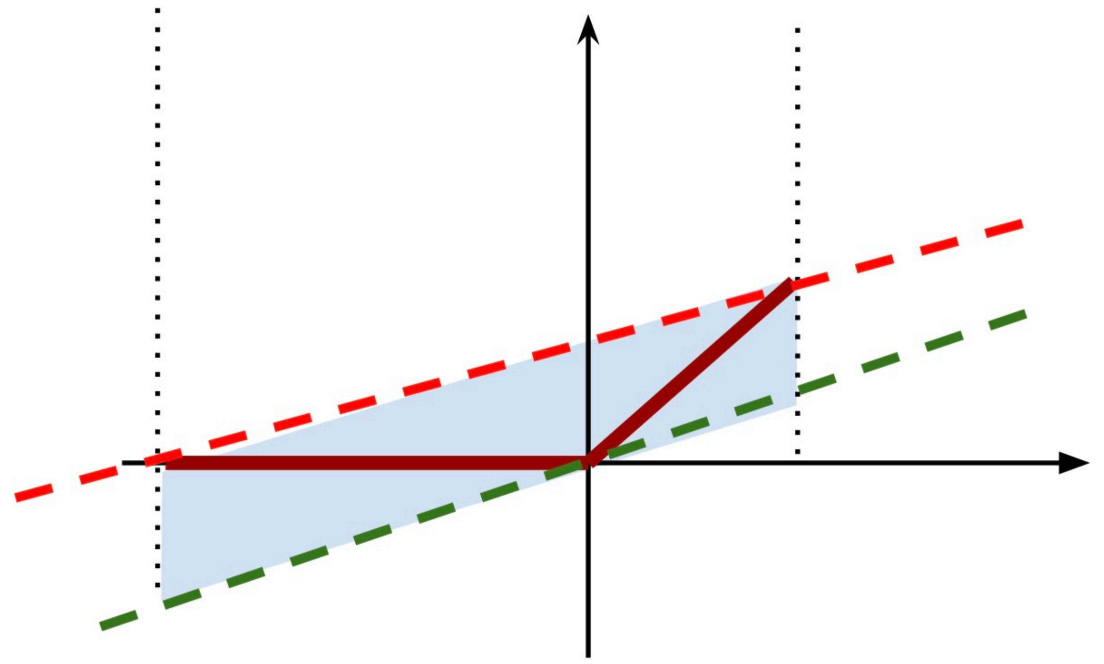
CROWN backward bound propagation

- Goal: find linear relationships between output and every hidden neuron



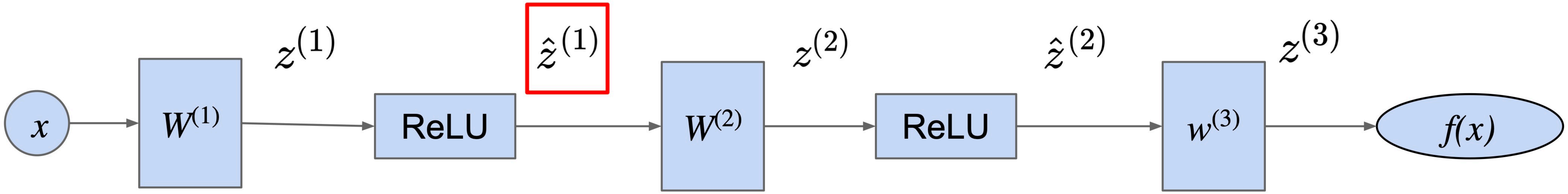
$$f(x) \geq w^{(3)\top} D^{(2)} z^{(2)} + \text{const.}$$

$D^{(2)}$ depends on the signs in $w^{(3)}$, and the linear relaxation of ReLU neuron to make the inequality hold



CROWN backward bound propagation

- Goal: find linear relationships between output and every hidden neuron



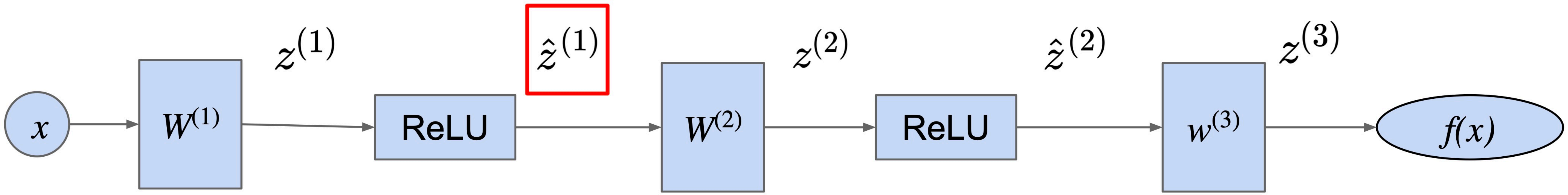
$$f(x) \geq w^{(3)\top} D^{(2)} \boxed{z^{(2)}} + \text{const.}$$

By definition $z^{(2)} = W^{(2)} \hat{z}^{(1)}$

The rest layers follow the same way of propagating bounds

CROWN backward bound propagation

- Goal: find linear relationships between output and every hidden neuron

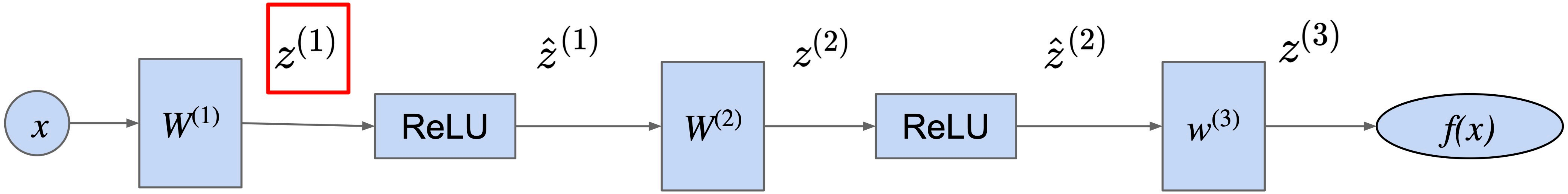


$$f(x) \geq w^{(3)\top} D^{(2)} W^{(2)} \hat{z}^{(1)} + \text{const.}$$

The rest layers follow the same way of propagating bounds

CROWN backward bound propagation

- Goal: find linear relationships between output and every hidden neuron



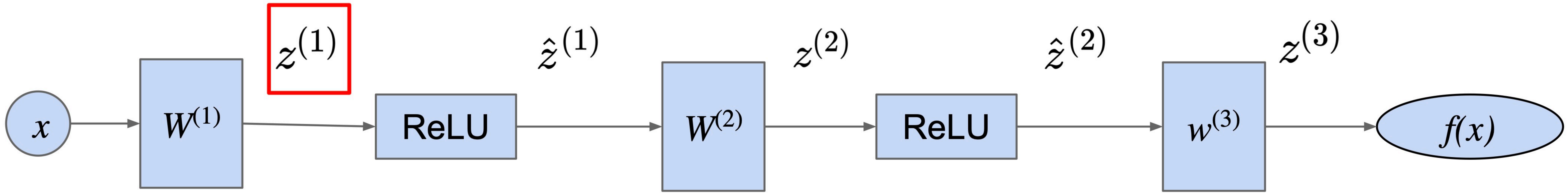
$$f(x) \geq w^{(3)\top} D^{(2)} W^{(2)} \boxed{D^{(1)}} z^{(1)} + \text{const.}$$

Based on the linear relaxation
of ReLU

The rest layers follow the same
way of propagating bounds

CROWN backward bound propagation

- Goal: find linear relationships between output and every hidden neuron



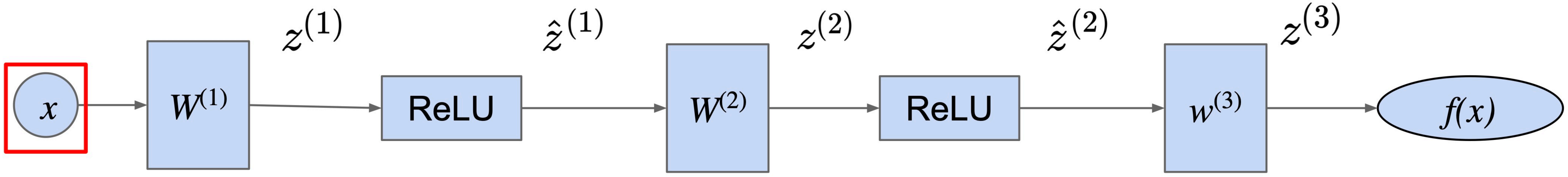
$$f(x) \geq w^{(3)\top} D^{(2)} W^{(2)} D^{(1)} \boxed{z^{(1)}} + \text{const.}$$

By definition $z^{(1)} = W^{(1)} x$

The rest layers follow the same way of propagating bounds

CROWN backward bound propagation

- Goal: find linear relationships between output and every hidden neuron, until we reach the input!

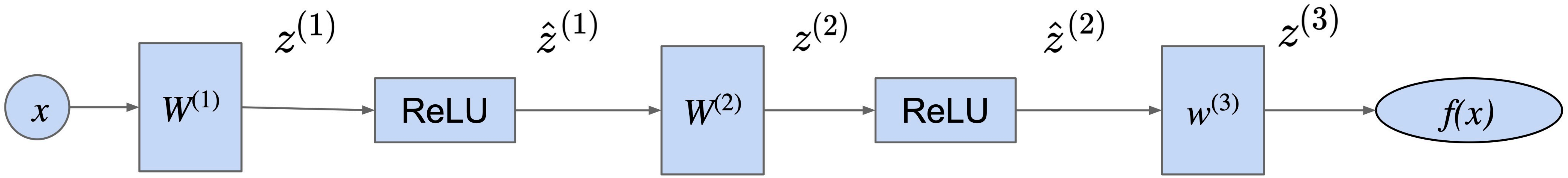


$$f(x) \geq w^{(3)\top} D^{(2)} W^{(2)} D^{(1)} W^{(1)} x + \text{const.}$$

The rest layers follow the same way of propagating bounds

CROWN backward bound propagation

- Goal: find linear relationships between output and every hidden neuron, until we reach the input!



$$f(x) \geq \boxed{w^{(3)\top} D^{(2)} W^{(2)} D^{(1)} W^{(1)}} x + \text{const.}$$

CROWN linear bound: $\min_{x \in \mathcal{C}} f(x) \geq \min_{x \in \mathcal{C}} \boldsymbol{a}_{\text{CROWN}}^{\top} x + c_{\text{CROWN}} := \min_{x \in \mathcal{C}} f_{\text{CROWN}}(x)$

Where $\boldsymbol{a}_{\text{CROWN}}$ and $\boldsymbol{c}_{\text{CROWN}}$ are functions of NN weights, and can be computed efficiently on GPUs in a backward manner

The CROWN lower bound

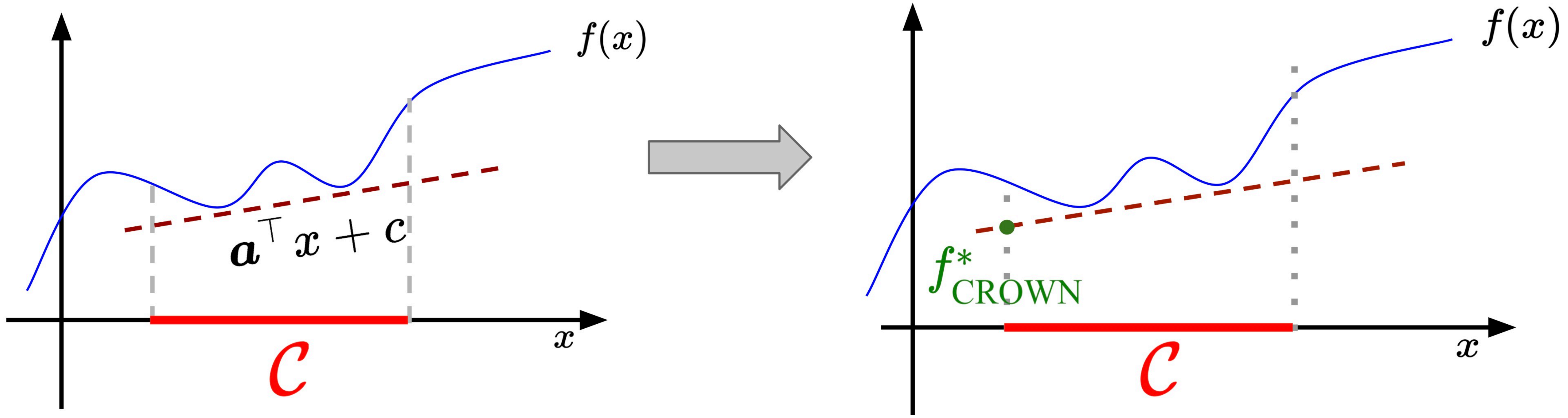
Linear Bound: $f_{\text{CROWN}}(x) = \boldsymbol{a}_{\text{CROWN}}^{\top} x + c_{\text{CROWN}}$

Final lower bound by solving an easier linear optimization problem:

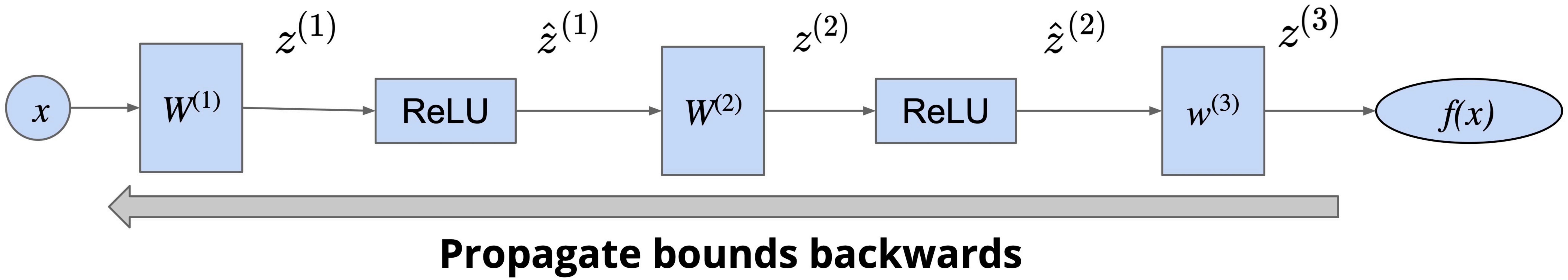
$$f^*_{\text{CROWN}} = \min_{x \in \mathcal{C}} \boldsymbol{a}_{\text{CROWN}}^{\top} x + c_{\text{CROWN}}$$

Simple closed form for ℓ_∞ norm perturbation $x \in \{x \mid \|x - x_0\|_\infty \leq \epsilon\}$

$$f^*_{\text{CROWN}} = -\|\boldsymbol{a}_{\text{CROWN}}\|_1 \epsilon + \boldsymbol{a}_{\text{CROWN}}^{\top} x_0 + c_{\text{CROWN}}$$



The CROWN lower bound



$$\min_{x \in \mathcal{C}} f(x) \geq \min_{x \in \mathcal{C}} \boldsymbol{a}_{\text{CROWN}}^{\top} x + c_{\text{CROWN}} := \min_{x \in \mathcal{C}} f_{\text{CROWN}}(x)$$